MSc. Thesis

Integrating Multi-Agent Task Allocation and Path Planning to Minimize Delays of Aircraft Engine-Off Towing Operations

Maartje Duzijn

Delft University of Technology

TU Delft

to70

MSc. Thesis

Integrating Multi-Agent Task Allocation and Path Planning to Minimize Delays of Aircraft Engine-Off Towing Operations

by

Maartje Duzijn

to obtain the degree of Master of Science
in Control & Operations at the Faculty of Aerospace Engineering, Delft University of Technology,
to be defended publicly on June 12th, 2023.

Student number:	4486285	
Project duration:	August 2022 - June 2023	
Thesis Committee:	Dr.ir. Bruno Santos	TU Delft, Assistant Professor (Chair)
	Dr. Barys Shyrokau	TU Delft, Assistant Professor (Examiner)
	Dr. Alexei Sharpanskykh	TU Delft, Assistant Professor (Supervisor)
	Ir. Patricia Martín-Fernandez	To70 Aviation, Senior Aviation Consultant (Supervisor)
	Ir. Malte von der Burg	TU Delft, PhD candidate (Supervisor)

Cover image obtained from: https://news.klm.com/klm-trials-sustainable-taxiing/
An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

Preface

Dear reader,

This thesis documents the work that has been carried out to obtain a Master of Science degree in Aerospace Engineering at Delft University of Technology. In this work, the implementation of tug-enabled taxiing operations at airports has been explored, both from an algorithmic and operational perspective. I hope that this thesis stimulates further discussions, research, and advancements in the field of tug-enabled taxiing, ultimately leading to a more sustainable and efficient aviation industry.

Upon reflecting on a year of dedicated effort, I am happy to say that I am proud of the final outcome that now lies before you. However, this achievement would not have been possible without the guidance and support of my supervisors. First of all, I am very grateful for the supervision of dr. Alexei Sharpanskykh during this thesis. Thanks for your continuous support, both on the content of my research and on a personal level. You have helped me to regain confidence in the project when I lost it and to reflect on what is most important in the process. I hope to keep in touch with you in the future!

Furthermore, I would like to thank Patricia Martin-Fernandez for her supervision and guidance from an operational perspective. Your experience and constructive feedback have not only been useful, but have also emphasized the possible operational value of this research, serving as a continuous source of motivation. In addition, I want to thank you for your availability and support whenever I encountered challenges, even with your own demanding workload. Hopefully, we will meet in Madrid in the near future!

Last but not least, I would like to express my appreciation to Malte von der Burg for his contribution to this work. Your critical eye for detail made sure that all loose ends were being questioned. Especially in the final stages of this project, your feedback was key in bringing all the elements together and allowing me to take pride in the final outcome. I wish you the best of luck in your ongoing pursuit of a PhD!

Finally, this document does not only mark the end of my thesis, but my time at Delft University of Technology as a student as well. Thanks to everyone who was part of this journey over the past 8 years, I would not have wanted to do it without you!

Maartje Duzijn
Delft, June 2023

Contents

List of Figures

List of Tables

List of Abbreviations

A-CDM	Airport Collaborative Decision Management.
A-SMGCS	Advanced Surface Movement Guidance and Control System.
AAS	Amsterdam Airport Schiphol.
ABFP	Action-Based Fictitious Play.
ACARS	Aircraft Communications and Reporting System.
ACI	Airports Council International.
ACO	Ant Colony Optimization.
AEON	Advanced Engine Off Navigation.
AGHT	Actual Ground Handling Time.
AGV	Automated Guided Vehicle.
AIBT	Actual In-Block Time.
AIP	Aeronautical Information Package.
ALDT	Actual Landing Time.
AMAPF	Anonymous Multi-Agent Path Finding.
AMS	Amsterdam Airport Schiphol.
AO	Aircraft Operator.
AOBT	Actual Off-Block Time.
APU	Auxiliary Power Unit.
ARDT	Actual Ready Time for Movement.
ASAT	Actual Start Up Approval Time.
ASRT	Actual Start Up Request Time.
ATA	Ant Task Allocation.
ATC	Air Traffic Control.
ATM	Air Traffic Management.
ATO	Air Transport & Operations.
ATOT	Actual Take Off Time.
AXIT	Actual Taxi-In Time.
AXOT	Actual Taxi-Out Time.
BADA	Base of Aircraft Data.
BLE	Broadcast of Local Eligibility.
CBAA	Consensus-Based Auction Algorithm.
CBBA	Consensus-Based Bundle Algorithm.
CBS	Conflict-Based Search.
CBS-TA	Conflict-Based Search with Task Assignment.
CD	Complex Dependencies.
CDM	Collaborative Decision Management.
CNP	Contract Net Protocol.
CPDLC	Controller-Pilot Data Link Communications.
CPDSP	Collaborative Pre-Departure Sequence Planning.
CT	Constraint Tree.
CTOT	Calculated Take Off Time.
CTPC	Cumulative Target Path Cost.
DNE	Do Not Enter.
DNP	Do Not Persist.
DNW	Do Not Wait.
ECBS	Enhanced Conflict-Based Search.

ECBS-CT Enhanced Conflict-Based Search for Continuous Time.
ECBS-TA Enhanced Conflict-Based Search with Task Assignment.
EDMS Emissions and Dispersion Modelling System.
EFS Electronic Flight Strips.
EIBT Estimated In-Block Time.
ELDT Estimated Landing Time.
EOBT Estimated Off-Block Time.
ETOT Estimated Take Off Time.
ETTT Estimated Turn-Round Time.
EXIT Estimated Taxi-In Time.
EXOT Estimated Taxi-Out Time.

FAA Federal Aviation Administration.
FIR Flight Information Region.
FPL Filed Flight Plan.
FSF Flight Safety Foundation.

GH Ground Handler.
GSE Ground Service Equipment.

IA Instantaneous Assignment.
IAI Israeli Aerospace Industries.
IATA International Air Transport Association.
ICAO International Civil Aviation Organization.
ICBS Improved Conflict-Based Search.
ID In-Schedule Dependencies.
IDMB Improved Distributed Market-Based.
IFR Instrument Flight Rules.

KDC Knowledge and Development Center.
KLM Koninklijke Luchtvaart Maatschappij.
KPI Key Performance Indicator.

MAMP Multi-Agent Motion Planning.
MAPD Multi-Agent Pickup and Delivery.
MAPD-TD Multi-Agent Pickup and Delivery with Task Deadlines.
MAPF Multi-Agent Path Finding.
MAPF-POST Post processing of MAPF plans.
MAPF-TN Multi-Agent Path Finding with Transit Networks.
MAS Multi-Agent System.
MATA Multi-Agent Task Allocation.
MILP Mixed-Integer Linear Programming.
MR Multi-Robot Tasks.
MRTA Multi-Robot Task Assignment.
MT Multi-Task Robots.
MTOW Maximum Take-Off Weight.
mTSP Multiple Traveling Salesmen Problem.
MTTT Minimum Turn-round Time.
MTWPS Modified Two-Part Wolf Pack Search.

NB Narrow-Body.
ND No Dependencies.
NextGen Next Generation Air Transportation System.
NMOC Network Manager Operations Centre.

OAP Optimal Assignment Problem.

PBS Priority-Based Search.

PSO Particle Swarm Optimization.
PT Priority Tree.
pTeSSB Probabilistic Temporal Sequential Single Bundle.
pTeSSI Probabilistic Temporal Sequential Single-Item.

RM Regret Matching.
RMCA Regret-Based Marginal-Cost Based Task Assignment.
RMCA(r) Relative Regret-Based Marginal-Cost Based Task Assignment.
RMO Runway Mode of Operation.
RPC Robot Path Cost.
RPK Revenue Passenger-Kilometer.
RPW Red Palm Weevil.
RWY Runway.

SAMP Single-Agent Motion Planning.
SAPF Single-Agent Path Finding.
SAS Smart Airport Systems.
SBB Sequential Bundle-Bid.
SCIPP Soft Conflict Safe Interval Path Planning.
SESAR Single European Sky ATM Research.
SET Single-Engine Taxiing.
SGA Sequential Greedy Algorithm.
SID Standard Instrument Departure.
SIPP Safe Interval Path Planning.
SIPPwrRT Safe Interval Path Planning with Reservation Table.
SR Single-Robot Tasks.
sSAP Selective Spatial Adaptive Play.
SSC Sequential Single-Cluster.
SSI Sequential Single-Item.
ST Single-Task Robots.
STN Simple Temporal Network.
STNU Simple Temporal Network with Uncertainty.

TA Time-Extended Assignment.
TA-H Task Assignment - Hybrid.
TA-P Task Assignment - Prioritized.
TAPF Target Assignment and Path Finding.
TeSSI Temporal Sequential Single-Item.
TOBT Target Off-Block Time.
TP Token Passing.
TPC Target Path Cost.
TPTS Token Passing with Task Swaps.
TSAT Target Start Up Approval Time.
TSP Traveling Salesmen Problem.
TTD Total Travel Delay.
TTOT Target Take Off Time.

UAV Unmanned Aerial Vehicle.
UBFP Utility-Based Fictitious Play.
UTARB Multi-UAV Task Allocation for RPW Combat Based on Bacteria Behaviour.

VHF Very High Frequency.
VRP Vehicle Routing Problem.
VTT Variable Taxi Time.

WB Wide-Body.
WTC Wake Turbulence Category.

XD Cross-Schedule Dependencies.

Scientific Paper

Integrating Multi-Agent Task Allocation and Path Planning to Minimize Delays of Aircraft Engine-Off Towing Operations

M.F. Duzijn* Ir. P.A. Martín-Fernandez† Dr. O.A. Sharpanskykh‡
Ir. M.F. von der Burg§

Delft University of Technology

Abstract

The Multi-Agent Pickup and Delivery (MAPD) problem has received significant academic attention, resulting in a variety of solution techniques that both assigns tasks to agents and finds a conflict-free routing plan for all agents in the system. However, limited research is done to investigate the added value of an integrated MAPD solving technique for real-world problems in which task allocation and path planning are performed simultaneously. The integration of the assignment and routing problem is relevant for the minimization of ground delay when implementing tug-enabled taxiing operations at airports to reduce the environmental impact of aircraft ground movements. In this work, a novel algorithmic framework is introduced that minimizes ground delay for tug-enabled taxiing compared to conventional taxiing operations by combining the Temporal Sequential Single-Item (TeSSI) allocation algorithm with Priority Based Search (PBS) and Safe-Interval Path Planning (SIPP) for path planning. Experiments were conducted to assess the performance of a coupled approach, where the assignment and routing problem are solved together, in comparison to a decoupled approach, where they are solved separately. The evaluation focused on ground delay for different towing fleet sizes. The findings indicate that the coupled approach does not offer significant advantages compared to a decoupled approach in the domain of aircraft engine-off towing operations: the decrease in total ground delay for a few specific cases was minimal, while the computational run times significantly increased across all scenarios. Key operational results highlight that, when tugs are limited, the primary contributor to the total ground delay per departing flight is the ramp delay caused by tug unavailability. Furthermore, the relationship between ramp delay and fleet size is not linear but exhibits exponential growth only after reducing the fleet size below a certain threshold of available tugs. These practical observations provide relevant insights for airports considering implementation of tug-enabled taxiing operations.

1 Introduction

The growth in air transportation demand over the past decades up to 2019 (pre-pandemic levels) has led to airports approaching capacity and environmental limits [1]. At the same time, the International Civil Aviation Organization (ICAO) does not only expect complete recovery of air passenger demand to pre-pandemic levels in the beginning of 2023, but also predicts growth of around 3% on 2019 figures [2]. Therefore, a need arises for airports to simultaneously increase capacity and sustainability of their operations. From an environmental point of view, significant gains are expected in the area of airport ground surface movements, since aircraft taxi-in and taxi-out times account for almost 30% of an airport's carbon emissions [3]. At the same time, one of Europe's largest hub airports, Amsterdam Airport Schiphol (AMS), has set the ambition to achieve emission-free ground surface operations by 2030 [4].

Therefore, existing research has investigated the possibilities of engine-off taxiing techniques to reduce the environmental impact of conventional taxiing operations, while maintaining at least a pre-pandemic capacity level [5]. A distinction between on-board and external engine-off towing techniques can be made. While on-board techniques are anticipated to yield greater reductions in emissions, currently the most promising option is an external technique that involves the use of a towing tug, as this method is the

*MSc Student Air Transport & Operations, Delft University of Technology (Delft)
†Senior Aviation Consultant, To70 (The Hague)
‡Assistant Professor, Delft University of Technology (Delft)
§PhD Candidate, Delft University of Technology (Delft)

only one that has received full certification thus far [6]. The certification holds for the TaxiBot concept, that involves towing of the aircraft from the ramp towards a decoupling location near the runway. Trials performed at AMS using TaxiBots show an average of 65% fuel reduction for runways identified as most promising for tug-enabled taxiing. However, the implementation of tug-enabled taxiing is considered to be a capital intensive concept, due to the purchasing, maintenance and operating costs of tugs [7]. Therefore, a need arises to analyze operational performance of the airport system for limited number of tugs available in terms of ground delay when using tug-enabled taxiing instead of conventional taxiing techniques.

Insights into the relationship between the amount of tugs available and the propagation of ground delay at an airport can be obtained by simulation. The geographically distributed nature of the tug-enabled taxiing problem combined with its dynamic nature, make the problem especially suitable for agent-based modelling and simulation [8]. Within the agent-based modelling paradigm, the tug-enabled taxiing problem can be described as a Multi-Agent Pickup and Delivery (MAPD) problem. This problem is a well-studied problem in the academic literature and involves assigning tasks to agents and finding a conflict-free routing plan for all agents that ensures successful completion of all tasks [9]. A distinction in solving approaches for an MAPD problem can be made between techniques that solve the allocation and routing problem separately (decoupled approaches) [9, 10, 11] and techniques that solve both parts of the problem simultaneously (coupled approaches) [12, 13, 14]. In theory, combining the assignment and routing problem using a coupled approach would yield improved solutions, as the assignment decisions would be based on real costs instead of relying solely on lower-bound costs using estimated pickup and delivery times [14]. However, limited research has been done on the application of coupled MAPD solving techniques on real-world applications in complex environments. Therefore, this work aims to contribute towards the development of solution techniques *to analyze the propagation of ground delay for tug-enabled taxiing operations using a limited fleet of towing tugs by designing and evaluating an approach for integration of task assignment with path planning using a multi-agent control architecture.*

In order to address the research goal, the following steps were performed. First, a concept of operations for conventional taxiing and tug-enabled taxiing was defined. Second, a Multi-Agent System (MAS) simulation model was created to investigate these two concepts, building upon existing research as a foundation [15, 16]. Next, a novel algorithmic framework was developed especially for airport ground surface movement operations that allows for the integration of the assignment and routing problem. The framework makes use of an adapted version of the Temporal Sequential Single-Item (TeSSI) [17] auction algorithm for allocation, because of its scalability, computational efficiency and ability to deal with tasks characterized by pick-up deadlines [18]. Previous research on the planning of vehicles in the tug-enabled taxiing problem has demonstrated the effectiveness of combining Priority Based Search (PBS) with Safe-Interval Path Planning (SIPP) for dealing with the routing problem [15, 16]. Therefore, in this study, an adapted version of PBS + SIPP is also implemented for path planning. After verification and validation of the MAS and algorithmic model, we analyzed 14 scenarios to evaluate the performance of the algorithmic framework in terms of ground delay for various fleet sizes and to provide operational insights related to the implementation of tug-enabled taxiing at large airports.

The structure of this paper is as follows: in section 2, the concept of operations for both conventional taxiing and aircraft engine-off towing will be described, as well as the airport that is used as a case study for analysis. Subsequently, the MAS used for simulation experiments is discussed in section 3. Next, the algorithmic model that is embedded into the MAS is elaborated on in section 4. Verification and validation procedures of the MAS is elaborated on in section 5. The experimental setup is then presented in section 6, and the corresponding results are discussed in section 7. Finally, a discussion of the study is presented in section 8 and the conclusions are discussed in section 9.

2 Case Study for Tug-Enabled Taxiing Operations

In this section, the conceptual model for simulating tug-enabled taxiing and its characteristics is described. First, a general concept of operations will be provided in subsection 2.1 for conventional taxiing and aircraft engine-off towing operations, including an outline of the most important assumptions related to these operations. Subsequently, the concept of tug-enabled taxiing at Amsterdam Airport Schiphol (AMS) specifically will be discussed in subsection 2.2. A list of all assumptions related to the concept of operations can be found in Appendix A.

2.1 Concept of Operations for Conventional Taxiing and Tug-Enabled Taxiing

In this section, an overview will be provided of the concept of operations for both conventional taxiing and tug-enabled taxiing. Figure 1 shows the procedures for outbound conventional taxiing that are modelled in this study. The Target Startup Approval Time (TSAT) as provided by Air Traffic Control (ATC) is used as starting point. To account for engine-start in the conventional scenario when Multi-Engine Taxiing (MET) is used, the start of the taxiing activity of the aircraft from its starting location towards the runway entry is delayed by 100 seconds. For inbound flights, it is assumed that the entire engine cool-down procedure is performed during the taxi-in movement.

Note that for both conventional taxiing and tug-enabled taxiing operations, we assume that aircraft can leave the ramp nose-first since no apron operations (pushback and push-pull maneuvers) are included in the simulation model. As a result, all ramps are modelled as meta-ramps that represent a group of aircraft ramps for a specific bay area, which do not require pushback or push-pull procedures.

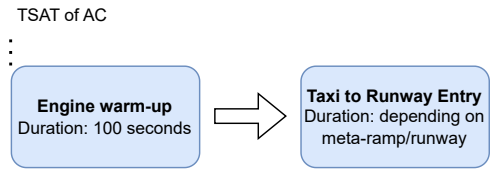


Figure 1: Overview of conventional taxiing operations using Multi-Engine Taxiing (MET).

In Figure 2, the 5 different stages of outbound tug-enabled taxiing operations are shown. The overall procedure for tug-enabled taxiing is based upon previous work on aircraft engine-off towing operations [15, 16]. Similarly as for MET operations, the starting time point of tug-enabled taxiing operations from the aircraft perspective is the TSAT. From the perspective of the tug, the operation starts with the movement of the tug from the tug parking facility to the ramp where the aircraft is located through service roads. When the tug is moving over the service road network and not coupled to an aircraft, the tug is controlled by the tug driver and drives in *solo mode*. The coupling maneuver of the tug to the aircraft starts if the tug is present at the ramp and not before $t_{\text{couple, start}}$ as defined in Equation 1, where t_{coupling} represent the duration of coupling (120 seconds). After coupling, the tug is driving in *pilot mode* (controlled by the pilot) over the taxiway network to a designated decoupling location in vicinity of the assigned runway. When arrived at the decoupling location, the aircraft-tug combination decouples, after which the aircraft resumes its journey towards the runway entry powered by its own engines. Depending on its schedule, the tug either returns to the parking facility or moves to a ramp to attend to the next outbound flight assigned through service roads.

In the remainder of this study, only outbound towing is considered, since the expected benefits of outbound towing are significantly higher than for inbound towing in terms of fuel reduction [19, 20, 21] and because implementation of inbound towing at airports is generally considered to be a very long-term goal [22].

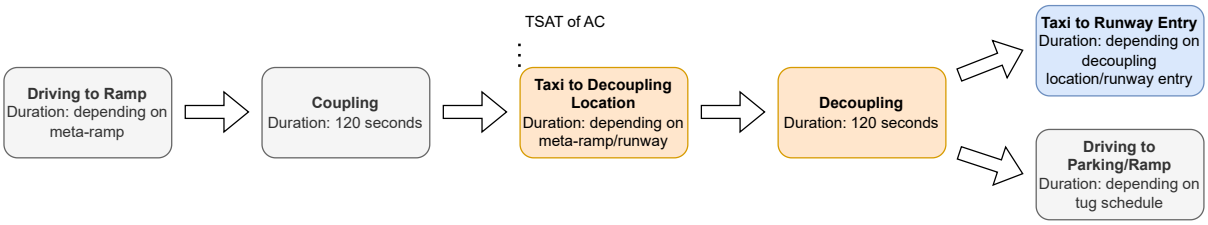


Figure 2: Overview of tug-enabled taxiing operations. Grey blocks represent maneuvers executed by the tug only, orange blocks represents maneuvers executed by the tug-aircraft combination and blue blocks represent maneuvers by the aircraft only.

$$t_{\text{couple, start}} \geq TSAT - t_{\text{coupling}} \tag{1}$$

When comparing conventional with tug-enabled taxiing operations, a number of differences can be identi-

fied. First, the maximum velocity of aircraft taxiing using own engine power is higher than the maximum velocity reached by towing tugs (60 km/h versus 45 km/h). As a result, taxi times are expected to increase when comparing tug-enabled taxiing with conventional taxiing operations. Furthermore, decoupling on the taxiway network is expected to disturb traffic flow around runway entries. Finally, the performance of tug-enabled taxiing operations is expected to be directly influenced by the number of available tugs, as the dependency on tugs to start the movement of the aircraft towards the runway can result in delays.

2.2 Tug-Enabled Taxiing at Amsterdam Airport Schiphol

In the remainder of this work, Amsterdam Airport Schiphol (AMS) is used as case study to evaluate the effects of implementing aircraft engine-off towing operations. AMS is one of Europe's largest hub airports in terms of infrastructure and passenger flow [23], providing for a challenging case in which the concept of aircraft engine-off towing operations will be tested under demanding circumstances. In addition, Royal Schiphol Group and its partners have performed several TaxiBot trials where single outbound flights were towed from the ramp to a designated runway, providing for relevant operational data [24, 25].

In this study, tug-enabled taxiing operations will be compared to conventional taxiing operations during an outbound and inbound peak. At AMS, a large variety of runway configurations exist. Based on feasibility studies regarding the implementation of tug-enabled taxiing operations at AMS [24, 25], it was concluded that a runway configuration for which runway 36L and/or 36C are used as departing runways is most promising. The reason for this is threefold: 1) both runways are characterized by long taxi-out times, offering the highest potential fuel savings; 2) infrastructure where decoupling operations can take place without blocking other traffic on taxiways is present near both runways; and 3) runway 36L has a preferred status for departing flights in peak hours, and therefore, is in use most of the time [25]. Based on this reasoning, Runway Mode of Operation (RMO) North will be studied in this work, where outbound flights depart from 36L and 36C, and inbound flights arrive at 06 and 36R (Figure 3).

For departing runways 36L and 36C, four decoupling locations per runway have been defined based on feasibility analysis of Royal Schiphol Group [25] and previous work done by Soomers [15] and Kamphof [16]. The exact locations and associated assumptions can be found in Appendix A.

(a) Inbound peak (A: 06 + 36R, D:36L)

(b) Outbound peak (A: 06, D: 36L + 36C)

Figure 3: Runway usage in RMO North (inbound peak and outbound peak): Polderbaan (36L), Aalsmeerbaan (36R), Kaagbaan (06) and Zwanenburgbaan (36C) in use.

3 Multi-Agent System

In this section, the Multi-Agent System (MAS) used for modelling tug-enabled taxiing operations is described in more detail. An overview of the high-level structure of the MAS is provided in Figure 4. The MAS consists of an environment and different types of agents, that are able to interact with the environment and with each other. The environment consists of the airport network and the flight schedule. The details on the elements of the environment are provided in subsection 3.1.

Two types of agents can be distinguished in the MAS: Central Agents and Individual Agents. The Airport Operations Agent, Tug Allocation Agent and Routing Agent can all be considered Central Agents that are responsible for the overall planning and coordination of all vehicles in the system.

Offline planning is performed for the duration of a planning window w_{plng} at a fixed planning frequency h_{plng}. In this planning window, the Central Agents are charged with defining the set of aircraft to route based on the flight data (Airport Operations Agent), the allocation of outbound flights to tugs (Tug Allocation Agent) and the actual routing of all vehicles avoiding any conflicts between them (Routing Agent). Individual Agents are agents that represent the actual vehicles moving in the system (Aircraft and Tug Agents) and are charged with execution of the plan developed by the Central Agents. In subsection 3.2, an explanation is given on the exact role of each agent in more detail.

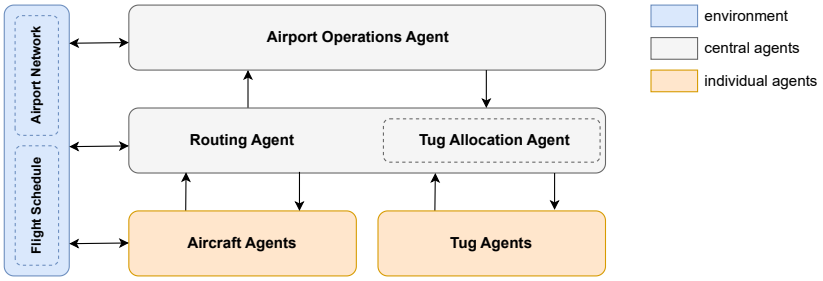


Figure 4: Overview of the structure of the multi-agent system used.

3.1 Environment Specifications

The environment of the MAS consists of two objects: the airport network graph and the historic flight schedule.

3.1.1 Airport Network

The layout of AMS is represented by a static graph $G = (V, E)$ consisting of 237 nodes and 390 bidirectional edges as shown in Figure 5. Given that apron operations are excluded from the model, aircraft ramps in a specific bay area are mapped to meta-ramps, shown in white in Figure 5 (for the exact mapping, refer to Appendix A). Connections between nodes are represented by different types of edges. All edges are bidirectional and can be traversed in both directions, except for a number of apron entries and exits (refer to Appendix A).

Taxiway and runway edges are both shown in black by a thin or thick line and obtained from their actual locations using geographical information. Crossings and turns are represented in a simplified manner, excluding actual turn angles. Restricted turns for aircraft on the taxiway network are included as layout constraints, as well as restrictions on the required heading for aircraft and tugs to enter decoupling locations. Finally, layout constraints are put in place that restrict movements over certain taxiways that are closed when operating RMO North. More information on the exact layout constraints can be found in Appendix A. Based on the layout constraints, the shortest feasible path between any two nodes in the network is precomputed to limit computational complexity during simulation.
Next to taxiways and runways, service roads are modelled as well and depicted by a red line. The service road infrastructure consists of the central service road at AMS and service road edges around departing runways 36L and 36C. Although parts of the service road infrastructure around these runways is currently non-existing, it is included in the model to allow for tug return movement after decoupling [25]. For more details on the assumptions related to the graph network, refer to Appendix A. Note that the graph network representation is fully accessible to the the Airport Operations Agent, Routing Agent and Tug Allocation Agent.

3.1.2 Flight Schedule

The flight schedule contains all relevant information on the flights that depart or arrive in the current planning window, including the flight ID, flight direction, ICAO Aerodrome Reference Code, the wake turbulence category, assigned ramp and assigned runway. For all flights, the assigned ramp is mapped to a meta-ramp node location. Based on the ICAO Aerodrome Reference Code and operational experts, a set of possible runway entries and exits is defined for each flight. In addition, the flight schedule contains the landing time for inbound flights and the TSAT for outbound flights. Note that in the flight schedule, only flights that depart from or arrive at a runway that is active in RMO North are included. Similarly,

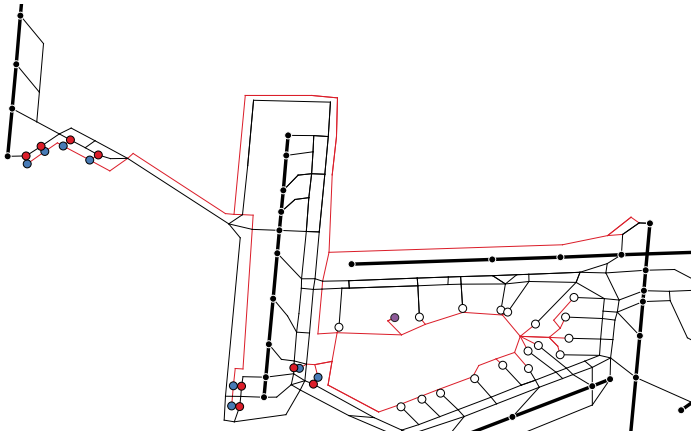


Figure 5: Part of graph network of AMS, showing taxiway edges (black, thin line), service road edges (red, thin line), runway edges (black, thick line), runway nodes (black), meta-ramp nodes (white), decouple nodes (red), all-clear nodes (blue) and tug base node (purple).

General Aviation flights are excluded. Additional assumptions, analysis and details on the input flight schedule can be found in Appendix B.

The flight schedule is considered deterministic and thus, can be classified as a static object in the environment of the MAS. During every replanning instance, the information in flight schedule is accessible for the Airport Operations Agent, Routing Agent and Tug Allocation Agent for the duration of the current planning window.

3.2 Agent Specifications

Next to the environment, a MAS is made up of agents, that can be described as autonomous actors in the environment. In this section, an overview will be given of all agents present in the system. Furthermore, the interaction between the agents is elaborated on as well. In Appendix A, a more elaborate description of the characteristics and properties for all agents can be found.

3.2.1 Airport Operations Agent

The role of the Airport Operations Agent is to define the set of flights to be routed for the current planning window w_{plng}. Based on the status of all vehicles provided by the Routing Agent, the Airport Operations Agent checks for each aircraft that is planned to spawn before the end of the planning window if it has arrived at its goal location yet. The output of the Airport Operations Agent (a set of flights to route) is communicated with the Tug Allocation Agent (outbound flights only) and the Routing Agent (all flights).

3.2.2 Routing Agent

The role of the Routing Agent is to execute the routing algorithm to find a conflict-free path for all vehicles to route in the current planning window. The Routing Agent receives information on the set of aircraft to route from the Airport Operations Agent. If we are considering a scenario where outbound towing is performed, the Routing Agent communicates the outbound flights in this set of aircraft to the Tug Allocation Agent and triggers the execution of the allocation algorithm. For every tug that is assigned to at least one outbound flight, the Routing Agent receives a set of maneuvers to be performed by the tug from the Tug Allocation Agent, based on the current position and status of the tug. For more information on the sequence of maneuvers to be executed by tugs, refer to section 2.

Once the total set of vehicles to route is defined, the Routing Agent creates a set of individual agents (Aircraft and Tug Agents) that represent the vehicles to route. Next, the Routing Agent triggers the path planning algorithm to find a conflict-free routing plan for all tugs and aircraft to route in the current planning window. For each vehicle, the routing plan describes the route and time for every maneuver that it has to perform. The working principle of the PBS + SIPP path planning algorithm is described in more detail in subsection 4.3. The Routing Agent communicates the solution found to all Aircraft and Tug Agents.

In the simulation model, the possibility exists to explore the consequences of multiple different allocations on the planning of paths for the vehicles to route in parallel. The details of the working principle will be explained in greater depth in section 4. The Routing Agent will identify the need for an additional allocation to be explored based on a set of conditions that is described in greater detail in section 4. Once the conditions are met, the Routing Agent triggers the Tug Allocation Agent to initiate the allocation algorithm and provides this agent with a set of constraints to prevent an allocation to be generated that has been explored previously. Based on the set of constraints provided by the Routing Agent, the Tug Allocation Agent generates a new allocation. Using the new allocation, the Routing Agent again generates a set of Individual Agents for routing, alongside the existing sets.

3.2.3 Tug Allocation Agent

The Tug Allocation Agent receives a set of outbound flights from the Routing Agent that is planned to spawn before the end of the current planning window and has not arrived at the goal location yet. In addition, the Routing Agent provides information on the status of the vehicles that have been routed so far, including the status of all tugs available for towing.

Based on this information, the Tug Allocation Agent defines an allocation of outbound flights to tugs for the current planning window. This is done by executing the TeSSI auction algorithm. At the beginning of every auction, the Tug Allocation Agent creates a Simple Temporal Network (STN) for every Tug Agent in the fleet of available tugs. The STN is used to keep track of the tug schedule and its previously assigned tasks. Based on its STN, the Tug Agent can determine what is the optimal position to insert a task up for auctioning into its current schedule. If the tug under consideration has already started a coupling, the task cannot be reallocated anymore and is inserted in its schedule as a fixed task. During every auction round, all Tug Agents submit a bid to the Tug Allocation Agent for every aircraft in the set of unallocated outbound flights. Then, a winning tug is determined that gets assigned the task to tow the aircraft from the meta-ramp location to a decoupling location in vicinity of the designated runway. The winning tug inserts the assigned task in its STN. This is repeated until all outbound flights are assigned to a Tug Agent. More details on the working principle of the TeSSI auction algorithm and the use of STNs is provided in section 4.

The final allocation is shared with the Routing Agent that is responsible for the routing of the agents based on the defined allocation.

3.2.4 Aircraft and Tug Agents

Both Aircraft and Tug Agents have the role to represent the actual vehicles that are routed in the current planning window. When a solution is found, the agents receive information on the route to execute in the next planning window from the Routing Agent. During execution, the agents move according to their routing plan and communicate their current status and position back to the Routing Agent.

Note that Tug Agents also participate in the auction process led by the Tug Allocation Agent. During this process, the Tug Agents receive information on the task that it gets assigned from the Tug Allocation Agent. In addition, Tug Agents provide the Tug Allocation Agent with their bid for all aircraft that are not yet assigned to any other tug.

4 Integration of Multi-Agent Task Allocation and Path Planning

The concept of tug-enabled taxiing operations for a finite fleet of tugs can be formalized as a Multi-Agent Pickup and Delivery (MAPD) problem. In MAPD problems, a set of agents $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ is charged with execution of a set of unexecuted tasks $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_m\}$ in a network. The network consists of an undirected graph $G = (V, E)$, whose vertices V correspond with node locations and edges E correspond with connections between locations. Every task $\tau_j \in \mathcal{T}$ is characterized by a pickup vertex $s_j \in V$ and a delivery vertex $g_j \in V$. Typically for MAPD problems, the total number of tasks in the system is greater than the number of agents [9], resulting in agents having to attend to a stream of tasks while avoiding collisions with other agents. Note that in this work, the MAPD problem is modelled in an offline setting, meaning that for a certain time interval or planning window w_{plng}, all characteristics of tasks scheduled in that time interval are known at the time of allocation [26].

A solution to the MAPD problem both defines an allocation of agents to tasks and a conflict-free routing plan for all agents in the system. In the existing literature, a distinction can be made in MAPD

solution techniques that solve task assignment and path finding consecutively (decoupled approaches) [9, 10, 11] and approaches that integrate allocation of tasks with path finding (coupled approaches) [12, 13, 14]. The main advantage of the latter approach is that the assignment choices are not only informed by lower-bound estimates of path costs associated with the execution of the task at hand, but reflect additional path costs resulting from interaction and conflict resolution between agents. However, the use of a coupled approach increases the computational complexity.

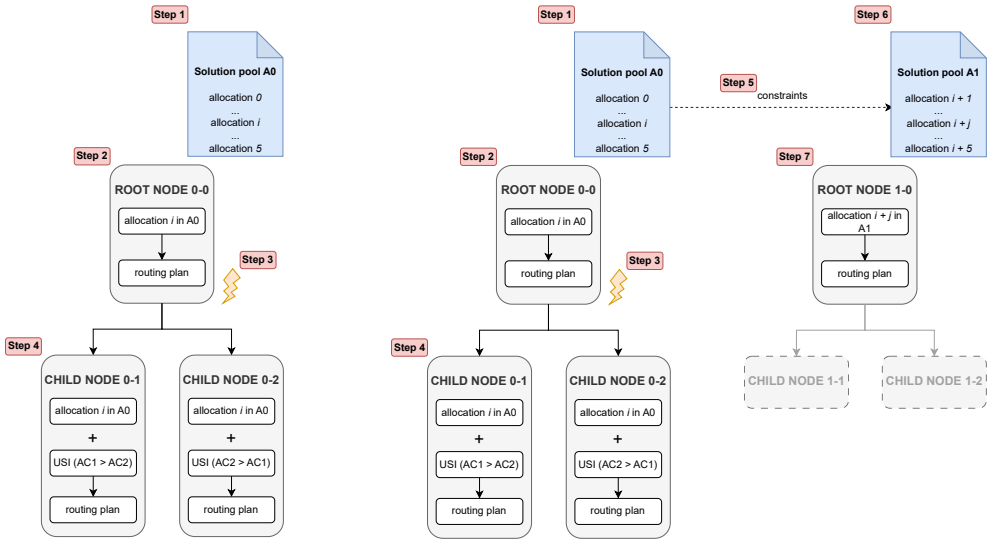
In this work, a novel algorithmic framework, Priority-Based Search including Task Allocation (PBS-TA), is developed that allows for the comparison of a decoupled and coupled MAPD solving technique in the context of tug-enabled taxiing operations. In this section, more details are provided on the general working principle of the PBS-TA framework (subsection 4.1), the allocation algorithm implemented (subsection 4.2) and the algorithm used for path planning (subsection 4.3). The pseudocode of the adapted versions of all algorithms can be found in Appendix A.

4.1 General Outline of the PBS-TA Framework

In this section, we present an outline of the PBS-TA framework that is capable of solving the MAPD problem at hand in both a coupled and decoupled manner (subsection 4.1.1). Within the framework, three different algorithms are tested on their algorithmic and operational performance with respect to delay minimization for aircraft engine-off towing operations. The characteristics of the three algorithms are discussed in subsection 4.1.2.

4.1.1 Integration of Task Allocation with Path Planning

In order to solve the task allocation and path planning problem in an integrated/coupled manner, the PBS-TA framework makes use of a search forest for path planning instead of a single search tree used in conventional search algorithms. The search forest principle is inspired by the work of Hönig et al. [13] and based on the idea that interaction between agents is not taken into account when assigning tasks. However, when interaction between agents is considered in path planning, the initial (optimal) allocation found might not result in a global optimum due to rerouting of agents with respect to their shortest paths. By using a search forest in the coupled approach, multiple possible task assignments and their resulting path conflicts can be explored simultaneously. Every additional tree in the forest analyzes the path planning problem for a next best allocation of agents to tasks. The addition of a tree to the search forest based on a yet unexplored allocation is triggered iff a root node of a previously added search tree is expanded and a conflict is found. As a consequence, a factorial explosion of all possible task assignments is avoided, since a new allocation is only explored on-demand [13]. Note that for the decoupled approach, only a single allocation (and thus, a single search tree) is explored in the path planning domain.



(a) Decoupled approach

(b) Coupled approach.

Figure 6: Overview of the working principle of the PBS-TA algorithmic framework for a decoupled (a) and coupled (b) approach.

In Figure 6, a visualization is shown of the working principle of the PBS-TA framework, for both the decoupled and coupled approach. In both approaches, the algorithm is initiated with an initial set of allocations $A0$ (step 1), containing a number of different assignments of all flights in the current planning window to tugs. The generation of the allocation solution pool will be elaborated on in subsection 4.2.3. From the solution pool, a preferred allocation i is chosen that is used to generate a routing plan for all vehicles in the root node 0 of the first search tree in the path planning forest (step 2) (refer to subsection 4.2.4 for more details on this step). If a conflict is found in the routing plan (step 3), two child nodes are generated that explore the consequences of resolving this conflict (step 4). Up until this point, we are executing a standard (decoupled) search procedure based on a single search tree. However, for a coupled approach, an additional root node is created as well and added as a new search tree to the forest. To do so, a new task allocation is needed and found in the following steps: first, a set of constraints is formulated based on the initial set of allocations $A0$ (step 5), to generate a new set of allocations $A1$ (step 6). Again, a preferred allocation is chosen from the set of allocations $A1$, which is used to generate a new routing plan in the newly created root node (step 7). Then, based on the cost of the two child nodes ($0-1$ and $0-2$) and root node $1-0$, the next node to expand is determined as the node with the minimum cost. If the next node to expand is the root node, the procedure is repeated starting from step 3. If the next node is a child node, the search is continued in a depth-first manner.

4.1.2 Algorithms to Evaluate

Based on the PBS-TA framework, three different algorithms are developed and evaluated in this study on their performance in the context of aircraft engine-off towing operations. Table 1 shows the main characteristics of the three algorithms to test. In order to assess the added value of a coupled approach for the integration of task assignment and path planning in the context of airport ground surface movement operations, a decoupled algorithm (DCPL) is used as a baseline and compared against two coupled algorithms (CPL-MIN and CPL-DIF). The coupled algorithms differ in the way how a preferred allocation from the allocation solution pool is chosen. More details on this will be provided in subsection 4.2.3.

All three algorithms make use of an altered version of the auction-based Temporal Sequential Single-Item (TeSSI) algorithm to generate allocations that assign outbound flights to tugs (subsection 4.2). Path planning is done in all three algorithms based on Priority-Based Search (PBS) [27] in combination with Safe Interval Path Planning (SIPP) [28], inspired on previous work related to agent-based modelling of aircraft engine-off towing operations [15, 16]. The implementation of the path planning algorithm will be discussed in detail in subsection 4.3.

Table 1: Characteristics of algorithms to evaluate on their performance for the application in airport ground surface movement operations.

Algorithm	**Approach**	**Size of allocation solution pool**	**Preferred allocation**
DCPL	Decoupled	5	Minimum cost
CPL-MIN	Coupled	5	Minimum cost
CPL-DIF	Coupled	5	Combination of minimum cost and degree of similarity

4.2 Adapted TeSSI for Allocation of Outbound Flights to Tugs

The modelling of tug-enabled taxiing operations requires an allocation method that remains efficient for an increasing number of agents and tasks, while not compromising on the quality of the solutions. In the field of Multi-Robot Task Allocation (MRTA), auctions are widely studied and implemented because of their scalability to large networks, performance in terms of solution quality and applicability to problems where the set of tasks to assign changes over time [29, 30, 31]. Therefore, an auction mechanism is used in which tugs are considered bidders and all outbound flights available in the current planning window are considered tasks to be executed. Since the tasks are not only characterized by spatial attributes (the location of the ramp with respect to the current position of the tug), but also by temporal attributes (the time the aircraft is scheduled to leave the ramp), a need arises for allocation mechanisms that are able to deal with tasks characterized by deadlines. In the work of Nunes and Gini [17], an auction mechanism is proposed that is able to handle temporal constraints for tasks, resulting in the Temporal Sequential Single-Item (TeSSI) algorithm. For every task in the set of unallocated tasks, all agents submit bids for

every task based on their capability of executing the task within the associated time window. In every auction round, one task is assigned to the agent with the best bid. The procedure repeats itself until all tasks are allocated [32].

In the remainder of this section, a number of important aspects of the adapted TeSSI algorithm will be discussed. First of all, the modelling of outbound flights as tasks with temporal constraints is elaborated on in subsection 4.2.1, as well as the mechanism that is used by tug vehicles to keep track of their schedule. Next, more details are provided on the allocation objective and associated bidding rules in subsection 4.2.2. Additionally, the design of the PBS-TA framework requires an efficient way of computing the next best allocation to use in the root node of an additional search tree in the path planning search forest. The setting of constraints and the generation of a next best allocation using the adapted TeSSI algorithm is further discussed in subsection 4.2.3. Finally, the determination of the preferred allocation from the solution pool is discussed in subsection 4.2.4.

4.2.1 Temporal Constraints of Tasks and Maintaining Tug Schedules

Similarly to the original implementation of TeSSI [17], the temporal constraints related to the tasks at hand are modelled as a simple temporal problem [33]. In such, each task τ is specified by a starting and finishing time point, S_τ and F_τ respectively. Both the duration and the earliest and latest starting time of the task are considered a characteristic of the task. When considering outbound flights as the tasks to be assigned, the earliest and latest starting time of the task are characterized by the interval $[TSAT - t_{couple}, \infty)$, where TSAT represents the Target Startup Approval Time and t_{couple} the time needed for coupling. We are interested in the propagation of ground delay and hence, no bound on the latest starting point is defined for the task. The duration of the task is defined as the time needed to perform coupling, move the tug-aircraft combination to a decoupling location near the assigned runway and perform decoupling. To account for unexpected delays en route, an additional buffer of 60 seconds is added to the duration of the task. Although there are multiple decoupling locations available for the two departing runways (subsection 2.2), a preferred location to perform decoupling can be determined for both runways, depending on the scenario to be analyzed. For the determination of the duration of a task, the preferred decoupling location for each runway is used as the goal location for the tug. Note that when planning paths, the tug might perform decoupling at a goal location different from the preferred decoupling location. The preferred decoupling location per runway is defined in Appendix A.

When an outbound flight is assigned to a tug, the starting and finishing time points corresponding to the flight are added to a Simple Temporal Network (STN). The STN is used to keep track of the schedule of the tug vehicle and allows the tug to find the best insertion position of new flights into its schedule before bidding on it. For a pair of starting and finishing time points in a STN, two types of constraints exist: travel time constraints and duration constraints. If both time points belong to the same task, a duration constraint is imposed and requires the start of a task not to be later than its finish time ($(F_\tau - S_\tau \in [DUR_\tau, \infty)$, where DUR_τ represents the duration of task τ). For a starting and finishing time point belonging to two different tasks, we impose travel time constraints that require the agent to only start the next task τ_{n+1} after finishing the previous task τ_n ($S_{\tau_{n+1}} - F_{\tau_n} \in [TT_{\tau_n,\tau_{n+1}}, \infty)$, where $TT_{\tau_n,\tau_{n+1}}$ represents the time needed to travel from the all-clear location corresponding with the preferred decoupling location to the ramp of the flight).

For every flight that has yet to be allocated, a copy of the current STN of the tug is made and the number of possible insertion points for the flight is determined. For every insertion position, both the travel time and duration constraints for all tasks in the schedule (including the flight yet unassigned) are added. Using Floyd-Warshall's all-pairs-shortest-paths algorithm, the consistency of the resulting STN is checked [33]. If the network is consistent, the shortest path in the network represents the network in which the starting time of each task is minimized. After having evaluated every possible insertion position of the task, a bid is calculated for the insertion position that minimizes the allocation objective. More details on the allocation objective and the associated bidding rules are provided in the next paragraph.

4.2.2 Allocation Objective and Bidding Rules

When considering fully outbound towing, the aim is to find an allocation of outbound flights to tugs that minimizes delay in off-block time due to unavailability of tugs for all aircraft. Therefore, the adapted TeSSI algorithm minimizes for the delay in arrival of the tug at the ramp of the assigned aircraft with respect to the deadline for the tug to be at the ramp. The deadline is defined as the TSAT of the assigned aircraft minus the time needed for coupling of the tug to the aircraft (Equation 1). In the context of aircraft engine-off towing operations, the set of unallocated tasks are considered to be all outbound flights

for which the TSAT is scheduled in the current planning window and for which no coupling activity has started yet. If a coupling activity has started, the corresponding outbound flight cannot be reallocated to another tug anymore.

For the set of unallocated tasks, all tugs submit their bid according to a shared bidding rule in order to prevent conflicting assignments [32]. In this study, tugs bid the difference in cost of their current schedule including already assigned tasks $\mathcal{K}$, and the cost of the schedule if task τ_m would be allocated (Equation 2). The cost of a schedule is calculated by summing the delay in arrival time at the ramp with respect to the deadline for arrival based on the TSAT at this ramp for all the tasks in its schedule (Equation 3). If the tug arrives at the ramp before the deadline, zero cost is used. When all agents have submitted their bids, the task with the minimum bid overall is assigned to the agent that submitted the corresponding bid. If two or more agents submit the same minimum bid, the agent with the bid that has been assigned the least tasks wins the auction round to ensure equal task distribution in a finite fleet of tugs. If both agents have the same amount of tasks assigned, tie-breaking is performed.

$$\text{bid}_{r_i,\tau_m} = \text{cost}_{r_i,\mathcal{K}\cup\tau_m} - \text{cost}_{r_i,\mathcal{K}} \qquad (2) \qquad \text{cost}_{r_i,\mathcal{K}} = \sum_k \max\left(0, t^k_{\text{arrival at ramp}} - t^k_{\text{deadline}}\right) \qquad (3)$$

4.2.3 Constraint Setting for Generation of Next Best Assignment

Based on the bidding rules defined in subsection 4.2.2, the TeSSI algorithm generates identical allocations in every auction when provided with identical input parameters. However, both the CPL-MIN and CPL-DIF algorithms require an efficient way to find a different, next best allocation when a new search tree is added to the path planning search forest. For deterministic allocation approaches, such as the Hungarian algorithm used in the original CBS-TA framework [13], a set of k best solutions can be found for which it is guaranteed that any solution not present in the set has a lower solution quality than the solutions that are part of the set [34]. However, since the TeSSI algorithm is not guaranteed to find the optimal solution in the first place, a novel constraint setting mechanism is proposed to find a next best allocation.

Figure 6 showed that a pool of allocation solutions is generated from which the preferred solution is chosen. In order to ensure that different allocations are created in the solution pool, a constraint is set for the task to assign in the first auction round for the next allocation to generate. Based on the previously generated allocation (for example, allocation i in pool $A0$), the task is found for which its assignment to a tug resulted in the highest increase of the objective value. This specific task is selected to be assigned first when generating allocation $i+1$ in pool $A0$. If the generated allocation is already present in the solution pool, infinite cost is assigned to the allocation. By imposing these constraints, it is guaranteed that all allocations in the solution pool will be unique if the size of the solution pool is less than the number of possible allocations based on the number of agents and tasks.

From the allocation solution pool, a preferred allocation is chosen that is used as input for the path planning algorithm. In the next subsection, the determination of a preferred allocation from the pool is further elaborated on.

4.2.4 Determination of Preferred Allocation from Allocation Solution Pool

Since the objective of the allocation of outbound flights to tugs is to minimize the delay in off-block time, the preferred allocation from the allocation solution pool is a solution which results in the least estimated amount of delay. Therefore, for the decoupled algorithm DCPL, the preferred allocation from the solution pool is the allocation for which the objective value is minimum. In other words, the allocation that results in the lowest sum of schedule costs for all tugs is used when planning paths.

As shown in Table 1, the two coupled algorithms CPL-MIN and CPL-DIF differ from each other in the determination strategy of the preferred allocation from the solution pool. Similarly to DCPL, the allocation in the solution pool with minimum cost is chosen as the preferred allocation for CPL-MIN. However, when exploring the consequences of a next best allocation when planning paths, we would prefer the next best allocation to be significantly different from the current allocation being explored. The reason for this is that a similar next best allocation could provide for similar constraints on the path planning problem, resulting in a roughly similar outcome in terms of path planning as for the current allocation. In other words, the added value of exploring multiple allocations in the PBS-TA framework is expected to be higher when significantly different allocations are being explored. Therefore, the total cost of every allocation in the solution pool when using the CPL-DIF algorithm is not solely based on the total amount of estimated delay, but also on the degree of similarity of the assignment with respect to all allocations explored in previous root nodes of the path planning forest. In order to measure the

degree of similarity, the cosine degree of similarity is calculated between the solution under consideration and all previously explored allocations [35]. More details on the calculation of the degree of similarity between two allocations is provided in Appendix A. Then, the total cost for every assignment in the solution pool is calculated using Equation 4,

$$\text{cost}_{\text{total}} = \text{cost}_{\text{delay}} \cdot \text{DOS} \tag{4}$$

where $\text{cost}_{\text{delay}}$ is the delay in off-block time [s] and DOS is the cosine degree of similarity [-]. The allocation with the minimum total cost in the solution pool is considered the preferred solution and used for the next search tree in the path planning search forest when using CPL-DIF. Note that for all three algorithms, a solution pool of 5 allocations is created.

4.3 Path Planning Based on PBS and Activity-Based SIPP

In this work, a routing algorithm inspired on the work of Soomers [15] and Kamphof [16] is implemented that consists of two levels: a high-level search based on Priority-Based Search (PBS) [27] and a low-level search based on Safe Interval Path Planning (SIPP) [28]. The high-level search finds a priority ordering among agents that resolves conflicts by setting constraints for the paths of lower-prioritized agents based on the paths of higher-prioritized agents. Then, the low-level algorithm searches for a path that results in the earliest arrival time at the goal location based on the constraints set in the high-level search for a lower-prioritized agent.

4.3.1 High-Level Search Using PBS

The goal of the high-level search is to find conflicts between the agents and resolve them at the earliest time [13]. The PBS-TA search forest starts with a single root node in which paths are planned for all agents based on the best assignment available of outbound flights to tugs. More details on the planning of individual paths using the SIPP algorithm will be provided in subsubsection 4.3.3. PBS maintains a priority tree and adds two child nodes whenever a conflict is detected in a parent node. Both child nodes inherit the existing priority ordering of their parent node, and add a priority couple that restricts one of the two agents to avoid the other's path. If a conflict is detected with a tug that is currently towing an aircraft, the resulting priority does not only affect the tug vehicle, but also applies to the aircraft being towed when travelling from the decoupling location to the runway entry. If the parent node in which a conflict is detected is a root node, an additional root node will be created, based on the next best allocation as provided for by the adapted TeSSI algorithm. No existing priority orderings are inherited by the newly created root node.

The resulting priority ordering in both child nodes is translated into unsafe intervals for all relevant locations in the network in which lower prioritized vehicles are not allowed to enter. More details on the construction of unsafe intervals can be found in subsubsection 4.3.2. The set of constraints is then passed on to the low-level path planning algorithm that constructs single-agent paths. Once new paths are found for the agents affected by the newly added priorities or the newly generated allocation, the solution cost of all newly added nodes is determined by summing the delay of all agents in the node. In this study, the total delay consists of ramp delay (for outbound flights, the delay in departure from the ramp with respect to their TSAT) and the delay en route (the delay in arrival at the goal location of the vehicle with respect to the earliest possible arrival time if interactions with other agents are neglected).

As for the original PBS implementation, a depth-first search is conducted until a node is expanded for which no conflicts are found in the current planning window [27]. In order to prevent the algorithm from exploring an unlimited amount of possible allocations, a maximum number of root nodes that can be added to the search forest is defined beforehand and set to 5 for this study (based on run time limits).

4.3.2 Construction of Safe Intervals

The priority ordering defined in the high-level search is translated in unsafe intervals for every location in the network that represent contiguous periods of time for which a prioritized vehicle occupies a location. Thus, deprioritized vehicles are not allowed to be present during an unsafe interval at that specific location. In addition, we add unsafe intervals to ensure runway separation for outbound flights based on Wake Turbulence Category (WTC) separation constraints (Appendix B).

For edge locations, unsafe intervals are defined for the edge in reversed direction for the duration of the traversal time in order to avoid head-on conflicts. At node locations, separation between vehicles is

maintained by defining unsafe intervals for the node based on the arrival and departure time of prioritized vehicles at the location, including a separation margin of 150 meters. The resulting unsafe interval starts at the time step for which the prioritized vehicle is first present within the separation distance from the node, and ends with the time step for which the vehicle is lastly present within the separation distance from the node. Note that the separation distance is measured along the path (and not using Euclidean distance).

Overlapping unsafe intervals are merged. The same holds for two unsafe intervals on an edge for which the time between them is smaller than the minimum time needed for the agent to traverse the edge, since traversal of this edge will not be possible for the deprioritized vehicle. Finally, the set of unsafe intervals is translated to a set of safe intervals and used in the low-level SIPP search to find single-agent paths.

Next to head-on conflicts and node conflicts, overtaking conflicts at edges are also restricted. Instead of using unsafe intervals, overtaking conflicts are restricted by adhering to the occupancy order of prioritized vehicles at an edge.

4.3.3 Low-Level Search Using SIPP

In the low-level search, single-agent paths are constructed that adhere to the safe intervals set by the high-level search and additional layout constraints (Appendix A), while minimizing for the travel time. When planning for single-agent paths, the input for the planning algorithm consists of an origin, destination and kinematic constraints of the vehicle. Based on the existing priority order and resulting unsafe intervals, SIPP generates states that consist of a (configuration, safe interval) pair. A single state represents a kinodynamically possible movement from the current state towards a location (configuration) in a certain safe interval and replaces what used to be many states: one for each time step in the safe interval. From the initial state (the safe interval in which the vehicle is at its starting location), a safe path is found for the agent using route and time as degrees of freedom.

During planning, all vehicles are assumed to be travelling at maximum velocity, except when unsafe intervals do not allow for this. If an agent would arrive in an unsafe interval when travelling at maximum velocity, the velocity is decreased in such a way that the vehicle will arrive as early as possible in the next safe interval at the specific location. Additionally, vehicles are not allowed to stand still anywhere on the taxiway and service road network, except for at the ramp or at a decoupling location.

The use of states instead of single time points allow for significant reduction of the search space, speeding up the path planning and making it more efficient [28]. In addition, SIPP allows for dealing with continuous time since safe intervals do not require time to be discretized into unit time steps. For more information on the working principle of SIPP, readers are referred to the work of Philips and Likhachev [28].

5 Verification and Validation

To ensure the accuracy and reliability of the simulation model, a variety of verification and validation procedures are performed. The conceptual model was validated by operational experts from To70 Aviation Consultancy. The model was developed in separate modules, allowing for independent unit testing of building blocks. Implementing exceptions is done to guarantee the course of internal processes is according to our expectations. Visual animations were used to ensure that routes were executed as intended, and small test scenarios were conducted to verify the model's behavior in various situations. These scenarios included the comparison of costs in the allocation and path planning domain, as well as the path planning for various activities to ensure correct implementation and adherence to kinematic constraints. As a results, the output of the model were checked to be according to our expectations. Finally, individual agent behavior was carefully tracked to ensure accuracy and continuity in the time domain.

6 Experimental Setup

Given the objective to evaluate the use of an integrated approach for allocation and path finding in the context of airport ground operations, experiments are performed that focus on analysis of algorithmic performance. In addition, experiments are performed that provide insights into the operational consequences of the implementation of tug-enabled taxiing at AMS. In subsection 6.1, the hypotheses to

be tested are stated, both for the algorithmic and operational performance of the PBS-TA framework. Second, the simulation plan is elaborated on in subsection 6.2. Finally, the key performance indicators (KPIs) that are used for evaluation of the hypotheses are discussed in subsection 6.3.

6.1 Hypotheses

This section focuses on the hypotheses that have been formulated and tested to assess the research objective. The first set of hypotheses relates to the comparison of results between the coupled and decoupled algorithms in the novel PBS-TA framework and is discussed in subsection 6.1.1. Next, a set of operational hypotheses is elaborated on in subsection 6.1.2. For all hypotheses, the total delay per flight is defined as the difference in arrival time of the aircraft at its destination in tug-enabled taxiing scenarios compared with the conventional taxiing scenario. For more information on the KPIs, refer to subsection 6.3.

6.1.1 Algorithmic Performance

With respect to the algorithmic performance of the PBS-TA framework, one hypothesis is formulated at the global (G_A) level that relates to the difference in solution quality between a coupled and decoupled approach.

G_A1 The sum of total delay per flight for all aircraft scheduled in the outbound peak is equal or lower when using a coupled approach than when using a decoupled approach.

- **Motivation**: according to literature [12, 13, 14], the expectation is that when task allocation and path finding are performed simultaneously (coupled approach), this will result in better overall solutions than when performing both procedures consecutively in a decoupled approach. Since the aim is to minimize ground delay, it is expected that on a global level, the sum of total delay is less when using a coupled algorithm compared to a decoupled algorithm.

6.1.2 Operational Performance

Similarly to the hypotheses on algorithmic performance, hypotheses related to the operational performance of the system are formulated at a global (G_O) level and concern the measurement and testing of total delays for flights when comparing tug-enabled taxiing with MET operations. Furthermore, different effects of tug-enabled taxiing operations are expected to be visible per departing runway. These expected effects are formulated as local (L_O) hypotheses.

G_O1 The total delay per flight for departing aircraft in tug-enabled taxiing scenarios increases with a decrease in available tugs.

- **Motivation**: when decreasing the number of available tugs, it is expected that aircraft will be delayed in their off-block time due to unavailability of tugs. As a consequence, aircraft are expected to arrive later at their final destination when compared to MET operations. Therefore, the total delay per flight is expected to increase for a decrease in number of tugs available.

G_O2 In tug-enabled taxiing scenarios with a similar number of tugs available, the total delay per flight for departing aircraft scheduled in the outbound peak is higher than for departing aircraft scheduled in the inbound peak.

- **Motivation**: during the outbound peak, the number of departing aircraft is higher than during the inbound peak. When having the same number of tugs available, more flights will be delayed during the outbound peak than in the inbound peak. In addition, the traffic density is expected to be increased on tracks used by departing flights in the outbound peak due to the higher traffic volume. Therefore, the total delay per flight for departing aircraft is expected to be higher during the outbound peak compared with the inbound peak.

L_O1 In tug-enabled taxiing scenarios during the outbound peak, the total delay per flight for aircraft scheduled to depart from 36L is higher than for flights scheduled to depart from 36C.

- **Motivation**: total delays for departing flights in the outbound peak are expected to be higher for 36L, since the route towards the runway is longer and thus, the chance on possible conflicts is larger.

6.2 Simulation Plan

In order to evaluate worst-case algorithmic and operational performance, flight data for one of the busiest days in terms of traffic volume at AMS (July 17th) of 2019 is used for simulation, in which RMO North was active. Based on the runway schedule throughout the day (refer to Appendix B), an outbound peak from 20:30-22:00h and inbound peak from 07:30-09:00h are defined and chosen to analyze. Any flights departing from or arriving at other runways that are not active in RMO North during the peak under consideration are removed from the flight schedule. Table 2 shows the number of departing and arriving flights during both the outbound and inbound peak. Any additional information and assumptions on the input data can be found in Appendix B.

Table 2: Number of departing and arriving flights per runway for the outbound peak (20:30-22:00h) and the inbound peak (07:30-09:00) for one of the busiest days in terms of traffic volume at AMS.

Flight Direction	Runway	Outbound Peak	Inbound Peak
Departures	36L	47	45
	36C	50	not active
	Total	97	45
Arrivals	06	46	43
	36R	not active	44
	Total	46	87

In order to measure and test the hypotheses as formulated in subsection 6.1, 14 scenarios are formulated and analyzed on their algorithmic and/or operational performance. Two baseline scenarios are specified that are used as a reference for system performance of MET operations in both the inbound and outbound peak (conventional operations). For all remaining scenarios, a coupling and decoupling duration of 120 seconds is used, based on previous research [15, 16] and operational data [24]. Kinematic characteristics of the vehicles can be found in Appendix B.

For analysis of the algorithmic hypotheses, 9 scenarios are generated for the outbound peak that differ in algorithm used (DCPL, CPL-MIN or CPL-DIF) and in available fleet size. The goal of the comparison of the selected scenarios is to gain insights in the added value of using a coupled approach compared with a decoupled approach for solving an MAPD problem in the context of airport ground surface operations. The comparison of the coupled and decoupled algorithms is deemed only useful for scenarios in which resources are limited. The reason for this is that a coupled approach is of no added value when the initial allocation results in no ramp delay (allocation cost of 0). In the case of tug-enabled taxiing operations, resources are considered limited if insufficient tugs are available to guarantee zero ramp delay for all departing flights. Therefore, the minimum number of tugs needed to ensure zero ramp delay is determined by forcing the decoupled algorithm to find an allocation that results in zero ramp delay for all departing flights. For the outbound peak, the analysis showed that 42 tugs are needed to guarantee zero ramp delay for all departing flights. Based on this number, three limited fleet sizes are chosen and determined to be 30, 21 (half) and 10 for the outbound peak. A summary of the characteristics of the algorithmic scenarios can be found in Table 3. After evaluation of the algorithmic scenarios, a conclusion is drawn on the algorithm most suitable for the application of aircraft engine-off towing operations. This algorithm is then used for analysis of the operational scenarios.

Concerning the analysis of the operational hypotheses, an additional 2 scenarios are formulated for the inbound peak. First, the number of tugs needed to guarantee zero delay in off-block time for all departing flights with respect to the flight schedule is found to be 21 during the inbound peak. Based on the limited fleet sizes evaluated in the outbound peak, the effects of using similar fleet sizes in the inbound peak are analyzed as well. Note that only fleet sizes smaller than the fleet size needed to ensure zero ramp delay in the inbound peak are evaluated. Table 4 shows the characteristics of the operational scenarios to evaluate.

6.3 Key Performance Indicators

Analysis and testing of the hypotheses is performed using a set of predetermined Key Performance Indicators (KPIs). Similarly as for hypotheses, KPIs are defined for different levels. Global KPIs (G) reflect performance of the model on a system level, while local KPIs (L) only reflect behaviour visible in certain parts of the airport. Finally, two KPIs related to sensitivity (S) reflect the difference in

Table 3: Summary of algorithmic scenarios to evaluate (9)

Algorithmic Performance Analysis			
Peak	outbound		
Algorithm	DCPL CPL-MIN CPL-DIF	DCPL CPL-MIN CPL-DIF	DCPL CPL-MIN CPL-DIF
Tug Fleet Size	10	21	30

Table 4: Summary of operational scenarios to evaluate for tug-enabled taxiing operations (6)

Operational Performance Analysis						
Peak	outbound				inbound	
Algorithm	DCPL/CPL-MIN/CPL-DIF					
Tug Fleet Size	10	21	30	42	10	21

performance for a scenario with limited tugs available, compared to the scenario with unlimited tugs available.

G1 Total Run Time (t_{total}): the run time needed to complete one entire simulation, expressed in minutes.

G2 Total Delay per Flight (d): the difference in arrival time of the aircraft at its goal location between the tug-enabled taxiing and conventional taxiing scenario, expressed in minutes.

G3 Route Delay per Flight (tt): the additional time required for an aircraft to reach its destination by comparing the duration it takes for the aircraft to travel from the starting location to the destination for tug-enabled taxiing versus MET operations, expressed in minutes.

L1 Ramp Delay per Flight (dr): the duration by which an aircraft's off-block time exceeds its planned off-block time (TSAT) as a result of a delayed arrival of a tug at the ramp, expressed in minutes.

S1 Difference in Total Delay per Flight (δd): the difference in total delay per flight for a scenario with limited tugs available, compared to a scenario with unlimited tugs available.

S2 Difference in Route Delay per Flight (δtt): the difference in route delay per flight for a scenario with limited tugs available, compared to a scenario with unlimited tugs available.

7 Analysis and Results

In this section, the results of the scenarios described in section 6 are presented, including their analysis. First, the results related to the algorithmic performance are discussed in subsection 7.1. In subsection 7.2, an outline is provided on the results of the evaluated scenarios from an operational point of view. For both the algorithmic and operational scenarios, the number of runs is determined to be 30 based on the coefficient of variation as function of the number of runs (Appendix C). All runs are performed on a 2.60 GHz Intel Core i7-2592M laptop with 8 GB RAM.

7.1 Algorithmic Performance Analysis

The emphasis of this section is on the analysis of the performance between a coupled and decoupled approach in the novel PBS-TA framework. Two algorithms that are based on a coupled approach (CPL-MIN and CPL-DIF) are compared against an algorithm based on a decoupled approach (DCPL). The main performance indicator is the total delay per flight with respect to MET operations for departing and arriving flights, $d_{\text{dep, out}}$ and $d_{\text{arr, out}}$ respectively. Furthermore, the run time needed for the execution of a single simulation (t_{total}) is used. As explained in subsection 6.2, the algorithmic analysis is performed for the outbound peak only.

7.1.1 Overall Trend Analysis

Table 5 shows for algorithms DCPL, CPL-MIN and CPL-DIF the medians, first quartiles and third quartiles for the run time per simulation t_{total} in minutes. Furthermore, the distribution characteristics of the total delay of departing and arriving flights in the outbound peak for various fleet sizes is shown, $d_{\text{dep, out}}$ and $d_{\text{arr, out}}$ respectively. In addition, the sum of the total delay per flight for all aircraft scheduled in the outbound peak is listed ($\sum_{\text{dep, arr}} d_{\text{out}}$). To quantify the added value of a coupled approach for allocation and path finding in the current application compared to a decoupled approach,

statistical tests are performed. The performance of both CPL-MIN and CPL-DIF is compared against DCPL, by evaluating the statistical difference in the distributions for the KPIs listed in Table 5 using the Wilcoxon Signed-Rank test [36]. The effect size of the statistical difference is evaluated using the Vargha-Delaney A-Test [37]. For more information on the statistical evaluation of the results, refer to Appendix C.

A decrease of the (sum of) total delay per flight of a coupled algorithm compared to a decoupled algorithm is shown in green, whereas an increase is shown in red. The magnitude of the statistical difference is indicated by the capital letters N (negligible), S (small), M (medium) and L (large) and through color shading. Darker shades indicate larger effect sizes, while lighter shades indicate smaller effect sizes. Results that do not show statistical differences are indicated in light-blue. In the remaining part of this section, the results will be discussed in the order of the KPIs as listed in Table 5.

Table 5: Comparison of algorithmic performance for algorithms DCPL, CPL-MIN and CPL-DIF. Significance level = $0.05/2 = 0.025$. Statistical difference between CPL-MIN/CPL-DIF and DCPL determined using the Wilcoxon Signed-Rank test.

Fleet size [tugs]	**KPI [min]**	**DCPL**	**CPL-MIN**		**CPL-DIF**	
		median (Q1, Q3)	median (Q1, Q3)	p-value, A-test	median (Q1, Q3)	p-value, A-test
10	t_{total}	12.95(12.74, 13.14)	52.0(51.50, 55.20)	N/A	56.87(52.07, 58.53)	N/A
	$d_{dep,\ out}$	13.72(2.82, 84.80)	13.67(3.30, 83.39)	0.70, (−)	13.91(3.02, 74.09)	0.94, (−)
	$d_{arr,\ out}$	0.0(−)	0.0(−)	0.59, (−)	0.0(−)	0.25, (−)
	$\sum_{dep,\ arr} d_{out}$	4252.5(−)	4137.7(−)	< 0.001 $(0.0, L)$	4160.2(−)	< 0.001 $(0.0, L)$
21	t_{total}	14.79(14.71, 14.89)	64.70(61.84, 66.54)	N/A	69.44(65.21, 72.36)	N/A
	$d_{dep,\ out}$	2.67(1.31, 9.10)	2.75(1.38, 10.28)	0.87, (−)	2.82(1.41, 8.28)	0.96, (−)
	$d_{arr,\ out}$	0.0(−)	0.0(−)	0.29, (−)	0.0(−)	0.29, (−)
	$\sum_{dep,\ arr} d_{out}$	936.0(−)	1000.4(−)	< 0.001 $(1.0, L)$	1033.5(−)	< 0.001 $(1.0, L)$
30	t_{total}	15.07(14.92, 15.44)	30.27(27.46, 31.76)	N/A	27.82(27.43, 31.09)	N/A
	$d_{dep,\ out}$	2.34(1.14, 2.72)	2.17(1.14, 2.72)	0.73, (−)	2.19(1.14, 2.72)	0.88, (−)
	$d_{arr,\ out}$	0.0(0.0, 0.024)	0.0(0.0, 0.024)	1.0, (−)	0.0(0.0, 0.024)	1.0, (−)
	$\sum_{dep,\ arr} d_{out}$	264.8(263.4, 264.8)	264.8(263.4, 264.8)	0.91, (−)	264.8(263.4, 264.8)	0.98, (−)

Run Time Performance

It is noted that significant differences in run time performance between both coupled algorithms and the decoupled algorithm exist. While the decoupled algorithm performs the entire simulation in 10 to 15 minutes for all scenarios, both coupled algorithms require 30 to 70 minutes to find a solution. The increase in run time for the CPL-MIN and CPL-DIF can be explained by the increase in the number of nodes that is explored when comparing both approaches. Whereas a single search tree is expanded in the decoupled approach, multiple search trees are explored in the coupled approach (for more details on the exploration of a search forest, refer to subsection 4.1). In Appendix C, a comparison is shown between both approaches for the number of nodes explored.

Total Delay per Flight: Individual Aircraft Level

With respect to the distribution of the total delay per departing/arriving flight, no statistical differences can be found for CPL-MIN and CPL-DIF compared to DCPL for all fleet sizes. Thus, on the level of individual aircraft, the integration of allocation and path planning in both coupled algorithms does not seem to be of benefit when aiming to minimize delay with respect to MET operations.

Sum of Total Delay per Flight: Global Level

Two interesting observations can be made when evaluating the sum of the total delay per flight for all aircraft scheduled in the outbound peak. First, the performance of both coupled algorithms is compared to each other. Based on the results in Table 5 it is concluded that independent of the number of tugs available, CPL-MIN provides a solution for which the sum of total delay is lower or equal to CPL-DIF. This leads us to believe that the CPL-MIN algorithm is more suitable for the current application than CPL-DIF. As explained in subsection 4.2.4, the selection of the allocation to explore in the path finding domain differs for CPL-MIN and CPL-DIF. Similarly to DCPL, CPL-MIN picks the allocation in the solution pool with minimum cost. For CPL-DIF, the selection of the allocation to explore in the path planning domain is not solely based on the cost of this allocation, but also on its cosine degree of similarity

with respect to previously explored allocations. It was expected that this would decrease the chances of the algorithm from getting trapped in a local minimum and to increase the chances of reaching a global minimum. However, the results in Table 5 show that this is not true. It turns out that the inclusion of the cosine degree of similarity in the cost calculations forces the algorithm to explore another allocation that is in fact located further away from the global minimum than the allocation with minimum cost. The results in Table 5 show that in terms of total delay on a global level, it is not desired to include the cosine degree of similarity in the cost calculation of the allocation in the solution pool. In other words, it is expected that the allocations in the solution pool are already close to the global minimum and therefore, no need exists to force the search area to be further expanded.

Second, we compare the best performing coupled algorithm (CPL-MIN) with the decoupled algorithm (DCPL) in terms of total delay per flight on a system level. For a fleet size of 10 tugs, we can see that CPL-MIN provides a solution where the summed delay for all flights with respect to MET operations is less than for the solution found by the decoupled algorithm. This is in line with our hypothesis item $\mathbf{G_A 1}$. Similarly, the results for a scenario with 30 tugs available show that CPL-MIN produces a solution that is at least equal to the solution provided by the decoupled algorithm, supporting hypothesis item $\mathbf{G_A 1}$ as well. However, in a scenario with 21 tugs available, CPL-MIN produces a solution that shows an increase in the sum of total delay per flight when compared with DCPL. Since this case is not in line with hypothesis item $\mathbf{G_A 1}$, we study the reason for this behaviour more in detail in the next subsection 7.1.2, by evaluating the difference in $\sum_{\text{dep, arr}} \text{d}_{\text{out}}$ between DCPL and CPL-MIN for multiple levels of tugs available.

7.1.2 Sensitivity Analysis: Total Delay per Flight and Tug Fleet Size

In Figure 7, the sum of the total delay per flight for all aircraft scheduled in the outbound peak is shown as a function of tug fleet size. Next to the fleet sizes already evaluated in the previous subsection, the algorithmic performance of the decoupled algorithm (DCPL) and best-performing coupled algorithm (CPL-MIN) is analyzed for an increased variety of number of tugs available. The results for DCPL and CPL-MIN are shown in blue and orange respectively.

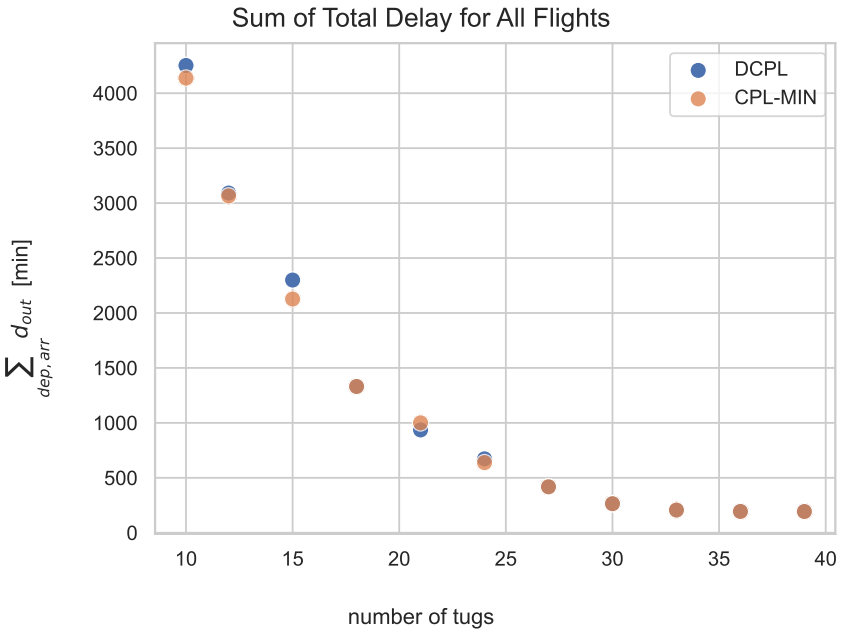


Figure 7: Comparison of the sum of total delay per flight for all aircraft scheduled in the outbound peak between DCPL and CPL-MIN for different tug fleet sizes.

Two interesting phenomena can be observed in Figure 7. First, it can be noted that for higher number of tugs available, the differences between DCPL and CPL-MIN for $\sum_{\text{dep, arr}} \text{d}_{\text{out}}$ become smaller. If having 25 tugs or more available, no benefits can be expected in terms of ground delay minimization when using a coupled algorithm compared to a decoupled algorithm. It is in line with our expectations that when having limited resources, specifically the allocation is of higher influence on the total delay arising in the system. As a result, differences between a decoupled and coupled algorithm can be observed more

frequently or become larger when having less tugs available.

Secondly, it can be seen that only for the scenario of 21 tugs, the coupled algorithm produces a worse solution in terms of total delay compared with the decoupled algorithm. For all other scenarios, the results in Figure 7 support hypothesis item $\mathbf{G_A1}$. A possible explanation for the increase in the sum of total delay when comparing CPL-MIN with DCPL for 21 tugs available, is the unknown effect of an allocation used in the current planning window on all future planning windows. For both the coupled and decoupled algorithm, a solution is found for a planning window of 30 minutes. Within this planning window, the objective is to reach a global minimum. However, the effect of the solution in the current planning window on all future planning windows is not taken into account. Therefore, a better solution for the current planning window found using CPL-MIN compared to DCPL does not guarantee that a better solution for the entire simulation is found. However, the results in Figure 7 show that this phenomena does not occur for the majority of the cases.

7.1.3 Conclusion on the Algorithm to Use for Operational Performance Analysis

Based on the comparison of the three algorithms, one algorithm is chosen to use for simulation and evaluation of the operational performance of the airport system for different number of tugs available. From the results analyzed in the previous sections, it is concluded that for all fleet sizes except 21 tugs, the use of CPL-MIN results in the lowest sum of total delay per flight. However, the decrease in total delay on a system level comes at the price of a significant increase in run time performance for CPL-MIN compared with DCPL. Furthermore, no statistical differences could be found when comparing the distribution of total delay per flight of both coupled algorithms with the decoupled algorithm.

In Table 6, the percentage decrease in the sum of total delay per flight as well as the percentage increase in run times are shown for CPL-MIN and CPL-DIF compared to DCPL. In the scenario where the biggest decrease in total delay per flight on a system level is observed (10 tugs when using CPL-MIN), the decrease is merely 2.7%. Since run times more than quadruple, the authors deem the use of DCPL for simulation and evaluation of the operational scenarios the most beneficial.

Table 6: Comparison of CPL-MIN/CPL-DIF with DCPL in terms of sum of total delay per flight and total run time. All values are percentages with respect to the DCPL algorithm.

KPI [%]	**Algorithm**	**Tugs**		
		10	21	30
$\Delta \sum_{\text{dep, arr}} d_{\text{out}}$	CPL-MIN	-2.7	6.9	0.0
	CPL-DIF	-2.2	10.4	0.0
Δt_{total}	CPL-MIN	301.5	337.5	100.9
	CPL-DIF	339.2	369.5	84.6

7.2 Operational Performance Analysis

In this section, a number of operational scenarios is analyzed to evaluate the performance of the airport system in terms of ground delay per flight upon implementation of tug-enabled taxiing operations. For both the outbound and inbound peak, several types of ground delay are recorded for scenarios in which the number of available tugs varies. In subsection 7.2.1, an overview is given of the global results for ground delay. Next, an in-depth analysis on the relation between ground delay and assigned runway for the outbound peak is discussed in subsection 7.2.2.

7.2.1 Overall Trend Analysis

In this section, the airport performance in terms of ground delay is discussed when implementing fully outbound towing. For varying number of tugs available, three types of ground delay are recorded: total delay, route delay and ramp delay. The total delay per flight is defined as the difference in arrival time of an aircraft at its destination when comparing tug-enabled taxiing with conventional taxiing techniques. The additional time required for an aircraft to reach its destination, referred to as taxi time increase per flight or route delay, is determined by comparing the duration it takes for the aircraft to travel from its starting location to its destination for both types of operations. Finally, ramp delay refers to the duration by which an aircraft's off-block time exceeds its planned off-block time (TSAT) as a result of a delayed arrival of a tug at the ramp. Additional KPIs such as tug waiting time and gate scheduling

conflicts are reported on in Appendix C.

In a scenario where unlimited tugs are available, initial analysis showed that 42 tugs are needed in the outbound peak to ensure zero ramp delay. In other words, if 42 tugs are available during the outbound peak, no aircraft will be delayed in its off-block time with respect to the historic flight schedule due to the unavailability of a towing tug. For the inbound peak, this number is determined to be 21 tugs. In the remainder of this section, the total delay, route delay and ramp delay will be discussed when having unlimited tugs available (42 and 21 tugs in the outbound and inbound peak respectively) and when limiting the number of resources.

Total Delay per Flight

Table 7 shows the medians, first quartiles and third quartiles for a set of KPIs related to the total delay per flight. Separate distributions are presented for the outbound and inbound peak, as well as for departing and arriving flights. All distributions are shown to be not normally distributed using the Shapiro-Wilk test [38]. Therefore, the statistical significance of the distributions for delay is determined using the Wilcoxon Signed-Rank test [36]. If a statistical difference between tug-enabled taxiing and MET operations is found, the effect size of the difference is determined using the Vargha-Delaney A-test [37]. Similarly as for the results discussed in section 7.1, the effect size is indicated by the capital letters N (negligible), S (small), M (medium) and L (large) and through color shading. Darker shades indicate larger effect sizes, while lighter shades indicate smaller effect sizes. Results that do not show statistical differences are indicated in light-blue.

Table 7: Comparison of the total delay per flight for various number of tugs available between tug-enabled taxiing and MET operations. Significance level for outbound peak = 0.05/4 = 0.0125. Significance level for inbound peak = 0.05/2 = 0.025. Statistical difference of the distribution is determined using the Wilcoxon Signed-Rank test.

KPI	**10 tugs**		**21 tugs**	
[min]	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$d_{arr,\ out}$	$0.0, (-)$	$0.80, (-)$	$0.0, (-)$	$0.056, (-)$
$d_{arr,\ in}$	$0.0, (-)$	$0.11, (-)$	$0.0, (-)$	$0.27, (-)$
$d_{dep,\ out}$	$13.72, (2.82, 84.80)$	$< 0.001, (0.70, M)$	$2.67, (1.31, 9.10)$	$< 0.001, (0.58, S)$
$d_{dep,\ in}$	$15.45, (2.77, 28.62)$	$< 0.001, (0.65, S)$	$2.68, (2.59, 2.90)$	$< 0.001, (0.53, N)$
$\delta d_{dep,\ out}$ (ref: 42)	$12.01, (0.98, 83.92)$	$< 0.001, (0.69, M)$	$0.0, (0.0, 6.58)$	$< 0.001, (0.56, N)$
$\delta d_{dep,\ in}$ (ref: 21)	$12.73, (0.0, 25.74)$	$< 0.001, (0.63, M)$	-	-

KPI	**30 tugs**		**42 tugs**	
[min]	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$d_{arr,\ out}$	$0.0, (0.0, 0.02)$	$0.076, (-)$	$0.0, (0.0, 0.03)$	$0.042, (-)$
$d_{arr,\ in}$	-	-	-	-
$d_{dep,\ out}$	$2.34, (1.14, 2.72)$	$< 0.001, (0.53, N)$	$1.97, (1.06, 2.72)$	$< 0.001, (0.52, N)$
$d_{dep,\ in}$	-	-	-	-
$\delta d_{dep,\ out}$ (ref: 42)	$0.0, (-)$	$0.52, (-)$	-	-
$\delta d_{dep,\ in}$ (ref: 21)	-	-	-	-

First, the total delay per flight for arriving aircraft is evaluated for both the outbound and inbound peak, $d_{arr,\ out}$ and $d_{arr,\ in}$ respectively. Based on the results presented in Table 7, we can see that no statistically significant differences are found between tug-enabled taxiing and MET operations for all scenarios. Therefore, we conclude that that tug-enabled taxiing operations are not of significant influence on ground movement operations for inbound traffic.

With regards to the total delay for departures $d_{dep,\ out}/d_{dep,\ in}$, our analysis reveals that departing flights experience positive total delay in reaching their goal location when compared with MET operations, regardless of the type of peak or the number of tugs available. While this delay is statistically significant across all scenarios, it is only relevant in size for 3 out of 6 scenarios in which fleet sizes are smallest. Furthermore, although the median of the distribution for the total delay decreases with increasing number of tugs for all scenarios, the most significant decrease can be seen when increasing fleet size from 10 to 21 tugs in both peaks. This result indicates that the distribution for total delay per flight is not linearly related to the fleet size.

Sensitivity analysis related to fleet size confirms this statement, showing that differences in total delay for the outbound peak $\delta d_{dep,\ out}$ are only relevant in size when comparing a fleet size of 42 tugs with 10 tugs. For the other two scenarios in the outbound peak, the results do not show statistical significant differences (42 tugs vs. 30 tugs) or the differences are negligible in effect size (42 tugs vs. 21 tugs). Therefore, hypothesis $\mathbf{G_O1}$ is rejected for the outbound peak, since the total delay per flight for departing aircraft in tug-enabled taxiing scenarios does not increase significantly in all cases for a decrease in available tugs.

When performing a similar analysis for the inbound peak, we can see that a decrease from 21 tugs to 10 tugs does result in a statistically relevant increase in total delay per flight. This result suggests that although less outbound flights are departing in an inbound peak, the total delay per flight is more sensitive and dependent on the number of available tugs than for an outbound peak. This result can be explained by the fact that only runway 36L is in use during the inbound peak, requiring longer taxi times and tug return times than runway 36C. Therefore, the built-up of delay is more steep in the inbound peak when limited tugs are available, compared to the outbound peak. To conclude, the results for the inbound peak in Table 7 support hypothesis $\mathbf{G_O1}$.

Finally, a statistical comparison between the distribution of total delay per departing flight in the outbound and inbound peak is performed. Since we are comparing two unpaired groups, the Mann-Whitney U-test is used to detect any statistical difference between the two distributions. The results of the statistical comparison are shown in Table 8.

It was expected to see higher total delays per departing flight in the outbound peak compared with the inbound peak for a similar number of tugs available (hypothesis $\mathbf{G_O2}$). The reasoning for this is based on the fact that more departing flights are scheduled during an outbound peak than an inbound peak. Therefore, if having the same number of tugs available, delays are expected to stack up more quickly in an outbound peak. However, the results in Table 8 show that no statistical difference between the total delay per departing flight exists in the inbound and outbound peak when having 10 or 21 tugs available. Therefore, it is expected that the stacking up of delays in the outbound peak is counteracted by the longer taxi times and tug return times for runway 36L in the inbound peak. Since only runway 36L is in use during the inbound peak, the average tug journey for every flight in the inbound peak is longer than in the outbound peak. As a result, the number of tugs available has more influence on the total delay per flight in the inbound peak than in the outbound peak. Thus, whereas the percentage of departing flights that is delayed in the inbound peak is lower, the total delay per flight is higher compared to the outbound peak for similar number of tugs available. It is expected that this phenomena is responsible for the lack of any statistical differences in the total delay per flight for both peaks and the rejection of hypothesis $\mathbf{G_O2}$.

Table 8: Comparison of the total delay per departing flight in the outbound peak and inbound peak for various number of tugs available. Significance level = 0.05/2 = 0.025. Statistical difference of the distribution is determined using the Mann-Whitney U-test.

KPI	**10 tugs**		**21 tugs**	
[min]	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$d_{dep,\ out}$	13.72, (2.82, 84.80)	0.41, (-)	2.67, (1.31, 9.10)	0.93, (-)
$d_{dep,\ in}$	15.45, (2.77, 28.62)		2.68, (2.59, 2.90)	

Route Delay per Departing Flight

Next, we consider the taxi time increase per departing flight tt_{dep} for both peaks. The results in Table 9 show that the median value for the increase in taxi time is higher in the inbound peak than in the outbound peak for all scenarios. In the inbound peak, all flights depart from the more remotely located runway 36L, whereas runway 36C (closer to the central parking area at Schiphol) is also used as departure runway in the outbound peak. Since the maximum speed for a tug-aircraft combination is lower than for single aircraft (approx. 45 km/h versus 60 km/h), increase in taxi time per flight is expected to be higher in the peak where the average taxi distance is higher as well.

Furthermore, the median value for the increase in taxi time per flight shows very limited variation for all scenarios within the same peak. This suggests that the taxi time increase is independent of the number of tugs, but can solely be attributed to: 1) difference in maximum speed for tug-aircraft combinations and single aircraft; and 2) the increase in decoupling times of the tug-aircraft combination at a decoupling location (120 seconds) compared with conventional engine warm-up operations at the

apron (100 seconds). For all scenarios with limited tugs available, the taxi time increase per departing flight is compared with the taxi time increase per departing flight in a scenario with unlimited tugs available. The distribution characteristics of the resulting parameter $\delta tt_{dep,\ out}/\delta tt_{dep,\ in}$ can be found in Table 9. Statistical analysis showed that all distributions for δtt_{dep} are symmetrical around zero and thus, no differences exist between the taxi time increase per flight across all scenarios. Therefore, the increase in taxi time per departing flight in both peaks is shown to be independent of tug fleet size. As a consequence, the authors conclude that the increase in traffic density on the taxiway network for larger tug fleet sizes is not of influence on the taxi time of departing aircraft.

Table 9: Comparison of the route delay per flight for various numbers of tugs available between tug-enabled taxiing and MET operations. Significance level for outbound peak = 0.05/4 = 0.0125. Significance level for inbound peak = 0.05/2 = 0.025. Statistical difference of the distribution is determined using the Wilcoxon Signed-Rank test.

KPI	**10 tugs**		**21 tugs**	
[min]	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$tt_{dep,\ out}$	2.06, (1.03, 2.77)	$< 0.001, (0.69, M)$	2.10, (0.93, 2.79)	$< 0.001, (0.68, M)$
$tt_{dep,\ in}$	2.68, (2.49, 2.98)	$< 0.001, (0.98, L)$	2.68, (2.59, 2.90)	$< 0.001, (0.99, L)$
$\delta tt_{dep,\ out}$ (ref: 42)	0.0, (−0.59, 0.64)	0.60, (−)	0.0, (−0.22, 0.18)	0.77, (−)
$\delta tt_{dep,\ in}$ (ref: 21)	0.0, (−0.23, 0.05)	0.36, (−)	-	-

KPI	**30 tugs**		**42 tugs**	
[min]	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$tt_{dep,\ out}$	2.19, (1.14, 2.72)	$< 0.001, (0.68, M)$	1.97, (1.06, 2.72)	$< 0.001, (0.68, M)$
$tt_{dep,\ in}$	-	-	-	-
$\delta tt_{dep,\ out}$ (ref: 42)	0.0, (−)	0.78, (−)	-	-
$\delta tt_{dep,\ in}$ (ref: 21)	-	-	-	-

Ramp Delay per Departing Flight

Lastly, we consider ramp delay. Ramp delay refers to the duration by which an aircraft's off-block time exceeds its planned off-block time (TSAT) as a result of a delayed arrival of a tug at the ramp. In Table 10, it can be seen that significant ramp delay occurs in the outbound peak when having 10 and 21 tugs available. The increase in ramp delay for both peaks is largest for the scenario in which the least amount of tugs is available, which is according to our expectations. Interestingly, no significant ramp delay is found in the outbound peak for the scenario in which the fleet consists of 30 tugs. Thus, it can be concluded that ramp delay is not linearly related to fleet size.

In addition, the median value of the distribution for ramp delay does not change for scenarios in which 21, 30 and 42 tugs are available and remains equal to zero. The significant increase in the median value for ramp delay when decreasing the fleet size from 21 tugs to 10 tugs suggests that a tipping point exists in the relation between ramp delay and fleet size.

Table 10: Comparison of the ramp delay per flight for various numbers of tugs available between tug-enabled taxiing and MET operations. Significance level for outbound peak = 0.05/4 = 0.0125. Significance level for inbound peak = 0.05/2 = 0.025. Statistical difference of the distribution is determined using the Wilcoxon Signed-Rank test.

KPI	**10 tugs**		**21 tugs**	
[min]	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$dr_{dep,\ out}$	13.20, (0.34, 82.77)	$< 0.001, (0.88, L)$	0.0, (0.0, 7.63)	$< 0.001, (0.69, M)$
$dr_{dep,\ in}$	12.46, (0.0, 25.95)	$< 0.001, (0.83, L)$	0.0, (−)	0.11, (−)

KPI	**30 tugs**		**42 tugs**	
[min]	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$dr_{dep,\ out}$	0.0, (−)	0.028, (−)	0.0, (−)	0.32, (−)
$dr_{dep,\ in}$	-	-	-	-

All results on the total delay, route delay and ramp delay are summarized in Figure 8. For both peaks, the variance of the total delay per departing flight increases for a decrease in tug fleet size. However,

this increase in total delay per flight does not seem linearly related to the number of tugs available. The reason for this is the significant increase in total delay per flight when decreasing the number of available tugs from 21 to 10, whereas decreasing the fleet size from 42 to 30 tugs does not have significant effect on the distribution of the total delay per flight.

In both the outbound and inbound peak, the distribution of route delay per departing flight does not show significant changes when varying the number of tugs available. Furthermore, note that the increase in total delay per flight for decreasing fleet sizes exhibits the same behaviour as the ramp delay per flight does. This result is an indication that the ramp delay is a dominant factor in the development of the total delay per flight.

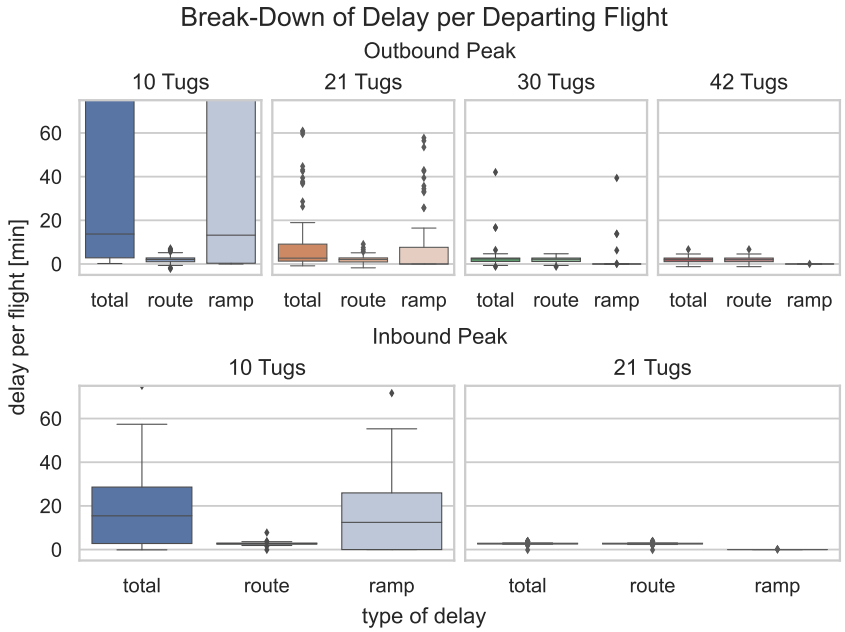


Figure 8: Break-down of delay for departing flights in both the outbound and inbound peak for various numbers of tugs available.

7.2.2 In-Depth Analysis: Delay per Departing Flight per Runway

In order to provide more insights on the effect of runway assignment on the ground delay for tug-enabled taxiing operations, the following section provides details on ground delay per departing flight in the outbound peak, split per runway. Similarly as to the previous subsection, the total delay, route delay and ramp delay per departing flight will be discussed.

Table 11: Comparison of the total delay per departing flight per runway for various number of tugs available between tug-enabled taxiing and MET operations. Significance level $= 0.05/4 = 0.0125$. Statistical difference of the distribution is determined using the Wilcoxon Signed-Rank test.

KPI [min]	**10 tugs**		**21 tugs**	
	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$d_{36L,\ out}$	$86.92, (34.442, 127.96)$	$< 0.001, (0.91, L)$	$6.75, (2.77, 27.46)$	$< 0.001, (0.67, M)$
$d_{36C,\ out}$	$3.64, (1.39, 8.78)$	$< 0.001, (0.58, S)$	$1.35, (0.84, 2.44)$	$< 0.001, (0.53, N)$
$\delta d_{36L,\ out}$ (ref: 42)	$84.11, (30.12, 124.88)$	$< 0.001, (0.91, L)$	$4.73(0.0, 24.89)$	$< 0.001, (0.65, S)$
$\delta d_{36C,\ out}$ (ref: 42)	$2.92, (0.0, 7.64)$	$< 0.001, (0.57, N)$	$0.0(0.0, 1.38)$	$< 0.001, (0.52, N)$

KPI [min]	**30 tugs**		**42 tugs**	
	median (Q1, Q3)	p-value (A-test)	median (Q1, Q3)	p-value (A-test)
$d_{36L,\ out}$	$2.72(2.53, 3.07)$	$< 0.001, (0.55, N)$	$2.72(2.59, 2.93)$	$< 0.001, (0.54, N)$
$d_{36C,\ out}$	$1.21(0.90, 1.63)$	$< 0.001, (0.52, N)$	$1.14(0.80, 1.51)$	$< 0.001, (0.52, N)$
$\delta d_{36L,\ out}$ (ref: 42)	$0.0, (0.0, 0.04)$	$0.65, (-)$	-	-
$\delta d_{36C,\ out}$ (ref: 42)	$0.0(-)$	$0.63, (-)$	-	-

Total Delay per Departing Flight per Runway
In Table 11, results are presented on the total delay per departing flight in the outbound peak, split per runway. The first thing that is noticed is that a large difference in the total delay per flight can be identified between the two runways, especially for smaller fleet sizes. In a scenario where 10 tugs are available, we observe relatively small delays of approximately 3.5 minutes for flights departing from 36C, whereas at least half of the flights departing from 36L are delayed by almost 87 minutes. Whenever the fleet size consists of 30 tugs or more, the differences in total delay for both runways with respect to MET operations can be considered negligible. However, comparison of the total delay per flight for both runways shows that independent of fleet size, the total delays per flight are higher for aircraft departing from 36L than for aircraft assigned to 36C. We will now look into the cause for these differences between total delay per departing flight per runway.

Route Delay per Departing Flight per Runway
When examining Figure 9 for the scenario with unlimited tugs available (42), we can observe a slight difference in taxi time increase per flight for runway 36L compared to 36C. This can be explained by the increase in travelling distance towards 36L. Since the maximum speed of tugs is slightly lower than that of aircraft using conventional engine taxiing techniques, an increase in travelling distance results in an increase in travelling time as well with respect to MET operations. Thus, for scenarios with zero ramp delay, the differences in total delay per flight per runway can be entirely attributed to the increase in taxi time when comparing both departure runways.

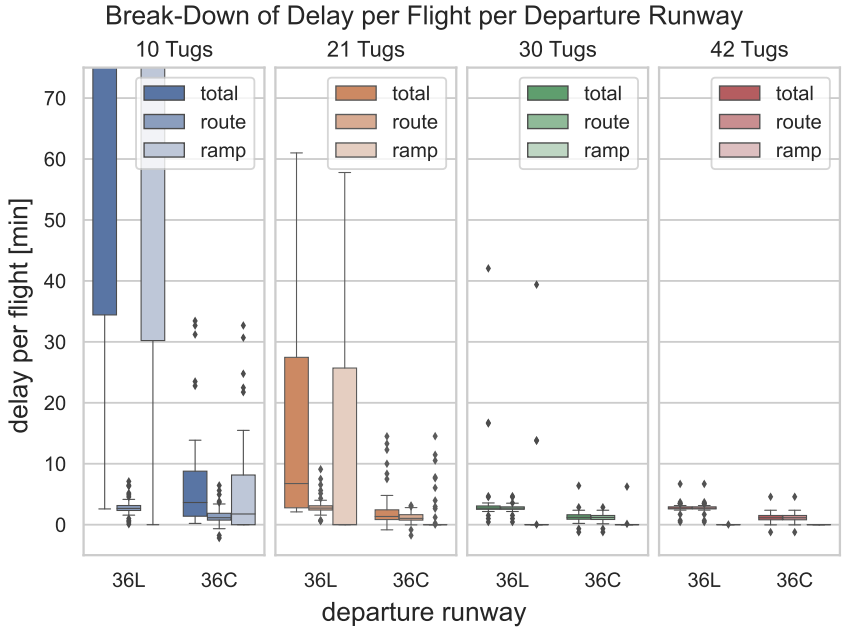


Figure 9: Break-down of delay for departing flights in the outbound peak, split per runway assigned and dependent on the number of tugs available.

Ramp Delay per Departing Flight per Runway
For scenarios in which limited tugs are available, it can be observed that large differences in ramp delay per runway exist. The results from Table 11 and Figure 9 show that the differences in total delay per flight between both runways increase for decreasing number of tugs available. For a scenario in which 10 tugs are available, a large statistical difference in total delay per flight $\delta d_{36L,\,out}$ can be observed for runway 36L when compared with unlimited tugs available, whereas this difference is small for runway 36C ($\delta d_{36C,\,out}$). Similar behaviour can be seen for a scenario with 21 tugs available. For a scenario with 30 tugs, no differences in total delay per flight are detected when comparing against a fleet size of 42 tugs.

In subsection 7.2.1, it was concluded that ramp delay is the dominant factor in the total delay per flight when considering scenarios in which limited tugs are available. The break-down of the delay per flight per runway in Figure 9 supports this conclusion. An increase of the median value and spread of the distribution of ramp delay per flight departing from 36L is reflected in an increase for median and

spread of the total delay per flight as well. This result is an indication that the differences in total delay per flight per runway are mainly caused by differences in ramp delay per runway.

A possible explanation for the large differences in ramp delay per runway concerns the implicit priority given to flights departing from 36C in the allocation phase. Due to the smaller taxi times and tug return times for runway 36C, the completion of tasks to be delivered at 36C takes less time than the tasks to be delivered at 36L. Therefore, adding a flight into the schedule of the tug that is departing from 36C will have less effect in terms of delay compared to a flight scheduled at 36L on all other flights that are already present in the schedule of a tug. Tug formulate bids for any outbound flight based on the additional delays arising in their schedule due to towing the flight to its designated runway. Since tasks are assigned in increasing order of costs, flights that have less effect on the delay in the schedule of the tug will be assigned first. As a consequence, it could be that only after the majority of the flights departing from 36C are assigned, the flights departing from 36L are allocated.

In Table 12, characteristics of the distribution are shown that represent in what round a flight departing from 36L or 36C is allocated in the TeSSI auction process, when having 10 tugs available ($i_{auction}$). The median value of this KPI for flights departing from 36L is indeed higher than that of flights departing from 36C, meaning that in general, aircraft scheduled to depart from 36C are assigned earlier in the TeSSI auction process. In addition, the distribution representing the index of a flight departing from 36L or 36C in the schedule of the tug ($i_{schedule\ tug}$) is provided for both runways in Table 12. When analyzing this distribution, it can be observed that aircraft departing from 36L are in general placed at a higher index in the schedule of the tug than aircraft departing from 36C. In other words, aircraft departing form 36L are in general placed more towards the end in the schedule of the tug. These results confirm the statement that an implicit priority is given to aircraft departing from 36C in the allocation process, resulting in large differences in the ramp and total delay per flight between runway 36L and 36C.

Table 12: Comparison of auction characteristics for flights departing from 36L and 36C. Statistical significance = 0.05/1 = 0.05. Statistical difference of the distribution is determined using the Mann-Whitney U-test.

KPI [min]	**10 tugs**		
	36L	36C	comparison
	median (Q1, Q3)	median (Q1, Q3)	p-value, (A-test)
$i_{auction}$	$17(7, 31)$	$15(7, 21)$	$< 0.001, (0.58, S)$
$i_{schedule\ tug}$	$7(5, 8)$	$2(1, 4)$	$< 0.001, (0.88, L)$

8 Discussion

In the previous section, results are presented for the set of scenarios analyzed to evaluate algorithmic and operational performance. The interpretation and broader implications of these results will be discussed in subsection 8.1, as well as the contributions to the academic field and recommendations for future work. In subsection 8.2, reflections on the assumptions and methodology will be provided.

8.1 Discussion on the Results

In this work, we have evaluated the performance of a novel algorithmic framework in the context of airport ground surface movement operations. First, focus will be put on the performance of the framework from an algorithmic point of view (subsection 8.1.1). Afterwards, the performance of the framework in terms of operational output is evaluated in subsection 8.1.2, focusing on how tug-enabled taxiing operations compare to MET operations in terms of ground delay for varying fleet sizes.

8.1.1 Discussion on Algorithmic Results

The outcome of this study shows that the PBS-TA framework is capable of solving the MAPD problem for aircraft engine-off towing operations at a major hub airport, using either a coupled or decoupled MAPD solving technique. Both approaches have shown to provide for a conflict-free solution for various number of outbound flights and various number of tugs. In terms of total solution cost, the results point

out that CPL-MIN provides for equal or lower total solution cost when compared with the DCPL algorithm in the scenarios analyzed. The maximum decrease in total solution cost was found to be 2.7% and becomes less for increasing the number of agents in the system. However, due to the significant increase of run times (for smaller fleet sizes, approximately 300% increase), the use of a coupled approach for this specific application is not preferred over the use of a decoupled approach.

From the results of this study, a proposal is drafted for scenarios in which the application of coupled approaches can be of added value. In general, it is expected that the use of a coupled approach is especially valuable in real-world problems for which the routing of agents is highly dependent on the allocation of tasks to agents. In the context of fully outbound towing, this is only the case to a limited extent. Both the pickup (gate area) and delivery locations (one or two departing runways) for all tasks are located relatively closely together. In other words, independent of the aircraft assigned, all tugs will have similar high-level routes from the pickup location to the delivery location. In combination with the fact that only limited taxiways exist that connect the pickup and delivery locations, conflicts arising on these routes are expected to be influenced only very limited by the exact allocation and more by factors such as runway throughput and capacity of all decoupling locations. Thus, exploring different allocations when using a coupled approach in this context is seen to have only limited effect on the total solution cost. Applications with sparsely located pickup and delivery locations are expected to be more suitable for the application of integrated approaches that solve task allocation and path planning simultaneously. Future research will have to further investigate this.

Moreover, the solution quality of the simulation is expected to increase if consequences of the current allocation on the future dynamics of the system are accounted for. If estimates of the effects of an assignment choice in future planning windows can be taken into account, we move one step closer to finding an allocation that is more likely to be near the global minimum. The inherently uncertain dynamics of airport ground surface movement operations add an extra complexity layer to this aspect. Future research should point out whether efficient allocation of tasks to agents can be done while accounting for propagating effects of current assignment choices.

8.1.2 Discussion on Operational Results

With regards to the operational performance of an airport upon implementation of tug-enabled taxiing operations, three general conclusions can be drawn. First, the results of this work show that the difference in delay for departing flights between tug-enabled taxiing and MET operations when having unlimited tugs available is caused by an increase in taxi time. The increase is caused by a lower maximum taxiing velocity of tug-aircraft combinations compared with aircraft and longer decoupling procedures compared with conventional decoupling of the pushback truck. It is shown that the taxi time for outbound flights is not affected by the size of the towing fleet. This result implies that it would be beneficial if the maximum velocity of the tug increases and/or decoupling times decrease. Future research should point out whether it is technically feasible to increase the speed for tug-aircraft combinations up to conventional taxiing speeds. For airports that consider implementation of towing operations, it is of importance to devote time and resources into training of personnel to ensure efficient decoupling operations.

Second, when having limited tugs available, ramp delay is a dominant factor in the total delay per departing flight. For decreasing number of tugs, the median value for both the ramp and total delay per flight do not change significantly, until a tipping point is reached. For both the inbound and outbound peak, this tipping point is in between 10 and 21 tugs available. Note that in this study, fully outbound towing is considered, meaning that all departing flights should be towed towards the runway. In order to limit ground delays and make more efficient use of tugs, it is suggested to devote attention in future research on the determination of a strategy on how to decide which aircraft will be towed and which aircraft will use conventional taxiing techniques, if having limited tugs available.

Third, analysis of the delay per runway when having limited tugs available showed that the delay per flight for aircraft departing from runway 36L increases significantly with respect to the delay for aircraft assigned to depart from 36C. However, from an environmental perspective, runway 36L is identified as the most promising runway for tug-enabled taxiing due to its remote location and as a result, longer engine-on times. The implicit priority given by TeSSI to flights departing from 36C is expected to be the reason for the significant increase in delay between 36L and 36C for decreasing number of tugs available. Therefore, a major improvement of the current PBS-TA framework concerns the performance of TeSSI when having to deal with a set of tasks in which two subsets of tasks exist, that exhibit large differences in cost per group. Especially for scenarios in which limited tugs are available, we can see "greedy" behaviour of TeSSI by first assigning all tasks with lower costs (flights departing from 36C). Since the

degrees of freedom decrease with every new auction round, suboptimal allocations are generated if tasks with higher costs are assigned at the end of each auction. This behaviour points out that TeSSI is not capable of taking all synergies between tasks in the system into account. For future work on similar problems, the authors suggest to investigate the tuning of TeSSI or the application of other allocation algorithms to improve this behaviour. As a consequence, a contribution could be provided on how to allocate flights to tugs such that aircraft departing from a runway that provides for the largest fuel savings are not delayed more than aircraft departing from other runways.

8.2 Reflection on Assumptions and Methodology

The proposed PBS-TA framework and algorithms have proved their capability to provide insights on the applicability of integrating allocation with path planning in the field of airport ground surface movements. In addition, this work has provided practical insights in the application of tug-enabled taxiing operations compared against conventional taxiing techniques. However, a number of practical limitations related to the assumptions or implementation of the methodology can also be identified. First, the role of Air Traffic Control (ATC) in the implementation of tug-enabled taxiing operations is left out of scope in this study. However, if tugs will be crossing taxiways while driving in solo mode, clearance is required in current operations. Future research should point out how the implementation of tug-enabled taxiing operations affects current ATC procedures and workload.

Furthermore, this study has looked into the operational consequences of tug-enabled taxiing operations for RMO North. It is suggested to explore other runway configurations in future research, and evaluate on the major operational differences for different runway configurations.

Finally, all information available to the agents in the system as well as the operations themselves are considered to be deterministic in this research. However, real-life airport ground surface movement operations are inherently dynamic and uncertain in nature. In this study, a fixed buffer time is included for the allocation of tasks. It would be of interest to evaluate in detail how buffer times can be used during planning in future research. In addition, uncertainty in the execution of the solution is considered to be out of the scope for this study. If a feedback loop is included in the model that allows for optimizing the allocation of flights to tugs based on real-time status information, the practical applicability of the model and framework would increase significantly.

9 Conclusion

In this study, a novel algorithmic framework is proposed that allows for comparison of conventional taxiing techniques with tug-enabled taxiing operations, while integrating multi-agent task allocation and path planning. The framework combines the Temporal Sequential Single-Item (TeSSI) auction algorithm with Priority-Based Search (PBS) and Safe Interval Path Planning (SIPP) to generate a search forest in which the consequences of an allocation in the path planning domain are explored. A simplified network representation of Amsterdam Schiphol Airport (AMS) is implemented, that is used as a case study for the evaluation of both the algorithmic and operational performance of the framework. For varying number of available tugs in an outbound and inbound peak, various types of delay per flight are used to evaluate algorithmic and operational performance of the proposed framework and airport system respectively.

Analysis showed that an integrated approach for allocation and path planning did result in a decrease of the sum of the total delay per flight for all aircraft when compared to an approach where allocation and path planning are performed separately. However, since this decrease was very minimal (maximum of 2.7% decrease) and due to a significant increase in required run times (between 100-300% increase), the added value of the coupled approach is considered to be minimal for aircraft engine-off towing operations. In general, the authors recommend to make use of a coupled approach in applications where pickup and delivery locations of tasks are not clustered, but sparsely located.

Upon evaluation of tug-enabled taxiing compared to MET operations, results show zero delay for all inbound flights. A total delay per departing flight ranging between 2 and 3 minutes (depending on the peak) is found in a scenario with unlimited tugs available, mainly caused by a lower maximum velocity of tug-aircraft combinations compared with aircraft and longer process times for decoupling procedures. When decreasing the number of available tugs, ramp delay becomes an increasingly dominant factor in the total delay per flight. Analysis shows that the median value of the ramp delay is not linearly related to the fleet size. Furthermore, the total delay per flight for runways that are located further away from the apron area is found to be more sensitive to the number of available tugs. With regards to providing practical insights to airports that consider implementation of tug-enabled taxiing operations, the authors

recommend to investigate how to determine which departing flights should be towed towards the runway and for which departing flights it is best to use conventional taxiing techniques, taking into account the interests of all stakeholders involved (airport, airlines, passengers and ground handlers).

References

[1] H. Udluft. "Decentralization in Air Transportation". Delft University of Technology, 2017. DOI: 10.4233/uuid:2af25aa3-be84-4d37-9332-fe2af319db1e.

[2] International Civil Aviation Organization. *ICAO forecasts complete and sustainable recovery and growth of air passenger demand in 2023*. Accessed on: May 22th, 2023. [Online]. Available: https://www.icao.int/Newsroom/Pages/ICAO-forecasts-complete-and-sustainable-recovery-and-growth-of-air-passenger-demand-in-2023.aspx. Feb. 2021.

[3] Airport Operators Association. *Aircraft on the Ground CO2 Reduction Programme*. Tech. rep. 2010.

[4] Royal Schiphol Group. *Sustaining Your World*. Accessed on: June 28th, 2022. [Online]. Available: https://www.schiphol.nl/en/schiphol-group/page/road-to-the-most-sustainable-airports/. 2022.

[5] Advance Engine Off Navigation (AEON). *Engine-off taxiing techniques*. Accessed on: April 4th, 2022. [Online]. Available: https://www.aeon-project.eu/engine-off-taxiing-operations/.

[6] M. Lukic et al. "State of the Art of Electric Taxiing Systems". In: *International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS) and International Transportation Electrification Conference (ITEC)*. Nov. 2018. DOI: 10.1109/ESARS-ITEC.2018.8607786.

[7] P. Vaishnav. "Costs and Benefits of Reducing Fuel Burn and Emissions from Taxiing Aircraft: Low-Hanging Fruit?" In: *Transportation Research Record* 2400.1 (2014), pp. 65–77. DOI: 10.3141/2400-08.

[8] J.L. Adler and V.J. Blue. "A Cooperative Multi-Agent Transportation Management and Route Guidance System". In: *Transportation Research Part C: Emerging Technologies* 10.5 (2002), pp. 433–454. DOI: 10.1016/S0968-090X(02)00030-X.

[9] H. Ma et al. "Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks". In: *Autonomous Agents and Multi-Agent Systems* (May 2017), pp. 837–845.

[10] K. Yakovlev et al. "On the Application of Safe-Interval Path Planning to a Variant of the Pickup and Delivery Problem". In: *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2020)*. 2020, pp. 521–528. DOI: 10.5220/0009888905210528.

[11] M. Liu et al. "Task and Path Planning for Multi-Agent Pickup and Delivery". In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*. Ed. by N. Agmon et al. May 2019.

[12] X. Wu et al. "Multi-Agent Pickup and Delivery with Task Deadlines". In: *Proceedings of the 14th International Symposium on Combinatorial Search (SoCS 2021)*. Vol. 12. 01. Association for the Advancement of Artificial Intelligence (AAAI), 2021, pp. 206–208.

[13] W. Hönig et al. "Conflict-Based Search with Optimal Task Assignment". In: *Proceedings of the 17th International Conference for Autonomous Agents and Multiagent Systems (AAMAS 2018)*. Ed. by M. Dastani et al. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 757–765. DOI: 10.5555/3237383.3237495.

[14] Z. Chen et al. "Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery". In: *IEEE Robotics and Automation Letters* 6.03 (2021), pp. 5816–5823. DOI: 10.1109/LRA.2021.3074883.

[15] J. Soomers. "Hierarchical Multi-Agent Path Planning For Delay Mitigation in Airport Engine-Off Towing System". Delft University of Technology, May 2022.

[16] J.G. Kamphof. "Design and Analysis of Tug-Enabled Engine-Off Taxiing Operation Using Hierarchical Multi-Agent Planning". Delft University of Technology, 2022.

[17] E. Nunes and M. Gini. "Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints". In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 2110–2216. DOI: 10.5555/2886521.2886614.

[18] M.F. Duzijn. "Literature Study: Integration of Multi-Agent Task Allocation and Path Finding in Aircraft Engine-Off Towing Operations". Delft University of Technology, June 2022.

[19] L. Mantecchini, M.N. Postorino, and E. Gualandi. "Integration Between Aircraft And Handling Vehicles During Taxiing Procedures To Improve Airport Sustainability". In: *International Journal of Transport Development and Integration* 1 (May 2016), pp. 28–42. DOI: 10.2495/TDI-V1-N1-28-42.

[20] N.J.F.P. Guillame. "Finding The Viability Of Using An Automated Guided Vehicle Taxiing System For Aircraft". Delft University of Technology, Apr. 2018.

[21] E.V.M. Van Baaren. "The Feasibility Of A Fully Electric Aircraft Towing System". Delft University of Technology, May 2019.

[22] To70 Aviation. *Sustainable Taxi at Schiphol Airport - FTS Study*. Tech. rep. 2020.

[23] Royal Schiphol Group. *Amsterdam Airport Schiphol Facts: What You Would Like To Know*. Accessed on: June 21th, 2022. [Online]. Available: https://www.schiphol.nl/nl/route-development/pagina/amsterdam-airport-schiphol-airport-facts/. 2022.

[24] Royal Schiphol Group. *Final Results Taxibot Proof of Concept*. Tech. rep. 2021.

[25] Royal Schiphol Group. *Feasibility Study Sustainable Taxiing at Schiphol*. Tech. rep. 2021.

[26] I.F.A. Vis. "Survey of research in the design and control of automated guided vehicle systems". In: *European Journal of Operational Research* 170.3 (2006), pp. 677–709. DOI: 10.1016/j.ejor.2004.09.020.

[27] H. Ma et al. "Searching with Consistent Prioritization for Multi-Agent Path Finding". In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. AAAI'19. AAAI Press, 2019, pp. 7643–7650. DOI: 10.1609/aaai.v33i01.33017643.

[28] M. Phillips and M. Likhachev. "SIPP: Safe Interval Path Planning for Dynamic Environments". In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5628–5635. DOI: 10.1109/ICRA.2011.5980306.

[29] A. Khamis ad A. Hussein and A. Elmogy. "Multi-robot Task Allocation: A Review of the State-of-the-Art". In: *Cooperative Robots and Sensor Networks 2015*. Ed. by A. Koubâa and J.R. Martínez-de Dios. Springer International Publishing, 2015, pp. 31–51. DOI: 10.1007/978-3-319-18299-5_2.

[30] P. Rizzo. "Auction-Based Task Allocation for Online Pickup and Delivery Problems with Time Constraints and Uncertainty". Delft University of Technology, Feb. 2021.

[31] M.B. Dias et al. "Market-Based Multirobot Coordination: A Survey and Analysis". In: *Proceedings of the IEEE* 94.7 (2006), pp. 1257–1270. DOI: 10.1109/JPROC.2006.876939.

[32] M. Lagoudakis et al. "Auction-Based Multi-Robot Routing". In: June 2005, pp. 343–350. DOI: 10.15607/RSS.2005.I.045.

[33] R. Dechter, I. Meiri, and J. Pearl. "Temporal Constraint Networks". In: *Artificial Intelligence* 49.1 (1991), pp. 61–95. DOI: 10.1016/0004-3702(91)90006-6.

[34] D. Eppstein. "k-Best Enumeration". In: *Encyclopedia of Algorithms*. Ed. by M.Y. Kao. Springer New York, 2016, pp. 1003–1006. DOI: 10.1007/978-1-4939-2864-4_733.

[35] J. Han, M. Kamber, and J. Pei. "2 - Getting to Know Your Data". In: *Data Mining*. Third Edition. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 39–82. DOI: 10.1016/B978-0-12-381479-1.00002-2.

[36] R.F. Woolson. "Wilcoxon Signed-Rank Test". In: *Wiley Encyclopedia of Clinical Trials*. John Wiley & Sons, Ltd, 2008, pp. 1–3. DOI: 10.1002/9780471462422.eoct979.

[37] A. Vargha and H. D. Delaney. "A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong". In: *Journal of Educational and Behavioral Statistics* 25.2 (2000), pp. 101–132. DOI: 10.3102/10769986025002101.

[38] S. S. Shapiro and M. B. Wilk. "An Analysis of Variance Test for Normality (Complete Samples)". In: *Biometrika* 52.3/4 (1965), pp. 591–611.

II

Literature Study

previously graded under AE4020

1

Introduction

The aviation sector is dealing with a number of conflicting challenges. On one hand, the expected growth rate for air transportation cannot be accommodated by the current infrastructure, that is already reaching its limits [1]. Especially airport congestion and ATC workload are identified as two major bottlenecks. The addition of infrastructure at airports to reduce congestion is often not only infeasible due to limited space available, but also adds to the complexity of operations resulting in increasing human workload. Therefore, a need arises to increase the efficiency of current ground surface operations. On the other hand, the urge for the airspace industry to become greener becomes more and more pressing. The current global aviation sector is responsible for almost 3 % of global greenhouse gas emissions, which is expected to increase further with increasing demand for air transportation [2]. From an airport point of view, almost 30 % of its emissions can be attributed to aircraft ground surface movements [3]. One of Europe's biggest hub airports, Amsterdam Airport Schiphol (AAS), has announced its ambition to provide for emission-free operations by 2030 [4].

In order to tackle the challenges of increasing demand for air transportation combined with the urge for the aviation sector to become more sustainable, a number of research projects have been launched, including the Advanced Engine Off Navigation (AEON) project. This research focuses on the implementation of more environmental-friendly concepts of taxiing. One of the promising concepts identified by the AEON project is the use of external towing tugs, such as the TaxiBot [5], that has shown to provide for almost 50% of fuel savings [6]. However, the expected downsides of TaxiBots are increased process times and increased traffic density, possibly leading to additional delays [7].

In order to study the effects of implementing TaxiBots at large airports in more detail, previous studies have been conducted on overall system performance using a multi-agent simulation model. Although this model has shown to provide for safe, realistic and efficient routing of aircraft and TaxiBots over the airport ground surface [8, 9], it has not considered the allocation of TaxiBots to aircraft. This MSc thesis will continue on the previous research by explicitly focusing on how to perform assignment of aircraft to TaxiBots in a manner that allows to adjust for real-time events.

Thus, the aim of this thesis is to elaborate the existing multi-agent model with a task allocation part, that allows for more realistic operations. In addition, focus will be put on how to integrate task allocation with path finding. From an academic point of view, especially the latter is of interest: to the best of our knowledge, no research before has implemented an integrated algorithm that combines task allocation and path finding in a real-world airport application before. The results of this study will therefore provide insights on how task allocation algorithms can be implemented in real-life pickup and delivery problems. From a practical point of view, it is of interest to see how the airport system is affected by the implementation of TaxiBots, especially in terms of throughput, delays and emissions. Amsterdam Airport Schiphol (AAS) will be used as a case study to generate results on the latter three items. Both goals are summarized into the following research objective: *"To design and evaluate an approach for combining task allocation with path finding for a cooperative fleet of TaxiBots to perform fully outbound towing at AAS, using a hierarchical multi-agent control architecture."*

In this literature study, the basis will be formed to tackle the aforementioned research objective. First, conventional ground surface operations will be investigated on in chapter 2. Special attention will

be paid on how ground surface operations at AAS take place, since this airport will be used as our case study. In chapter 3, developments in the field of ground surface operations will be discussed, with a focus on various aircraft engine-off taxiing techniques. This chapter will be concluded with the definition of the research gap to be fulfilled in this study. Next, based on previous work, a concept of operations regarding the implementation of TaxiBots at AAS will be defined and elaborated on in chapter 4. Based on the concept of operations, a set of system requirements is developed, and a suitable modelling technique is defined. In chapter 5, chapter 6 and chapter 7, the subject of task allocation, path finding and the combination of the two will be discussed respectively. Based on a literature review and trade-off as provided in these three chapters, a combination of algorithms is chosen to implement in the remainder of the thesis. A formal research proposal is then defined in chapter 8, including research questions, methodology and time plan.

2

Conventional Airport Surface Movement Operations

In order to understand the context and relations of the problem under consideration in this study, it is of importance to gain a basic understanding of the airport ground movement problem. In essence, this problem deals with scheduling and routing aircraft from the gate to the runway (and vice versa) in a safe and efficient manner [10]. In this chapter, an overview will be given of ground surface operations in general (section 2.1). First, the key components of ground surface operations from the aircraft point of view will be discussed, followed by an elaboration on the Airport Collaborative Decision Management (A-CDM) system used to monitor ground surface operations from an airport and control point of view. A zoom-in on taxi operations specifically will be provided, including a description on involved stakeholders and interaction between them.

In section 2.2, ground surface operations at Amsterdam Airport Schiphol (AAS) in particular will be discussed, focusing primarily on the available infrastructure (runways, the taxiway network, service roads and the apron area). Although airports share many similarities, AAS has been chosen as the case study for this thesis due to several reasons: first, the airport is among one of the largest airports in Europe, both in terms of layout and passenger flow [11]. Next to that, Schiphol has already performed several trials and investigations into the subject of sustainable taxiing, providing for relevant practical data (refer to chapter 4 for more details on these studies).

2.1. General Overview of Ground Surface Operations

This section deals with the context in which the airport ground movement problem should be placed. The subsection 2.1.1 provides a general overview of the key components in the ground surface operations viewed from an aircraft's perspective. Next, the Airport Collaborative Decision Management (A-CDM) system will be elaborated on in subsection 2.1.2, providing insights in how ground surface operations are managed from an airport and control perspective. Specific attention will be paid to the part of taxi operations within the A-CDM system. Finally, the stakeholders within the airport surface movement operations and interactions between them will be discussed in subsection 2.1.4.

2.1.1. Process Flow in Ground Surface Operations

The operation flow of aircraft in airports is visualized in Figure 2.1 [12]. After having completed the turnaround process for a departing flight, the aircraft waits for instructions by Air Traffic Control (ATC) on the assigned runway and taxi route. When clearance is provided, the aircraft is pushed back and starts moving from the apron area through the taxiway to the runway. When arrived at the runway entrance, the aircraft waits for ATC clearance to take off.

When arriving, the aircraft follows ATC instructions on exiting the runway to the taxiway. The taxi route and assigned gate are determined prior to the aircraft entering the taxi system. As soon as the aircraft arrives at the gate, the turnaround process can start. A more detailed description on the taxi operations is given in subsection 2.1.3.

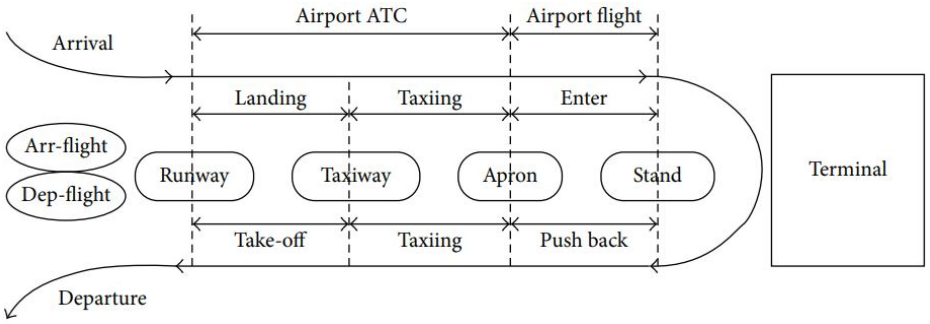


Figure 2.1: General overview of flow in ground surface operations. Taken from [12].

During the entire process of the aforementioned ground movements, different stakeholders are involved. More details on stakeholders are provided in subsection 2.1.4. In order to ensure efficient use of resources and cooperation of all parties involved based on accurate and high-quality information, EUROCONTROL has introduced the concept of Airport Collaborative Decision Management (A-CDM). The main goals are to "improve operational efficiency, predictability and punctuality to the ATM network and airport stakeholders" [13, p. 1-1]. In subsection 2.1.2, a general overview of the key principles of this approach will be given.

2.1.2. Airport Collaborative Decision Management (A-CDM)

The A-CDM system consists of six key elements [13]: Information Sharing, the Milestone Approach, Variable Taxi Time, Collaborative Pre-Departure Sequence, CDM in Adverse Conditions and Collaborative Management of Flight Updates. A brief description of these elements will be elaborated on below.

1. **Information Sharing:** in order to achieve common situational awareness among all relevant stakeholders, the most basic requirement is the sharing of accurate and high-quality information with all parties involved. This includes information from aircraft operators/ground handlers, the airport itself, network operations, ATC and other service providers (think of meteorological information).
2. **The Milestone Approach:** in different phases of the ground operation seen from the perspective of an aircraft, a sequence of different events can be identified, corresponding with so-called milestones. These milestones are used to track the progress of the turn-around process and provide for an enhanced situational awareness for the parties involved.
3. **Variable Taxi Time:** instead of making use of a standard default value for the expected time for aircraft to taxi, the A-CDM approach recommends using realistic individual taxi times. As a result, ATC is able to optimize push back, taxi and take off sequence to reduce queuing and taxiway congestion.
4. **Pre-Departure Sequencing:** a complementary method to reduce queuing at the runway, is to determine a more optimal sequence of departing flights based on the progress in turn-around. This is opposed to the frequently used principle of "first come, first served", where the aircraft is cleared by ATC based on the order of finishing the turn-around process. Instead, when ATC can decide to for example postpone pushing back an aircraft in some cases, a more regulated traffic flow with less waiting times at the runway may be the overall (improved) result.
5. **Adverse Conditions:** by taking into account the probability of adverse conditions that can be foreseen with reasonable accuracy, the impact of situations of reduced capacity can be predicted more easily. This would allow for managing these situations of reduced capacity more optimally and returning to normal operating conditions more swiftly.
6. **Collaborative Management of Flight Updates:** in order to provide for more accurate estimates of arriving flights to A-CDM airports and to improve departure slot management, the next key element concerns sharing information (Flight Update Messages) between A-CDM airports and EUROCONTROL Network Managers.

2.1.3. Taxi Operations Viewed in the Light of A-CDM

In this section, an elaboration will be provided on how conventional ground and taxi operations are handled described from the perspective of the A-CDM approach. First, the most important events related to taxiing from the Milestone Approach will be elaborated on. Next, two key elements of the A-CDM approach that aim for decreasing of delays through optimizing taxi operations are described: Variable Taxi Time (VTT) and Pre-Departure Sequencing.

In the baseline model of this thesis, no unexpected events causing delay are considered. Although the baseline model might be extended to deal with uncertainty depending on available time and resources, no further elaboration on the key element related to Adverse Conditions will be provided in this literature study. Similarly, since the focus of this thesis will be on AAS as an individual airport without considering interaction between other airports, no further details are provided on the key element related to Collaborative Management.

The Milestone Approach

Within the Milestone Approach, three different phases of the flight can be distinguished: inbound, turnaround and outbound. Within these three phases, different milestones can be defined, as shown in Figure 2.2. From the perspective of taxiing, the milestones from the Actual Landing Time (ALDT) to the Actual Take Off Time (ATOT) are especially of interest (milestone 6 to 16). Below, a description will be given of these milestones, based on [13], in order to gain a general understanding of the sequence of events related to taxiing operations.

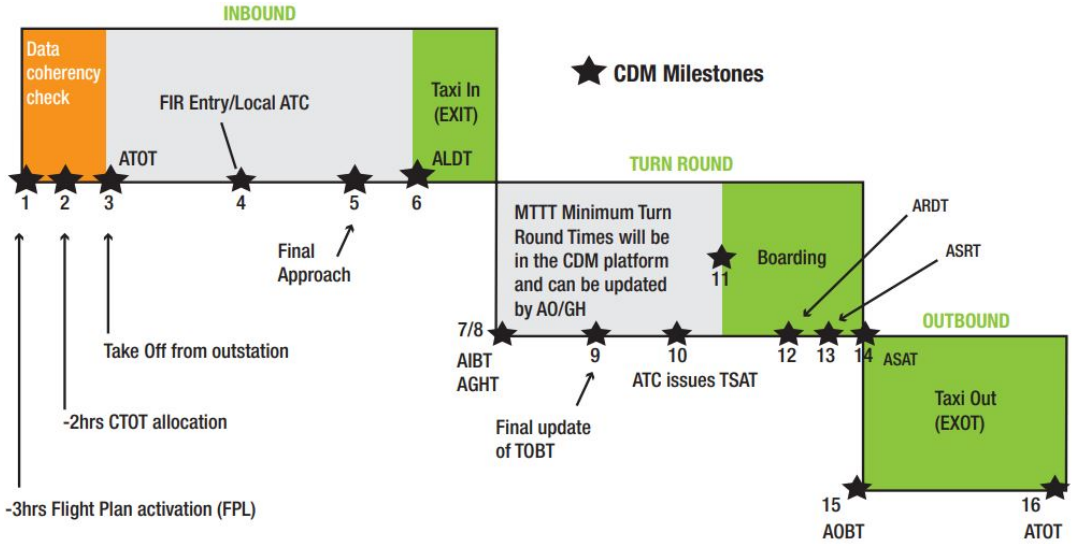


Figure 2.2: Overview of Milestone Approach, including all milestones for different phases of the flight. Taken from [13].

- **INBOUND**
  - **Milestone 6 - Landed**: as soon as the aircraft touches down on the runway, the Actual Landing Time (ALDT) is determined and saved to the CDM system. The aircraft exits the runway and an estimate for the Estimated Taxi-In Time (EXIT) is used to determine Estimated In-Block Time (EIBT). Based on the ALDT, the Target Off-Block Time (TOBT) and Target Take Off Time (TTOT) are updated accordingly.
  - **Milestone 7 - In-Block**: once the aircraft has arrived at its parking position (Actual In-Block Time (AIBT)), again the TOBT and TTOT are updated. In addition, the AIBT triggers the start of milestone 8.
- **TURN ROUND**
  - **Milestone 8 - Ground Handling Started**: as soon as the aircraft is parked (AIBT), the turnaround process can start. Based on the Estimated Turn-Round Time (ETTT), the TOBT and TTOT are updated.

- **Milestone 9 - Final Confirmation of TOBT**: based on the operational situation, a final estimation of the TOBT is provided by the Aircraft Operator (AO)/Ground Handler (GH) t minutes before the Estimated Off-Block Time (EOBT) (where t is a parameter agreed upon for each airport specifically). An accurate TOBT is of importance to ATC to be able to optimize the pre-departure sequence.
- **Milestone 10 - TSAT Issued**: ATC issues the Target Start Up Approval Time (TSAT) to the pilots once the flight is scheduled as part of the pre-departure sequence. This provides for guidance for the ground crew to manage the turn-around process accordingly. Again, the TTOT might be updated based on the issuing of the TSAT.
- **Milestone 11 - Boarding Starts**: this milestone corresponds with the moment that passengers physically start boarding the aircraft.
- **Milestone 12 - Aircraft Ready**: the Actual Ready Time for Movement (ARDT) corresponds with the moment that boarding is completed and the boarding bridge is removed, all doors are closed, the push-back vehicle is connected (if push-back is required) and that the aircraft is ready to move directly upon instructions from ATC.
- **Milestone 13 - Start Up Requested**: based on request of the pilot to ATC, clearance is asked to start up the engines. This milestone correspondes with the Actual Start Up Request Time (ASRT). Provided that the ARDT was on time, it is now the responsibility of ATC to ensure that the flight meets its Calculated Take Off Time (CTOT).
- **Milestone 14 - Start Up Approval**: after the request to start-up, ATC approves start up, corresponding with the Actual Start Up Approval Time (ASAT). After the approval, the aircraft starts up, is pushed back and starts taxiing towards the designated runway.

- **OUTBOUND**
  - **Milestone 15 - Off-Block**: the moment that the aircraft actually starts push-back corresponds with the Actual Off-Block Time (AOBT). Based on the AOBT, the ATOT is updated considering the Estimated Taxi-Out Time (EXOT).
  - **Milestone 16 - Take Off**: once the aircraft takes off, the ATOT is saved to the CDM system.

Table 2.1: Definition of metrics used in the milestones approach related to taxiing. Based on [13].

Metric	Definition
Actual Taxi-In Time (AXIT)	Actual In-Block Time (AIBT) - Actual Landing Time (ALDT)
Actual Taxi-Out Time (AXOT)	Actual Take Off Time (ATOT) - Actual Off-Block Time (AOBT)
Estimated Taxi-In Time (EXIT)	Estimated In-Block Time (EIBT) - Estimated Landing Time (ELDT)
Estimated Taxi-Out Time (EXOT)	Estimated Take Off Time (ETOT) - Estimated Off-Block Time (EOBT)

Several important times can be identified in the Milestone Approach related to taxiing of aircraft. These metrics and their definitions according to [13] have been summarized in Table 2.1. Based on these definitions, it becomes apparent that minimizing the difference between the actual taxi time and the estimated taxi time (both for inbound and outbound flights) is of importance to limit delays as much as possible.

Variable Taxi Time

In order to provide for taxi times as accurate as possible, the key element of Variable Taxi Time (VTT) is incorporated in the A-CDM approach. This will not only enable ATC to optimize the pre-departure sequence and thus, reduce runway queuing, but also allows to improve CTOT compliance [13]. Especially at medium to large airports where multiple runway configurations and lay-out of terminal buildings can affect taxi time for arriving and departing flights significantly, it is recommended to adopt the approach of VTT [13]. Instead of using default taxi times solely based on runway configuration (e.g. landing runway X corresponds with t minutes taxi-in time), the following additional factors should be taken into account [13]:

- **Operational Expertise**: the parties involved in ground operations (e.g. ATC, AOs, GHs) have great experience in the movement of aircraft, and therefore, can provide for valuable information regarding the estimation of taxi times.
- **Aircraft Type/Category**: since the type of aircraft can be of significant influence on the taxi time, this factor should be involved in the estimation of the taxi time.
- **Historical Data**: based on statistical data on average taxi times for specific aircraft types, the default taxi time can be improved. In this case, a differentiation should be made for different runway configurations, stands (or groups of stands) and different periods of the year/day (e.g. weekdays vs. weekends).
- **Operational Conditions**: default taxi times are independent of for example weather conditions. When these operational conditions are taken into account, more reliable estimates can be provided. In addition, when queuing near the runway frequently occurs, it is necessary to add additional time to account for these delays.
- **Advanced Monitoring Systems**: with the use of more advanced tools, such as Advanced Surface Movement Guidance and Control System (A-SMGCS) that include both a routing and surveillance service, accurate taxi times can be provided and updated more dynamically [14].

Pre-Departure Sequencing

Another key element of the A-CDM approach to decrease queuing at the runway (next to VTT), is to make use of pre-departure sequencing. One of the CDM airports that has implemented this technique is AAS. In Figure 2.3, the process of Collaborative Pre-Departure Sequence Planning (CPDSP) that is implemented at Schiphol is graphically depicted. If a flight is IFR, planned to leave on one of the main runways and has a known TOBT, the earliest possible TTOT' is determined taking into account the situation-specific EXOT. The actual TTOT is then determined not on a 'first come, first served'-basis, but determined through an optimized take-off sequence. This sequences in its turn leads to an optimized TSAT for each individual flight by subtracting the EXOT from the TTOT [15].

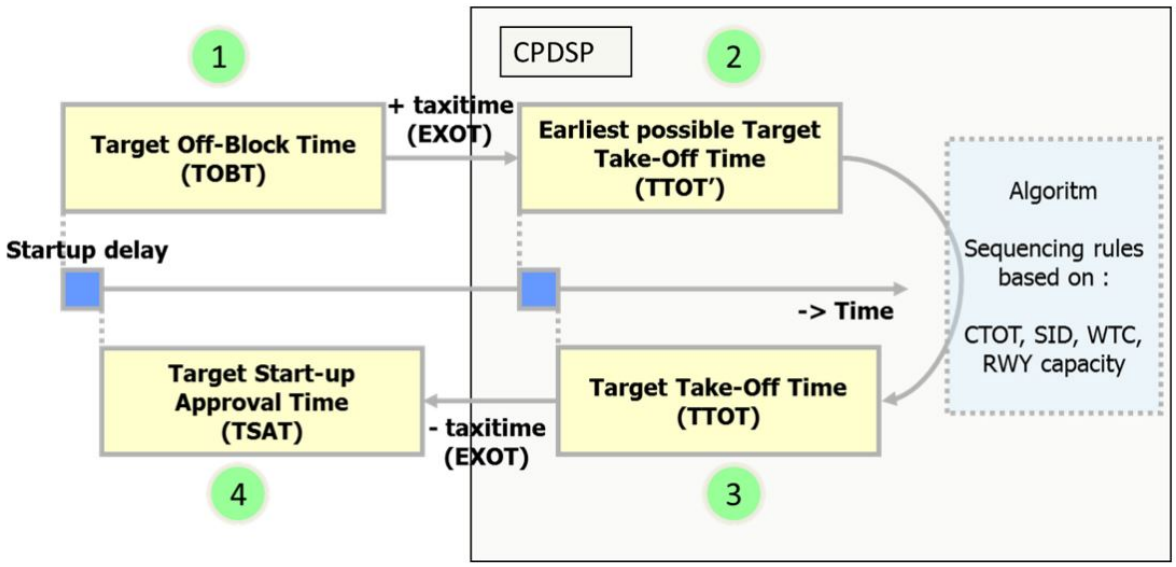


Figure 2.3: Graphical representation of Collaborative Pre-Departure Sequence Planning (CPDSP) implemented at Schiphol Airport (CDM airport) [15].

2.1.4. Communications and Interactions Related to Ground Surface Operations

In subsection 2.1.2 and subsection 2.1.3, it has become apparent that many different stakeholders are involved in the operational process related to ground handling at airports. In order to provide clarity on the interactions and responsibilities of these stakeholders, a visual representation of this system

has been provided in Figure 2.4, based on previous work within the research group Air Transport & Operations (ATO) at Delft University of Technology [16, 17, 18, 19].

First, a description will be given of the involved stakeholders, also called agents in the context of a socio-technical system representation. After having specified the goals and responsibilities of all agent types involved, their interactions will be described.

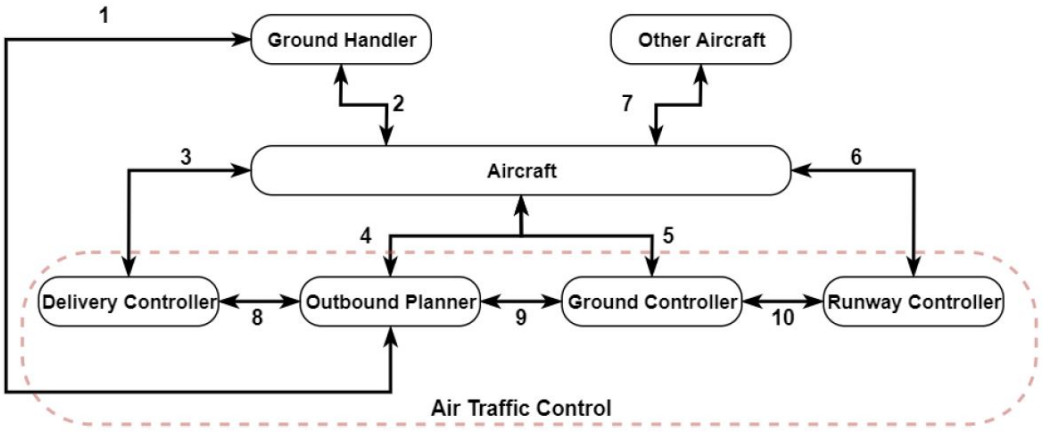


Figure 2.4: Graphical representation of a CDM airport as socio-technical system. Based on work by [16, 17, 18, 19].

Agent Types

When representing the ground operations of an airport as a socio-technical system, several different types of agents can be identified. Below, a description of the involved stakeholders in the context of ground surface movement is given.

- **Aircraft**: the overall goal in the field of ground surface movements is to make sure that aircraft are guided efficiently from their starting point to their destination in a safe manner. As a result, aircraft are at the center of the socio-technical system representation. Aircraft have properties like weight, wingspan, speed and acceleration limits etc. and are controlled by their flight crew. The flight crew bears the responsibility of complying with ATC instructions and the Rules of Air defined by ICAO [20].
  In the context of ground surface movement, a distinction can be made between arriving and departing aircraft. For arriving aircraft, their goal is to travel from the runway to the assigned gate in an optimal manner (in terms of fuel consumption and/or travel time) and comply with the airline schedule, for departing aircraft vice versa.

- **Ground Handler**: the process of ground handling is concerned with optimizing the turn-around process, including e.g. cleaning, refueling, passenger boarding and more. Therefore, the agent 'Ground Handler' does not refer to a single person/entity, but rather represents all persons involved in the turn-around process (e.g. cleaners, tow truck operators, baggage handlers and fueling services).
  The ground handler is responsible for minimizing the Minimum Turn-round Time (MTTT). Based on interaction with the aircraft/flight crew and its own insights, the ground handler provides for the final update of TOBT (Milestone 9, see subsection 2.1.3).

- **Delivery Controller**: the delivery controller provides for clearance on the flight plan submitted by departing aircraft. After checking the flight plan, the delivery controller communicates either through radio or data link an assigned runway and Standard Instrument Departure (SID), which is a route that the aircraft takes after take-off. After confirmation of the provided information by the flight crew, the aircraft will be considered in the pre-departure sequence.
  Any information regarding assignment of Calculated Take Off Time (CTOT) by EUROCONTROL's network operations is communicated by the delivery controller to the respective aircraft. In addition, it is checked by the delivery controller whether the clearance request by an aircraft which is assigned a CTOT is actually within the assigned CTOT window.

- **Outbound Planner**: once the aircraft is ready for movement (Milestone 12), the flight crew requests the outbound planner for start-up approval (Milestone 13). It is the responsibility of the outbound planner to regulate traffic flow by granting start-up clearances based on the congestion levels of the airport (expected queues at runways or conflicts resulting from clearances provided to adjacent aircraft). Using tools like Collaborative Pre-Departure Sequence Planning (CPDSP) (subsection 2.1.3), a TSAT sequence is determined and start-up approval is provided to the specific aircraft.
- **Ground Controller**: the goal of the ground controller is to ensure safe and efficient operations of all ground vehicles. This includes giving instructions related to push-back and taxiing to departing and arriving aircraft, but also providing guidance to all other vehicles related to the turn-around process (e.g. fueling services, luggage handling trucks). Avoidance of conflicts in the bay area and on the taxiway network is one the main responsibilities of the ground controller. In addition, the ground controller is responsible for adhering to the pre-departure sequence established by tools such as CPDSP.
  Note that all movements on runways (including crossing of active runways by aircraft or ground vehicles) are overseen by the runway controller.
- **Runway Controller**: all operations on or close to active runways are controlled by the runway controller. The goal of the runway controller is to ensure that these operations are executed both in a safe and efficient manner. The responsibilities include providing clearance for departing aircraft to take-off based on the pre-departure sequence, as well as providing clearance for arriving aircraft to land on their assigned runways. Maintaining separation between departing and arriving aircraft is also part of the runway controller's responsibilities and is based on Wake Turbulence Category (WTC). Furthermore, ground vehicles crossing active runways must request for clearance of the runway controller to do so.
- **Network Manager Operations Centre (NMOC)**: NMOC is part of EUROCONTROL and provides for optimization of traffic flows in the European air traffic network [21]. This is done by processing updates regarding TOBT and TTOT of all flights departing from CDM airports, and monitoring traffic load with available airspace capacity.

Interactions Between Agents

Based on the links as shown in Figure 2.4, a description of the interactions between the different stakeholders is given below.

1. **Ground Handler** → **Outbound Planner**: based on the progress in turn-around, the ground handler provides a final update on the TOBT t minutes before the EOBT (where t is a parameter agreed upon for each airport specifically) to the outbound planner.
2. **Ground Handler** ↔ **Aircraft**: the ground handler and aircraft keep each other updated on the progress in turn-around, both on processes outside the aircraft such as fueling (ground handler) and processes inside the aircraft such as boarding (aircraft). Furthermore, the aircraft notifies the ground handler of any issues related to TOBT or TSAT which they receive from ATC.
3. **Delivery Controller** ↔ **Aircraft**: the flight crew requests departure clearance from the delivery controller. After checking the flight plan, the delivery controller grants this clearance and informs the flight crew of the assigned departure runway and SID. In its turn, the flight crew provides clearance and instructions confirmations to the delivery controller. Furthermore, the delivery controller instructs the flight crew to contact the outbound planner. All communications between the aircraft and delivery controller take place through Very High Frequency (VHF) radio, the Aircraft Communications and Reporting System (ACARS) or the Controller-Pilot Data Link Communications (CPDLC) system.
4. **Outbound Planner** ↔ **Aircraft**: the flight crew requests start-up clearance from the outbound planner through radio communication once the aircraft is ready for departure. In case the start-up request falls within the TSAT window, the outbound planner grants start-up approval or forwards the aircraft to the ground controller in case push-back is required. Furthermore, the outbound planner provides the flight crew with airport and weather related information and requests the flight crew to contact the ground controller.

5. **Ground Controller ↔ Aircraft**: the flight crew contacts the ground controller to request for push-back clearance through radio communication. The ground controller grants this clearance if push-back of the aircraft under consideration will not interfere with push-back procedures of neighboring aircraft. Once push-back is completed, the flight crew requests the ground controller for taxi instructions. During taxiing, the ground controller can provide the flight crew with additional instructions, e.g. to solve conflicts. Note that the flight crew is expected to confirm or read back all instructions given by the ground controller.
   Arriving aircraft that have exited the runway also fall within the responsibilities of the ground controller. In some cases, the ground controller may request arriving aircraft to wait at a remote holding position if the assigned gate is still occupied.

6. **Runway Controller ↔ Aircraft**: the runway controller is responsible for providing clearances to aircraft related to lining up on the runway, taking off, landing and crossing active runways. This is both done through radio communication and by means of lights on taxiways (such as stop-bars on taxiway crossings). The flight crew has to confirm all the instructions issued by the runway controller.

7. **Other Aircraft ↔ Aircraft**: aircraft should maintain separation based on visual inspection and instructions received from ATC. More elaboration on interaction between aircraft on the taxiway network and the type of conflicts is provided in subsection 2.2.2 and section 4.2 respectively.

8. **Delivery Controller ↔ Outbound Planner**: after clearance of the flight plan by the delivery controller, the responsibility for the flight is handed over to the outbound planner by means of transferring Electronic Flight Strips (EFS). If changes occur with respect to assigned runway or departure route which require new clearance, the outbound planner might hand over the responsibility over the flight back to the delivery controller.

9. **Outbound Planner ↔ Ground Controller**: once start-up clearance is given by the outbound planner, responsibility for the flight is handed over to the ground controller. Since EFS are used for handing over the responsibility, the outbound planner can monitor the workload of the ground controller and decide whether a new aircraft can be forwarded. If the aircraft is suddenly not able to start with push-back, it might occur that the ground controller hands over the responsibility back to the outbound planner.

10. **Ground Controller ↔ Runway Controller**: when an aircraft approaches the runway holding point or needs to cross an active runway, the responsibility of the flight is handed over from the ground controller to the runway controller by means of transferring EFS. Similarly, when an arriving aircraft is leaving the runway or when an aircraft has crossed an active runway, responsibility of the flight is handed over from the runway controller to the ground controller.

2.2. Ground Surface Operations at Amsterdam Airport Schiphol (AAS)

Now that a general overview is provided on ground surface operations, more details will be described on the ground surface operations at AAS, specifically on its infrastructure used for these operations. The specific runways and runway configurations are elaborated on, as well as the taxiway and service road network. Finally, the apron area is discussed.

2.2.1. Runways

At AAS, six runways are present that are used to facilitate all inbound and outbound flights (Figure 2.5). Note that the Oostbaan is not used for commercial flights, but for general aviation only [9]. The runway usage at AAS is subjected to a number of restrictions that can be found in the Annual Runway Usage Forecast [22]. For example, the Polderbaan can only be used for departures in Northern direction (36L) and arrivals from Southern direction (18R), due to the presence of Hoofddorp at the south of the Polderbaan. Similarly, the Aalsmeerbaan can only be used for departures in Southern direction (18L) and arrivals from Northern direction (36R).
In the same manual [22], preferred sequences of runways are defined, both for daytime and nighttime. The preferred sequences are defined with respect to restrictions on noise and emissions and consist

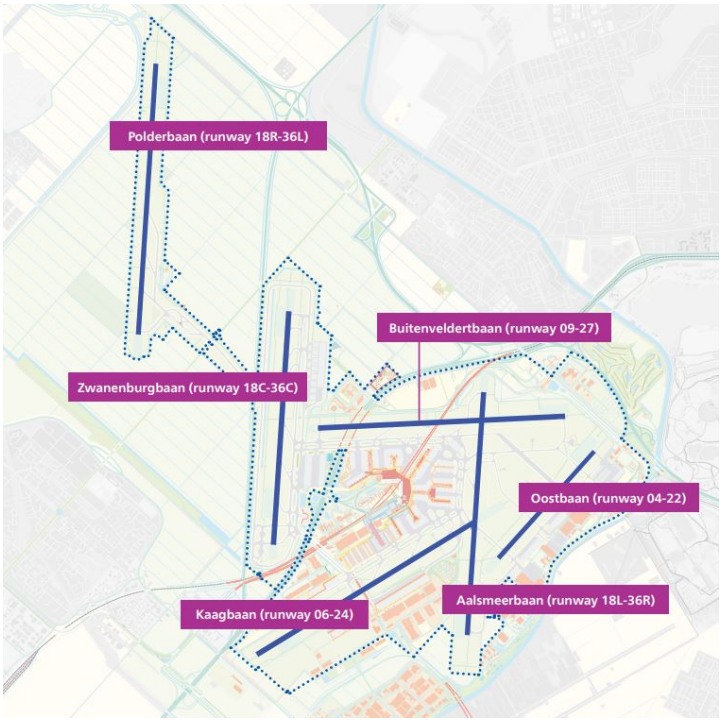


Figure 2.5: Configuration of runways at AAS, including their names and locations. Taken from [6].

of combinations of arrival and departure runways. A summary of all preferred sequences is given in Table 2.2. Note that the actual runway usage will not always adhere to one of the preferred sequences, due to for example weather conditions.

Table 2.2: Preferred sequences of runway usage during the day and depending on visibility conditions [22].

Time	Visibility Conditions	Preferred Sequence	Arrival		Departure	
Day (06:00h - 23:00h)	Good, during daylight period	1	06	(36R)	36L	(36C)
		2	18R	(18C)	24	(18L)
		3	06	(36R)	09	(36L)
		4	27	(18R)	24	(18L)
	Good	5a	36R	(36C)	36L	(36C)
		5b	18R	(18C)	18L	(18C)
	Good or marginal	6a	36R	(36C)	36L	(09)
		6b	18R	(18C)	18L	(24)
Night (23:00h - 06:00h)	-	1	06	-	36L	-
		2	18R	-	24	-
		3	36C	-	36L	-
		4	18R	-	18C	-

2.2.2. Taxiway Network

The taxiway network at AAS is used to connect the gates with runways and is controlled by ATC. At AAS, a central taxiway surrounds the piers, visualized in Figure 2.6. It consists of two unidirectional taxiways, Alpha (clockwise direction) and Bravo (counterclockwise). Although more efficiency might be achieved when allowing for bidirectional traffic on both taxiways, adhering to the prescribed directions is enforced by ATC due to limited cognitive capacity of humans [9]. Furthermore, a final segment called Quebec is a single taxiway used to connect the rear ends of both Alpha and Bravo. It can be used in

both directions, but only allows for unidirectional traffic [9]. Currently, taxiway Quebec is being extended to a bidirectional taxiway [23].

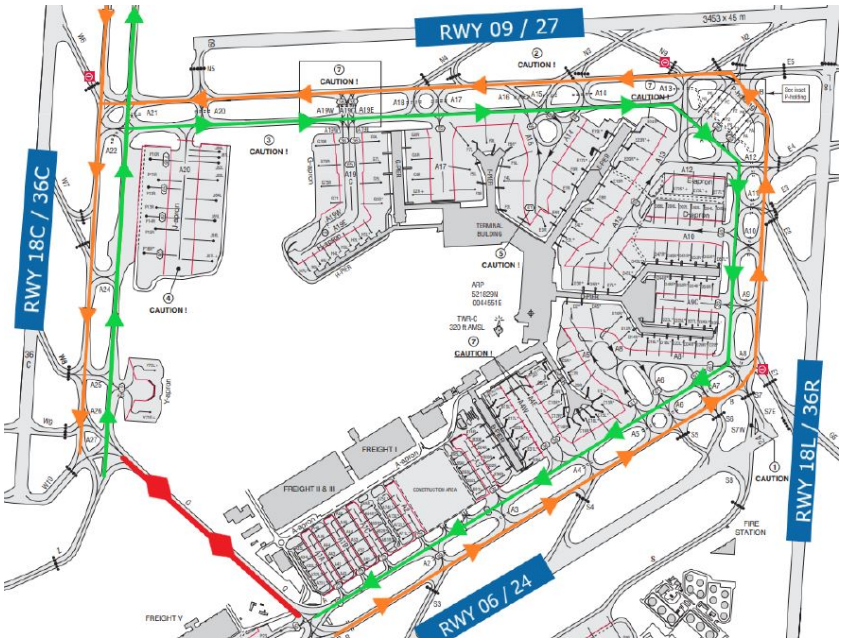


Figure 2.6: Center taxiway network at AAS, where taxiway Alpha (clockwise) is shown in green, taxiway Bravo (counterclockwise) in orange and Quebec in red. Taken from [9].

As stated previously, the taxiway network is under control of ATC by providing aircraft specific instructions on which routes to take. Next to the instructions of ATC, pilots need to adhere to a set of predefined rules by ICAO (taken directly from [20]):

- 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.

2.2.3. Service Road Network

Whereas aircraft make use of the taxiway network to move over the airport, other Ground Service Equipment (GSE) vehicles might also have to move over the airport. Therefore, a specific service road network is put in place as well. According to the Schipholregels [24], vehicles on the air side of the airport have to adhere to the following set of rules:

- 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 that an aircraft being towed by a TaxiBot to the runway, is not classified as a ”towed aircraft”, since it is formally taxiing [6]. In order to avoid confusion on this, it was noted by Schiphol that a procedure should be developed that clearly distinguishes TaxiBotting aircraft from towed aircraft [7].

2.2.4. Apron Area

The final infrastructural component of importance concern the aprons and gates. At AAS, 93 gates are present, divided over 7 piers, as shown in Figure 2.7.

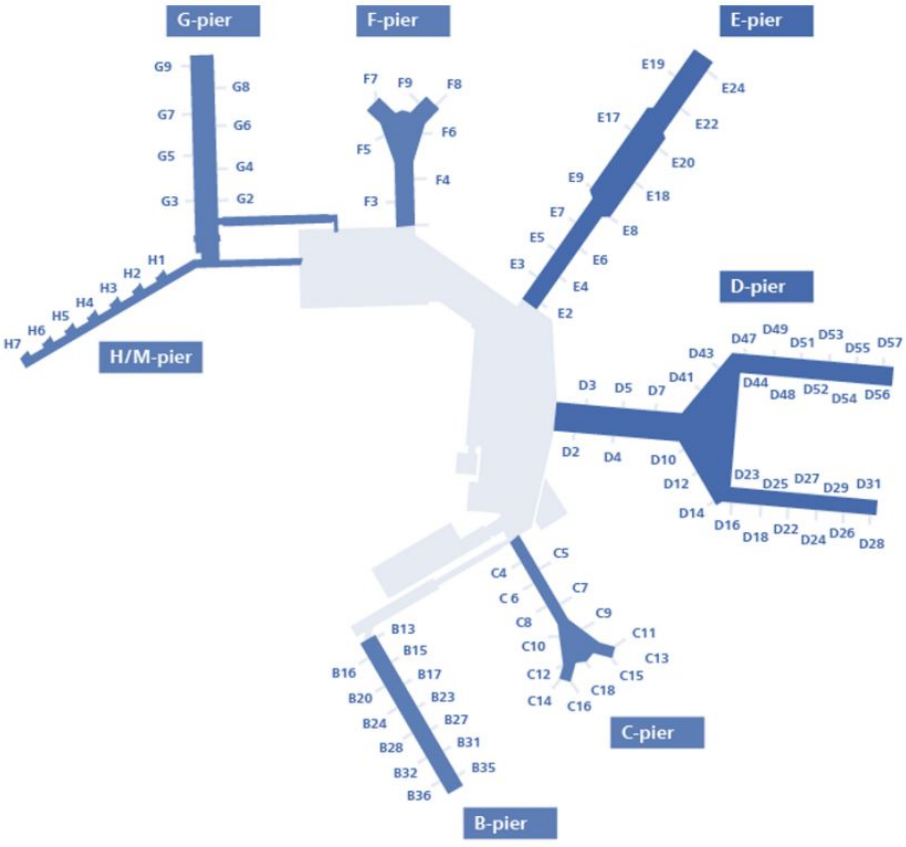


Figure 2.7: Overview of the existing piers and gates at AAS. Taken from [9].

As already mentioned in subsection 2.1.1, arriving flights arrive at the gate, after which the turnaround process starts. At AAS, a maximum turnaround time of 170 minutes is handled for most narrow bodies (CAT 4 or lower), whereas most wide bodies are allowed a maximum turnaround time of 210 minutes (CAT 5 of higher) [25].

After having turnaround completed, the pushback procedure is initialized. For most gates at AAS,

aircraft are parked nose-in, meaning that the aircraft needs to reverse to leave the gate. Since this is not possible, a pushback truck is needed to perform the pushback procedure. One exception is the B-pier, where aircraft can leave the gate autonomously [26]. Both the pushback procedure and pushback path are highly standardized procedures, which are described in detail in [27]. Once the aircraft crosses the so-called red clearance line (represents the border between ATC controlled area and the aircraft stand, refer to Figure 2.8), engine start-up is allowed. Note that clearance is always needed to cross the red line [27]. Engine start-up can either happen during or after the pushback procedure, depending on the required duration of both engine start-up (dependent on engine type) and the pushback procedure itself (dependent on pushback path, where a maximum speed of 3 kts is maintained [6]).

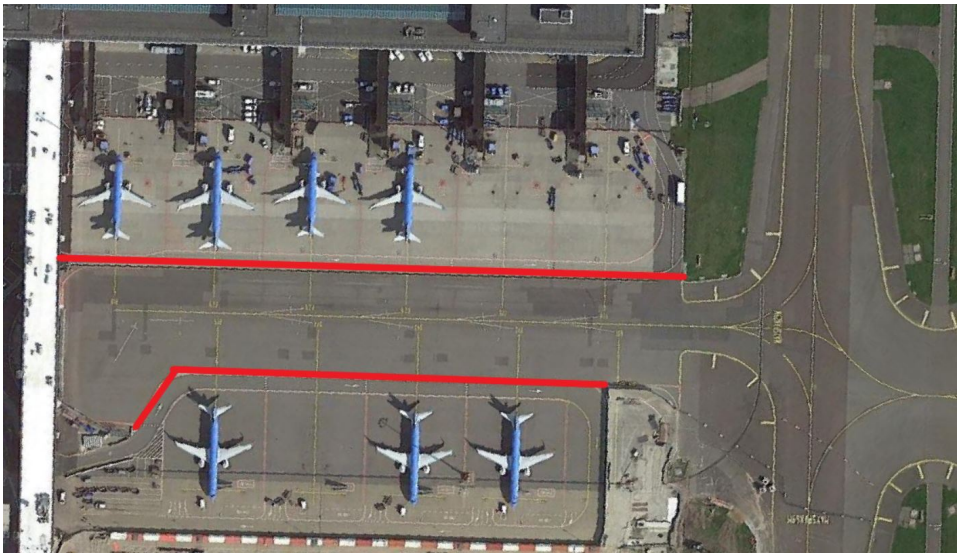


Figure 2.8: Overview of west-side of B-pier, indicating the red clearance line. Taken from [9].

3

Developments in Airport Surface Movement Operations

A general overview of airport surface movement operations has been provided in the previous chapter, with a special focus AAS. Although current ground surface operations are both safe and efficient, challenges are always present, as in any field of study. The major challenges that are related to airport surface movement operations are summarized in section 3.1. Based on these challenges, a number of developments can be identified in the sector, which will be outlined in section 3.2. Special focus will be put on the project of SESAR called Advanced Engine Off Navigation (AEON), focusing on aircraft engine-off taxiing techniques. A promising alternative to conventional engine-on taxiing is the TaxiBot, which is an external towing vehicle. The characteristics, expected effects of implementation of the TaxiBot at AAS and findings of previous research on TaxiBot implementation will be elaborated on. Finally, the research gap that this thesis will try to fill is described in section 3.3.

3.1. Challenges within Ground Surface Operations

One of the biggest challenges within the airspace industry in general concerns the expected growth rate of demand for air transportation. In 2019, figures showed an expected annual increase of approximately 4% in Revenue Passenger-Kilometer (RPK) [28]. Despite the fact that the COVID-19 pandemic has caused for a serious fallback in the entire air travel industry, it is expected that passenger levels return to 2019-levels in 2024 and keep increasing from there on [29].

The increase in demand for air transportation cannot be accommodated with the current infrastructure, which already is reaching its capacity limits [1]. Especially hub airports are expected to form bottlenecks for the overall air traffic management system within Europe [10] due to a combination of two reasons: airport congestion and limits on ATC-workload [1, 30, 31]. Regarding airport congestion, the normal approach for increasing capacity would be to increase the physical infrastructure by constructing new terminals, runways and surface area for taxiing [31]. However, besides the fact that the realization of large-scale infrastructure projects at airports is often limited by available space [1], it also adds to the complexity of operations which increases human workload [30, 31]. As a result, the net benefits of increasing physical infrastructure to reduce airport congestion are limited, which calls for optimization of current airport ground operations using the existing space [10]. In addition, workload of ATC is per definition limited since exceeding the mental capacity of an air traffic controller (by increasing the amount or complexity of traffic) increases the risk of human error with the result of potential unsafe operations [30].

Next to the expected growth rate of demand in air transportation, another major challenge in the air transport industry concerns sustainability and the urge for aviation to become greener. Seen from the perspective airport surface movement operations, a study into the carbon footprint of Heathrow Airport has shown that approximately 30% of the carbon dioxide emissions can be attributed to aircraft ground movements, including taxiing and the use of the Auxiliary Power Unit (APU) [3]. Looking from the aircraft point of view, it is estimated that short haul fleets operating a single-aisle aircraft use up

to 6% of the total fuel consumption during taxiing only [32]. Besides, in combination with the problem of airport congestion, aircraft burn an unnecessary additional amount of fuel because of waiting times in ground operations [33]. The latter suggests a large potential gain when increasing the efficiency of ground surface operations, both from an operational as well as environmental point of view.

When it comes to tackling the challenges of increasing demand and reducing emissions in ground surface operations, international initiatives such as the Single European Sky ATM Research (SESAR) in Europe and Next Generation Air Transportation System (NextGen) in the United States aim to contribute to this. Among their main goals is to increase overall efficiency of the air traffic management system while reducing the environmental impact of flights on the environment [34, 35]. A promising project related to these goals is called Advanced Engine Off Navigation (AEON) and is concerned with the implementation of more environmental-friendly concepts of taxiing, such as TaxiBot, E-Taxi and single engine taxiing [5]. A more elaborate description of this project is given in section 3.2.

Another research direction that SESAR focuses on is to investigate to what extent automation and decision support systems can provide for reduction of ATC-workload [36]. The potential of this research direction is endorsed by various entities, including Royal Schiphol Group, which has announced a program called Autonomous Airside Operations. This program aims for making all vehicles and associated processes fully autonomous by 2050 [37]. As will become apparent in section 3.2, introducing aircraft engine-off taxi operations allows for moving towards more autonomous processes related to management of airport surface movement operations. Despite the fact that moving towards autonomy will have significant influence on current ATM systems and procedures, the implications of decentralization on ATM and ATC is considered to be out of the scope for this thesis and will therefore not be elaborated on in more detail in this literature study.

3.2. Developments in Aircraft Engine-Off Taxiing at Airports

In the remainder of this thesis, focus will be put on taxiing specifically as part of ground surface operations at airports. As identified by [10], the airport ground movement problem is a research area of past and current interest, since it is highly related to airport capacity and therefore, to the overall capacity of the entire air transportation system. In addition, previous research has shown large potential in reducing the environmental impact of the taxiing phase by considering so-called engine-off taxiing techniques, which reduce engine-on time of aircraft [30, 38, 39, 40].

As mentioned before, the Advanced Engine Off Navigation (AEON) project of SESAR in particular focuses on the implementation of these engine-off taxiing techniques which reduce environmental impact. The AEON project considers three different methods to reduce engine-on time of aircraft: Single-Engine Taxiing (SET) solutions, non-autonomous taxiing solutions and autonomous taxiing solutions. Below, a short description of these technologies will be provided, based on [5]. In this context, autonomous taxiing techniques refer to solutions that are installed on-board and allow for the aircraft to taxi independently. Non-autonomous taxiing techniques are solutions that entail an external towing device [41].

- **Single-Engine Taxiing (SET)**: this concept involves the use of half the number of installed aircraft engines in order to generate propulsion when taxiing. Although the expected reduction in fuel consumption is significant (approximately 25%), the solution is not suitable for all taxiways and weather conditions, since using SET-techniques influences manoeuvrability and balance of the aircraft. In addition, SET-techniques are more suitable for taxi-in procedures in comparison with taxi-out procedures, due to a higher workload for the pilots related to engine cooldown when warming up and lower predictability in taxi duration when departing (due to possible runway queuing).

- **Non-Autonomous Taxiing**: as stated before, non-autonomous taxiing techniques involve the use of an external towing vehicle. A promising concept is the TaxiBot, which is a pilot-controlled pushback truck that is capable of performing taxi operations by towing the aircraft at regular taxi speeds [42]. Fuel savings during taxiing are expected to be around 50-80% according to the manufacturer. However, three major challenges are still to be tackled: from a technical point of view (how to reduce decoupling time), from an infrastructural point of view (where to create decoupling locations) and from an ATC-point of view (how to incorporate the extra traffic in the ground controller's workload).

- **Autonomous Taxiing**: when considering autonomous taxiing techniques, this involves systems that are internally embedded in the nose or main landing gear and provide for propulsion of the aircraft without the use of the main engines. Examples of such systems are the E-Taxi system or WheelTug. A reduction between 50-85% of fuel consumption is expected, although no live trials have been performed on these internal systems.

From an environmental point of view, either autonomous or non-autonomous taxiing techniques have the highest potential in reducing emissions. Therefore, focus will be put on a comparison between autonomous (internal, or on-board systems) and non-autonomous (external systems) taxiing techniques. Several studies have been performed in comparing these two techniques [43, 44]. Both studies confirm that the greatest reduction in greenhouse gasses is achieved when using on-board systems. In addition, using on-board systems will simplify push-back procedures, since no external towing device is needed.

However, on-board systems add significant weight to the aircraft, possibly reducing environmental benefits for long-haul flights [41]. Next to that, maintenance costs are expected to be increased for internal systems [45] and due to required changes in aircraft architecture, considerable efforts are expected in terms of certification and legislation [43, 46]. Especially the latter is considered to be decisive: up until to this date, the TaxiBot concept is the only alternative taxiing technique that is fully certified and considered in practice for engine-off taxiing [18, 42]. Previous related work [18, 47] has also identified the TaxiBot as the alternative taxiing technique with the highest potential. Therefore, in the remainder of this thesis, the concept of non-autonomous taxiing using TaxiBots will be considered.

3.2.1. Technical Characteristics of the TaxiBot Concept

The TaxiBot is an external towing truck for aircraft ground movements developed by Israeli Aerospace Industries (IAI). The drive line of the tractor consists of two diesel engines that power two separate electric generators. The generators supply 8 electric motors attached to the 4 wheels, capable of providing for a maximum power of 500 kW with maximum achievable torque of 45 kNm [43]. Two types of TaxiBots have been developed: a Narrow-Body (NB) and Wide-Body (WB) version. Technical specifications of both types are provided in Table 3.1. In Figure 3.1, a front and side view are provided on the NB TaxiBot.

As mentioned before, the NB TaxiBot is only certified for the B737 and A320 family. However, when considering AAS, these aircraft types account for approximately 54% of the total number of flights [7].

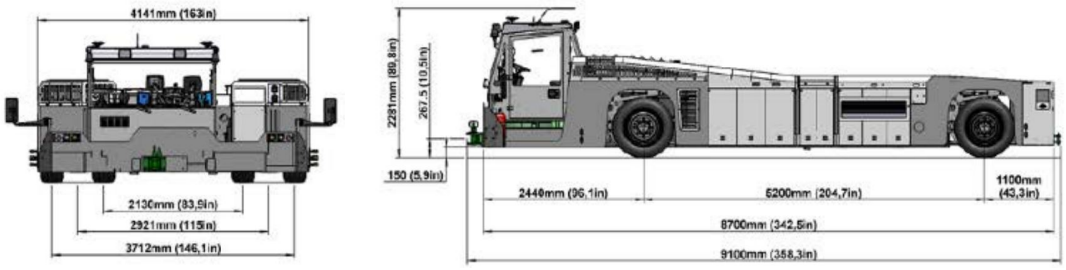
Table 3.1: Technical specifications of NB and WB TaxiBots.

Technical Specifications		**NB TaxiBot**	**WB TaxiBot**
Designed for:		Single-aisle aircraft [48] - A318 to A321 - B737 to B757	Twin-aisle aircraft [49] - A330 to A380 - B767 to B747
Certified for:		B737 and A320	n.a.
Maximum speed [km/h]	Pilot mode	43 [42]	
	Solo mode	30 [6]	
Normal turning speed [km/h]		18.5 [6]	
Purchasing costs [USD]		1.5 million	3 million

In Figure 3.2, the coupling of the TaxiBot to the nose landing gear of an aircraft is shown. The TaxiBot slightly lifts the front wheel and locks it in position on a rotating turret platform [43]. Note that after push-back out of the bay area, control is switched from the TaxiBot driver to the pilot [6]. This is an important characteristic of the TaxiBot due to safety and accountability regulatories [43]. Controlling the TaxiBot by the pilot is done through sensors installed at the platform that detect the steering angle of the nose landing gear. In a similar fashion, braking is also accounted for [43].

3.2.2. TaxiBot Operations Compared with Conventional Taxi Operations

The introduction of TaxiBots on AAS will have significant impact on the current ground surface operations. First, an overview of the operational flow will be outlined for ground surface operations when



(a) Front view of NB TaxiBot. (b) Side view of NB TaxiBot.

Figure 3.1: Technical drawings of NB TaxiBot. Taken from [6].

Figure 3.2: Visualization of the working principle of coupling an aircraft nose landing gear to a TaxiBot. Taken from [50].

deploying TaxiBots, including the main differences with respect to conventional ground surface operations.

When deploying TaxiBots at AAS, the ground surface operations for outbound towing will consist of the following procedures for TaxiBots: pushback, TaxiBotting, decoupling and a TaxiBot return movement [6]. Note that in this section, only outbound towing will be considered. For more details on the reasoning behind this choice, refer to section 4.3.

1. **Pushback**: after loading the aircraft onto the TaxiBot, pushback from the gate stand will occur, similar to conventional operations. In normal taxi operations, the pushback truck will be decoupled from the aircraft after push back and the engines of the aircraft will be started, a process normally taking about 100 seconds [6]. When using a TaxiBot, both decoupling and starting of the aircraft engines will be postponed until reaching a decoupling location. After having pushback completed, control of the TaxiBot driver will be handed over to the pilot (taking about 30 seconds [6]) and the aircraft will start moving towards the runway and a corresponding decoupling location.

2. **TaxiBotting**: TaxiBotting is defined as the movement of an aircraft on the ground by means of a TaxiBot providing power, while being controlled by the pilot. The aircraft will move from the apron area to a decoupling location corresponding with the assigned runway during TaxiBotting. During TaxiBot trials performed at AAS in 2020, engine start was not performed in parallel with TaxiBotting or uncoupling. However, since Schiphol announced the intention to develop a procedure for starting of the engines during TaxiBotting, it is assumed in this study that engine warm-up is finished at the moment that the TaxiBot driver gives the all-clear signal.
   Normal taxiing aircraft have a top speed of 30 knots, whereas TaxiBots have a top speed of 22 knots. As a result, the process of taxiing itself takes slightly longer, especially on long straight tracks (e.g., taxi segment towards runway 36l) [6].

3. **Decoupling**: when arrived at the decoupling location, the decoupling process consists of the following steps [6]:

   - Braking of aircraft by pilot at decoupling location
   - Communication between pilot and TaxiBot driver to determine unloading moment of TaxiBot
   - Unloading of TaxiBot

- Stepping out of TaxiBot by driver to unplug communication cable
- Driving of TaxiBot to all-clear point and signalling all-clear signal to pilot
- Requesting of taxi clearance by pilot to ATC and starting to taxi the remainder of the route to runway once clearance received

The decoupling process was measured by Schiphol to take 150 seconds during trials [6]. According to Smart Airport Systems (SAS) (product owner of the TaxiBot), decoupling durations vary between 90 and 120 seconds [8].

4. **TaxiBot Return Movement**: after decoupling, the TaxiBot will have to move from the decoupling location to its next mission, which is either a next outbound flight or its base location. Compared with conventional operations, this is an extra step regarding taxiing that is currently not present.

Assumptions on TaxiBot Operations Compared with Conventional Taxi Operations

Based on the outline of TaxiBot operations, several assumptions are made on specific operational characteristics of implementing TaxiBots at AAS. These assumptions only relate to operational characteristics of an individual TaxiBot, whereas assumptions on the global concept of operations will be defined in chapter 4.

1. **Engine Start**: it is assumed that a procedure for starting the engines during TaxiBotting/decoupling is developed, resulting in fully warmed-up engines at the moment that the decoupling procedure is finished. As a result, the aircraft can start moving immediately once the all-clear signal is provided by the TaxiBot driver. Currently, performing engine start-up during taxiing is not allowed due to safety reasons, since the pilot will have to monitor the engine warm-up process while taxiing [7]. However, as Schiphol has already declared that it is necessary to develop a procedure that will allow engine start-up during TaxiBotting [6], this assumption is considered to be valid.
   It is assumed that both engine warm-up and cool down time is 3 minutes. This is based on an average of the Boeing 737-800 operator [7].

2. **Coupling Time**: coupling of the TaxiBot to the aircraft at the gate has to be considered. However, no sources could be found that provide operational data on the duration of coupling at the gate. If the duration of coupling remains unclear after consultation of operational experts, a similar time for coupling as for decoupling will be incorporated in the model.

3. **Decoupling Time**: a decoupling time of 120 seconds is assumed in this thesis. The result is based on different trials performed by SAS, which pointed out that 120 seconds was the worst case scenario for the entire decoupling process [8]. Although trials performed by Schiphol resulted in an average decoupling time of 150 seconds, it is assumed that with increased training and experience this time can be brought down by at least 30 seconds, since trials where only performed on seven outbound flights [6].

3.2.3. Expected Effects of Implementing TaxiBots at AAS

A number of previous studies has been conducted in order to quantify the effect of the expected benefits and challenges of the deployment of TaxiBots at AAS. As mentioned earlier, Schiphol itself conducted a number of trials to assess the operational implications of using TaxiBots for outbound towing [6]. The results of these trials provide for practical data on the use of TaxiBots, but do not consider the effects of implementing an external towing system on system performance level. Therefore, a simulation study has been conducted by the aviation consultant To70 [51]. In this research, it has been investigated what the potential savings in terms of delay and fuel consumption are when using fully outbound towing with TaxiBots for the two most common Runway Mode of Operation (RMO) at the busiest day of 2019 with 1520 movements. Based on the results of the simulation study, a number of expected benefits and challenges can be formulated.

Expected Benefits of Introduction of TaxiBots at AAS

The introduction of TaxiBots in the ground surface movement system is expected to create two major benefits: reduced environmental impact of aircraft ground movements and increased capacity [51].

- **Reduced Environmental Impact**: instead of towing the aircraft just outside the apron area, aircraft are towed as close to the runway as possible while using TaxiBots. During the TaxiBotting, the main aircraft engines are not needed for thrust, but power is provided by the TaxiBot itself. Therefore, the time that the main engines are running is significantly reduces when using TaxiBots compared with conventional operations. However, due to the fact that the main engines still need to be warmed up (or cooled down for inbound flights) and the APU needs to be running during TaxiBotting, fuel savings will not reach 100% (see Figure 3.3, [7]).
  Based on trials for 7 outbound flights at AAS, the observed fuel savings were observed to be around 50% compared to conventional taxiing. However, usage of runway 24 was over-represented in these trials, which is characterized by shorter taxi times. Using the actual runway mix is expected to increase gains [6]. Especially runway 36L seems to be very promising for sustainable taxiing from a fuel reduction point of view, due to the combination of very long taxi times and its preferred status for departing flights [6].
- **Increased Capacity**: due to the fact that aircraft can start moving faster after pushback, congestion in the bay area is reduced and capacity in the apron area can be improved. Upon discussion with operational experts, another advantage of TaxiBots related to capacity became apparent. Since gate capacity is one of the limiting factors at Schiphol, performing outbound holding (outbound flights leaving their stand at ARDT instead of waiting at the gate until their departure slot is available) could potentially increase capacity at the gate for inbound flights. In conventional operations, this would result in additional fuel burn by aircraft operators, since engines will be running idle when performing outbound holding, for example while waiting on a holding platform. However, this problem can be tackled using TaxiBots, since it will not cost any additional fuel and increase gate capacity for inbound flights.

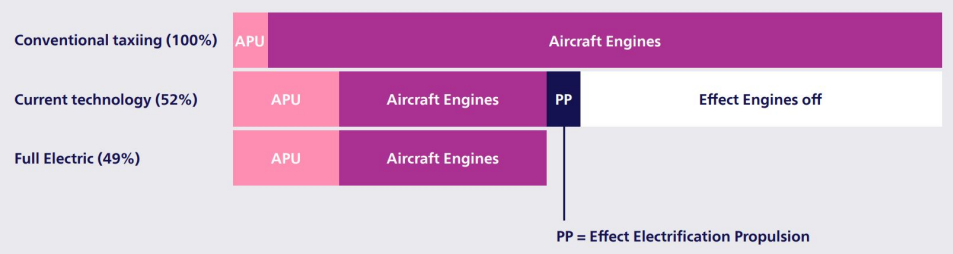


Figure 3.3: Indicative visualization of potential fuel savings when using TaxiBots for outbound towing, both for the current technology (diesel driven) and a fully electric version. Percentages shown are calculated based on an average of 14 minutes taxi time. Taken from [7].

Expected Challenges of Introduction of TaxiBots at AAS

Despite the benefits, a number of challenges is also expected when TaxiBots are incorporated in the ground movement system at AAS. On the level of individual TaxiBot performance, Schiphol has concluded based on its trials that the ”concept is infeasible in its current form, but might be feasible with adjustments” [7, p. 39]. These adjustments concern infrastructural, procedural and technical changes.

On the overall system performance level, the following expected challenges were identified [7]:

- **Increased Process Times**: due to the lower maximum speed while taxiing using TaxiBots and increased decoupling duration with respect to normal uncoupling after pushback, the process time is expected to increase. Especially for hub and spoke carriers like KLM, it is of utmost importance that any additional delays are prevented as much as possible. The reason for this is twofold: on one hand, connecting flights need to arrive on schedule in order for passengers to catch transfer flights, since the business model of KLM depends on these connecting passengers through Schiphol. On the other hand, since most aircraft depart and arrive multiple times a day, delay of already a couple of minutes could lead to the deployment of an additional aircraft, large delays or cancelled flights [6].
  Another side-effect of increased process times is a potential decrease in runway capacity. If uncoupling durations take longer than minimal runway separation requirements, the possible number of departing flights per hour diminishes [6].

- **Increased Traffic Density**: since an additional movement is introduced in the ground movement system (the return movement of a TaxiBot), traffic densities are expected to increase. Not only may this result in additional delays on the taxiway network, it is also expected that workload of ATC increases.

Especially from an ATC perspective, it was concluded that implementation of TaxiBots in its current form at AAS would not be possible while maintaining capacity, on-time performance, safety and workload [7]. However, the simulation study performed by To70 showed significant differences in effect of increasing process times on system throughput, largely depending on factors such as the runway configuration used and the configuration of decoupling locations. Note that the increase in traffic density has not been investigated in the simulation study by To70, since TaxiBot return movement has not been modelled. In addition, an infinite number of TaxiBots available at the gate was assumed [51].

To conclude, based on the proposed adjustments and initial outcomes of the trials performed by Schiphol [6] and the simulation studies performed by To70 [51], the need arises to further investigate the conditions under which the implementation of TaxiBots has minimal effects on the airport system performance. In the next section, an overview will be given on previous research performed based on the outcomes of the initial feasibility study performed by Schiphol and To70.

3.2.4. Previous Academic Research on Implementing TaxiBots at AAS

Based on the findings of the initial feasibility study performed by Schiphol and partners and the conclusions of mainly ATC related to the decreased capacity of the airport system, steps in the direction of autonomy to tackle the aforementioned issues have been suggested. One of the first studies to look into the effects of implementing self-driving technology to enable autonomous taxi systems at airports, is the work done by Morris et al. [30]. It was concluded that implementing towing tugs might increase operational complexity. However, the authors suggest that these effects could be partially or completely mitigated by the introduction of autonomous decision making in the vehicles. Although the outcome of the research performed by Chua et al. [52] show that an increase in the workload on ATC was observed upon introduction of autonomous tugs, other studies show that distributed coordination and planning for autonomous vehicles within airport ground surface movements allow for safe and efficient operations [18].

The concept of distributed coordination and planning has been further investigated by Soomers [8] and Kamphof [9]. Both studies focus on the implementation of a hierarchical control structure that guides autonomous towing tugs over the surface of AAS to perform outbound towing under various operational scenarios. In their work, a realistic representation of the runways, the taxiway network, the service roads, holding platforms and gates is included in a graph layout of Schiphol. Especially the addition of bay areas and actual pushback paths makes their model different from previously existing representations of AAS. Even for the high level of complexity of the network, both studies validate the aforementioned findings of Benda [18] and show that a hierarchical distributed control architecture is capable of finding collision-free solutions for the routing of both scheduled flights and TaxiBots, adhering to kinematic constraints of all vehicles [8, 9].

In his work, Soomers investigated how the use of decoupling locations and durations influences ground delays [8]. It was concluded that ground delays are minimized when making use of both holding platforms and decoupling locations on taxiways. Furthermore, each decoupling configuration has an individual maximum capacity, determined by each location and decoupling duration. Once capacity of the decoupling configuration is exceeded, queues emerge and delays are seen to be stacked up. Interestingly, inbound flights were not significantly affected by outbound towing. Important limitations in his study were the exclusion of other airport processes that are linked to uncoupling operations, such as the use of CTOT slots for pre-departure sequencing, runway capacities or pushback operations. Next to that, the problem was modelled in an offline setting using time windows of one hour, not allowing for testing the control architecture on real-time performance nor for evaluating the influence of stacking delays over time. Finally, although TaxiBot return movements were modelled, it was assumed that an infinite number of TaxiBots was available for outbound towing at the gate, excluding the allocation of TaxiBots to departing aircraft from the model.

In order to study the effect of pushback operations and especially engine warm-up, Kamphof included pushback paths and speeds into the model to compare the duration of conventional taxi operations with duration of TaxiBot operations [9]. Similarly as for the study performed by Soomers [8], it was

concluded that departing flights suffer from additional ground delays due to decoupling durations and lower taxiing speed with a TaxiBot compared to normal taxiing. However, performing engine warm-up during TaxiBotting decreases both bay area and total outbound taxi time, especially for larger engine warm-up times. Similarly as in the model of Soomers, task allocation of TaxiBots was not incorporated, as well as the TaxiBot return movement [9].

3.3. Concluding Remarks and Research Gap

Due to the increasing awareness related to the environmental impact of the aviation sector and the growing demand for air transportation, the need arises to develop sustainable solutions that allow for efficient airport operations. Within the field of airport operations, the ground surface movement problem has been identified as a promising research area, since it is highly related to overall airport capacity and has shown large potential for reducing environmental impact by using engine-off taxiing techniques. One of the most promising engine-off taxiing techniques is the TaxiBot developed by IAI, an external towing tug that is capable of towing aircraft from the gate to the runway and v.v. at reasonable taxiing speeds.

Several studies have done preliminary analyses on the implementation of TaxiBots at AAS. In trials performed with a single TaxiBot at Schiphol in 2020, useful operational data was obtained, including a set of necessary conditions and changes to be implemented in order to consider the concept of fully outbound towing feasible. However, trials were only performed with one TaxiBot and therefore, no conclusions could be drawn of the influence on high-level system performance when external towing tugs would be implemented at AAS. In a simulation study performed by To70, effects on high-level system performance were investigated by modelling the concept of fully outbound towing at Schiphol. Significant differences in the effect of increasing process times on system throughput were observed, largely depending on factors such as the configuration of decoupling locations.

Therefore, two studies have been performed based on the initial findings, investigating in greater detail how both decoupling locations/durations and pushback operations influence overall system performance respectively [8, 9]. Although both studies distinguish themselves by using a realistic representation of the layout of Schiphol including a high level of detail and complexity, it was assumed that infinite TaxiBots are available for outbound towing. In addition, allocation of TaxiBots to aircraft was not considered and the work did either not consider modelling the problem in an online setting [8] or taking the TaxiBot return movement into account [9]. Due to these assumptions, it is yet unclear how allocation of TaxiBots to aircraft should be performed in general and in an online setting specifically; and how it affects high-level system performance.

To conclude, an area that has not received enough academic attention is how to perform assignment of TaxiBots to aircraft for fully outbound towing in an online setting, while using a hierarchical distributed control architecture. From a practical point of view, this proposed research direction is of interest since it enables to investigate how the allocation of TaxiBots to aircraft affects system performance in terms of capacity and delay. From an academic point of view, this research adds value in the field of combining task assignment with path planning in a computationally efficient manner to provide for real-time operations.

Based on the defined research gap, a detailed concept of operations will be developed that define how implementation of the TaxiBot at AAS will take place and which modelling assumptions should be made.

4

Defining a Concept of Operations for Deploying TaxiBots at AAS

In chapter 2, an overview on current ground surface operations is provided, with special attention for the ground surface operations at AAS. Then, the developments related to ground surface operations were discussed in chapter 3. From this chapter, it was concluded that the use of TaxiBots could provide for reducing the environmental impact of ground surface operations. After having defined a research gap related to the implementation of TaxiBots at AAS, combined with a justification for future research on this subject, a concept of operations can be determined. In this chapter, relevant literature will be reviewed on different models and modelling assumptions related to the implementation of a tug-enabled engine-off taxi system at airports. As a guideline, a set of operational and tactical issues is used, as defined by Vis [53] that has to be addressed when considering the design of an Automated Guided Vehicle (AGV) system.

- Network Layout
- Traffic Management and Resolving Conflicts
- Location of Pick-Up and Delivery Points
- Fleet Composition and Characteristics
- Control of AGVs

In section 4.1 to section 4.5, each item will be elaborated on. This includes a general description of the item under consideration based on the work done by Vis [53], followed by a more detailed description based on concepts of operations discussed in papers that review the operational consequences of implementing a tug-enabled taxiing system at airports [18, 30, 31, 47, 58, 52, 59, 60, 61, 54, 55, 56, 57, 7, 6, 8, 9]. Using this work as a reference, a set of assumptions related to each item will be defined. All together, this defines the use case that will be studied in this thesis regarding the implementation of TaxiBots at AAS. A final summary is provided in section 4.6, including the final concept of operations, scope of the research, resulting system requirements and a modelling technique used in the remainder of the thesis to simulate the defined concept of operations.

4.1. Network Layout

The network of a system defines the paths along with AGVs can travel. Usually, the network is defined by arcs (paths or links) and nodes (intersections of paths, e.g. pickup and delivery locations). When defining the network, choices have to be made on the direction of travel along arcs. These can be unidirectional (travel is allowed only in one direction) or bidirectional (travel is allowed in both directions) [53]. Unidirectional arcs allow for easier control of the system, since no opposite traffic is allowed. However, bidirectional arcs allow for shorter paths since shorts cuts may be possible. In order to combine the advantages of both unidirectional and bidirectional flow, multiple lane paths can be used, introducing

multiple paths with opposite traffic flow on the same arc [53].

The network layout is of direct influence on the system performance: it affects travel time for vehicles, the number of vehicles required and the degree of possible congestion [53].

4.1.1. Airport Representation

In the context of a tug-enabled taxiing system, the network layout mainly concerns the representation of the airport in the system. Not all papers specify the exact representation of the airport [58, 52], but the vast majority considers a graph-based lay-out using nodes and edges to represent the airport infrastructure. Airport characteristics such as location of gates, runway entrances, spots or intersections are represented by nodes. Edges represent the possible paths for vehicle movement between these locations.

Since Amsterdam Airport Schiphol (AAS) will be considered as use case in this thesis, a review of research based on implications for AAS [18, 60, 61, 55, 56, 7, 8, 9, 51] will be considered when discussing network layout.

The majority of the differences in network layout between the considered papers are related to the simplifications made with respect to real infrastructure at AAS. The most elaborate model is the one used by Soomers and Kamphof [8, 9], mainly distinguishing itself due to a very realistic representation of the taxiway network. This includes multiple runway entrances/exits instead of one entrance/exit per runway; realistic turning paths (avoiding right-angle turns); and constraints on allowable directions, velocities and accelerations per segment. In addition, the service road network is incorporated, as well as apron operations. Previous research mostly did not take these operations into account, meaning that push-back from the gate was not considered. As a consequences, gates are modelled using a single meta-gate node representing an apron area, including multiple gates. This means that for the majority of the models, aircraft movements are only modelled from their respective apron to the assigned runway and vice versa [18, 60, 61, 55, 56]. In [9], all gates are modelled using separate nodes, taking into account the modelling of push-back operations as well.

4.1.2. Runway Configurations

As explained in section 2.2, AAS has 5 runways that can be used in different combinations. All studies that consider AAS as their case study, base their input data on historic flight information for one or multiple days of operation. When evaluating this input data, it can be seen that the majority of the departing flights use runways 36L, 36C, 24 and/or 18L [18, 60, 61, 55]. For arriving flights, common runways to use are runway 06/24, runway 18R, runway 18C or runway 36R [60, 61, 55].

The commonly-used runways for departing and arriving flights correspond with the most frequently used Runway Mode of Operation (RMO) at AAS [7]. These are RMO North and RMO South. In Figure A.3 and Figure 4.2, the runway use in the various RMOs during the arrival and departure peak and transition phase is shown. Both the studies conducted by Schiphol itself [7] and the studies by Soomers [8] and Kamphof [9] focus on evaluating the TaxiBot concept for these two RMOs.

(a) Arrival peak (36L/06 + 36R) (b) Transition (36L + 36C/06 + 36R) (c) Departure peak (36L + 36C/06)

Figure 4.1: RMO North (arrival peak, transition and departure peak), including the corresponding runways. Taken from [8].

(a) Arrival peak (24/18C + 18R) (b) Transition (18L + 24/18C + 18R) (c) Departure peak (18L + 24/18R)

Figure 4.2: RMO South (arrival peak, transition and departure peak), including the corresponding runways. Taken from [8].

4.1.3. Concluding Remarks on Network Layout

The network representation of AAS as proposed by Soomers and Kamphof [8, 9] will be used in this study due to its high level of detail and agreement with reality. Figure 4.3 shows a visual representation of AAS in the model of Soomers, with zoomed in sections provided in Figure 4.4 [8].

The network consists of nodes and edges, where nodes represent gates, uncoupling points, parking facilities, runway entrances/exits and intersection locations. Two unidirectional edges connect adjacent nodes, either representing runways, taxiway centerlines or service roads. Next to type, edges also contain information on dimensions, position and operational constraints, such as closing of segments, stop bars (for crossing of runways) and the RMO. Due to these operational constraints, the network can be considered a dynamic object.

With respect to RMOs, operations for days with the two runway combinations that are used most often at AAS are evaluated. These concern RMO North and RMO South (Figure A.3 and Figure 4.2). This is in accordance with the studies done by Schiphol and To70 [51] and the work done by Soomers [8] and Kamphof [9].

Assumptions

1.1 **Service Road Infrastructure**: for all terminal and air side service roads, it is assumed that the roads are wide enough to allow for bidirectional traffic of TaxiBots between themselves (when not towing an aircraft). Based on trials performed by Schiphol using TaxiBots at AAS [6], in reality not all terminal and air side service roads allow for this bidirectional traffic. An example is the area between D and E concourse. Therefore, this assumption may result in lower levels of traffic present on taxiways than in reality, since TaxiBots might have to use the taxiway network to make a return movement instead of the service roads.

1.2 **Taxiway Infrastructure**: the construction of an extra taxiway lane on taxiway Quebec is assumed to be completed, allowing for bidirectional traffic flow on this specific road. Currently, taxiway Quebec can only be used in one direction, depending on instructions from ATC. However, it is expected that the infrastructural expansion of Quebec will be finished towards 2025 [23]. Since it is not expected that the TaxiBot concept will be fully deployed before then, this assumption can be considered valid.
In addition, wingspan restricted taxiways as provided by the Aeronautical Information Package (AIP) are adhered to [9].

1.3 **Direction of Taxiways**: while in reality taxiway Alpha and Bravo have standard flow directions (section 2.2), these limitations are lifted in this work. The result is that traffic is allowed to move in both directions on the taxiways, as long as edge conflicts are avoided. For ATC, this may result in a less well-organized environment, possibly increasing workload. On the other hand, when considering distributed control of TaxiBots using automated systems, this does not necessarily have to be true. However, the result of this assumption on ATC workload will have to be validated in future work.

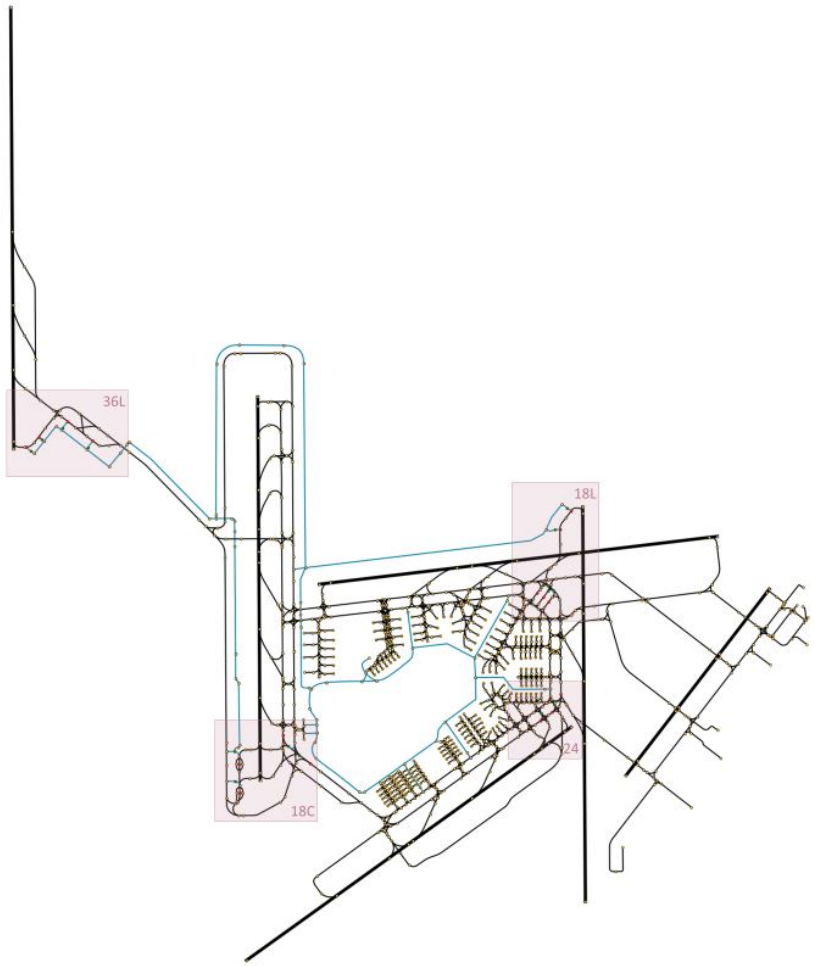


Figure 4.3: Network layout of Amsterdam Airport Schiphol (AAS) used by Soomers [8]. Including: taxiway centre lines (thin black line), runways (thick black line) and service roads (thin blue line).

4.2. Traffic Management and Resolving Conflicts

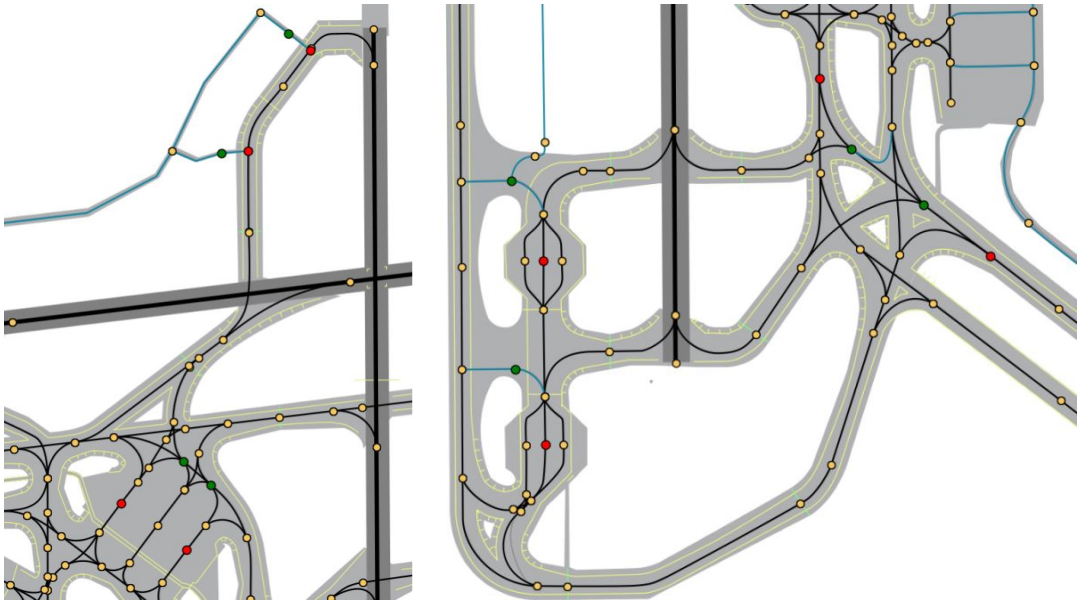
If two vehicles are moving on the same edge in opposite directions on a single lane path, the vehicles are forced to stop in front of each other and blocking of the path occurs. With respect to system performance, it is suggested by Vis [53] to avoid collisions instead of solving upon detection.

Three different categories can be defined with respect to deadlock and collision avoidance [53]. First of all, the layout of the network can be designed in such a way that collisions and deadlocks are avoided. Next, use can be made of so-called non-overlapping control zones, where only one vehicle at the time is allowed to enter and travel through the zone. Finally, deadlock and collision avoidance can also be tackled when defining routing strategies. In this thesis, the focus will be on the third option.

4.2.1. Prediction and Avoidance of Collisions and Deadlocks

When considering collisions and deadlocks in the context of aircraft taxi operations, three main types of conflicts can be determined as defined by Smeltink et al. [62]. In Figure 4.5, the three types of conflicts (intersection, rear-end and head-on) are visualized [62].

- **Intersection Conflict**: two aircraft use the same taxiway intersection at the same time.
- **Rear-end Conflict**: two aircraft use the same edge at the same time, in similar direction. This will result in a conflict if the rear aircraft has a higher speed.
- **Head-on Conflict**: two aircraft use the same edge at the same time, in opposite direction. This will result in a conflict if multiple lane paths are not introduced in the system.



(a) Network layout of AAS at runway 18L. (b) Network layout of AAS at runway 18C.

Figure 4.4: Zoomed-in sections of network layout of Amsterdam Airport Schiphol (AAS) used by Soomers [8]. Including: taxiway centre lines (thin black line), runways (thick black line), service roads (thin blue line), stop bars (thin green line), regular nodes (yellow), uncoupling locations (red) and all-clear points (green).

(a) Crossing. (b) Rear-end. (c) Head-on.

Figure 4.5: Graphical representation of an intersection conflict (a), rear-end conflict (b) and head-on conflict [62].

As stated previously, using collision avoidance techniques instead of collision detection and solving consecutively improves the overall system performance [53]. To backup this claim, it is noted that the majority of the relevant literature stated before makes use of conflict avoidance strategies by imposing separation constraints when determining and/or evaluating (possible) routes [18, 30, 31, 59, 61, 54, 8, 9]. A more elaborate description on how conflict avoidance can be done while finding optimal paths, can be found in chapter 6.

Only in the work of Van Baaren [60], interference with other traffic is not considered. In the work of Salihu, Lloyd and Akgunduz [57], intersection conflicts are not considered. For the models used by Tindemans [55] and Kroese [56], only separation between aircraft among themselves. However, conflicts between TaxiBots among themselves and between aircraft and TaxiBots is not taken into account. Soomers and Kamphof [8, 9] assume that separation between TaxiBots does not have to be maintained on roads wide enough to allow for bidirectional traffic.

4.2.2. Separation Constraints

Different studies use different types of separation constraints. A distinction can be made between time-based and distance-based separation. When using distance-based separation, Benda [18] assumes a required separation distance of 50 meters between TaxiBots themselves and between TaxiBots and aircraft when travelling on taxiways. Aircraft need to maintain 150 meters of separation. When travelling on service roads, the required separation distance for TaxiBots is 35 meters. Soomers [8] uses significant smaller values and assumes 20 meters of required separation between aircraft themselves and between aircraft and TaxiBots, excluding a 5 meter safety margin. In the study performed by To70,

a separation distance of 60 meters is maintained [51]. All other research do not specify the distances required for separation (if used at all).

Time-based separation is mainly used for separation on runways between arriving and departing flights and is dependent on aircraft size. Based on Wake Turbulence Category (WTC), time-based separation is maintained [59, 8, 9]. In Table B.4, the different timing criteria are listed for arriving and departing flights.

Table 4.1: WTC separation in seconds for different aircraft categories. Based on RECAT-EU wake separation rules [63].

Leader		Super Heavy A	Upper Heavy B	Lower Heavy C	Upper Medium D	Lower Medium E	Light F
Super Heavy	**A**	80	100	120	140	160	180
Upper Heavy	**B**	80	80	80	100	120	140
Lower Heavy	**C**	80	80	80	80	100	120
Upper Medium	**D**	80	80	80	80	80	120
Lower Medium	**E**	80	80	80	80	80	100
Light	**F**	80	80	80	80	80	80

4.2.3. Concluding Remarks on Traffic Management Strategies

In this thesis, interference between traffic will be taken into account. Based on the different types of conflicts as described before, a set of definitions can be formulated describing when vehicles are in conflict with each other, based on the work done by Soltani et al. [54].

1. Two vehicles that are trailing each other on the same edge in the same direction, are not considered to be in conflict when they maintain separation distance.
2. Vehicles that interact on intersections are not considered to be in conflict when they maintain separation distance.
3. Vehicles that travel on the same edge in opposite direction are considered to be in conflict (for exceptions, refer to assumptions).

Assumptions

2.1 **Modelling of Ground Vehicles**: in this thesis, only aircraft and TaxiBot movements are modelled. The movement of other ground vehicles, such as luggage handling or catering services, are excluded. This will mainly result in simplifications regarding the apron area, where other ground handling vehicles will also be present, and possibly affecting the optimality of the path traversed by TaxiBots.
Schiphol has announced that it strives for autonomous ground operations by 2050 and thus, the interaction between TaxiBots and ground handling vehicles will have to be studied in future work. However, since it is assumed that the TaxiBot is controlled by a human driver at this point, the avoidance of conflicts with other ground handling vehicles can be expected to be handled by the driver, since this is similar to driving a car on the road.

2.2 **Interaction Between Vehicles**: separation between aircraft themselves and between aircraft and TaxiBots are maintained, while separation between TaxiBots themselves on bidirectional service and taxi roads is not guaranteed. Similar to the previous assumption, the assumption that the TaxiBot is controlled by a human driver allows for relaxation of separation control between TaxiBots themselves.

2.3 **Aircraft Types Under Consideration**: all aircraft available in the flight data that is used for simulation will be mapped into one of the six categories of the RECAT-EU system [63], based on their MTOW. This results in a simplification of actual weight and size of aircraft. However, since this study aims to provide a preliminary insight in how different task allocation strategies influence system performance, the actual aircraft specifics are of minor importance at this research stage.

2.4 **Separation Distances Between Vehicles**: with respect to separation required between arriving and departing flights, different aircraft will be mapped to one of the six categories available in the

RECAT-EU system provided by EUROCONTROL [63]. Subsequently, the required separation between the different aircraft will be based on the values in Table B.4.
Regarding separation distances on taxiways and service roads, it is assumed that both aircraft between themselves and aircraft and TaxiBots should maintain 60 meters separation [51]. Although this is not strictly specified in airport manuals, it is deemed a valid assumption upon consultation of operational experts.
Finally, separation distances in the apron area are considered the same as on taxiways and service roads, with the additional requirement that the entire pushback path of aircraft should be free when performing this maneuver [9].

4.3. Location of Pick-Up and Delivery Points

When designing a network layout, one has to determine the location of pick-up and delivery points. Pick-up and delivery points could refer to points where an actual load has to be picked up and/or delivered, but also to inspection/charging stations or machines that have to be visited by the AGV. The choice of the locations is of importance since it is of influence on the system performance in terms of e.g. total distance travelled or waiting times [53].

4.3.1. Type of Flights Considered for Towing

When considering pick-up and delivery points in the context of tug-enabled taxi operations, a distinction can be made between outbound and inbound taxiing. For departing aircraft (outbound taxiing), the pick-up location will be the aircraft stand near the gate and the delivery location will be a decoupling location on the route towards the assigned runway. The opposite will be the case when considering arriving aircraft (inbound taxiing).

In the majority of the research on the implementation of a tug-enabled taxiing system, only towing of outbound flights is considered [18, 58, 52, 59, 61, 7, 8]. One of the reasons for this is that the potential for emission savings for taxi-out is higher than for taxi-in, since the amount of carbon dioxide emitted during outbound taxiing is almost doubled compared with inbound taxiing [58]. This can be attributed to two factors: a higher weight of departing aircraft compared with arriving aircraft [60] and on average, higher taxi-out times compared with taxi-in times (based on major US hub airport operation) [61].

In [54, 57], different strategies related to towing for inbound and outbound taxiing are considered. A baseline scenario (no towing at all) is compared with different approaches related to deployment of tugs (e.g. both inbound and outbound always using tugs; outbound using tugs and inbound using main engines; inbound and outbound using tugs if tug is available). As concluded by Soltani et al. [54], the deployment of tugs both for inbound and outbound taxiing based on availability of tugs results in the most economical solution, while providing for 95% reduction in carbon dioxide emissions. Similar results are obtained by Salihy, Lloyd and Akgunduz [57]. In addition, it is shown that when deploying tugs based on availability also for inbound flights, a carbon dioxide reduction of around 70% is achieved, compared to 50% reduction when only deploying tugs for outbound flights (with respect to conventional taxi operations). However, it has to be noted that a fully electric towing system is assumed in both studies.

4.3.2. Centralized and Decentralized (De)Coupling

As explained in subsection 3.2.2, the coupling and decoupling for respectively inbound and outbound flights near the runway is a procedure that takes time. Therefore, the location of these coupling/decoupling locations has to be chosen wisely in order to avoid blocking of other traffic as much as possible. In addition, fuel savings are also highly influenced by the location of these points, since the position of the points with respect to the assigned runway determines the distance that remains for the aircraft to taxi with its main engines.

In a TaxiBot simulation study executed by Schiphol and To70 in 2020 (only considering outbound towing), two main strategies were defined for positioning the decoupling points: at central locations, or at de-central locations [7]. Central locations are considered to be so-called P-platforms (parking areas), not dedicated to a specific runway but strategically located to be accessible from multiple runways. Expected advantages are flexibility and the possibility to decouple outside main taxiways, not blocking any other traffic during decoupling. The downside is that fuel savings are expected to be lower, since approximately half of the entire taxi operation will still have to executed using the aircraft's main engines

[7].

On the other hand, when considering de-central decoupling locations, the aircraft is being towed as close to the runway as possible. Therefore, de-central locations will be dedicated to one runway. It is expected that fuel savings will be higher. However, when performing decoupling in a decentralized manner, this procedure will have to take place on the main taxiway, potentially resulting in queuing at runways [7].

As a result of the simulation study, it was concluded that using a centralized distribution of decoupling locations was not a favourable concept of operations to be implemented at AAS [7]. This had two reasons, namely negligible fuel savings (due to increased taxi distances because of the central locations) and increased airport congestion (due to exceeding the capacity of the P-platforms, which are also used as holding) [7]. Therefore, it is suggested to adopt a concept of operations with de-central decoupling procedures at Schiphol. These findings are confirmed by the research of Soomers [8], which concludes that using a mixed strategy (using decentralized decoupling locations and when capacity allows for it, use centralized decoupling locations) is optimal in terms of minimizing for delays.

4.3.3. Possible Decoupling Locations at AAS

In addition to (de)coupling strategies, the actual locations for possible (de)coupling will also have to be considered from an infrastructural point of view. Possible locations for decoupling are identified by Schiphol after careful consideration of all possibilities and making a distinction between locations that are invalidated provisory (can in the future be validated after a specific update) and invalidated indefinitely (no actions will be undertaken to validate the decoupling location) [7]. Currently, no possible decoupling location is suitable for direct use. The decoupling locations that are disqualified for actual use until action is taken are summarized in Table 4.2. The necessary actions vary from adding markings or lights to changing procedures to make return movements possible (also refer to subsection 4.1.3). Note that decoupling locations that are invalidated indefinitely are not further considered in this work.

Table 4.2: Overview of possible decoupling locations, identified after feasibility study by Schiphol [7].

Decoupling Point	Runway	Comment
P1	Central/18L	Add marking and/or lighting to indicate stopping point
P3	Central/18L	Add marking and/or lighting to indicate stopping point
P4/P5	36C	1. Add marking and/or lighting to indicate stopping point 2. Return movement not possible
P6/P7	36L	1. Add marking and/or lighting to indicate stopping point 2. Return movement not possible
18C-C Northbound	18C	1. All-clear point TaxiBot is too far away, create new all-clear point 2. Add marking or lighting to indicate stopping point
18C-D Northbound	18C	1. All-clear point TaxiBot is unsafe to reach, create new all-clear point 2. Add marking or lighting to indicate stopping point
09 Westbound (A20)	09	1. Decoupling point at busy taxiway (flow risk) 2. Add marking or lighting to indicate stopping point
09 Eastbound (A20)	09	1. Decoupling point at busy taxiway (flow risk) 2. Add marking or lighting to indicate stopping point
27 at N1	27	1. Decoupling point at busy taxiway when in use (flow risk) 2. Add marking or lighting to indicate stopping point 3. Return movement not possible
24 Southbound	24	1. Decoupling point at busy taxiway (flow risk) 2. Add marking or lighting to indicate stopping point 3. All-clear point with blast risks
24 Northbound (A7)	24	1. Decoupling point at busy taxiway (flow risk) 2. Add marking or lighting to indicate stopping point 3. All-clear point with blast risks
18L E6	18L	1. Decoupling point at busy taxiway (flow risk) 2. Add marking or lighting to indicate stopping point 3. All-clear point TaxiBot is hard to reach, create optimized route to all-clear point

4.3.4. Concluding Remarks on Location of Pick-up and Delivery Points

In this study, fully outbound taxiing is considered, implying that all departing flights will have to be assigned a TaxiBot to tow them from the gate to a designated decoupling location. Inbound flights use their own engines to taxi from the runway to the designated gate.

Based on the work done by Soomers [8], a mixed strategy concerning the choice of decoupling locations will be used, combining both decoupling locations at holding platforms and on taxiways.

Assumptions

3.1 **Departing Flights**: in this thesis, only outbound taxiing is considered when deploying TaxiBots. This means that departing flights will be towed from the gate to a decoupling location near the designated runway, while all arriving flights use their own engines to taxi from the runway to the assigned gate.
The reason for only considering outbound towing is that emission savings are higher for outbound taxiing compared with inbound taxiing. Since all inbound flights use their own engines for taxiing, extra waiting times for arriving flights that need to be coupled to a TaxiBot are not considered, including the allocation of TaxiBots to arriving flights. The assumption of only considering outbound towing is considered to be valid since Schiphol itself has identified inbound taxiing as a ”very long-term goal” [51, p. 9].

3.2 **Decoupling Locations**: the decoupling locations as mentioned in Table 4.2 are assumed to be suitable to use for the decoupling procedure. In addition, all required infrastructural changes to make the decoupling locations feasible are assumed to be in place. As mentioned above, this is currently not the case until various measures have been taken. However, Schiphol has started investigating the feasibility of these measures [7]. For now, it is assumed that the required infrastructural and procedural updates are feasible and can be executed to make the decoupling locations suitable for operations. If part of these measures turn out not to be feasible, less decoupling points will be available, possibly affecting traffic flow, airport congestion and fuel savings.

3.3 **Runway Scheduling and Gate Allocation**: in this study, the assignment of aircraft to gates (pick-up point) and runways (delivery point) is considered to be out of the scope. This means that runway scheduling and gate allocation is assumed to be provided as an input to the model as part of the flight schedule, based on historic data. As a result, the computational complexity of the model is decreased, not accounting for the planning and allocation of aircraft to their starting and ending point. In addition, real-time changes of assignment of aircraft to gates/runways is not considered.

4.4. Fleet Composition and Characteristics

To make sure that all tasks in the system are completed in time, a minimum number of vehicles required has to be determined. However, from a financial point of view, the number of vehicles should not be overestimated. Besides, a (too) large number of vehicles in the system could cause unnecessary congestion.

Vis [53] has identified several factors that are of influence when determining the optimal fleet size for an AGV-system. These are copied below.

- Type of drive
- Costs of the system
- Capacity of the vehicle
- Speed of the vehicle
- Number of units to be transported and points in time at which units can be or need to be transported
- Layout of the network system (see section 4.1)
- Traffic congestion (see section 4.2)

- Number and location of pick-up and delivery points (see section 4.3)
- Vehicle dispatching strategies (see section 4.5)

4.4.1. Type of Drive

In several applications of AGV systems, electrically powered vehicles are used. As a result, battery charging has to be considered when designing such a system. The time required for charging can heavily affect system performance in terms of number of vehicles required, throughput of the system, congestion and costs [53].

Although the TaxiBot, which is diesel-driven, is the only certified towing tug on the market up until this date, Schiphol already identified in its feasibility study (2020) that a carbon-neutral driveline should be developed for the TaxiBot [7]. Several previous studies assume fully electric towing tugs running on batteries, taking into account charging time [47, 60, 61, 54, 55, 56, 57]. However, since for this study the current version of the TaxiBot will be implemented as towing tug, a more into-depth analysis on battery charging is considered beyond the scope of this thesis.

4.4.2. Costs of the System

With respect to the fleet size of a tug-enabled taxi system, some past research assumes an infinite number of vehicles available or does not take into account how a finite number of vehicles available influences system performance [18, 30, 31, 58, 7, 6, 8]. For the remaining studies, costs of the system are the driving factor when it comes to determining an optimal solution. Three main approaches can be distinguished when it comes to taking costs into account and are summarized below. The work done by Chua et al. [52] is the only study that focuses on the effect of a tug-enabled taxiing system on the workload for ground control.

- Research that primarily aims to reduce the environmental impact of a finite number of towing tugs available and thus, minimizes for fuel cost or fuel consumption [47, 59, 60]
- Research that primarily aims to reduce delays introduced by a finite number of towing tugs available and thus, minimizes for taxi time [55, 56]
- Research that aims to reduce the overall cost of implementing an alternative taxi system, taking into account fuel cost, operating cost and delay cost (for a finite fleet) [61, 54, 57]

Based on the different approaches, an outline is provided on how the fuel costs, delay costs and tug purchasing/operating costs are estimated. Studies that assume infinite vehicles available are taken into account when providing this outline.

Fuel Cost

When considering fuel cost, both fuel consumed by the aircraft and fuel consumed by the towing vehicle (if not electrically powered) have to be taken into account.

- **Jet Fuel Cost**: when considering aircraft fuel consumption, both the fuel usage of the engines and the APU have to be taken into account. Different studies use different ways to estimate these fuel usages. The work done by Postorino, Mantecchini and Gualandi [58] makes use of the Emissions and Dispersion Modelling System (EDMS) based on EUROCONTROL Base of Aircraft Data (BADA) to calculate aircraft engine emissions based on characteristics of air side aircraft operations and engine characteristics. Similar approaches use aircraft and engine characteristics from the ICAO Aircraft Engine Emission Databank in combination with calculated/measured performance profiles [59, 60, 61, 54, 57].
  In their feasibility study related to the TaxiBot concept, Schiphol uses as a rule of thumb a fuel flow of 7.5 kg/min per engine and 2 kg/min for the APU. These values are indicative and based on the Boeing 737NG [6].
- **TaxiBot Fuel Cost**: in the studies that take into account the fuel consumption of diesel-driven TaxiBots [59, 58], the fuel consumption of the TaxiBot is estimated by computing the product of unit fuel consumption depending on TaxiBot type and the total duration of operations. No distinction is made between time when the TaxiBot is towing an aircraft and when it is driving in solo mode.

Values for fuel consumption range from 0.28 kg/min (NB) [59, 58] to 1.03 kg/min (WB) [59]. According to Guillaume [61], using a constant fuel rate per unit time is not sufficient. Therefore, he uses an approach based on Newton's equations of motion, taking into account the performance profile, friction forces and drag forces to calculate fuel consumption for each specific part of the trajectory of a TaxiBot.
When performing measurements with a NB TaxiBot at Schiphol, a fuel consumption of 0.65 kg/min was found when in pilot mode [6]. However, when comparing the current technology (using a fossil driveline) with a fully electric towing system, an indicative reduction of approximately 3% in fuel usage can be obtained (based on an average taxi-out time of 14 minutes). In the work done by Salihu, Lloyd and Akgunduz [57], a similar reduction in fuel consumption was found, resulting in omitting the fuel consumed by TaxiBots from the analysis.

Delay Cost

When speaking about aircraft delays, various definitions and interpretations of delay exist. In addition, one can express delays in terms of minutes or in terms of money. In this section, focus will be put on how to express delays in monetary terms.

One of the studies that extensively investigates delay cost, is the work done by Guillaume [61]. According to Cook [64], a relation can be found between the tactical delay cost and the MTOW at given delay durations (5, 15, 30, 60, 90, 120, 180, 240 and 300 minutes). Equation 4.1 shows the relation, where the variables m and c can be found in the work of Cook [64]. Guillaume shows that Equation 4.1 can be used to estimate at-gate delay costs, consisting mainly of extra costs for crew and passengers (e.g. re-booking and compensation), if extra costs related to fuel are already taken into account in the model [61]. If this is not the case, both at-gate and en-route tactical delay costs should be taken into account.

$$cost_{delay} = m \cdot \sqrt{MTOW} + c \tag{4.1}$$

Tug Purchasing and Operating Cost

Guillaume [61] calculates an hourly depreciation cost of the vehicle to account for purchasing and operating costs in his model. The hourly depreciation cost can be found by converting purchasing costs, driver costs and maintenance costs to an hourly values. In order to do so, the estimated vehicle amortization is 5 years, while using the vehicles for 18 hours/day [65]. Currently, the NB and WB TaxiBot purchasing costs are 1.5 million USD and 3 million USD respectively [65]. Driver costs are expected to be 40 USD/h and maintenance costs can be estimated by adding 7.5% of the new value to the depreciation cost of the vehicle [65]. In addition, a salvage value of the vehicle of 10% is assumed after being operated for 5 years [61]. Linear depreciation is used to find an hourly depreciation cost for both types of TaxiBots.

4.4.3. Capacity and Speed of the Vehicle

Intuitively, the throughput of a tug-enabled taxi system will also be dependent on the technical specifications of the towing tractor. In subsection 3.2.1, an overview of the technical specifications of a TaxiBot is given, including capacity and speed constraints. These constraints differ between the two types of TaxiBots that are available: a Narrow-Body (NB) and Wide-Body (WB) version. Most research focuses on the implementation of NB TaxiBots or do not differentiate between the two types and assumes all flights can be towed by a single type of TaxiBot [18, 30, 31, 58, 54, 56, 57, 8, 9]. Some research take into account the differences between NB and WB TaxiBots into the constraints of the model [59, 61].

4.4.4. Number of Units To Be Transported and Associated Time Frame

When considering the number of units to be transported and their associated time frame for completion in the context of a tug-enabled taxi system, this mainly concerns the flight schedule. The construction of the flight schedule is described in detail for the majority of the studies [18, 47, 59, 60, 61, 55, 56, 8, 9] and usually contains the following parameters:

- **Aircraft Type**: refers to the type of aircraft, such as A380. For example, the type can be used to extract information on weights [55], or to map it to a specific aircraft category used in the model [8, 9].

- **Flight Direction**: defines whether the aircraft is arriving or departing.
- **Starting Node**: depending on the flight direction, the starting node can either be an assigned (meta-)gate (departing flight) or the assigned runway exit (arriving flight).
- **Ending Node**: depending on the flight direction, the ending node can either be an assigned runway (departing flight) or assigned (meta-)gate (arriving flight). Depending on the complexity of the network layout, the runway entrance already is predetermined (for departing flights), or the runway entrance is optimized for. As far as we know, the model used by Soomers and Kamphof [8, 9] is the only model that optimizes for runway entrances.
- **Starting Time**: depending on the flight direction, the starting time refers to the moment the aircraft leaves the gate (departing flight) or the moment the aircraft leaves the runway (arriving flight). For some studies, both the scheduled and block departing/arrival times are included [61, 55].

Note that for all studies under consideration, runway scheduling and gate allocation is not considered part of the scope of the research. On the contrary, it is assumed that the flight schedule already includes an assigned gate and runway. This is also included in the assumptions related to pick-up and delivery points in section 4.3.

Some studies take pre-departure sequencing into account. In the work done by Baaren [60], next to a start time, an ending time is included as well. For departing flights, this means that the starting time refers to the earliest moment the aircraft is allowed to leave the gate, and the ending time is the scheduled take-off time. Soomers and Kamphof [8, 9] includes a more simplified form of pre-departure sequencing, by using the TSAT time as starting time for departing aircraft.

4.4.5. Concluding Remarks on Fleet Composition and Characteristics

In this thesis, a finite number of vehicles will be available for outbound towing. The deployment of these vehicles will be optimized in terms of environmental impact and delays. As a result, travel time will be one of the most important parameters to consider, since this both covers fuel consumption (environmental impact) and delays.

In this context, both delays on the taxiways and at the gate will be considered. Delay is considered to be the increased average duration of aircraft taxi time when compared to conventional operations, including additional waiting time at the gate for departing flights due to unavailability of a TaxiBot for towing with respect to the original flight schedule. This includes decoupling time of the TaxiBot from the aircraft at decoupling locations. Both the aircraft taxi time for conventional and TaxiBot scenarios will be simulated, allowing for an objective comparison of the two situations.

Assumptions

4.1 **Diesel-Driven TaxiBot**: the TaxiBot is considered to be diesel-driven, and capable of performing operations consecutively for 18 hours per day [65] without the need to refuel. Although Schiphol has announced that it could and should decarbonize the drive-line of the TaxiBot (which currently is diesel-driven) [7], no timeline for this ambition is available. Therefore, it is assumed that this will be a very long-term project, approving the belief that the TaxiBot will remain diesel-driven in the near future.

4.2 **Fuel Cost of Jet Engines**: in this thesis, the simplified numbers used by Schiphol in their feasibility study related to fuel use of engines will be used as well (7.5 kg/min per engie and 2 kg/min for APU use). Although these numbers are a great simplification of actual fuel use, the aim of this thesis is not to quantify the exact environmental benefits of using TaxiBots, but merely to optimize the allocation of TaxiBots in terms of environmental impact. It is assumed that using indicative figures at this stage of the research on assignment of TaxiBots to aircraft provides for enough accuracy.

4.3 **Fuel Consumption Due To Engine Warm-Up**: as explained in subsection 3.2.2, engine warm-up will be performed during taxiing with a TaxiBot. Since engine warm-up will have to be performed both in the conventional taxi situation and when deploying TaxiBots, it is not of a difference when we are mainly interested in the difference in fuel consumption between both cases. Therefore, the fuel consumption due to the engine warm-up will not be taken into account in the analysis.

4.4 **Fuel Cost of TaxiBot**: the values used by Schiphol for fuel consumption of the TaxiBot as a rule of thumb will be used (0.65 kg/min of diesel). Since these values are directly obtained through measurements during trials, they are assumed to be representative for the actual fuel consumption.
No differentiation between driving in solo mode or pilot mode will be made in this study. As a result, the estimated fuel consumption will be higher than in reality, since it is expected that the fuel usage will be less when the TaxiBot is driving in solo mode. However, since the fuel usage of the TaxiBot accounts for only 3% of the total fuel consumption when comparing it with a conventional scenario, this assumption is expected not to be of significant influence when comparing both scenarios and thus, can be considered valid.

4.5 **Delay Cost**: when expressing delay at the gate in monetary terms, the assumptions as described in subsection 4.4.2 will be used.

4.6 **Tug Purchasing and Operating Cost**: when including tug purchasing and operating cost as an hourly depreciation cost, the assumptions as described in subsection 4.4.2 will be used.

4.7 **Types of TaxiBots Under Consideration**: no differentiation between the two different types of TaxiBots will be considered. As a reference, the NB TaxiBot will be used. Currently, this type of TaxiBot is certified for towing the B737 and A320 aircraft, accounting for approximately 54% of the total flights at Schiphol [7]. The B737 and A320 families fit within CAT-D in the RECAT-EU system ('upper medium') [63]. Assuming that the NB TaxiBot in theory should be capable of towing all aircraft in CAT-D to CAT-F, this accounts for approximately 80% of all flights at Schiphol (based on conversations with experts). All flights in CAT-A to CAT-C should theoretically be towed by a WB TaxiBot.
As a result of not differentiating between the different types of TaxiBots, the output parameters of the model will not be representative for real-life scenarios, such as number of vehicles required, the allocation of TaxiBots to aircraft and even the total taxi time required. However, since this study is a first look into allocation of TaxiBots and different strategies, it can be considered out of the scope of this research to take different types of TaxiBots into account. Future work should evaluate how the use of different types of TaxiBots influences the model output.

4.8 **Kinematic Constraints of Vehicles**: in this study, it is assumed that all modelled vehicles have a maximum acceleration and deceleration level, as well as speed limits on straight segments and in turns. These values can be found in the work of Soomers [8] and Kamphof [9].

4.9 **Speed Limits of TaxiBot**: the maximum speed of the TaxiBot while driving in solo mode is increased from 30 km/h to pilot mode speed. This assumption is based on recommendations for future operations of Schiphol [6]. Since the normal taxi speed of aircraft and aircraft with TaxiBot (55 km/h and 43 km/h respectively) is significantly higher than the 30 km/h, the TaxiBot is not able to blend into the taxiway system [6]. Therefore, increasing the speed limit to pilot mode speed when driving in solo mode, will result in a more realistic simulation of reality.

4.10 **Pre-Departure Sequencing**: the model will not take departure slots for runway use into account, but instead will use the TSAT from historic data as the time that an aircraft will have to leave the gate. Since this is the way that Schiphol takes into account the process of pre-departure sequencing (subsection 2.1.3), it is assumed that using the TSAT as input to the model will be a realistic measure to decrease runway queuing.

4.11 **Deviations from Flight Schedule**: in the baseline model, the flight schedule is considered to be deterministic, meaning that no deviations from the schedule will occur in the execution. The only exception is that due to an unavailability of TaxiBot, delay at the gate for departing flights might occur with respect to the TSAT (see also assumption item 4.12 mentioned next).
When considering model extensions, the inclusion of random delay to model disruption is considered (also refer to assumption item 5.1). For more details on the methodological steps to be taken in the modelling process, refer to chapter 8.

4.12 **Minutes Delay at Gate**: when considering delay at the gate, the delay in minutes will be measured with respect to the TSAT time. This is the time that is used as an input for the model

and corresponds with the time that the aircraft will have to leave the gate. Refer to assumption item 4.10 why the TSAT is used.

4.5. Control of AGVs

In order to satisfy all demands in the AGV-system as efficient as possible in a safe manner, a control policy should be set in place. This policy should include a strategy for dispatching the vehicles to loads, routing and scheduling of the vehicles and dispatching idle vehicles to waiting locations [53].

A determining factor in choosing the set-up of such a control policy, is the difference between an offline and online controlled system. When all transportation demands including origin, destination, release time and transportation time are known beforehand for a certain time frame, the system can be classified as an offline system. If the complete set of tasks is not known a priori and/or decisions have to made real-time due to the stochastic nature of the system (e.g. high degree of uncertainty), the system will have to be controlled online [53].

4.5.1. Dispatching of Vehicles

When considering the assignment of vehicles to loads, Vis [53] describes two different approaches for the dispatching problem: work centre initiated dispatching vs. vehicle initiated dispatching. Viewing the dispatching problem from a work centre point of view, means that a vehicle is selected from a set of idle vehicles to perform a task. In the vehicle initiated dispatching problem, a vehicle that is not assigned to a task has to choose a task from a set of open requests.

The dispatching of vehicles only plays a role in problems that assume a finite number of vehicles available. Within those studies, the majority models the problem as a Vehicle Routing Problem (VRP) using a Mixed-Integer Linear Programming (MILP) approach, implying a work centre initiated dispatching point of view [47, 60, 55, 56, 61, 54, 57]. In the work done by Khammash, Matecchini and Reis [59], a general-purpose simulation tool is used to model the problem, not stating explicitly whether the dispatching is work centre or vehicle initiated.

4.5.2. Routing and Scheduling of Vehicles

Similarly to dispatching strategies, routing and scheduling of vehicles is mostly done using linear programming techniques, by modelling the problem as a VRP [30, 31, 58, 60, 61, 54, 55, 56, 57]. The three main exceptions are the work done by Benda [18], Soomers [8] and Kamphof [9], which use an agent-based approach.

According to Vis [53], VRPs can be categorized as static problems, meaning that the route is determined in advance and that the solution will be carried out as planned. Opposite to this are dynamic problems, where the routing of vehicles is based on real-time information and changes in the system are taken into account when planning [53]. Despite the stochastic nature of the airport environment and related ground operations, all studies under consideration model the routing and scheduling of vehicles as a static problem [18, 30, 31, 58, 60, 61, 54, 56, 57, 8, 9]. One exception is the study done by Tindemans [55], who takes into account uncertainties in arrival and departure times of aircraft by generating operating times based on probability density functions. However, this study still assumes an offline version of the problem, since it is assumed that the flight schedule for an entire day (including the uncertainties) is known beforehand.

More details on suitable modelling techniques used for dispatching, routing and scheduling in the field of ground surface operations will be provided in subsection 4.6.3. A review of the state-of-the-art algorithms on dispatching and routing/scheduling can be found in chapter 5 and chapter 6 respectively.

4.5.3. Positioning of Idle Vehicles

After having finished a task, a vehicle can either immediately get assigned a new task or can become idle. If a vehicle becomes idle, it has to be positioned in the system in such a way that it can efficiently reach a new task and does not block tasks or routes for other (active) vehicles [53]. Three commonly applied rules for positioning idling vehicles are mentioned by Vis [53]:

- **Central Zone Positioning Rule**: idling vehicles are routed towards centrally located parking areas and can respond to new tasks once arrived at these buffer locations.
- **Circulatory Loop Positioning Rule**: when having completed a task, empty vehicles are routed

on one or more loops in the network and wait from there for new tasks to fulfill.

- **Point of Release Positioning Rule**: once a vehicle becomes idle (after having completed a task at a destination), it remains at its last point of release until a new task is assigned.

When considering the positioning of idling towing tugs, almost all studies assume that after completion of a towing assignment, the tug moves back to a parking or holding location [18, 30, 31, 47, 58, 52, 60, 61, 54, 8, 9]. For some studies, the possibility of directly moving towards a new departing aircraft is included [30, 31, 47, 60, 61, 54].

With regards to the studies using AAS as use case, the location of the parking facility is not always specified [61, 7, 6]. In studies that have specified these locations, the parking locations for towing trucks used by KLM near the G-pier (WB) and the B-pier (NB) [18], the P-platforms P1 and P2 [60] and a dead-end node in the north of the central service road network near the H-apron [8, 9] are used as parking locations for the towing trucks.

4.5.4. Concluding Remarks on Control of AGVs

The focus of this study will be on evaluating different TaxiBot dispatching strategies in the context of aircraft ground movements. Different dispatching, routing and scheduling algorithms will therefore be elaborated on more extensively in chapter 5 to chapter 7. However, some specific control-related assumptions will be elaborated on below.

Assumptions

5.1 **Perfect Execution**: in the baseline model of this thesis, it is assumed that the execution of the plan generated by the model is perfect and that no unexpected delays or disruptions will occur. In reality, airport ground operations are inherently stochastic by nature, and therefore, this assumption results in a simplification of real operations. However, since this study is one of the first to investigate allocation of a finite fleet of TaxiBots to outbound aircraft, it is considered valid to focus merely on the planning phase.
When time and resources allow to do so, it is planned to incorporate a type of unexpected events into the model by introducing some random variable representing delay. For more details on the methodological steps to be taken in the modelling process, refer to chapter 8.

5.2 **Online Problem**: the problem will be modelled in an online setting using intervals, meaning that new information on arriving and departing flights will arrive in the system in time intervals. Since in real-life operations the TSAT is issued approximately 10 minutes before the EOBT, a time horizon of 10 minutes look-ahead is used in this thesis. However, it is assumed that for this entire interval, all the information is available and no changes will occur. In reality, this is not the case (item 4.11 in section 4.4 and item 5.1).

5.3 **Waiting Locations TaxiBots**: similarly to the model used by Soomers [8], the main parking location for the TaxiBots is located at a dead-end node in the north of the central service road network near the H-apron.

4.6. Final Concept of Operations

Based on the literature review and associated assumptions made throughout the previous sections, a final concept of operations can be formulated. This concept of operations can be seen as a high-level summary of all assumptions made earlier and is described in subsection 4.6.1. In addition, the scope of the research is also set and described.

Next, a set of system requirements is formulated, based on the concept of operations and described in subsection 4.6.2. These requirements are used to provide for the mapping of the domain (deploying TaxiBots at AAS) to the modelling technique, which is elaborated on in subsection 4.6.3.

4.6.1. Summary

In this study, the deployment of TaxiBots to make fully outbound towing at AAS possible, will be studied. The main goal will be to minimize the travel time, taking into account both environmental impact in terms of fuel consumption and delay with respect to conventional airport ground and taxi operations. All inbound flights will taxi from the runway to their designated gate using their own engines.

Actual flight schedules for past operating days with the two most frequently used RMOs will be used to evaluate the feasibility of the concept. Note that the flight schedule is considered to be deterministic in the baseline model, meaning that no unexpected deviations or disruptions will occur in the execution. Future model extensions might include unexpected events by including a random variable representing delay. In addition, both runway scheduling and gate allocation are considered outside the scope of this study, meaning that the assigned runway and allocated gate are based on historic data and available in the flight schedule. A simplified version of pre-departure sequencing is taken into account to avoid runway queuing, by using the TSAT as the time that an aircraft will have to leave the gate.

From an operational point of view, interaction between different vehicles is taken into account by ensuring separation between all aircraft and between aircraft and TaxiBots. Coupling of the TaxiBot to the assigned aircraft will happen at the gate, decoupling will take place on an assigned decoupling location. Both centralized and decentralized decoupling is possible, implying that all possible (future) decoupling locations at AAS are available for the model to choose from. Apron operations will be included in the model. The main parking/waiting location for TaxiBots is located near the H-apron, on the north of the central service road network. Note that no differentiation in type of TaxiBot is made (Narrow-Body and Wide-Body versions). In this study, only the use of the NB TaxiBot will be considered, assuming that this type is certified to tow all aircraft.

Scope of Research

When considering the aforementioned concept of operations for the deployment of TaxiBots at AAS, several boundaries regarding the scope of this research can be identified in order to keep realistic research goals in terms of resources and time.

- **Operational Feasibility**: the main focus of this study is to evaluate the operational feasibility of deploying TaxiBots at AAS, while economic feasibility is investigated to a lesser extent. Both technical feasibility and legal feasibility of the concept are not taken into account.

- **Commercial Airlines**: in this research, only taxi operations for commercial passenger flights are considered, excluding cargo flights, military flights and general aviation.

- **Human TaxiBot Drivers**: it is assumed that enough trained crew is present to handle the TaxiBots at AAS. Besides, no additional time has been accounted for switching drivers during operating hours because of ending shifts.

- **Human-Machine Interactions**: although it is assumed that TaxiBots will be controlled by a human driver, the resulting human-machine interactions are considered to be outside the scope of this research. Instead, it is assumed that no disruptions arise due to human factors, implying that humans execute all commands perfectly.

- **Air Traffic Control**: the consequences of implementing a tug-enabled taxiing system on ATC is not considered in this research, both in terms of workload and procedural changes as well as technical changes of the current ATM systems. In addition, legal feasibility of the concept in relation to ATC is not considered in this research.

- **Environmental Impact**: the environmental impact of deploying TaxiBots in comparison with conventional taxi operations will be determined only in terms of fuel consumption, excluding quantification of greenhouse gas emissions and aircraft noise.

- **Nominal Operating Conditions**: this study focuses on nominal operating conditions, meaning that all non-nominal operations are excluded from the analysis. This includes for example de-icing operations and low visibility conditions.

- **Operating Data**: due to the COVID-19 pandemic, operating data will be based on the year 2019 (instead of 2020 or 2021) to account for a more realistic scenario for the coming years. Expected growth of the aviation sector with respect to 2019 is not taken into account and thus, the feasibility of the concept will be evaluated based on travel levels in 2019.

4.6.2. System Requirements

Based on the concept of operations and associated assumptions, a set of model requirements can be formulated. These requirements will form the basis for the evaluation of the suitability of a modelling paradigm and more specifically, which algorithms related to dispatching, routing and scheduling are suitable to study the implementation of TaxiBots at AAS.

Note that this set of system requirements is not meant to be a complete and all-encompassing set of requirements, but rather can be considered an extension of the aforementioned set of assumptions.

1. The airport nodal network shall represent the lay-out of Amsterdam Airport Schiphol (AAS), including apron areas, service roads, taxiways and runways.
2. The flight schedule shall match a real flight schedule at AAS for operating days where only the two most frequently employed RMOs are in use.
3. The objective of the model shall be to minimize the total taxi travel time of all outbound and inbound flights.
4. Conflict and collision avoidance strategies for both aircraft and TaxiBots shall be in place.
5. The model shall be able to deal with unexpected delays and disruptions.
6. The model shall be able to be used for real-time operations and in an online setting.
7. The model shall be able to generate results on performance for both the high-level system, as well as for the local level (considering specific runways and/or taxi segments).
8. The model shall be completed within the limited time and resources available.

4.6.3. Modelling Paradigm

Based on the concept of operations and resulting system requirements as formulated above, a choice for a modelling paradigm has to be made. In this section, the characteristics and benefits of multi-agent systems will be discussed, including its applicability to the problem under consideration specifically. Special attention will be paid to coordination between agents. All information provided in these sections is based on the lecture series on ”Agent-Based Modelling and Simulation in Air Transportation” of Sharpanskykh [66, 67, 68].

Characteristics of the Agent-Based Modelling Paradigm

Within the agent-based modelling paradigm, a model is specified as a Multi-Agent System (MAS), which is ”a set of agents interacting in the environment to solve problems, achieve goals, or execute tasks that are difficult or impossible for a single agent” [66, p. 22]. Within this definition, three key components of MASs can be distinguished: the environment, agents and interaction [66].

- **Environment**: the environment of a MAS consists of all objects that are not agents. However, agents can interact with the environment or act upon (changes in) the environment. The environment can be classified along three axes:
  - **Deterministic vs. Non-Deterministic**: in a deterministic environment, a specific action always has the same effect that is known beforehand. In other words, no uncertainty in the consequences of actions is considered.
  - **Static vs. Dynamic**: in a static environment, no changes occur except the changes that are a consequences of actions of agents. In dynamic environments, the environment itself may change over time, unrelated to agent behaviour.
  - **Accessible vs. Inaccessible**: in an accessible environment, all agents always have access to the complete state of the entire system.
- **Agents**: agents in a MAS are autonomous actors in the environment. They are able to perceive this environment and act upon changes in it, based on their behavioural properties. Agents can also include some form of rationality, that allows them to not only be driven by observations, but also by internal states. In that case, one speaks of the agent characterized by cognitive properties as well.

- **Interaction**: when considering interaction, both interaction among agents and between agents and the environment is considered. Social abilities of agents allow them to communicate and coordinate with each other. In addition, agents might have the ability to reason about behaviour or internal states of other agents. More on coordination between agents is elaborated on in the next section.

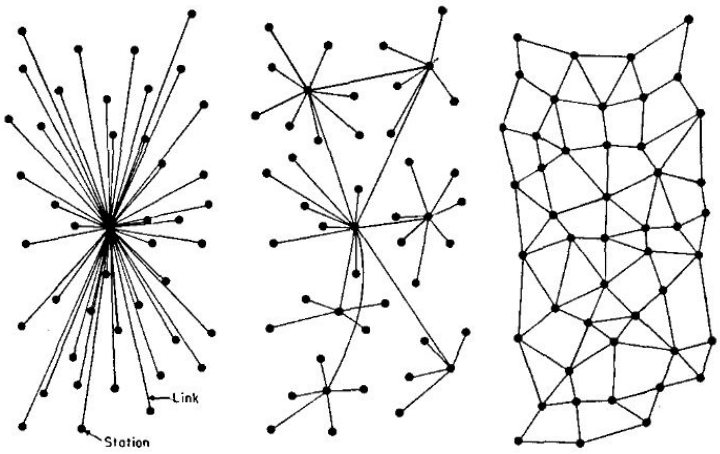
Benefits of Agent-Based Modelling Paradigm

As mentioned before, the agent-based modelling paradigm is due to its characteristics suitable to apply in the general transportation domain [69]. When examining the system requirements described in subsection 4.6.2 in more detail, it can be concluded that also in this context, the agent-based modelling paradigm is very applicable and suitable. In accordance with Adler and Blue [70], Sharpanskykh also concludes that agent-based modelling is a suitable technique especially for geographically open distributed system [66]. When considering system requirement item 1, it becomes apparent that the problem is geographically dispersed over an airport environment. Since agent-based modelling allows for explicit interaction between components and agents [66], the implementation of collision and conflict avoidance strategies (system requirement item 4). Furthermore, with regards to system requirement item 5 and item 6, it is noted by Sharpanskykh that MASs are especially suitable for dynamic, uncertain environments that allow for analyzing of emergent behaviour [66]. Finally, since agent-based models are inherent to represent multiple scales of analysis [66], a MAS is deemed suitable to generate performance results on both the high-level, as well as the local level.

Although the agent-based modelling paradigm seems very suitable based on the majority of the system requirements, special attention should be paid to item 8, concerning limited resources. One of the disadvantages of agent-based modelling, is that it may be (very) time-consuming to create an agent-based model [66]. Therefore, in the modelling phase of this thesis, the available time and resources should be monitored closely to prevent serious delay from happening.

Control Structures in an Agent-Based Network

Different type of control structures can be distinguished in networks, as shown in Figure 4.6. In a centralized network, information about the state of the entire network is known at the central node. When moving from a centralized network to a more decentralized or even distributed network, differences mainly arise due to differences in this information availability. In a decentralized network, nodes only have information about local conditions in their connected cluster of nodes, whereas in a distributed network, no hierarchical structure is present. All nodes are equally connected and all have the same amount of information available. With less information available in decentralized and/or distributed networks, this often leads to a faster response to changes since decision authority and thus, processing capacity is distributed over more resources. However, local optimization may not result in a global optimal solution. Since in a centralized network information about the state of all nodes is available at a single point, global optimization of the entire network is allowed [1].



(a) Centralized (b) Decentralized (c) Distributed

Figure 4.6: Graphical representation of centralized (a), decentralized (b) and distributed (c) networks [71].

Concerning current control in the field of ATM, this is mainly done in a centralized manner. Several studies suggest a shift from the current centralized control paradigm to a more decentralized system, including Udluft [1]. The reason for this is that it could increase the available capacity by moving away decision authority and consequently, the number of tasks, from a centralized control unit to distributed entities in the system. In the context of air traffic management, ”control” can be visualized using networks consisting of nodes (elements where information is processed, or agents) and links (elements that connect nodes and represent transfer of information, or interactions between agents) [1].

In their work, Soomers [8] and Kamphof [9] propose a hierarchical multi-agent control architecture, using both centralized and decentralized approaches at different levels of the system to combine the advantages of both control structures. In this hierarchical control architecture, planning and conflict avoidance is performed on the central level, whereas plan instruction, execution and monitoring is performed at a local level. A visualization of the control structure is presented in Figure 4.7. Since the work of Soomers [8] and Kamphof [9] show that the proposed architecture is capable of providing for safe and efficient ground surface operations, the same control architecture will be used in this thesis.

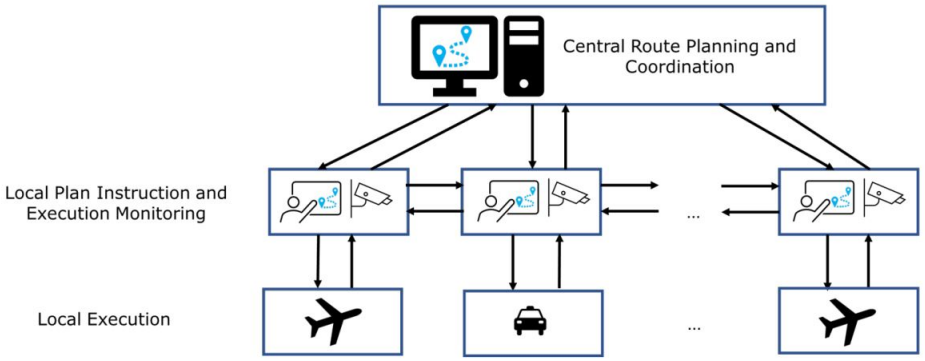


Figure 4.7: Visualization of the hierarchical multi-agent control architecture used in the work of Soomers and Kamphof. Taken from [9].

Coordination Between Agents

When considering coordination between agents, two types of settings can be considered: a cooperative setting and competitive setting (Figure 4.8). Intuitively, one can understand that when cooperating, agents try to combine their efforts to accomplish a common goal, whereas agents try to maximize their own benefits at the expense of others in a competitive setting.

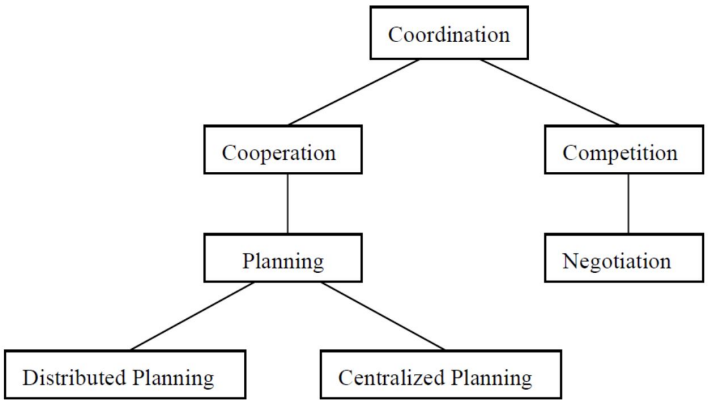


Figure 4.8: Classification of different types of coordination in a MAS. Taken from [67].

In this context, a cooperative fleet of TaxiBot is considered to perform outbound towing at AAS. As a consequence, all TaxiBot agents will work together to maximize the defined objective. More on the specifics on the objective of the fleet of TaxiBots can be found in chapter 5.

As can be seen in Figure 4.8, when considering a cooperative setting among agents in a MAS, a

multi-agent planning problem arises. In this problem, the goal is to find a plan for each individual agent such that both global and individual goals are met by means of coordination between the set of agents [68]. The coordination can either be set up in a centralized manner or in a distributed/decentralized manner, as discussed in the previous section.

The general multi-agent planning problem usually consists of six phases, outlined below [68]. Note that not all phases have to be part of the multi-agent planning problem. In addition, a seventh step related to replanning may be incorporated [68].

1. **Global Goal or Task Refinement**: if necessary, the global goal is divided into a set of sub-goals that can be assigned to individual agents.
2. **Task Allocation**: the set of (sub-)goals or (sub-)tasks is assigned to agents.
3. **Coordination Before Planning**: in this phase, a set of rules or constraints is defined that restrict the behaviour of agents, usually aimed at preventing the production of conflicting plans.
4. **Individual Planning**: each individual agent makes a plan to perform the assigned task(s). In this phase, coordination among agents may be involved.
5. **Coordination After Planning**: all individual plans of agents are coordinated to check for conflicting plans.
6. **Plan Execution**: after resolving of conflicts, plans are executed. This phase may involve coordination as well.

Concluding Remarks on Modelling Paradigm

Within the general traffic and transportation domain, agent-based approaches are widely used to model and simulate control [69]. According to a study done by Adler and Blue [70], the agent computing paradigm can provide significant benefits in problem domains that 1) are geographically distributed; 2) have a dynamic nature; and 3) require flexible interactions of subsystems. Since transportation systems are in general spatially distributed and characterized by highly dynamic environments that require communication on different levels, agent-based approaches are considered to be suitable [69]. Similarly, when modelling TaxiBots at AAS, we are also dealing with a geographically distributed problem (system requirement item 1) that is inherently dynamic and uncertain by nature (system requirement item 5 and item 6). Based on the previous review of the agent-based modelling paradigm, it is concluded that this modelling technique is suitable to use in the remainder of this thesis.

This claim is backed up by the fact that existing research on distributed control in aviation in general [1, 16, 17, 72] and on the implementation of external towing tugs at airports [18, 8, 9] is largely based on the agent-based modelling paradigm.

Using the aforementioned enumeration related to the multi-agent planning problem as a guideline for the remainder of this report results in the following structure: first, the problem of assigning agents to tasks will be discussed in chapter 5. Next, finding a set of paths to execute the task assignment will be elaborated on in chapter 6. Finally, special attention will be paid to the combination of task allocation with path finding in chapter 7.

5

Multi-Robot Task Assignment

Up to this point, the domain of the problem under consideration in this thesis has been described, concluded with a fully defined concept of operations in chapter 4. Based on this concept of operations, a Multi-Agent System (MAS) can be set up. One of the phases in this MAS will be the assignment of tasks to agents, or the allocation of a finite fleet of TaxiBots to all outbound flight at AAS.

Therefore, in this chapter, the subject of task allocation will be further explored, starting with a taxonomy of allocation problems in section 5.1. Next, a broad overview of different task assignment approaches will be provided in section 5.2, including a trade-off of these approaches with respect to applicability in the TaxiBot domain. Based on the trade-off, two approaches will be elaborated on in more detail: greedy approaches in section 5.3 and auction-based approaches in section 5.4. Both sections will be concluded with a review on which algorithm is deemed most suitable to use for allocating TaxiBots to outbound flights.

5.1. Classification of Allocation Problems

In the work done by Gerkey and Mataric [73], it becomes apparent that existing task allocation problems can be categorized along three different dimensions. These dimensions are visualized in Figure 5.1 and explained below.

- **Single-Task Robots (ST)** vs. **Multi-Task Robots (MT)**: in the case of ST problems, each robot can execute at most one task at a time, whereas in MT problems, some robots can perform multiple tasks simultaneously.
- **Single-Robot Tasks (SR)** vs. **Multi-Robot Tasks (MR)**: in the case of SR problems, each task can be fulfilled by exactly one robot, whereas in MR problems, multiple robots are needed to fulfill one task.
- **Instantaneous Assignment (IA)** vs. **Time-Extended Assignment (TA)**: in the case of IA problems, the available information related to tasks, robots and the environment only allows for instantaneous allocation of tasks to robots, whereas in TA problems, the number of tasks exceeds the number of robots and/or more information is available (such as how tasks are expected to arrive over time), allowing for reasoning of agents about future allocation of tasks.

Based on the concept of operations as described in section 4.6 and the associated assumptions in chapter 4, the problem that will be studied in this thesis can be classified as an ST-SR-TA problem according to the definitions above. Each TaxiBot can only tow one aircraft at a time (ST), each aircraft can be towed by exactly one TaxiBot (SR) and based on the windowed approach of making the information in the flight schedule available to the agents, TaxiBot agents are allowed to consider future allocations to outbound flights up to a certain extent (TA).

Although the taxonomy used by Gerkey and Mataric is widely used and considered ground-breaking at the time, in the work done by Korsah, Stentz and Dias [74] it is argued that this classification is incomplete since it is limited to systems with independent tasks. An example given by the authors that is

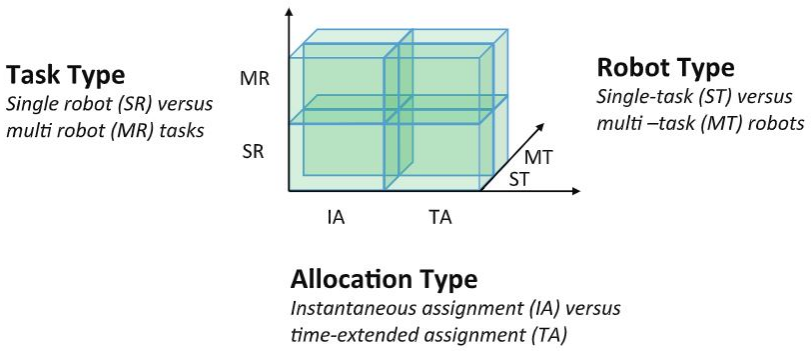


Figure 5.1: Visualization of the dimensions along which Gerkey and Mataric [73] categorize different task allocation problems. Based on work done by Korsah, Stentz and Dias [74].

notably excluded, concerns multi-robot routing systems in which robots have to perform various spatially distributed tasks. Obviously, it could improve overall system performance if the tasks that are located close together, are executed by the same robot. In this case, the total utility of the robot performing these clustered tasks might not equal the sum of the utilities for executing them one by one. Therefore, Korsah, Stentz and Dias propose another taxonomy called iTax to explicitly incorporate the degree of interrelated utilities and task constraints [74]. Since the assignment of a TaxiBot to aircraft depends on which aircraft at which location is dropped off before, taking interrelated utilities into account is relevant for this subject.

The authors propose a two-level taxonomy, where the first level relates to the degree of interdependence of agent-task utilities. A description of the four different options is given below [74]. The second level uses the taxonomy used by Gerkey and Mataric mentioned before [73].

- **No Dependencies (ND)**: in this case, the utility of an agent performing a task is completely independent of all other tasks and agents in the system, which commonly occurs when the utility function of an agent is determined by its capabilities or proximity to a certain task.
- **In-Schedule Dependencies (ID)**: in this case, the utility of an agent performing a task is dependent on the other tasks assigned to the agent, which commonly occurs in time-extended allocation problems. Another case in which In-Schedule Dependencies occur is when agents are capable of performing multiple tasks simultaneously.
- **Cross-Schedule Dependencies (XD)**: in this case, the utility of an agent performing a task does not only depend on its own schedule, but also on the schedule of other agents. A common example is when tasks consists of multiple sub-tasks that need to be performed in a predetermined order by different agents, in some cases even simultaneously. A key difference between In-Schedule Dependencies and Cross-Schedule Dependencies is that for ID, agents can individually optimize their schedule, while for XD this can only be done in coordination with other agents.
- **Complex Dependencies (CD)**: in this case, the goal is to allocate complex tasks to the available agents. Complex tasks have multiple possible decompositions, meaning that not only the question of *who* is going to perform a task and *when*, but also *which* set of sub-tasks should be performed. Similar to Cross-Schedule Dependencies, an agent's utility is dependent on the schedules of other agents.

When considering task allocation in the context of assigning TaxiBots to outbound flights, the utilities of agents are interrelated based on In-Schedule Dependencies. Since tasks can be performed by single agents independently of the other agents in the system, no cross-schedule or complex dependencies exist. However, the utility of performing a task for a single agent is indeed dependent on the other tasks in its schedule. For example, towing a certain aircraft to a certain runway influences the utility of towing future aircraft: some might be located closer to the end position of the agent then others. In addition, since outbound aircraft have task deadlines (times that towing should start), previous tasks of agents affect whether they can make it in time to future outbound flights. Thus, their utility is affected

by previous tasks in their schedule.

To conclude, the allocation problem related to assigning TaxiBots to outbound flights at AAS can be considered a ID [ST-SR-TA] problem. Therefore, in the remainder of this chapter, focus will be put on solution techniques that apply to this variant of the task allocation problem.

5.2. Overview and Comparison of Solution Techniques

In this section, an overview of solution techniques applicable to the IDST-SR-TA problem is provided. Based on the survey of Korsah, Stentz and Dias [74], a distinction between optimization-based and market-based approaches can be made. Both solution techniques will be elaborated on below, followed by a trade-off in subsection 5.2.3.

5.2.1. Optimization-Based Approaches

When dealing with optimization, one aims to find mathematically a solution that maximizes or minimizes a certain function or variable subject to a set of constraints [75]. Based on the set of constraints, a set of feasible solutions to the problem exist, and the optimal solution is then defined by a set of criteria formulated in the form of an objective function [76].

Within the optimization domain, two main approaches can be distinguished: deterministic and stochastic approaches [76, 77]. In the sections below, a general description of these two approaches will be provided, including relevant literature on the application of such approaches in the domain of MRTA.

Deterministic Approaches

Generally speaking, deterministic approaches apply a method for solving optimization problems to end up with a global or an approximately global solution [78]. The guarantee of optimality is based on the fact that deterministic optimization approaches are strictly repeatable: based on a certain initial condition, the approach will follow a set of strict and rigid procedures, resulting in the same result every time the optimization is performed [76, 79].

One of the most well-known task allocation problems in optimization research is formulated as the Optimal Assignment Problem (OAP) and can be characterized as a one-shot ST-SR-IA problem. This problem is generally composed of r amount of robots and t amount of tasks, where t is at most equal to r. This method can be solved in $O\left(rt^2\right)$ time by using linear programming optimization algorithms, such as the Hungarian method [80]. Although these methods have shown to perform computationally well when scaled to hundreds of robots and tasks [73], a prerequisite of these methods are the costs related to each rt combination, which can be unavailable or computationally expensive to produce [81].

In the domain of MRTA, several studies have applied the Hungarian method in more realistic versions of the assignment problem. Examples of this are the study done by Mills-Tettey, Stent and Dias [82], which efficiently recalculates the optimal assignment of tasks to robots in a dynamic environment where costs associated for each robot-task combination may change.

Another interesting variation of the OAP is the variant where the number of tasks to be performed exceeds the number of available robots or problems in which the arrival of new tasks is known beforehand. When previously assigned robots can be reassigned after the arrival of new tasks, the problem reduces to an instance of the iterated ST-SR-IA [73]. The iterated version of the ST-SR-IA problem can both be solved deterministically (by rerunning the Hungarian algorithm every time a new task enters the system) or greedily (refer to the section below). When robots cannot be reassigned after the arrival of new tasks, the problem can be classified as ST-SR-TA. Since problems in this class are strongly NP-hard, meaning that no guarantee for an optimal solution within polynomial time can be given, deterministic approaches are not suitable to tackle the OAP for ST-SR-TA problems. However, various greedy approaches have been formulated based on the previous described deterministic approaches to find (bounded) suboptimal solutions. These will be elaborated on in the following section.

Within the domain of MRTA, the allocation of agents to tasks can also be formulated as a Multiple Traveling Salesmen Problem (mTSP). The mTSP is based on the original Traveling Salesmen Problem (TSP), in which a salesmen has to visit n number of cities in the most efficient way (e.g. shortest route, shortest time) before returning back home. When considering mTSP, the main difference with TSP is that not one, but multiple salesmen are given. Various studies have formulated the MRTA problem as a mTSP, due to its analogy with assigning robots to tasks in the most efficient way [83].

For solving the mTSP, several classical optimization techniques have been considered, such as Mixed-Integer Linear Programming (MILP) [84, 85]. Exact solution methods such as Branch-and-Bound, Brach-and-Cut and Dynamic Programming have been used [86] that provide guarantees to reach optimal solutions. However, as shown by Martin et al. [77], the number of feasible discrete solutions for the considered ST-SR-TA MRTA problem can be approximated by Equation 5.1 [1] Already for 5 robots and 5 tasks, the number of feasible solutions goes up to 244,140,625, which is several orders of magnitude larger than the proposed limit for using exact methods (less than 10^5 solutions according to Equation 5.1) [77].

$$N_{sol} = (r \cdot t)^{t+1} \tag{5.1}$$

Greedy Approaches

For various MRTA problems, the number of tasks *t* often exceeds the number of robots *r*, not allowing to deterministically compute a solution using the OAP formulation. In addition, new tasks might arrive in an online fashion to the system, requiring time-extended scheduling of tasks for each robot. Both problems are instances of the ST-SR-TA problem, which is known to be strongly NP-hard. However, as mentioned before, when previously assigned robots can be reassigned, the problem reduces to an instance of the iterated ST-SR-IA. The iterated version of the ST-SR-IA problem can be solved using a deterministic approach as explained before, or by the use of the Broadcast of Local Eligibility (BLE) assignment algorithm [87]. The greedy BLE algorithm is known to be 2-competitive for the OAP, meaning that in the worst case scenario, the algorithm will produce a solution with the benefit of at least half of the benefit of the optimal solution [73]. However, it is noted by the authors of [73] that the greedy algorithm ”works extremely well on [typical MRTA] problems” [73, p. 945].

When previously assigned agents are not allowed to be reassigned, the ST-SR-TA problem can be approximately solved by reducing it to an online instance of the ST-SR-IA problem. This problem can be solved sequentially by first optimally solving the initial assignment problem (e.g. using the Hungarian method) and afterwards greedily assigning remaining tasks in an online fashion when the agents become available. The latter can be done by using the MURDOCH algorithm, which prescribes to assign a new task upon introduction to the most fit agent to execute this task [73]. Note that if re-allocation of robots is not allowed, the MURDOCH algorithm is known to produce the best possible outcome for any online assignment problem [73]. This ST-SR-TA approximation algorithm is 3-competitive in the worst case for the offline OAP, but performance approaches optimality when the difference between the number of of initially presented tasks and robots decreases.

One of the works that implements an online version of the Hungarian method in the context of pickup and delivery problems is the research done by Ma et al. [88]. They use the algorithm CENTRAL that combines task allocation and path planning for robots with pickup and delivery tasks in a warehouse setting. Although the algorithm is capable of performing efficient task assignment every time a task enters the system, the running times are considerably large for reasonable number of agents and tasks (e.g. around 400 seconds for 20 agents and 500 tasks, all added with a frequency of 2 tasks per timestep). However, since CENTRAL both performs task allocation and path planning, the run time cannot be solely attributed to the task allocation mechanism. It remains unclear how the computational time should be divided over the task allocation and path finding part respectively.

Stochastic Approaches

Due to scalability issues with deterministic approaches for optimization problems in the domain of task allocation, several heuristic or stochastic methods have been developed to find solutions in reasonable time frames. Stochastic approaches are characterized by the integration of random factors, meaning that solutions are generated by combining heuristics and randomness and not guaranteed to produce the exact same result every time [76, 79]. Although stochastic approaches are in general more efficient and flexible than deterministic approaches, the solution quality of stochastic approaches cannot be guaranteed [78].

Two main approaches can be distinguished within the domain of stochastic optimization approaches: trajectory-based and population-based algorithms [76]. Trajectory-based methods make use of a single agent or solution, moving through the solution space in a discrete fashion by adhering to certain

[1] The determination of the number of feasible solutions is based on the fact that the problem is modelled mathematically using discrete variables $u_i(n)$, representing the nth allocated task to robot i [77].

guidelines. A better solution is always accepted, whereas a not-so-good solution can be accepted with a certain probability. Examples of trajectory-based algorithms are tabu search and simulated annealing [75]. In the work done by Juedes et al. [89], a simulated annealing approach is used to allocate tasks to processors based on the workload of each task. Through experiments, it was shown that the simulated annealing algorithm generated comparable results with respect to a greedy algorithm (analytically proven to provide a solution of at least 41% of the optimal). Although results went up using a larger number of iterations (40,000), this also increased run time significantly: around 700 seconds for the allocation of 20 tasks to 10 processors, compared to 170 seconds for 10,000 iterations.

The work done by Zhang, Collins and Shi [90] shows that both the run time and solution quality for a simulated annealing approach are also highly dependent on the cooling ratio. For smaller cooling ratios, run time increase significantly, but so does the solution quality (with respect to greedy algorithms) and v.v. In experiments where 3 robots are used to perform 50 tasks, the solution is approximately 12% increased with respect to a greedy algorithm, while run times are 200 times higher (around 20 seconds versus 0.1 second for a greedy algorithm). Note that this applies to a simulated annealing approach using a smaller cooler rate. Finally, it was also tested by these authors how the simulated annealing algorithm performs in an online environment. In the case of 6 robots and 50 tasks, the number of required iterations has increased sixfold. It is not clear how this relates to run time, but based on other research, it is expected that run times increase at least sixfold as well.

Finally, the research of Hussein and Khamis [91] compares two optimization-based methods (a simulated annealing algorithm and genetic algorithm) with a market-based approach in the context of MRTA. For large scale problems (including 15 robots and 50 tasks), they show that the simulated annealing algorithm reaches the best solution of the three approaches in approximately 20 seconds.

Among the reviewed studies related to trajectory-based stochastic approaches, it can be concluded that large variations in run times arise. However, for all outcomes it holds that no guarantees on the bounds for optimality can be provided. In addition, generally speaking, it can be concluded that the large number of iterations required for a trajectory-based algorithm does not allow for scalability to TaxiBot applications in an online environment [81, 86].

Next to trajectory-based approaches, population-based approaches can be used to find solutions to the task allocation problem. These include amongst others genetic algorithms, swarm optimization approaches or evolutionary-based algorithms. Compared to trajectory-based approaches, population-based approaches use multiple agents or solutions instead of a single one to explore the search space to find a near-optimal solution [76].

Genetic algorithms are widely applied to solve problems formulated as an instance of the mTSP [92, 93, 94, 95, 96, 97, 98]. The principle of the genetic algorithm relies on the Darwinian theory of evolution and survival of the fittest. Based on combining promising solutions and applying mutations to the combined solution, the search space is discovered [99]. Although all research apply different variations of the genetic algorithm, a common characteristic is that several thousands of iterations are generally needed to reach stable solutions (3-5 robots and 20-70 tasks). In addition with the fact that most research assume task allocation problems of the ST-SR-IA class and ignore any time-extended assignment of robots to tasks, seems to make genetic algorithms not suitable to apply in the domain of TaxiBot scheduling.

However, several exceptions were found with regards to the significant running times needed for convergence. In the work done by Martin et al. [77], a genetic algorithm is used to allocate 5 robots (heterogeneous fleet) to 15 inspection tasks in a solar power plant represented by a meshed warehouse-like grid. While the algorithm was able to present near-optimal solutions, the total average computation time was just 3.5 seconds. Furthermore, the research took temporal constraints into account by modelling the problem as an instance of the ST-SR-TA class. However, no guarantees on optimality of the genetic algorithm can be provided [77]. In the work done by Edison and Shima [100], a genetic algorithm is used to solve the integrated task assignment and path finding problem for UAVs. Roughly speaking, the run time of the algorithm was $\frac{1}{60}$ of the entire scenario duration, meaning that for a scenario of 2 minutes, the algorithm took around 2 seconds.

Another research implemented a decentralized version of the genetic algorithm to a situation where a number of tasks are distributed over a grid of 100 x 100, to be performed by a selection of robots [101]. For a scenario where 45 tasks were to be executed by 5 robots, the algorithm did not only produce the best solution compared to other state-of-the-art heuristics, it was also shown to converge rapidly (within

5 seconds after completion of the first iteration). However, no graph-based layout was used, possibly increasing the complexity of the problem. In addition, the online arrival of new tasks to the system has not been evaluated.

Next to genetic algorithms, swarm-based approaches can be found in the literature that tackle the MRTA problem. Similarly as for genetic algorithms, often a large number of iterations is required to produce converging solutions. An example of this is presented in the work done by Du et al. [102], which combines task allocation based on the model of honeybees (where agents specialize in certain tasks) with the Ant Colony Optimization (ACO) algorithm that is based on the shortest-path finding capability of ants to their food using pheromones. The result is the so-called Ant Task Allocation (ATA) algorithm. Even for a simple grid-like environment, several thousands of iterations are needed to produce converging solutions. Another research makes use of the ACO algorithm in the context of gate assignment [103]. Although the algorithm produces results relatively fast (on average 3 seconds) for the assignment of 10 aircraft to 6 available gates, the computational time increases significantly when the number of aircraft is increased to 30 (on average 80 seconds). Similar examples can be seen in research on applying Particle Swarm Optimization (PSO) to cooperatively assign UAVs to military tasks [104], combining PSO with hill-climbing and neighborhood search techniques to assign towing trucks to taxiing aircraft [105] and using a evolutionary-based algorithm to assign UAVs to tasks, while avoiding dead-locks and taking time-sensitive uncertainties into account [106].

Some interesting exceptions to the large number of required iterations for swarm-based approaches can be found in the literature. An example of this is the work done by Kurdi et al. [107] and uses a bacteria-inspired approach to allocate UAVs to perform tasks in a cooperative setting. During foraging, bacterial colonies show swarm movement based on the release of attracting and repelling chemicals. Using this behaviour as a source of inspiration, UAVs are assigned to tasks related to detection and treatment of palms in plantation areas. Especially with respect to scalability and increasing number of robots, the Multi-UAV Task Allocation for RPW Combat Based on Bacteria Behaviour (UTARB) algorithm outperforms several other state-of-the-art heuristics significantly. For a set of 64 tasks and varying robot numbers (2, 4, 8, 16 and 32), all calculations could be performed within 30 seconds while maximizing throughput, where other heuristics need at least several minutes and in most cases multiple hours. However, no applications to an online setting are tested.

5.2.2. Market-Based Approaches

Whereas optimization-based approaches focus on exploring a finite solution space based on a set of constraints, market-based approaches are inspired by the market trading concept and are based on economic theory [76].

Two main approaches can be distinguished: approaches based on auctions and based on game theory. According to economic theory, an auction is a mechanism in which activities can be distributed among participants according to their bids and bid evaluation criteria [76, 108]. On the other hand, game theoretic approaches model agents and their objectives in a game setting, allowing for agents to maximize their own utility based on knowledge about the other agents and the environment [86].

Auction-Based Approaches

Ever since the beginning of history, auctions have been used to allocate resources among a group of participants [109]. Generally speaking, the process of task allocation when considering auction mechanisms can be split up in four stages: announcement, submission, selection and contract [76, 108].

- **Announcement**: a central auctioneer announces the available tasks in the system and informs the robots on the specifics related to the tasks.
- **Submission**: based on its capability to perform the task and the objective function, each individual agent submits a bid to the central auctioneer representing the individual utility.
- **Selection**: depending on the optimization strategy, the central auctioneer evaluates the received bids and chooses the winning agent.
- **Contract**: the winning agent is offered a contract by the central auctioneer to commit to its bid and to perform the task under consideration.

In some cases, no central auctioneer is present, allowing tasks to be allocated based on the submitted bids and a consensus algorithm. Auctions of this type are therefore referred to as consensus-based auctions [110]. Furthermore, a distinction can be made between open-cry auctions, where all bids are publicly available to all agents involved, and sealed-bid auctions, where bids are only submitted to the auctioneer. In most MRTA applications, use is made of sealed-bid auctions [110].

Although it might seem counter-intuitive to use a competitive mechanism in a cooperative setting where the goal is to maximize the entire system performance, it is noted by Gerkey and Mataric that agents are not considered to be selfish: ”the most basic motivation is to do useful work” [109, p. 761]. Thus, greedy or dis-honest behaviour will not occur, and self-interested actions will only be seen if this is due to limited resources that impact the quality of work (e.g. low battery levels might result in refusing to take on new tasks) [109].

Auction-based approaches are widely studied and implemented in the field of MRTA [81, 86, 108]. The reason for this is that they are scalable to large networks, while remaining computationally efficient [76, 81, 86, 109, 111]. This is mainly due to the local characteristic of auction-based approaches: by relying on local information and/or self-interest of agents, efficient solutions for large scale problems can be found [112]. Furthermore, auctions are especially suitable for online problems where new tasks arrive continuously to the system [76, 81, 113] and auctions allow for flexibility in team objectives, agent characteristics and specifics of the environment [81]au.

Several types of auctions can be distinguished. When we consider auctions with a central auctioneer, a popular approach is the class of Sequential Single-Item (SSI) auctions, in which agents submit bids for all unallocated tasks, but only one task is assigned in each bidding round [114]. SSI auctions combine the advantages of combinatorial auctions (every agent bids on bundles of tasks, taking synergies between tasks into account) and parallel auctions (tasks are allocated simultaneously in single-round auctions). As a result, SSI auctions allow for more optimal solutions (taking synergies between tasks into account resulting in a smaller sum of travel distances) that are computationally inexpensive (small number of bids and fast winner determination) [114, 115, 116]. Furthermore, it can be analytically proven that when using SSI auctions, the sum of travel distances can be a factor 1.5 larger than the optimal solution and at most a factor 2 larger. This holds for both exact determination of the travel distance or heuristics that approximate the travel distance, allowing for obtaining solutions in polynomial-time [114].

Various improvements have been suggested in the literature regarding the use of SSI [114], as well as an elaboration of the SSI algorithm taking temporal constraints (time windows for execution of tasks) into account, known as the Temporal Sequential Single-Item (TeSSI) algorithm [117]. Another variation are the so-called Sequential Single-Cluster (SSC) auctions, where tasks that are geographically located close to each other are bundled into clusters that agents bid on. By bundling the tasks, the number of required bids and communication overhead reduces, which improves scalability of the algorithm [118].

Regarding consensus-based auctions where no central auctioneer is present and tasks are allocated based on predefined agreements, the Consensus-Based Bundle Algorithm (CBBA) is a promising approach, providing for a 50% optimality guarantee [119]. In the past years, several improvements and extensions have been developed, including the handling of time-sensitive tasks [86], the implementation in highly dynamic environments [116] and reallocation of new tasks arriving to the system [120].

Game Theoretic-Based Approaches

When tackling the MRTA problem using game theory, the interaction between agents is inspired on a game, where all agents aim to maximize their own utility based on knowledge of behaviour and characteristics of other agents and the environment [86]. Although similarly to auctions, a competitive mechanism such as a game can be used to maximize entire system performance, enforcing cooperation between agents in game theoretic-based approaches is much more difficult than for auction-based approaches. The reason for this is that the formulation of an objective on the level of each individual agent which ensures a global optimum, is a challenging task [116].

However, some studies have investigated how to apply game theory in a cooperative setting. An example is the work done by Arslan, Marden and Shamma [121], in which a set of autonomous vehicles is expected to optimally assign themselves to a set of tasks (from a global perspective). Different negotiation mechanisms are evaluated, including Action-Based Fictitious Play (ABFP), Utility-Based Fictitious Play (UBFP) and Regret Matching (RM). Although these negotiation mechanisms guarantee

convergence towards a stable and conflict-free solution (also called a pure strategy Nash equilibrium in the field of game theory), the obtained solution is in most cases far from optimal [86, 121]. Empirically, the negotiation mechanism based on Selective Spatial Adaptive Play (sSAP) shows near-optimal behaviour with ”arbitrarily high probability” [121, p. 592]. However, for convergence towards a stable solution in a large-scale problem, almost 1500 iterations are required, resulting in significant computation times that do not allow for online problems. In addition, the sSAP algorithm only solves IA problems and do not consider target assignment over a time horizon, making the approach unsuitable for online problems [121].

In more recent research, some promising studies implement game-theoretic approaches to the task allocation problem. In the work done by Cui, Guo and Gao [122], an initial task assignment is computed using an auction-based approach, followed by the possibility of task reallocation using game theory. The results show that for a fixed number of robots (10 agents) and increasing number of tasks, running times of the algorithm increase. However, up to 100 targets, the computational time of the algorithm remains below 1 second. This is achieved by only allowing up to three robots to participate in one negotiation procedure. Cost reduction with respect to the initial solution is achieved, however, no bounds on solution quality are provided.

Similar results regarding computational performance are achieved in the work of Smyrnakis, Gu and Veres [123]. For varying robot team sizes (ranging from 10 to 50 robots), running times of the algorithm stay below the 0.1 second. However, the study only examines performance of problems in the ST-SR-IA class and do not consider online arrival of new tasks to the system. Although strict online problems tackled by a game-theoretic approach could not be found in the literature, several studies consider dynamic environments, such as changing workloads of tasks over time. Examples of this are task reallocation in a multi-stage setting [124] or task reallocation based on growing workload of tasks over time [125]. Both studies show that game-theoretic remain computationally efficient in dynamic environments and are scalable to situations similar to towing of aircraft by TaxiBots. In addition, in the work done by Wu and Shang [124], an upper (optimal) and lower bound (2-competitive) is provided for the end result of the algorithm based on game theory.

5.2.3. Comparative Evaluation of Solution Techniques

Based on the review of task allocation approaches described in subsection 5.2.1 and subsection 5.2.2, a trade-off is done to select suitable approaches for the assignment of TaxiBots to outbound flights. Based on the concept of operations and system requirements described in chapter 4 and subsection 4.6.2 respectively, a set of trade-off criteria has been determined to evaluate the different allocation approaches on. Scoring will be done using a range from 1 to 3 (worst to best).

- **Solution quality**: this criteria concerns the degree of optimality of the solution produced by the approach under consideration. A score of 3 is given when the approach guarantees to provide the optimal solution, a score of 2 is given when a bounded (suboptimal) solution is produced and a score of 1 corresponds with solutions that are unbounded (corresponds with system requirement item 3).

- **Scalability / Computational efficiency**: when an approach remains efficient for increasing number of agents and tasks, a higher score on scalability is rewarded. This trade-off criteria relates to system requirement item 2. Computational efficiency is measured in terms of run time of the algorithm and is related to the real-time requirement of the system (item 6). According to Ma et al. [88], real-time corresponds with computational times less than 1 second. Approaches that in general provide solutions within this time frame are rewarded a score of 3. When approaches are able to provide results within 30 seconds, a score of 2 is given. Approaches with larger computational times are rewarded a score of 1.
  Due to the fact that scalability and computational efficiency are very much related to each other, both scores are averaged into a single score (equally weighted). The notation in Table 5.1 is as follows: average score (score on ”Scalability”, score on ”Computational efficiency”).

- **Flexibility**: this criteria measures the flexibility of an approach to adapt to varying team objectives, agent characteristics or constraints originated from the environment.

- **Complexity**: in order to reflect the system requirement related to the limited available time and

resources in this project (item 8), a criteria related to complexity is introduced and measures the complexity of implementing the approach under consideration and building upon previous work.

- **Applicability**: depending on the degree of applicability of the approach to the task allocation problem of TaxiBots, a score is given. Applicability concerns to what extent approaches have been tested on large-scale, real-world applications involving cooperative task assignment. Limited simulation testing in few application areas scores a 1, extensive simulation testing in multiple application areas scores a 2 and a score of 3 is awarded when approaches are not only extensively tested in a simulation environment, but also tested on real hardware. This criteria can be linked to the objective of the model as formulated in system requirement item 3.
- **Suitability for online use**: regarding the suitability of approaches to apply to problems in an online setting, a score of 1 is given when no/very limited studies are found that use the approach under consideration in an online setting. When the approach is applied occasionally to online problems or in dynamic environments (for example, requiring task reallocation), a score of 2 is awarded. Finally, when the approach is widely applied to online problems, the highest score of 3 is given. This criteria is based on the system requirement related to real-life operations (item 2) and real-time execution (item 6).

In Table 5.1, the resulting scores for all approaches on each criteria is presented. Regarding solution quality, only deterministic approaches guarantee to provide the optimal solution to a problem, hence, a score of 3 is given. For all approaches except stochastic approaches, bounds on the optimality of solutions can be analytically proven.

When it comes to scalability and computational efficiency, it is clear that deterministic approaches are not suitable for this domain due to the exponentially increasing numbers of variables when increasing the number of agents and tasks. Both auctions and game-theoretic approaches are very suitable to apply in larger problems and are still capable of providing solutions within 1 second based on reviewing the literature. For greedy approaches, the number of variables generally also increases with increasing problem-size, however, they are known for their computational efficiency. With respect to stochastic approaches, they are especially suitable for large-scale problems, but in general provide solutions for problems with a scale of 3-7 robots and 10-30 tasks in a multiple of ten seconds, therefore, scoring a 2 on computational efficiency.

Concerning flexibility of the selected approaches, it can be noted that all optimization-based approaches are equally flexible. This is due to the fact that both objective functions and constraints can be altered relatively easy to reflect agent characteristics and environmental/task constraints. Auctions share this characteristic as well. For game-theoretic approaches in a cooperative setting, it is a challenging task to align individual utility functions for agents with the global team objective. Therefore, flexibility of game-theoretic approaches receives a low score.

When it comes to complexity, both greedy approaches and auction approaches are scored the highest, due to their intuitive concept that can be easily translated to the TaxiBot situation. For deterministic and stochastic approaches, the concept may be slightly more complex, however, significant literature is available to be able to grasp the approach within the available time and resources. Although game-theoretic approaches have also been widely discussed in literature, the concept of deriving at Nash equilibria in a cooperative setting is significantly more complex than the other approaches under consideration.

Concerning applicability of the approaches to the problem under consideration, it can be seen that all approaches except for game-theoretic approaches are awarded a score of 3. The reason for this is that all approaches have been extensively tested in simulation environment in multiple application areas and large-scale problems. However, the application of game-theoretic approaches to large-scale cooperative problems is limited and therefore, a score of 1 is given.

Finally, the approaches are evaluated on their suitability for online use. Notably, both auction-based approaches and greedy approaches are scored highest, due to the fact that they are widely applied to (and especially developed for) problems where new tasks arrive to the system in an unknown manner. Some examples of research using stochastic and game-theoretic approaches also make use of problems in a time-extended setting or dynamic environments, but to a far lesser extent than the two previous ones. Finally, no research has been found that applies deterministic approaches to an online setting. This can be explained due to the fact that ST-SR-TA type of problems are classified as NP-hard, meaning that an optimal solution cannot be found within polynomial time [73].

As can be seen from Table 5.1, both greedy and auction-based approaches turn out to be the most suitable approaches when it comes to task assignment in the context of TaxiBot deployment at AAS. Therefore, in the remainder of this chapter, a selection of specific greedy and auction-based approaches will be reviewed in detail.

Table 5.1: Scoring on trade-off criteria for different approaches on task allocation

	Optimization-Based Approaches			**Market-Based Approaches**	
	Deterministic Approaches	Greedy Approaches	Stochastic Approaches	Auction-Based Approaches	Game Theoretic-Based Approaches
Solution quality	3	2	1	2	2
Scalability / Computational efficiency	1 (1, 1)	2.5 (2, 3)	2.5 (3, 2)	3 (3, 3)	3 (3, 3)
Flexibility	3	3	3	3	1
Complexity	2	3	2	3	1
Applicability	3	3	3	3	1
Suitability for online use	1	3	2	3	2
Total	13	16.5	13.5	17	10

5.3. Further Elaboration on Greedy Approaches

In this section, more details will be provided on three specific greedy algorithms: the previously mentioned Broadcast of Local Eligibility (BLE) and MURDOCH algorithm, as well as a greedy algorithm specifically taking temporal constraints into account.

5.3.1. Broadcast of Local Eligibility (BLE) Algorithm

When tasks arrive in an online fashion to the system and the system allows for reallocation of previously assigned robots, the ST-SR-TA type of problem reduces to an iterated instance of ST-SR-IA [73]. This instance can be solved by the use of the Broadcast of Local Eligibility (BLE) algorithm which is described in the work done by Werger and Mataríc [87]. In essence, this algorithm (re)calculates all robot-task utilities when at least one robot becomes unassigned, by considering all (assigned and unassigned) tasks. For the robot-task pair that results in the highest utility, task t is assigned to robot r until no robot remains unassigned.

The algorithm is known to be 2-competitive for the optimal solution of the OAP [73].

5.3.2. MURDOCH Algorithm

Contrary to the BLE algorithm, the MURDOCH algorithm is useful when robots that are previously assigned to a task cannot be reassigned [73]. The MURDOCH algorithm originally was proposed for problems of the class ST-SR-IA, in which newly arriving tasks to the system have to be assigned and is based on the original work as proposed by Gerkey and Matarić [109]. For problems of the class ST-SR-TA, it is proposed by the same authors in later work [73] to initially assign all robots to tasks in an optimal way, using for example the Hungarian method. All remaining tasks are assigned in an online fashion to the robot that is most fit (according to some predefined metrics) and currently available. Although this algorithm strictly is an auction-based method [109] (newly arriving tasks are allocated using a basic auction scheme based on the CNP protocol), it is placed under the greedy approaches since it is initiated using a deterministic method followed by a greedy way of assigning all remaining and newly arriving tasks. The altered version of the MURDOCH algorithm is known to be 3-competitive [73].

5.3.3. Flexibility-Based Task Assignment

In a very brief paper, Wu et al. [126] propose a framework that integrates task assignment with path planning for offline problems. Although this framework will be discussed in more detail in chapter 7, the notion of how the authors propose to determine the next task to be assigned will be discussed

in this section. In their problem, the tasks in the set of tasks $\mathcal{T}$ are characterized by a deadline d_i, thus, explicitly taking temporal constraints into account. The goal is to maximize the number of tasks completed before the associated deadline.

In order to take the deadline of a task into account in the assignment phase, Wu et al. [126] introduce a measure called flexibility f_i, which is defined according to Equation 5.2 where $c_{k,i}$ represents the earliest timestep that any agent $r_k, r_k \in \mathcal{R}$ can complete the unassigned task t_i. Thus, the flexibility f_i represents the margin that is available for execution of a task with respect to its deadline. In the set of unassigned tasks with non-negative flexibility values, the next task i^* to be assigned is the task with the least value f_i: thus, the most urgent task is assigned (Equation 5.3, [126]).

$$f_i = d_i - \min_{r_k \in \mathcal{R}} c_{k,i} \tag{5.2}$$

$$i^* = \underset{f_i \geq 0}{\operatorname{argmin}}\, f_i \tag{5.3}$$

In section 7.3, a more elaborate description is provided on how the most suitable agent k^* is found to fulfill the task i^*, including results on the success rate of the algorithm (i.e. the number of tasks completed before their deadline). Unfortunately, no other results in terms of optimality were provided. However, due to the fact that this greedy algorithm explicitly takes deadlines of tasks into account, makes it a suitable candidate to apply to the problem of allocating TaxiBots at AAS.

5.3.4. Concluding Remarks on Greedy Task Allocation

In the previous sections, three greedy approaches have been reviewed that can be used for task allocation in the MAS to be designed. In this section, a qualitative reasoning will be provided based on the trade-off criteria presented in subsection 5.2.3 to choose the most suitable greedy technique for allocation of TaxiBots to outbound flights at AAS.

As will be explained in section 8.3, in a first version of the model to be designed, no reallocation of tasks will be considered. Therefore, the BLE algorithm is not suitable to use for implementation in a baseline model, since it explicitly requires reallocation of tasks to be possible. Due to the fact that the MURDOCH algorithm provides for a boundary on suboptimality of the solution and is specifically designed for dealing with online problems [73, 109], the framework of the MURDOCH algorithm is preferred over the offline algorithm as presented in subsection 5.3.3. However, it is noted that using flexibility as a measure to determine the most-fit robot seems to be promising, especially since general bidding rules and more specifically, rules on how to deal with temporal constraints, are not defined in the original paper introducing the MURDOCH algorithm [109]. Therefore, based on the findings in literature, the following greedy approach in task allocation is proposed: (1) define an initial, optimal allocation of TaxiBots to aircraft by means of the Hungarian method; (2) for newly arriving flights to the system, use the measure of flexibility to determine path costs for free TaxiBots; (3) assign the most-fit robot to the most urgent task (read: aircraft with earliest TSAT time); (4) repeat steps 2 and 3 until no aircraft remain unassigned.

5.4. Further Elaboration on Auction-Based Approaches

In this section, more details will be provided on three specific types of auction algorithms: Sequential Single-Item (SSI) auctions, the SSI version with temporal constraints included (TeSSI) and Sequential Single-Cluster (SSC) auctions. Note that consensus-based auctions are not considered in more detail based on recommendations by Chen in previous related work on the automation of ground handling at airports [72]. It was pointed out that using multiple local auctions in ground handling may lead to inconsistent situational awareness, often leading to solutions far from optimal. Since common situational awareness is of a prime concern in the context of aircraft engine-off towing operations as well, it was chosen not to consider auctions that have a distributed nature in more detail. In addition, as will become apparent throughout the next section, it was shown by previous studies that TeSSI outperforms consensus-based approaches significantly in run time performance, which is another primary concern in this research.

5.4.1. Sequential Single-Item (SSI) Auctions

In Sequential Single-Item (SSI) auctions, all tasks are initially unallocated. Tasks are assigned based on the bids of robot for each task. In every auction round, every robot bids on each unallocated task. The central auctioneer allocates the task to the robot with the best bid. After winner determination, the procedure repeats itself until all tasks are allocated [115].

The first study to formalize the type of SSI auctions, is the work done by Lagoudakis et al. [127]. In their work, they specify a multi-robot routing problem by a set of robots $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ and a set of targets $\mathcal{T} = \{t_1, t_2, ..., t_m\}$. A strictly positive and symmetric [2] cost function $c(i, j)$ is assumed, denoting the costs to travel from location i to j. Furthermore, the Robot Path Cost (RPC) associated with robot r concern the costs related to the total path travelled by the robot, including all allocated targets. The Target Path Cost (TPC) of target t relate to the increase in total path costs for robot r to visit target t from the initial location of robot r [127].

The goal of a multi-robot routing problem is to assign robots to tasks in such a way that all tasks are fulfilled, while taking the team objective into account. In the work of Lagoudakis et al., three different team objectives are considered [127]. When considering $\mathcal{A} = \{A_1, A_2, ..., A_z\}$ as a partition of the set of targets, where A_i is the set of targets allocated to robot r_i, the team objectives can be formally defined as follows:

- MINISUM : $\min_{\mathcal{A}} \sum_j RPC(r_j, A_j)$. The objective is to minimize the sum of all path costs associated with all robots.
- MINIMAX : $\min_{\mathcal{A}} \max_j RPC(r_j, A_j)$. The objective is to minimize the sum of the path cost for the robot with the largest costs.
- MINIAVE : $\min_{\mathcal{A}} \frac{1}{m} \sum_j CTPC(r_j, A_j)$, where the Cumulative Target Path Cost (CTPC) are the costs associated with visiting all targets in A_i for robot r_i from its current position. This leads to the objective to minimize the average TPC for all robots.

Based on the previously mentioned team objectives, a general bidding rule can be derived and formulated as follows:

> Robot r bids on unallocated target t the difference in performance for the given team objective between the current allocation of targets to robots and the allocation that results from the current one if robot r is additionally allocated target t [127].

Using this general bidding rule, a set of specific bidding rules corresponding with the three team objectives can be formulated. Let $\mathcal{S} = \{S_1, S_2, ..., S_n\}$ be the current partial allocation of tasks t to the robots, then the bidding rules can be defined as follows [127]:

- BIDSUMPATH : $RPC(r_i, S_i \cup \{t\}) - RPC(r_i, S_i)$. According to this bidding rule, robot r_i should bid the increase in path cost when adding task t to its partition S_i.
- BIDMAXPATH : $RPC(r_i, S_i \cup \{t\})$. According to this bidding rule, robot r_i should bid its total path cost associated to its current allocation including the addition of task t.
- BIDAVEPATH : $CTPC(r_i, S_i \cup \{t\}) - CTPC(r_i, S_i)$. According to this bidding rule, robot r_i should bid the increase in cumulative task path costs when adding task t to its partition S_i.

In order to find the RPC and CTPC, robot r_i has to find the optimal path through all target in S_i, which corresponds with an instance of the TSP. Since this is an NP-hard problem, Lagoudakis et al. [127] propose to use the cheapest insertion heuristic that finds the cheapest position to insert task t in the existing path in terms of path costs. Assuming that the approximation of path costs is no worse than the cheapest insertion heuristic, the performance ratios as summarized in Table 5.2 for various team objectives and bidding rules hold [127]. The performance ratios are expressed as an upper and lower bound, meaning that the resulting factor is a maximum and minimum factor respectively away from the optimal solution.

Despite the performance guarantees of SSI auctions, no guarantees on completeness can be provided. This is due to the fact that not all synergies between tasks are taken into account because of auctioning tasks in multiple rounds [81].

In the past years, several adjustments have been made to SSI auctions to improve the solution quality while still maintaining computational efficiency needed for online task allocation problems. According to the work of Rizzo [81], the most important improvements are the inclusion of rollouts and making use of the K-swaps procedure. Both phenomena will be elaborated on below.

[2] A symmetric cost function means that travel costs for both directions along the same edge are equal.

Table 5.2: Summary of performance bounds for different team objectives and bidding rules for n robots and m targets. The bounds are presented as a ratio of rule performance over optimal performance [127].

Bidding Rule	**Team Objective**					
	MINISUM		MINIMAX		MINIAVE	
	Lower	Upper	Lower	Upper	Lower	Upper
BIDSUMPATH	1.5	2	n	$2n$	$\frac{m+1}{2}$	$2m$
BIDMAXPATH	n	$2n$	$\frac{n+1}{2}$	$2n$	$\Omega(m^{1/3})$	$2m$
BIDAVEPATH	m	$2m^2$	$\frac{n+1}{2}$	$2m^2n$	$\Omega(m^{1/3})$	$2m^2$

SSI with Rollouts

In basic SSI auctions, each robot evaluates the effect of inserting task t in its current partial allocation, without considering the allocation of remaining tasks. In the work done by Zheng, Koenig and Tovey [128], it is suggested to make use of so-called rollouts: instead of using the team costs resulting from the partial assignment of target t, the team costs corresponding with the *complete* allocation are used to evaluate the partial allocations of tasks. As a result, robot r_i uses the costs associated with the complete allocation (of all m targets) that would follow if it was assigned task t to calculate its bid. A downside of this technique is that several sets of SSI auctions need to be run instead of just one, increasing computational times. Therefore, it is suggested to only use rollouts in the first few rounds of SSI auctions when large amounts of targets are still to be allocated. Experiments done by the authors show that using rollouts only in the first three rounds already improves the solution significantly, while computation times are still in the order of tens of seconds (especially for the combination of large number of robots and small number of tasks) [128].

The K-Swaps Procedure

In order to capture synergies between tasks that are already allocated and tasks that are on auction in a later round [81], the possibility of swapping tasks is introduced by Zhen and Koenig [129]. In their work, they introduce the set $\{T_r\}_{r\in\mathcal{R}}$, representing the solution before any task exchanges and $\{T_r^{'}\}_{r\in\mathcal{R}}$ as the solution after task exchanges. Three different types of exchanges can be formalized [129]:

- **Out Swap**: agent r_k transfers task $t \in T_r$ to some other agent $r_k^{'}$, written as $(r_k, -, t, -)$.
- **In Swap**: agent r_k receives task $t^{'} \notin T_r$ from some other agent $r_k^{'}$, written as $(r_k, -, -, t^{'})$.
- **Exchange Swap**: agent r_k and agent $r_k^{'}$ exchange task $t \in T_r$ from agent r_k to agent $r_k^{'}$ and task $t^{'} \in T_{r'}$ from agent $r_k^{'}$ to agent r_k, formalized as $(r_k, r_k^{'}, t, t^{'})$.

When an in and out swap can be combined into an exchange swap, the pair of in and out swaps is called a resolvable pair. The exchanges of tasks among agents are described using the variable s^k, denoting a partial k-swap for agents $\mathcal{R}(s^k) \subseteq \mathcal{R}$. The partial k-swap contains a set of out swaps (tasks transferred from agents in $\mathcal{R}(s^k)$ to agents not in $\mathcal{R}(s^k)$); a set of in swaps (tasks received from agents not in $\mathcal{R}(s^k)$ to agents in $\mathcal{R}(s^k)$); and a set of compact exchange swaps (tasks exchanged between agents that are both in $\mathcal{R}(s^k)$). A partial k-swap is said to be complete if and only if it contains only compact exchange swaps. Note that the value k is the number of exchange swaps present in s^k.

In their work, Zhen and Koenig [129] present both a centralized and distributed approach to executing the K-swaps procedure. In the centralized approach, a central planner constructs all profitable complete k-swaps for $1 \leq k \leq K$ for a given solution, where K is a user-defined value.

1. The set $\mathcal{P}$ of all profitable k-swaps for $1 \leq k \leq K$ is initialized to empty, as well as the set $\mathcal{S}_{planner}$, containing all the partial k-swaps that are constructed.
2. Each agent $r \in \mathcal{R}$ constructs all possible swaps that only contain itself bounded by K, meaning that the so-called partial zero-swap consists of at most K in swaps, at most K out swaps, no exchange swaps and at least one in or out swap. All possible partial zero-swaps are then send to the central planner.

3. In K rounds, the central planner adds all received partial zero-swaps to $\mathcal{S}_{planner}$ and:
   (a) Creates partial k-swaps by combining a partial g-swap s^g and partial h-swap s^h, if and only if s^g and s^h form a combinable pair.
      - If the resulting pair s^k is profitable and complete, it is added to the set $\mathcal{P}$.
      - If s^k is not profitable, but bounded by K, it is added to the set $\mathcal{S}_{planner}$.

As noted by the authors, the centralized approach might result in a communication or computation bottleneck. Therefore, a distributed approach was introduced. The main difference between the distributed and the aforementioned centralized approach, is that the agents themselves decide whether a partial zero-swap that only contains itself is profitable and subsequently, only send the profitable partial zero-swaps to other agents. Based on a predefined order among the agents, global partial k-swaps are found using the own partial zero-swaps combined with partial zero-swaps received from other agents [129].

Both a greedy and roll-out approach are introduced to implement the above procedures in sequential auctions. In the greedy approach, the distributed algorithm is used to determine all profitable complete k-swaps and then selects the swap that results in the highest gain for the system. When using the roll-out approach, all profitable complete k-swaps are again found using the distributed algorithm, but instead of choosing the swap with the highest gain, they are all hypothetically executed. Then, the greedy approach is hypothetically executed on this hypothetical solution, after which the swap is chosen that resulted in the smallest overall team cost.

When comparing the greedy and roll-out approaches with each other on a 51 x 51 size grid representing an office environment, it was concluded by the authors that both greedy and roll-out approaches significantly contribute in reducing the overall solution cost. For larger values of K, the team cost was reduced more but at the expense of more required run times. Especially the greedy approach looks promising for the application of TaxiBotting, since this algorithm is capable of improving the initial solution cost by more than half (from 1070.5 to 443.3) in only 0.01 seconds for $K = 1$ [129].

5.4.2. Temporal Sequential Single-Item (TeSSI) Auctions

When tasks need to be completed in a specific time window, temporal constraints are introduced in the assignment problem. In the work of Nunes and Gini [117], SSI auctions are extended to include temporal constraints, resulting in the Temporal Sequential Single-Item (TeSSI) auction algorithm. TeSSI distinguishes itself from previous work on including temporal constraints in auctions because of its capabilities to handle overlapping time windows and changing starting times of tasks [81].

If we consider a set of robots $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ and a set of targets $\mathcal{T} = \{t_1, t_2, ..., t_m\}$, all $t \in \mathcal{T}$ have an earliest starting time (ES_t), latest starting time (LS_t) and a duration (DUR_t). Based on the combination of the earliest or latest starting time and duration, both an earliest and latest finish time can be determined (EF_t and LF_t respectively). Similarly as for SSI auctions, agents submit bids for all tasks available. Subsequently, the task is allocated to the agent with the best bid and the bidding continues until the set of tasks $\mathcal{T}$ is empty. Tasks that are not executable for any robot are added to the set of unallocated tasks ($\mathcal{T}_{unalloc}$), guaranteeing the termination of TeSSI (since the set $\mathcal{T}$ will be empty at some point).

In order for robots to determine whether their current schedule allows them to execute task t on auction, each robot stores its task schedule as a Simple Temporal Network (STN). The STN consists of time points related to the start and finish time of tasks, an origin time point as a reference value (assigned zero) and two types of constraints between pairs of time points: duration constraints and travel time constraints. Duration constraints are applicable to time points that correspond to the same task and ensure that the start of a task does not occur after its finish time: $(F_t - S_t \in [DUR_t, \infty))$. Travel time constraints are imposed when one time point corresponds with the end of a task and the other time point with the start of a task. This type of constraints ensure that an agent can only start the next task t after having finished its previous task t': $(S_t - F_{t'} \in [TT_{t',t}, \infty))$. An example of a STN with three tasks and associated duration and travel time constraints is shown in Figure 5.2.
For every task t on auction, all agents try to insert the task into their STN by propagating it using the Floyd-Warshall algorithm without making it inconsistent, meaning that no negative cycles occur [117]. When multiple insertion options exist, the resulting objective value for each position is calculated and the one that minimizes the overall objective is returned. The individual bid of each agent on task t is

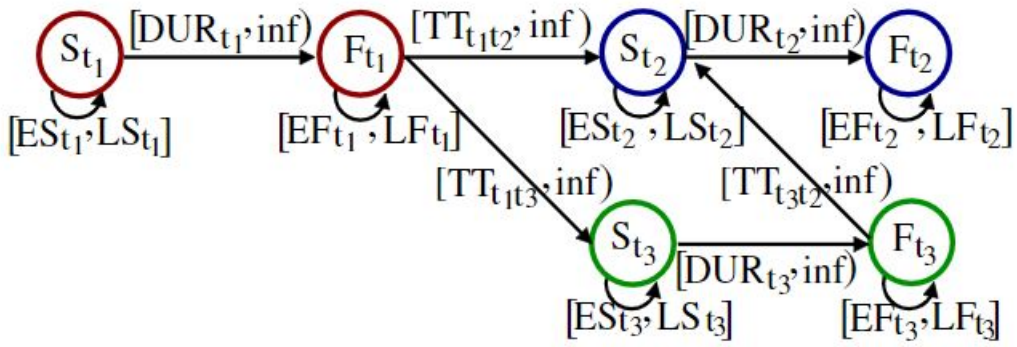


Figure 5.2: Example of a STN with three tasks and associated duration and travel time constraints. The reference value of zero is omitted from the figure [117].

based on the lowest objective value. Similarly as for SSI auctions, the task t^* among all tasks that yields the lowest bid is awarded to agent r^*. Next, the procedure is repeated for the set of tasks excluding task t^*. Note that no agent except for agent $r*$ need to recalculate their bidding, since their schedules have not changed.

The solutions generated by TeSSI are not guaranteed to be complete or optimal, due to similar reasons as why SSI auctions cannot provide any guarantees on completeness or optimality [117]. However, in the work of Nunes and Gini [117], it is shown that the TeSSI algorithm consistently outperforms both a greedy algorithm and a version of a consensus-based algorithm that handles time windows in terms of number of tasks allocated and number of robots used. This was shown by conducting several experiments, one in which 100 tasks arrive in a dynamic fashion to the system and have to be fulfilled by a team of 10 agents. Furthermore, it is shown that the computation time of TeSSI is competitive with that of the greedy algorithm, and two orders of magnitude smaller than that of CBBA (0.43 seconds versus 98.9 seconds respectively) [117].

Probabilistic Temporal Sequential Single-Item (pTeSSI) Auction

In order to deal with uncertain task durations, Rizzo [81] introduced the Probabilistic Temporal Sequential Single-Item (pTeSSI) algorithm, based on the original TeSSI algorithm. For each task t, agent r_k calculates the minimum distance and time needed to complete the task. However, the duration of the task is not given as a constant, but as a random uniformly distributed variable, with upper and lower bounds known beforehand. Note that Rizzo assumed a uniformly distributed duration, but that any kind of distribution can be implemented in the algorithm.

Similarly as for TeSSI, the agent tries to fit a task t on auction into its temporal network, called Simple Temporal Network with Uncertainty (STNU). A valid insertion point satisfies the constraint that the earliest delivery time of the new is no larger than the latest delivery time of the subsequent task. Based on the insertion position, the risk of not being able to execute the new schedule is determined. If this risk remains below a certain agent threshold, the agent calculates its bid for the task at the specific insertion point. After having evaluated all possible insertion points, the lowest bid is submitted. On a system level, the probability that the resulting schedules will not be dispatched successfully is minimized, next to the original MINIMAX or MINISUM objectives [81].

In the evaluation of pTeSSI compared with TeSSI, it was shown that pTeSSI is a more efficient and effective task allocation algorithm when dealing with uncertain task duration. The pTeSSI not only allocated more tasks, but also runs significantly faster than the TeSSI algorithm. This is a surprising result, since the worst-case complexity was shown to be equal for both algorithms [81]. Especially the latter characteristic makes the pTeSSI suitable to apply to the TaxiBot problem, since real-time operations require fast enough running times of the algorithm.

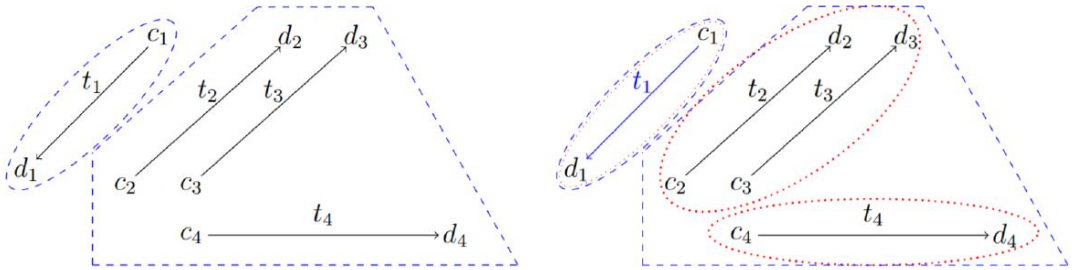
5.4.3. Sequential Single-Cluster (SSC) Auctions

As mentioned earlier, when using SSI auctions (or variations thereof such as the TeSSI algorithm), some synergies among tasks are taken into account, but not all of them. As a result, the associated team costs when using SSI auctions are in general higher than when all tasks would be allocated in one round, as is the case in full combinatorial auctions. These types of auctions allow agents to take

all synergies among tasks into account resulting in an optimal solution. However, due to the increased computational complexity and higher run times, full combinatorial auctions are not suitable to use in an online task allocation problem that requires real-time solutions. In order to take more synergies between tasks into account when compared with basic SSI auctions without increasing computational efficiency, Koenig et al. [130] proposed to assign $k > 1$ additional tasks among agents in each round during a SSI auction. In this altered version which is called a Sequential Bundle-Bid (SBB) auction, agents bid on bundles of at most k tasks. Although experiments show that in general, SBB provide for reduced computational and communication costs by several orders of magnitude compared with fully combinatorial auctions [130], SBB auctions cannot be easily implemented in online problems where tasks arrive dynamically to the system [81].

Another approach that makes use of bundling of tasks, is an extension of the SBB auction introduced by Heap and Pagnucco [118], called Sequential Single-Cluster (SSC) auctions. This type of auction is specifically designed for dynamically appearing tasks, making it especially useful for task allocation problems in an online setting. Similarly as for SBB auctions, individual agents calculate their bidding costs for each unassigned cluster of tasks and submits its lowest bid for any one cluster. Bidding costs are based on the path costs for each robot to incorporate the tasks in cluster C in its current allocation of tasks, eventually making use of a cheapest insertion heuristic. After each bidding round, a previously unassigned cluster C is assigned to the robot that results in the lowest increase of overall team costs. After having allocated all clusters, each robot finds the path that minimizes the distance travelled to complete all allocated tasks [118].

Before the auction begins, all tasks are assigned to one cluster. Clustering of tasks is based on the associated pickup and delivery locations of tasks. First, tasks that have geographically closely positioned pickup locations are clustered, in which new clusters based on delivery locations are formed. An example of this way of clustering is shown in Figure 5.3. Although this type of clustering is shown to be effective by Heap and Pagnucco in an environment where pickup and delivery locations are scattered, it is questionable if clustering based on these characteristics will have any benefit in an airport environment where pickup locations are relatively clustered (apron area) and delivery locations widely spread over the environment (decoupling locations near runways). In addition, as already mentioned by Rizzo [81], when considering pickup and delivery problems with deadlines, an additional temporal layer in which tasks are clustered according to their deadlines has to be added to make it suitable for our problem.



(a) Clustering based on pickup locations

(b) Clustering based on delivery locations

Figure 5.3: Example of clustering in a SSC auction as proposed by Heap and Pagnucco [118]. In Figure 5.3a, first clustering based on pickup locations is done. Subsequently, as shown in Figure 5.3b, new clusters within the existing clusters are formed based on delivery locations. Taken from [81]

When it comes to dynamically arriving tasks to , a new task t is instantaneously assigned to a robot $r_i \in \mathcal{R}$. This robot can be chosen randomly or based on agent characteristics, such as the robot that has completed the least amount of tasks to distribute workload. After this assignment, the robot has two options: local replanning of its path to incorporate the new task t in its route; or initiate a repeated auction to globally reallocate the task to another robot. When global reallocation is chosen, all agents cluster their uncompleted tasks, excluding the tasks that are being executed at the moment of auctioning. In the work of Heap and Pagnucco [118], the auction is then executed in a fully distributed manner, meaning that each individual agent sends its bids for all available clusters to all other agents. Based on a globally aligned winner determination strategy, each individual agent determines for each cluster the winner independently. Variations to a fully distributed auction could be that the agent that initiates the global reallocation, acts as a central auctioneer, reducing communication and computation

overhead.

In their work, Heap and Pagnucco [118] studied the performance of the SSC auctions in an office-like environment with 10 robots and 60 tasks, both for the MINISUM and MINIMAX objective. When comparing local replanning with global reallocation strategies in terms of team costs and computation time for robots with a capacity of one, it is shown that global reallocation of tasks results in lower team costs especially for the MINIMAX objective (ranging between 18.7 and 36.5%), whereas marginal differences for the MINISUM objective are observed, in favor of global reallocation (ranging from 4.2 to 18.5%). However, this comes at a cost when comparing computational times for both strategies. In the case where 75% of tasks is unknown upfront, local replanning results in a mean overall cumulative task allocation computation time of 41 seconds, whereas global reallocation almost requires triple of the time (119 seconds) [118].

5.4.4. Concluding Remarks on Auction-Based Task Allocation

In this section, the three auction-based algorithms that are investigated in the previous sections, will be reviewed on suitability to apply to the problem under consideration in this thesis. Note that the use of auction-based approaches for the allocation of TaxiBots to outbound flights in an online setting is assumed to be proven in subsection 5.2.3, and that this holds for all three algorithms. Thus, the qualitative reasoning in this section will focus on specific aspects of the problem that is expected to be dealt differently with in the three auction algorithms.

First of all, the assignment of TaxiBots to aircraft will include deadlines: as mentioned in section 4.6, the TSAT based on historic flight schedules will be used as the time that an aircraft *has* to leave the gate in order to be able to depart in its departure slot at the assigned runway. From the selection of auction algorithms, TeSSI is the only algorithm that already has a temporal layer included.

In a first version of the model, no disruptions or unexpected events will be considered, as described in chapter 4. However, one possible extension that is considered is the addition of uncertainty and delay in the model (section 8.3). Similarly as for temporal constraints, the TeSSI algorithm is the only auction algorithm that already has been altered to deal with uncertainty [81]. This is another reason for the preference to use TeSSI as an allocation algorithm in the baseline model already.

In addition, when compared with SSC auctions, it is noted that computational times are significantly smaller for TeSSI. Furthermore, the relevance of using clustered tasks based on their pickup and delivery location is questioned for the specific problem under consideration. The reason for this is that all pickup locations (gates) are geographically located closely together, whereas the delivery locations (decoupling locations near runways) are relatively dispersed. Since it is a given that TaxiBots will always return from the runway to the apron area for their next mission, it is expected that clustering of tasks will not be of added value. If inbound towing would be considered as well, clustering *is* expected to improve the efficiency of the overall system. However, since this is not the case and only outbound towing is considered, TeSSI is preferred to use as an initial task allocation mechanism over a variant of a SSC auction.

6 Multi-Agent Motion Planning

Having considered the allocation of TaxiBots to aircraft when implementing outbound towing in chapter 5, the next step is to focus on motion planning of the vehicles in the Multi-Agent System (MAS). In this chapter, the planning of agents' paths to execute the tasks they have been assigned is considered.

First, the concept of path finding in general will be elaborated on, introducing the Multi-Agent Path Finding (MAPF) instance and a selection of classical MAPF solution techniques (section 6.1). Next, the application of the MAPF instance to real-world problems will be discussed in section 6.2, including difficulties associated with this application. In the same section, different algorithms will be elaborated on that tackle the aforementioned difficulties. Finally, one of the reviewed algorithms will be chosen to use in the remainder of this thesis for planning paths of all vehicles in the MAS representing aircraft engine-off towing operations.

6.1. Path Finding in General

In this section, first a description of the single-agent path finding problem will be given, including an explanation on the working principle of one of the most well-known solving algorithms for path finding: the A* algorithm. Next, the single-agent path finding problem is generalized to a Multi-Agent Path Finding (MAPF) instance in subsection 6.1.2. A selection of solution techniques to solve the classical MAPF problem will be elaborated on in subsection 6.1.3.

6.1.1. Shortest Path Finding Problem

When considering a single agent, the shortest path finding problem can be described as finding the path in a graph from a start node to a goal node with the least amount of costs [131]. Cost can be expressed by the weight of edges, representing distance, time or any other variable of interest. Dijkstra [132] developed a shortest path finding problem, which still forms the basis of most current shortest path finding algorithms.

One of the most well-known extensions to Dijkstra's algorithm is the A* algorithm [133]. In this algorithm, a best-first search is performed guided by the function $f(n) = g(n) + h(n)$, where $g(n)$ represents the cost related to the shortest known path of the starting node to the current node n. Then, $h(n)$ represents a heuristic value of the expected path costs from node n to the goal node. If $h(n)$ never overestimates the costs of the shortest path from n to a certain goal node, that is, if $h(n)$ is admissible, then the A* algorithm is guaranteed to be optimal and complete [133]. One of the most commonly used heuristics is the Euclidean distance, neglecting all obstacles [81, 9].

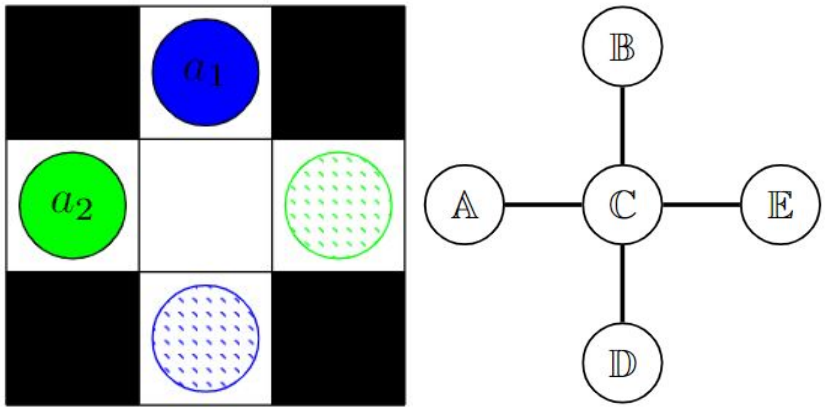
6.1.2. Classical Multi-Agent Path Finding (MAPF) Problems

When extending the single-agent path finding problem to an instance where multiple agents need to find a shortest path, one speaks of a Multi-Agent Path Finding (MAPF) problem. Using the definition as proposed by Stern et al. [134], a classical MAPF problem instance consists of a tuple $\langle G, s, t \rangle$ where $G = (V, E)$ is an undirected graph whose vertices V correspond with node locations and edges E correspond with connections between locations, where n agents can move along. Each agent i has a source vertex $s : [1, ..., n] \rightarrow V$ and target or goal vertex $t : [1, ..., n] \rightarrow V$, that represent the predefined

task allocation. Note that both source vertices and goal vertices are pairwise different [135].

In classical MAPF problems, time is assumed to be discretized, meaning that kinematics (such as finite accelerations/decelerations) are neglected. In every time step, agents can perform a single action $a : V \rightarrow V$ such that $a(v) = v'$, meaning that if agent k is currently at vertex v and performs action a, it will be in vertex v' in the next time step [134]. Two types of actions can be distinguished: either to wait (staying at the same location during a time step), or to move (moving from current vertex v to an adjacent vertex v', where $(v, v') \in E$). Finally, a single-agent plan can be defined as a sequence of actions for agent i $\pi = (a_1, ..., a_j)$ that start in the starting vertex s_i and makes sure that agent k ends at its goal vertex t_i at the arrival time T_i, which is the minimal time step T_i such that for all time steps $t = T_i, ..., \infty$ $\pi_i(t) = t_i$. Formally, the latter can be expressed as the path of agent i: $\pi_i = \langle \pi_i(0), \pi_i(1), ..., \pi_i(T_i), \pi_i(T_i + 1), ... \rangle$ [135]. A solution to the MAPF problem can then be defined as a set of n single-agent plans [134].

An example of a classical MAPF problem is given in Figure 6.1 [135]. White cells are traversable, black cells represent objects. Agents are defined as fully colored circles, with their corresponding target vertices represented by similarly colored hatched circles. Using the graph as depicted in Figure 6.1, the problem can be formally described as follows: agent a_1 with $s_1 = \mathbb{B}$ and $t_1 = \mathbb{D}$, agent a_2 with $s_2 = \mathbb{A}$ and $t_2 = \mathbb{E}$ and an optimal solution $\{\pi_1 = \langle \mathbb{B}, \mathbb{C}, \mathbb{D} \rangle, \pi_2 = \langle \mathbb{A}, \mathbb{A}, \mathbb{C}, \mathbb{E} \rangle\}$ [135].



(a) Grid representation of MAPF problem. (b) Graph representation of MAPF example

Figure 6.1: Example of an MAPF problem, where agents a_1 and a_2 are represented with fully colored circles. Their corresponding goal vertices t_1 and t_2 are represented using similarly colored hatched circles. Taken from [135].

Assumptions Related to Classical MAPF Problems

When referring to a ”classical” MAPF problem, one usually refers to a MAPF problem with the following set of assumptions:

1. No unexpected events that introduce delay of any kind are considered [136].
2. The duration of every action (move to an adjacent vertex over an edge or wait at current vertex) takes exactly one time step [137].
3. During one time step, each agent only occupies one single location [137].
4. Time is discretized into time steps [137].

Over the past decades, a lot of extensions to the MAPF problem have been proposed, both when considering the problem definition itself and when it comes to relaxation of the assumptions mentioned above. Concerning the latter, a discussion of MAPF in real-world applications will be discussed in section 6.2. A selection of the extensions in the problem definition itself will be elaborated on in more detail in section 7.1.

Before diving into the application of MAPF in real-world settings, first the three types of solution techniques to the classical MAPF will be elaborated on in more detail in the next section.

6.1.3. Classical MAPF Solution Techniques

The problem of MAPF has been extensively studied over the past few decades, resulting in a wide variety of solution approaches. Since the focus of this thesis is mainly on task allocation and integrating task allocation with path finding, not all solution approaches available in the domain of path finding will be reviewed in detail. Based on previous work of MSc students working on the subject of path finding in multi-agent systems [19, 8, 81, 138, 72, 9], focus is put on a selection of search-based solvers. A description of the well-known A* algorithm has already been provided in subsection 6.1.1, since this algorithm forms the basis of the majority of MAPF solution techniques. Based on the outcomes of comparing different MAPF solvers [19, 8, 81, 138, 72, 9] and recommendations [8] in previous related work, two approaches are chosen to highlight in more detail: Conflict-Based Search (CBS) and Priority-Based Search (PBS). The working principle of both algorithms will be discussed in greater detail, including extensions that make them more suitable for larger problems.

Note that no elaboration on performance characteristics of the discussed algorithms will be provided in this section. Experiments showing effectiveness and efficiency of the algorithms in real-world applications will be elaborated on in more detail in section 6.2, where some of the assumptions related to classical MAPF problems as provided above are lifted.

Approaches Based on Conflict-Based Search (CBS)

Sharon et al. [131] were the first authors to introduce the Conflict-Based Search (CBS) algorithm. The key idea of CBS is to split a problem where multiple agents have to find paths into multiple single-agent pathfinding problems. This is done by finding paths based on a set of constraints. If collisions exist among these paths, new constraints are added that resolve the conflicts. As mentioned before, the A* algorithm is exponential in the number of agents (k). However, CBS is only linear in the graph size and thus, more efficient than (variations of) the A* algorithm for large number of agents [131].

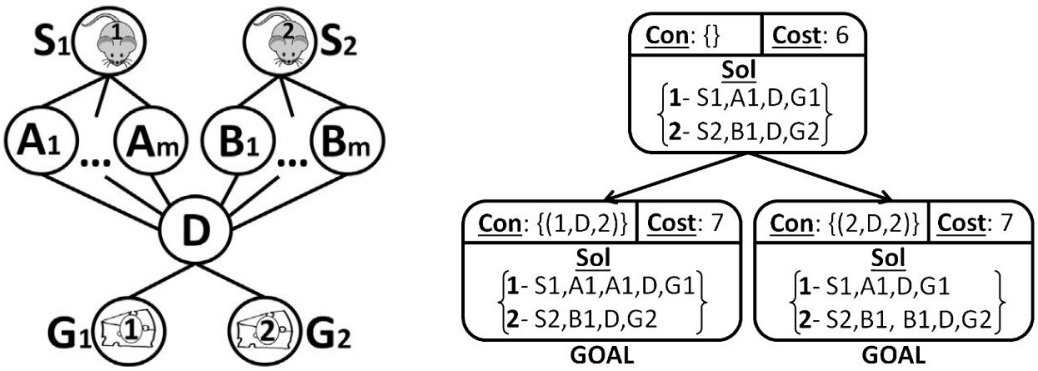
The CBS algorithm consists of two levels. In the high-level, conflicts are detected and constraints are added to resolve the conflicts. Subsequently, paths are found based on the set of constraints in the low-level of the search. While searching for conflicts, CBS makes use of a Constraint Tree (CT) with nodes consisting of a set of constraints, a set of k paths that are consistent with the constraints in the node corresponding with agent r_k and the total cost of the current solution. If the solution of a node is valid, e.g. no conflicts occur between paths of agents, the node is described to be the goal node.

However, if the solution includes conflicts, child nodes are added to the root node that include constraints that resolve the conflict. If conflict $C_n = (r_k, r_{k'}, v, \tau)$ arises, two constraints need to be added that ensure either r_k or $r_{k'}$ not being present at (v, t). To guarantee optimality, both possibilities are included in the CT (in programming language also referred to as the list OPEN), after which the low-level search is invoked that finds the shortest path using A* for each agent based on the constraints, but ignoring any other agent. Then, the resulting paths are evaluated in the high-level on conflicts and the procedure starts again. Note that CBS always expands the CT node with the smallest cost [131].

In Figure 6.2 [131], an example is shown of a MAPF instance where two mice have to reach two goal locations. The corresponding CT is shown in Figure 6.2b and is initialized with a root node, containing the two shortest paths for the two agents (found by the low-level search) and the corresponding solution cost. No constraints are included in the root node. After validation of the two paths, a conflict is found for both agents: $C_1 = (s_1, s_2, D, 2)$. As a result, two child nodes are created: one for restricting both agents to be present at vertex D at time step 2. In the low-level search, new shortest paths for both agents are found. While validating the newly found paths, no conflicts are found for both child nodes and thus, both child nodes are declared a goal node. Using trivial tie-breaking due to similar costs for both possibilities, the left node is selected as the final optimal solution.

The CBS framework is proved by Sharon et al. [131] to provide for optimal solutions. However, for large problem instances, runtimes of the algorithm become excessive, motivating the idea to trade solution quality for better computational performance. One of the most promising variations of the CBS algorithm for path finding in the context of complex and large multi-agent systems according to previous work [19, 8, 138, 72, 9], is Enhanced Conflict-Based Search (ECBS) and introduced by Barer et al. [139].

The key idea of ECBS is to not expand nodes any further that are close to the optimal solution. This is done by using an additional list of nodes (FOCAL) next to the already existing OPEN list in the high-level search, and an OPEN$_i$ and FOCAL$_i$ list in the low level for each agent r_i affected by the constraints originating in the high-level search. As mentioned before, OPEN contains all root and



(a) Example of a MAPF problem where two mice (s_1 and s_2) need to reach their goal locations g_1 and g_2 respectively. Taken from [131].

(b) Example of a constraint tree used in the solution technique CBS. Taken from [131].

Figure 6.2: Example of a MAPF problem where two mice (s_1 and s_2) need to reach their goal locations g_1 and g_2 respectively. The CT represents the working principle of the CBS algorithm. Taken from [131].

(expanded) child nodes. The FOCAL list is a subset of OPEN and only contains nodes that are within a certain factor ω from the optimal solution. Let $f_{min}(i)$ be the minimal cost in OPEN$_i$ when searching in CBS's low level for a path for agent r_i. Then, the lower bound on the f-value (solution cost) for a node n in the CT can be expressed using Equation 6.1 [139]. Similarly, the upper bound of the paths considered for further expansion is based on f_min and factor ω, as shown in Equation 6.2 [139].

$$LB(n) = \sum_{i=1}^{k} f_{min}(i) \tag{6.1}$$

$$UB = f_{min} \cdot \omega \tag{6.2}$$

Instead of only passing the cost of a combination of paths from the low-level to the high-level ($n.cost$), in the ECBS algorithm also the lower bound $LB(n)$ is passed along. The list FOCAL is then determined using the expression as stated in Equation 6.3 [139]. As a result, all nodes that are in FOCAL are within a factor ω from the optimal solution, making ECBS a bounded suboptimal solver for classical MAPF instances.

$$FOCAL = \{n | n \in OPEN, n.cost \leq LB \cdot \omega\} \tag{6.3}$$

Approaches Based on Priority-Based Search (PBS)

Instead of searching for new solutions based on conflicting paths, priority based solvers determine a priority among the involved agents and plan collision-free paths subsequently in the priority order [9]. A promising priority-based solver as determined by Soomers [8], Chen [72] and Kamphof [9] is the Priority-Based Search (PBS) algorithm introduced by Ma et al. [140].

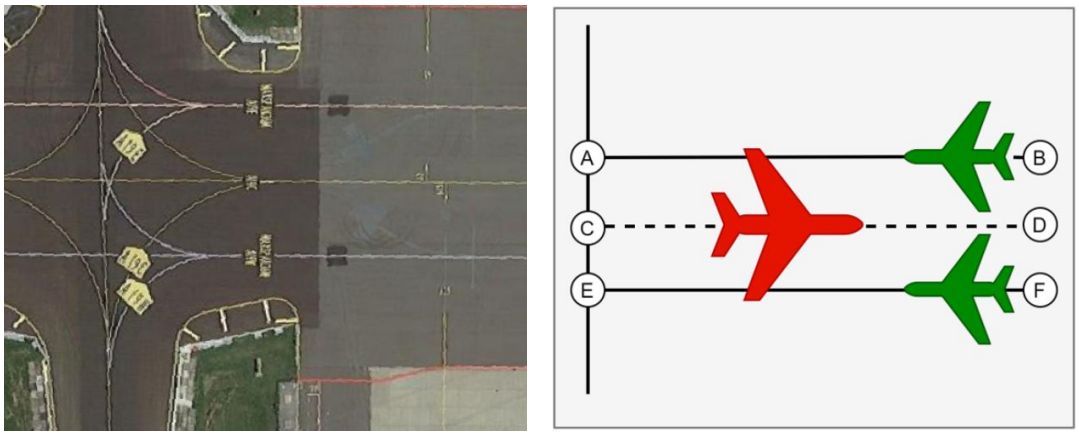
Similarly as to CBS, PBS is a two level search algorithm. However, instead of using a best-first search approach as adopted in CBS, the PBS algorithm uses a depth-first search approach. In a high-level Priority Tree (PT), the algorithm stores in every PT node a priority order, sum of costs and paths for all agents. No constraints are added to PT nodes, as opposed to the CT nodes in CBS. Using the priority order stored in the root node, PBS runs a low-level search and finds paths for all agents, where the paths of agents with lower priority are planned after the paths for agents with higher priority are determined. Any conflicts resulting from the low-level search are to be resolved by adding priorities in the high-level. This is done by expanding the root node into two child nodes: one in which agent r_k has priority over $r_{k'}$ and one with reversed priority. In the low-level, new paths are found with the updated priority order. This procedure is repeated until a conflict-free solution for all individual agents is found.

6.2. MAPF in Real-World Applications: Multi-Agent Motion Planning (MAMP)

In subsection 6.1.2, a number of assumptions has been listed that apply for classical MAPF solving algorithms. However, when applying these algorithms in real-world scenarios, the aforementioned assumptions often will not hold [141]. This is also the case for applying classical MAPF algorithms to

the problem of routing aircraft and TaxiBots over an airport ground network. According to Chen [72] and Kamphof [9], five main differences can be identified when it comes to classical MAPF instances and airport ground surface operations.

- **Perfect Plan Execution** (related to assumption item 1): as stated by Ma et al. [141], agents often have imperfect plan executions causing the assumption of perfect plan execution not to be valid in real-world problem settings. In addition, unexpected events are inherent to the field of airport ground surface operations, which underlines the importance of taking uncertainty into account.
- **Uniform Edge Traversal Times** (related to assumption item 2): in the grids used in classical MAPF instances, edges are assumed to be uniform, meaning that traversing from edge i to edge j, $i, j \forall i, j \in E$ will take one time step exactly. However, when considering a graph layout of e.g. AAS, edges may not be uniform in length. Although this issue can be easily solved by using weighted edges that represent travel time in a classical MAPF graph [134], the notion of non-uniform edge traversal time is an important difference between classical MAPF instances and real-world airport ground surface operations.
- **Kinematics of Agents** (related to assumption item 2): in classical MAPF instances, the actions of an agent (either move to an adjacent vertex or wait at the current vertex) always take exactly one time step. In addition to non-uniform edge traversal times as discussed before, the actual kinematics of agents is another factor that does not allow for this assumption to hold in real life [142]. For example, aircraft are subject to finite accelerations and decelerations and maximal turning velocities and therefore, cannot be realistically modelled without kinematic constraints.
- **Point Agents** (related to assumption item 3): using a graph layout in classical MAPF instances, it is assumed that agents are represented by points occupying a single location on the graph. However, as pointed out by Kamphof [9], this assumption might result in collisions between aircraft on certain taxiways (Figure 6.3).
- **Discretization of Time** (related to assumption item 4): in classical MAPF instances, time is discretized into time steps. However, since actual time is continuous and agents are restricted in their kinematic movements, modelling time using discrete time steps is in most real-world applications not sufficient nor accurate enough [72, 9].



(a) Example of wingspan restricted taxiway at AAS.

(b) Schematic representation of wingspan restricted taxiway at AAS.

Figure 6.3: Example of a situation in which modelling agents without shapes in classical MAPF instances would lead to collisions in real-life airport ground surface operations. Taken from [9].

In literature, several approaches have been suggested that tackle the relaxation of the assumptions as listed in subsection 6.1.2 to make the classical MAPF problem suitable to apply in real-life problems with experiments on hardware. Examples of these approaches include adapted versions of CBS to account for geometric shapes of agents [143], the use of conflict intervals to allow for planning in continuous time [136], post processing of MAPF plans to account for imperfect plan execution and

kinematic constraints of agents (referred to as MAPF-POST) [142] and forward simulation of paths of agents to include kinematics [18, 17].

It is pointed out by Kamphof [9] that the assumptions related to shapes of agents and non-uniform edge traversal times can be relatively easily accounted for by the addition of extra constraints. However, the inclusion of agent kinematics, the use of continuous time and allowing for imperfect plan execution are significantly harder to overcome. The reason for this is that both MAPF-POST and forward simulation do not take kinematics into account in the planning phase, but account for this by modifying the output of a classical MAPF algorithm. Since it is expected that the restrictive kinematics of aircraft (due to their weight) is of high influence on the actual execution, it is preferable to make use of an approach that takes kinematics into account already in the planning phase [9].

A type of approach that does not suffer from the aforementioned disadvantages are motion-based approaches, also referred to as Multi-Agent Motion Planning (MAMP) [144]. More explicitly, MAMP is "the task of finding conflict-free kinodynamically feasible plans for agents in a shared environment [while] efficiently reason[ing] with continuous time" [144, p. 44]. Previous work on airport ground surface operations [8, 72, 9] has shown that using a motion-based approach for path finding in large and complex multi-agent systems is a successful approach to take into account the kinematics of agents and deal with continuous time. In combination with the fact that the main focus of this thesis is on task allocation and the integration of task allocation with path finding (and not on path finding specifically), this literature study is limited to reviewing recommended MAMP algorithms for path finding based on the outcomes of aforementioned previous work.

One of the algorithms that forms the basis in this domain is Safe Interval Path Planning (SIPP), which is a single-agent path finding algorithm. In subsection 6.2.1, SIPP and useful extensions will be elaborated on. Next, the integration of SIPP with other search-based MAPF solvers will be discussed to account for interaction between agents in motion planning in subsection 6.2.2.

6.2.1. Single-Agent Motion Planning with SIPP

As mentioned above, the SIPP algorithm forms the basis within the field of MAMP solvers and will therefore be elaborated on first. In the remainder of this section, two extensions of SIPP will be discussed: Safe Interval Path Planning with Reservation Table (SIPPwrRT) and Soft Conflict Safe Interval Path Planning (SCIPP).

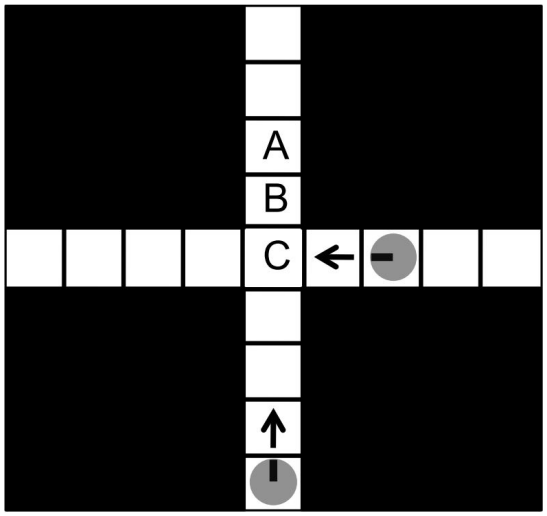
Safe Interval Path Planning

The Safe Interval Path Planning (SIPP) algorithm was first introduced by Philips and Likhachev [145] and plans paths for a single agent based on a predetermined priority ordering using a variant on the A* algorithm. Agents that are higher prioritized are treated as dynamic obstacles in the environment with whom collision should be avoided. The novelty of SIPP is that it makes use of [configuration, interval] pairs to search the state-space instead of [location, time step] pairs. In this context, two types of intervals can be distinguished: safe intervals (periods of time for a configuration in which no collisions with dynamic obstacles occurs) and conflict intervals (periods of time for which the configuration is in collision with a dynamic obstacle for each time step in the interval) [145]. Since the number of safe intervals is in general significantly smaller than the number of safe time steps (that are comprised in the safe interval), the state space is significantly smaller as well when using SIPP, due to the fact that a single state (represented by the [configuration, interval] pair) replaces what used to be many states (for each time step a [location, time step] pair).

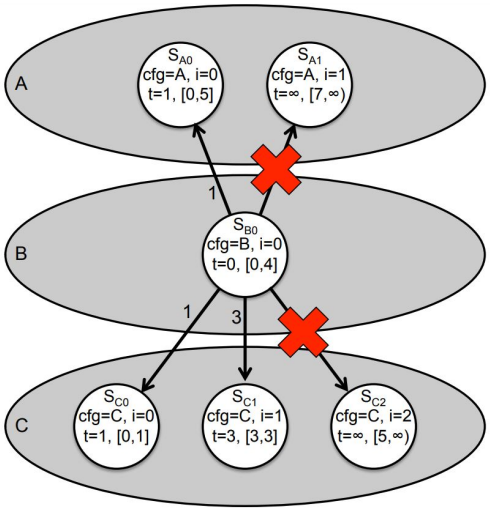
Let the [configuration, interval] pair define a state s, where the configuration is a set of non-time variables describing the agent's state, such as heading and position. For each state s, a cost function representing the g-cost ($g(s)$), a heuristic score ($h(s)$) and a cost function $c(s, s')$ representing the cost of transitioning from state s to s', is saved. When the algorithm is initialized, it first examines the graph and creates a spatial timeline constituting of safe and conflict intervals based on the trajectories of dynamic obstacles (assumed to be known beforehand, Figure 6.4). Afterwards, a modified version of the A* algorithm is run, in which a function $M(s)$ is defined to return possible movements from the current state s. For each possible

Figure 6.4: Example of a timeline constituting of safe and conflict intervals. Taken from [145].

movement, the earliest possible time of arrival based on the safe interval of the new configuration is determined. The different possibilities to move to adjacent vertices are referred to as successors in the vertex under consideration. Note that for moving to new configurations, arrival in the new safe interval is assumed to occur as early as possible and successors with the lowest cost (f-values) are expanded first.



(a) Example of an environment with two dynamic obstacles and three configurations (A, B, C). The agent initially is located at configuration B.



(b) Expansion of configuration B based on the environment as represented in Figure 6.5a. White circles represent states, consisting of: the name (first line); the configuration and its safe interval (second line); and the earliest known time the state can be reached and the corresponding safe interval (third line). The arrows represent transitions from one configuration to another, where the red crosses indicate infeasible transitions.

Figure 6.5: Example of an expansion using the SIPP algorithm. Taken from [145].

An example of expansion of configurations using the SIPP algorithm is shown in Figure 6.5. In Figure 6.5a, the environment with two dynamic obstacles and three configurations is shown, where the agent is initially located in configuration B. After initialization of the algorithm, the safe intervals for the current configuration (B) and adjacent configurations (A, C) are determined. When expanding S_{B0}, two possible successors are found for configuration A: the agent can move their immediately ($t = 1$, within safe interval of S_{A0}) or wait until $t = 7$ to move there. However, for the latter, it has to wait in B until $t = 6$, which is not in the safe interval of S_{B0}. Therefore, the expansion of state S_{A1} is not a feasible state. A similar reasoning is used to expand the states in C, resulting in two feasible successors (S_{C0} and S_{C1}). The cost for the successors is determined based on the time steps that the new state can be reached. All feasible expansions will be added to the OPEN list and the state with the lowest cost will be expanded first [145].

Safe Interval Path Planning with Reservation Table

Based on the idea of the space-time A* algorithm that keeps a reservation table for each cell indicating all safe intervals of that cell, Ma et al. [146] propose to extend the SIPP algorithm with a reservation table as well, resulting in the Safe Interval Path Planning with Reservation Table (SIPPwrRT) algorithm. In this context, a reservation table is referred to as a list that contains all reserved or collision intervals of the cell under consideration in increasing order of the earliest time step that the cell is considered not to be safe. When a new path is planned, new entries to the reservation table are added and redundant entries are deleted to keep the table small in size. According to the authors, the use of reservation table increases the efficiency of SIPP, since the algorithm does not have to iterate through all the paths of each dynamic obstacle separately to determine the safe intervals for a specific cell.

In addition, the authors also propose a method to handle continuous agent movements and include shapes of agents using SIPPwrRT. In the work of Ma et al. [146], an agent r_k with a volume are represented as circular object using a radius R_k. In order to avoid conflicts between agents, the distance D between the center points of two agents should at least be larger than $R_1 + R_2$. This distance is converted into a time offset δT, representing the time that should be added to the conflict interval

to restrict any agent accessing the cell. For different cases (agents moving in the same direction, in orthogonal direction and in opposite direction), Ma et al. [146] defined three formulas that represent the time offset to prevent agents from overlapping conflict intervals. Note that a similar collision avoidance system is used in the model of Soomers [8] and Kamphof [9] (refer to subsection 6.2.2).

The authors prove that SIPPwrRT finds collision-free paths for all agents and guarantees optimality given a certain priority order [146]. When using the algorithm in a pickup and delivery context (combined with Token Passing (TP), refer to subsection 7.2.3), the SIPPwrRT is claimed to be complete for all well-formed Multi-Agent Pickup and Delivery (MAPD) problems (subsection 7.1.3).

Soft Conflict Safe Interval Path Planning

In order to use SIPP in a FOCAL search algorithm, Cohen et al. [144] developed a low-level bounded suboptimal version of SIPP, referred to as Soft Conflict Safe Interval Path Planning (SCIPP). This version of SIPP is used in the MAMP solver Enhanced Conflict-Based Search for Continuous Time (ECBS-CT), elaborated on in subsection 6.2.2.

Based on a starting state, goal state, a list of hard constraints for each cell and a reservation table specified in the high-level search, the SCIPP finds a ω-suboptimal feasible path from start to goal. Hard constraints are represented using [cell, time step] pairs and soft conflicts are represented in a [cell, time interval] pair that describes the time interval that the cell is swept by agents higher in priority. From the FOCAL list, the algorithm chooses the node with the lowest cost first to expand, where the cost is based on a conflict heuristic h_c representing the number of soft conflicts in the path.

6.2.2. Multi-Agent Motion Planning with SIPP

As mentioned in subsection 6.2.1, SIPP is a low-level single-agent motion planning algorithm. In order to use it in a multi-agent system, the SIPP algorithm has to be combined with a high-level search algorithm to solve Multi-Agent Motion Planning (MAMP) instances. Whereas the nodes in MAPF problem represent physical locations, the MAMP problem is posed on states, which specify e.g., the location, orientation and velocity of agents [144]. Edges then are used to represent feasible motions from one state to another of arbitrary duration. Finally, the set of edges that represent a sequence of feasible, collision-free motions is referred to as a plan that leads an agent from its starting state to its goal state.

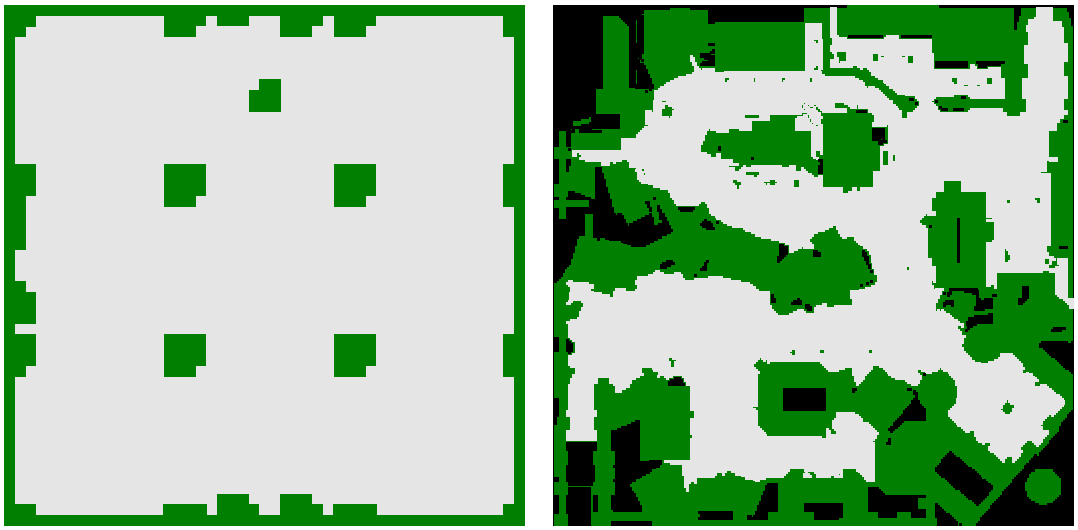
Two promising MAMP algorithms that are considered and/or implemented in previous related work [8, 72, 9], are an altered version of ECBS to deal with continuous time (Enhanced Conflict-Based Search for Continuous Time (ECBS-CT)) and a combination of SIPPwrRT with PBS.

Enhanced Conflict-Based Search for Continuous Time

In the work of Cohen et al. [144], an altered version of the ECBS algorithm is proposed that is capable of dealing with continuous time, resulting in the Enhanced Conflict-Based Search for Continuous Time (ECBS-CT) algorithm used for solving MAMP instances. Similarly to the discretized version of ECBS, a suboptimality factor ω is used to determine which nodes are placed from the OPEN list into the FOCAL list (refer to subsection 6.1.3 for more details on the working principle of ECBS). Using the low-level SCIPP search algorithm to find paths for all agents, conflicts are detected in the high-level search in the form of a [cell, time interval] pair. In order to resolve the conflict, a constraint (c, τ) is applied, where c represents the cell in which the conflict takes place and τ represents a single time step in the conflict interval. According to Cohen et al. [144], a single time point rather than a time interval is chosen to guarantee the suboptimality of ECBS-CT. Based on the constraints, expansion of the root node is performed, resulting in two child nodes. In the low-level search, a reservation table is updated with the specified constraints and paths are found for the two successor nodes. Both child nodes are added to the OPEN list, and to the FOCAL list if they satisfy the conditions as well. In order to efficiently keep track of the time intervals that represent cell c being occupied by a number of agents, Cohen et al. [144] suggest to make use of an interval map. This data structure allows for efficient detection of the earliest conflict in the high-level search [144].

When evaluating ECBS-CT for MAMP instances with varying number of agents in two different environments (Figure 6.6) and for different number of motions possible from one state to another (5 and 13 *motion primitives* respectively), Cohen et al. [144] concluded that ECBS-CT is a suitable algorithm to find solutions and scales better to larger numbers of agents in obstacle-rich maps than other solvers. In addition, solutions are produced with suboptimality guarantees that are significantly smaller than the suboptimality bound ω imposed. Although run times are relatively long for the application of the algorithm to online problems (around 20 seconds for 50 agents in the Den520d environment as presented in

Figure 6.6), Kamphof [9] pointed out that several significant differences exist between the implementation of Cohen et al. [144] and the intended implementation in the context of ground surface operations. The main difference is that Cohen et al. [144] considered a cell-based environment in which swapping of intermediate cells during a motion from one configuration to another was taken into account, leading to a sizable reservation table. However, for an airport environment and ground surface operations, the reservation table is expected to be much smaller, since we consider only segments (taxiways) instead of individual cells. Therefore, ECBS-CT is considered to be a suitable candidate to implement in this research.



(a) Arena map with 49x49 cells.

(b) Den520d map with 256x257 cells.

Figure 6.6: Two example environments used to test the ECBS-CT algorithm. Taken from [147].

SIPPwrRT Combined With PBS

In the work of both Soomers [8] and Kamphof [9], airport ground surface operations at AAS are studied, where aircraft and TaxiBots are routed from a predetermined starting location (gate) to a goal location (runway). For this application, a novel combination of PBS (high-level) and SIPPwrRT (low-level) is proposed [8, 9]. Their proposal is based on the fact that although PBS scores good on scalability, run time and success rate, it is not directly applicable in the context of airport ground surface operations due to the lack of taking kinematics of agents into account, as well as its assumption of discretization of time. On the other hand, SIPP-based approaches explicitly handle the two latter difficulties efficiently.

In their implementation of a MAMP solver, two levels are distinguished: a high-level search based on PBS (refer to subsection 6.1.3) and a low-level path planning search based on SIPPwrRT (refer to subsection 6.2.1). The PBS algorithm finds a priority ordering among agents that results in the lowest total sum of costs, assumed to be total taxi times. By translating the priority order in graph reservations (reserving specific edges in a specific time interval for movement of higher prioritized agents) based on the kinematics of aircraft and TaxiBots, constraints are generated that are used by the single-agent path planning algorithm (an adapted version of SIPPwrRT). In order to provide for path planning of newly arriving aircraft to the system in an online manner, Kamphof performed replanning every h_{plng} minutes for the next ω_{plng} minutes [9].

The translation of a priority ordering into a set of graph reservations is done by using three inputs: the shapes of agent r_{k1} and r_{k2}; the velocity profile of agent r_{k1}; and the distances between nodes in the graph layout. Similarly as done by Ma et al. [146], agents are modelled as circular discs with radius $R_k = \max(W, L)$, where W represents the width of the agent and L its length [8, 9]. Using the combined shape of both agents and a safety margin d_s, a planning radius R_p is determined using Equation 6.4. For each edge, the planning radius is then used to determine four relevant time points used for setting constraints related to safe and unsafe (conflict) intervals for the edge under consideration. Figure 6.7 visualizes the definition of these four time points related to the planning radius: τ_1 represents the time of entering edge AB in node A; τ_2 represents the time of leaving node A while entering edge AB; τ_3

represents the time of entering node B while leaving edge AB; and τ_4 represents the time of leaving edge AB in node B [8, 9].

$$R_p = R_{k1} + R_{k2} + d_s \tag{6.4}$$

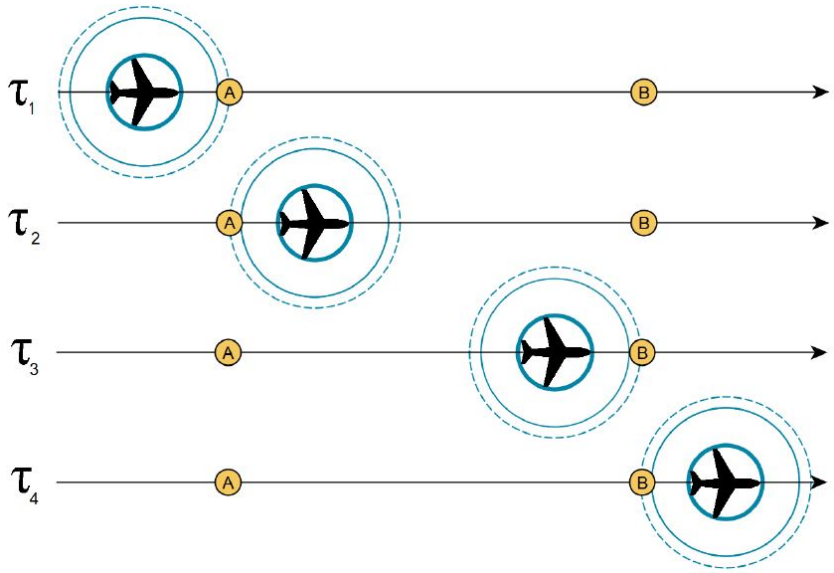
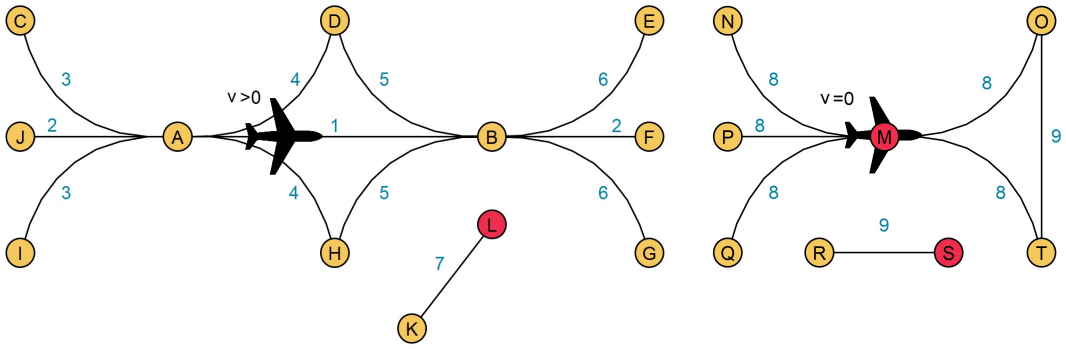


Figure 6.7: Visualization of definition of time points τ_1 to τ_4, related to edge traversal of agents on edge AB. The thick blue line represents the shape of the agent, the thin blue line represents the combined shape of both agents r_{k1} and r_{k2} and the dashed blue line represents the planning radius. Taken from [8].

After having defined the relevant time points for every node on the path of agent r_k, all connected edges to the path and all non-connected edges within the planning radius are evaluated. For all these edges, conflict intervals are determined based on the previously determined time points. Two classification systems are used: one for non-zero velocities of agent r_k, and one when agent r_k is waiting (velocity equal to zero). The classification systems are visualized in Figure 6.8 and the descriptions of edge and node types are summarized in Table 6.1, including the corresponding conflict intervals [8]. Note that the conflict intervals are defined by three variables: the starting time of the conflict interval, the ending time of the conflict interval and the type of limitation. Concerning the latter, three types of limitations are implemented: Do Not Enter (DNE) (restrict entering the edge), Do Not Persist (DNP) (restrict presence at the edge) and Do Not Wait (DNW) (restrict waiting at the node) [8]. Next to conflict intervals, regular constraints are implemented to prevent agents overtaking each other on the same edge.



(a) Classifications of edges based on agent r_k traversing over edge AB with non-zero velocity.

(b) Classifications of edge based on agent r_k waiting in node M with zero velocity.

Figure 6.8: Classification systems for edge traversal of agent r_k, where red nodes represent possible waiting locations for agent r_k. The numbers correspond with the edge types as presented in Table 6.1. Taken from [8].

Table 6.1: Description of edge and node types, corresponding conflict intervals and regular constraints for the edge classification systems as depicted in Figure 6.8 [8].

Type	Description	Conflict Interval in Direction of Traversal	Regular Constraints	Conflict Interval in Opposite Direction of Traversal
1	Traversed edge	$[\tau_1, \tau_2, \text{DNE}]$	If entered before τ_1, leave before τ_3 If entered after τ_2, leave after τ_4	$[\tau_1, \tau_4, \text{DNP}]$
2	Any other edge on path within the planning radius	-	-	-
3	Incoming edge at start of traversed edge	-	-	-
4	Outgoing edge at start of traversed edge	$[\tau_1, \tau_2, \text{DNE}]$	If entered before τ_1, leave before τ_3 If entered after τ_2, leave after τ_4	$[\tau_1, \tau_4, \text{DNP}]$
5	Incoming edge at end of traversed edge	$[\tau_1, \tau_4, \text{DNP}]$	-	$[\tau_1, \tau_4, \text{DNP}]$
6	Outgoing edge at end of traversed edge	-	-	-
7	Any other non-connected edge to the path within the planning radius	$[\tau_1, \tau_4, \text{DNP}]$	-	$[\tau_1, \tau_4, \text{DNP}]$
8	Edge connected to waiting location	$[\tau_1, \tau_2, \text{DNE}]$	-	-
9	Any other non-connected edge to the waiting location within the planning radius	$[\tau_1, \tau_2, \text{DNP}]$	-	$[\tau_1, \tau_2, \text{DNP}]$
Node L	Possible waiting location within the planning radius from any point on current edge	$[\tau_1, \tau_4, \text{DNW}]$	-	-
Node M	Current waiting location	$[\tau_1, \tau_2, \text{DNW}]$	-	-
Node S	Possible waiting location within the planning radius from any point on current edge	$[\tau_1, \tau_2, \text{DNW}]$	-	-

Once all conflict intervals based on the paths of agents higher in priority have been determined, a set of safe intervals and a set of regular constraints are defined. Then, in the low-level search, a feasible path and corresponding velocity profile are found. From an initial state, the agent tries to move as quickly to a neighboring node as possible, while adhering to the imposed constraints and safe intervals from the high-level node. When no conflict intervals or other restrictions are present on the neighboring edges of the next node, optimization for traversal time is performed by accelerating and decelerating at the beginning and end of the edge respectively. In the case of conflict intervals being present at neighboring edges of the next node, arrival at exactly the start of the safe interval is being aimed for by optimizing the maximal final velocity on the edge. This is done by accelerating or decelerating at the end or start of an edge respectively.

However, in some situations, the agent's kinematic properties do not allow for adhering to limitations related to future edges or nodes in the path. An example of this is that an agent might not have enough braking power to satisfy a conflict interval on a next edge or node when travelling at maximum velocity. Therefore, an anticipated-motion search is implemented in the SIPPwrRT algorithm. From the currently being *explored state*, the connected node that can be reached with the highest velocity is called the *prime state*. Then, the prime state is further explored for anticipation, meaning that all edges within the braking distance away from the prime state are evaluated. For each node, it is checked whether this

node can be reached through a feasible motion from the *prime state*. If the edge requires an unfeasible motion, it is checked whether this edge can be reached not from the prime state but from the *explored state*. Since distance is increased, more freedom with respect to kinematics is created. If the state that required an unfeasible motion from the prime state still requires an unfeasible motion from the explored state, this state is discarded as successor. All other options (the prime state and all states that are unfeasible from the *prime state* but feasible from the *explored state*) are included as possible successors [8, 9].

6.2.3. Concluding Remarks on MAMP Algorithms

In order to tackle the difficulties mainly related to agent kinematics and discretization of time when applying classical MAPF solution techniques to real-world applications, two Multi-Agent Motion Planning (MAMP) solution algorithms have been presented: ECBS-CT and SIPPwrRT combined with PBS. In this section, a reasoning will be provided on which MAMP algorithm is going to be used in the remainder of this thesis for motion planning of vehicles on the airport taxiway network and surrounding infrastructure.

Based on previous work [8, 72, 9], the general class of motion-based approaches are deemed suitable to provide for path planning in real-world MASs. When evaluating the system requirements as mentioned in subsection 4.6.2, it can be noted that both MAMP solvers are capable of dealing with the objective to minimize the total taxi time of all inbound and outbound flights (item 3). Similarly, both algorithms proved to efficiently avoid conflicts and collisions among agents (item 4) and are suitable to apply in online problem settings (item 6). When considering the requirement on available time and resources, a preference arises for the use of SIPPwrRT combined with PBS, since this algorithm is already implemented in the hierarchical multi-agent control architecture developed by Soomers [8] and Kamphof [9] that will be used in this thesis as well.

7

Integrated Task Allocation and Path Planning

In the two previous chapters, both allocation of agents to a set of tasks and path finding from a starting location to a target location has been discussed respectively. When considering the towing of aircraft by TaxiBots, both techniques will have to be combined: not only do TaxiBots need to be assigned to outbound aircraft, also conflict-free paths will have to be defined for all moving vehicles in the system (aircraft, TaxiBotting aircraft and TaxiBots).

In this chapter, the combination of task allocation and path finding will be discussed. First, a general description of the combined task allocation and path finding problem will be given, also called the Multi-Agent Pickup and Delivery (MAPD) problem (section 7.1). Next, a selection of studies will be elaborated on in section 7.2 that use a decoupled approach to solve the MAPD problem by separately and consecutively solving the task assignment problem and path finding problem. Finally, in section 7.3, two algorithms will be discussed that solve the task assignment and path finding problem simultaneously, allowing for guarantees on optimality. Again, the latter two sections will be concluded with a review on which approach to combine task allocation and path finding is deemed most suitable to apply in the context of TaxiBotting.

7.1. Extensions of the Classical MAPF Problem to Include Task Allocation

When considering the description of classical MAPF problems in subsection 6.1.2, it can be noted that the assignment of agents to tasks is assumed to be a given. In addition, all agents are assigned only one task (or destination) and the problem terminates when all agents have reached the location of this one task. Furthermore, the number of agents equals the number of tasks.

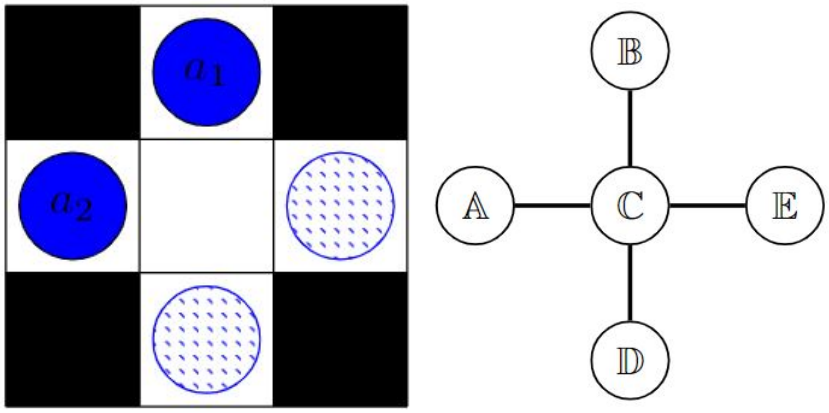
When considering real-world applications, several characteristics are not incorporated into this classical MAPF formulation, as described in section 6.2. However, this section focuses primarily on the inclusion of kinematics and continuous time into the classical MAPF problem, but does not incorporate other operational characteristics of real-world applications. Examples of this are situations where the assignment of agents to tasks is not determined beforehand, where agents are grouped into teams or where agents are constantly engaged with newly arriving tasks to the system. In the following subsections, extensions of the classical MAPF problem will be discussed, eventually leading to the definition of the problem that will be studied in more detail in this chapter: the Multi-Agent Pickup and Delivery (MAPD) problem. Note that the notation as described in subsection 6.1.2 will be assumed a baseline and supplemented when necessary.

7.1.1. Anonymous Multi-Agent Path Finding (AMAPF) Problem

In the Anonymous Multi-Agent Path Finding (AMAPF) problem, the goal is similar to that of MAPF problems: a set of agents has to move to a set of target vertices. However, in contrary to the classical MAPF problem, every agent can be assigned to every target: it does not matter which agent fulfills

which task [134]. Note that the task allocation is thus not a predetermined given to the problem. Given a set of agents $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ and a set of targets $\mathcal{T} = \{t_1, t_2, ..., t_n\}$, the assignment of targets to agents can be expressed as a one-to-one mapping $\sigma : t_{i'} = \sigma(r_i)$. As a consequence, an AMAPF plan consists not only of a path π_i to each agent r_i, but also includes the assignment σ of targets. A solution to the AMAPF problem is an AMAPF plan for which all the paths of agents are collision-free [135].

An example of an AMAPF problem is shown in Figure 7.1. Similarly as for the MAPF example shown in Figure 6.1, white cells are traversable and objects are represented using black colored squares. Agents r_1 with $s_1 = \mathbb{B}$ and r_2 with $s_2 = \mathbb{A}$ and a set of targets $t_1 = \mathbb{D}$ and $t_2 = \mathbb{E}$ are given. An optimal solution to this problem would be $\{\pi_1 = \langle \mathbb{B}, \mathbb{C}, \mathbb{E} \rangle, \pi_2 = \langle \mathbb{A}, \mathbb{A}, \mathbb{C}, \mathbb{D} \rangle\}$, where $\sigma(r_1) = t_2$ and $\sigma(r_2) = t_1$ [135].



(a) Grid representation of AMAPF problem. (b) Graph representation of AMAPF example

Figure 7.1: Example of an AMAPF problem, where agents a_1 and a_2 are represented with fully colored circles. A set of goal vertices t_i and t_j are represented using colored hatched circles. Taken from [135].

7.1.2. Target Assignment and Path Finding (TAPF) Problem

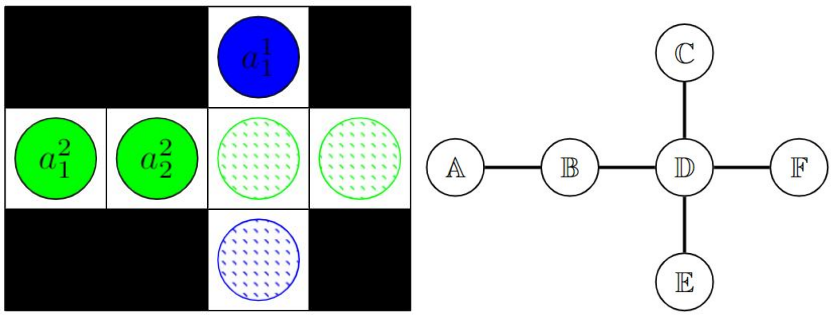
In the Target Assignment and Path Finding (TAPF) problem, the set of n agents $\mathcal{R}$ is partitioned into K disjoint teams $\text{team}_1, ..., \text{team}_K$. Each team_k consists of J_k agents $\text{agent}_1^k, ..., \text{agent}_{J_k}^k$ and is assigned with J_k targets $t_1^k, ..., t_{J_k}^k$. Every agent r_i^k is assigned a unique starting vertex s_i^k. Similarly as for the AMAPF problem, the assignment of targets to team_k is given as a one-to-one mapping $\sigma^k : t_{i'}^k = \sigma^k(r_i^k)$. Again, targets and agents are interchangeable, meaning that any one agent r_i^k can be assigned any target t_i^k that is in the partition of targets belonging to the team_k [135].

Note that the TAPF can be viewed as a generalization of both the MAPF and AMAPF problem [135]. When the number of teams in the TAPF is equal to the number of agents ($K = n$), the problem turns into a MAPF problem where each agent is part of its own ”team”, assigned a single target. When the number of teams is equal to one ($K = 1$), the problem becomes an instance of the AMAPF problem, where a target can be assigned to any agent in the one team present [135].

In the example shown in Figure 7.2, two teams of agents are presented: team_1 consisting of agent r_1^1 with $s_1^1 = \mathbb{C}$; and team_2 consisting of agent r_1^2 with $s_1^2 = \mathbb{A}$ and agent r_2^2 with $s_2^2 = \mathbb{B}$. The target vertex given to team_1 is $t_1^1 = \mathbb{E}$ and the target vertices given to team_2 are $t_1^2 = \mathbb{D}$ and $t_2^2 = \mathbb{F}$. A collision-free, optimal solution is given by $\{\pi_1^1 = \langle \mathbb{C}, \mathbb{D}, \mathbb{E} \rangle, \pi_1^2 = \langle \mathbb{A}, \mathbb{A}, \mathbb{B}, \mathbb{D} \rangle, \pi_2^2 = \langle \mathbb{B}, \mathbb{B}, \mathbb{D}, \mathbb{F} \rangle\}$, where $\sigma^1(r_1^1) = t_1^1$, $\sigma^2(r_1^2) = t_1^2$ and $\sigma^2(r_2^2) = t_2^2$ [135].

7.1.3. Multi-Agent Pickup and Delivery (MAPD) Problem

The last extension of the classical MAPF problem that will be discussed is the Multi-Agent Pickup and Delivery (MAPD) problem. Similarly as for the MAPF, AMAPF and TAPF, the problem consists of a set of agents $\mathcal{R} = \{r_1, r_2, ..., r_n\}$ and a set of unexecuted tasks $\mathcal{T} = \{\tau_1, \tau_2, ..., \tau_m\}$. Every target $\tau_j \in \mathcal{T}$ is characterized by a pickup vertex $s_j \in V$ and a delivery vertex $g_j \in V$. However, the number of agents is not necessarily equal to the number of targets. Furthermore, the set of tasks $\mathcal{T}$ changes dynamically: new tasks can be added to the system over time [135]. In other words, the MAPD problem can be seen as a ”lifelong” version of the one-shot MAPF problem [88], where agents have to fulfill a stream of



(a) Grid representation of TAPF problem. (b) Graph representation of TAPF example

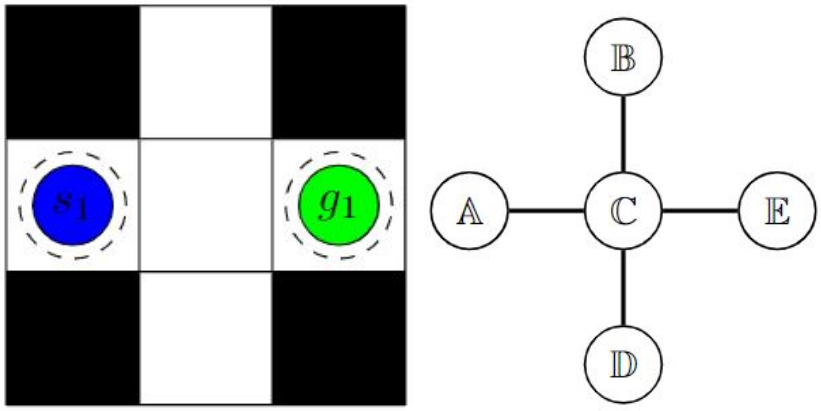
Figure 7.2: Example of an TAPF problem, where agents a_1^1, a_2^1 and a_2^2 are represented with fully colored circles, corresponding with two different teams. A set of goal vertices are represented using colored hatched circles, where the color corresponds with the team of agents that has to reach the given goal vertex. Taken from [135].

tasks while avoiding collisions with other agents and the environment. Similarly as for the AMAPF and TAPF problem, any free agent that is currently not executing a task, can be assigned to any task in the set $\mathcal{T}$. Furthermore, reallocation of agents can be done for tasks until agents have reached the pickup location of the unexecuted task: after that point, reallocation of the given task is not allowed anymore [88].

The goal in MAPD problems is to finish all tasks as quickly as possible while avoiding collisions between agents. Two metrics that are frequently used to evaluate the effectiveness of MAPD algorithms are the following [135]:

- **Makespan**: the earliest time step when all tasks are finished.
- **Service time**: the average number of time steps needed to finish each task from the moment it has arrived in the system.

In Figure 7.3, two agents r_1 (blue) and r_2 (green) are shown with colored circles, with starting positions $\pi_1(0) = \mathbb{A}$ and $\pi_2(0) = \mathbb{E}$. A single task τ_1 with $s_1 = \mathbb{A}$ and $g_1 = \mathbb{E}$ is shown, where both the pickup and delivery vertices are represented by dashed circles. A solution to the problem would be to assign task τ_1 to agent r_1, following path $\pi_1 = \langle \mathbb{A}, \mathbb{A}, \mathbb{C}, \mathbb{E} \rangle$. Agent r_2 is assigned path $\pi_2 = \langle \mathbb{E}, \mathbb{C}, \mathbb{B} \rangle$ [135].



(a) Grid representation of MAPD problem. (b) Graph representation of MAPD example

Figure 7.3: Example of an MAPD problem, where agents r_1 and r_2 are represented with fully colored circles. Task τ_1 is added at time step 0 to the system and is characterized by a pickup vertex s_1 and delivery vertex g_1, represented by dashed circles. Taken from [135].

Well-Formed Multi-Agent Pickup and Delivery (MAPD) Instances

In the example shown in Figure 7.3, the path assigned to agent r_1 and r_2 is called a solution since it results in a bounded service time for all tasks in the system. In general, an MAPD algorithm is said to

be able to solve an MAPD problem if and only if the resulting service time or makespan for all tasks is bounded [135]. However, not all MAPD instances are solvable: Figure 7.4 is an example of an MAPD problem that is not solvable, since neither agent a_1 nor a_2 can execute task τ_1 with pickup vertex s_1 and delivery location g_1 [88].

According to Ma et al. [88], if MAPD instances are well-formed, the problem is guaranteed to be solvable. Before explaining the definition of well-formed, two additional sets are defined. Let the set of endpoints V_{ep} be the set of all initial locations of agents, all pickup and delivery locations and any additional possible parking locations. The set of task endpoints V_{tsk} is the set of all possible pickup and delivery locations and the set of non-task endpoints is the set $V_{ep} \setminus V_{tsk}$ [88]. Then, the definition of well-formed can be expressed as follows:

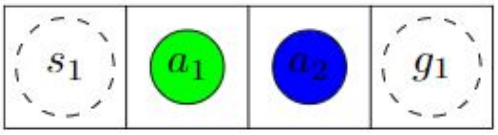


Figure 7.4: Example of MAPD instance that is not solvable. Agents a_1 and a_2 are represented by fully colored circles. Task τ_1 is characterized by pickup location s_1 and delivery location g_1. Taken from [88].

> A MAPD instance is well-formed iff a) the number of tasks is finite, b) there are no fewer non-task endpoints than the number of agents, and c) for any two endpoints, there exists a path between them that traverses no other endpoints [88].

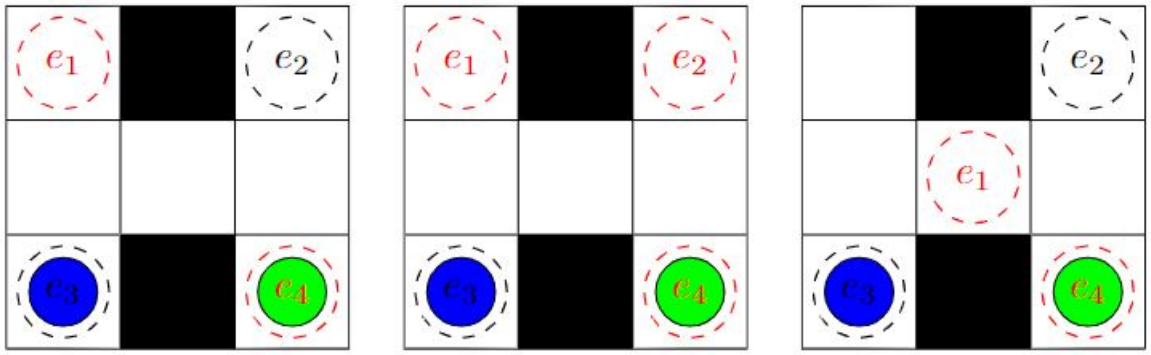


Figure 7.5: Examples of MAPD instances. White cells are traversable, black cells are blocked. Agents are represented using fully colored circles. Red and black dashed circles are task endpoints and non-task endpoints respectively. Taken from [88].

Figure 7.5 shows three MAPD instances. The left MAPD instance is well-formed. The middle MAPD instance is not well-formed, since the number of non-task endpoints (black dashed circles) is larger than the number of agents. The right MAPD instance is neither well-formed, since no path exists between endpoints e_2 and/or e_4 and e_3 that does not traverse over endpoint e_1 [88].

In order to solve well-formed MAPD problems, both task allocation and path finding will have to be performed. In the following two sections, both decoupled (first allocating tasks and subsequently defining collision-free paths for all agents) and coupled approaches (performing task allocation and path finding simultaneously) will be elaborated on in more detail.

7.2. Decoupled Approaches to Solve MAPD Problems

In Multi-Agent Pickup and Delivery (MAPD) problems, agents have to both assign themselves to tasks and find collision-free paths to execute these tasks. In general, all MAPD are considered to be lifelong: the total number of tasks in the system is larger than the number of agents causing agents to attend to a stream of tasks [88]. However, a distinction can be made between online and offline MAPD problems. For offline problems, all tasks are known upfront, whereas for online problems, tasks can arrive at any time to the system [88].

As explained before, the MAPD problem requires both task assignment and path finding. In this section, an overview of research will be given that make use of decoupled algorithms in which agents first assign themselves to tasks and thereafter computes its own collision-free path based on the provided global information [88]. Distinctions will be made between offline and online algorithms

7.2.1. Greedy Task Allocation Combined With SIPP (Offline)

In the work done by Yakovlev et al. [148], the performance of various task allocation strategies is reviewed, combined with the Safe Interval Path Planning (SIPP) algorithm (refer to subsection 6.2.1 and subsection 6.2.2 for more details on SIPP). All task allocation strategies are based on the principle of iterating over the robots and assigning each robot a task using a `select()` function. For this function, three different variations are implemented, where the most basic version randomly assigns a sub-goal out of the set of unexecuted sub-goals (`random`). A more intelligent approach makes use of the Euclidean distance between all unassigned sub-goals and the location of the current robot and greedily assigns the target for which the distance is minimal (`without obs`). Finally, in order to be able to take obstacles into account, the cost of executing a certain task is based on finding the shortest path between the start location of the robot and the sub-goal under consideration using the A* algorithm (**`with`** `obs`) (refer to subsection 6.1.3 for more details on the A* algorithm) [148].

For a varying number of agents, the effectiveness of the MAPD algorithm is tested on 100 scenarios in which 164 unique start, sub-goal and goal locations are randomly distributed over a grid-like warehouse environment. In addition, two variations were compared: one in which the agents are allowed to make cardinal moves and one in which any-angle moves are allowed. In Figure 7.6, the results are shown for the various configurations and scenarios. As can be seen, including estimates of path costs based on path finding algorithms such as A* in the allocation phase, results in significant improvements in terms of makespan. Furthermore, run times remain within 1 second for 32 agents when making use of the most sophisticated task allocation strategy, arguably allowing for real-time execution of the system.

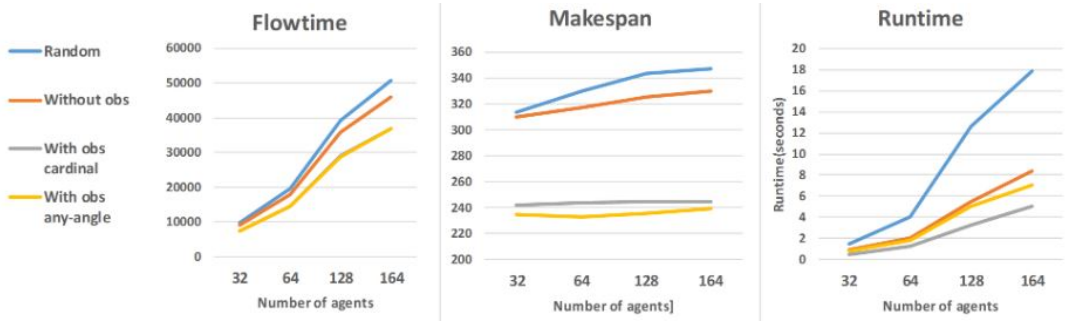


Figure 7.6: Experimental results showing effectiveness of MAPD algorithm for varying number of agents and task allocation strategies, based on the work of Yakovlev et al. [148].

7.2.2. TA-Prioritized and TA-Hybrid (Offline)

One of the first studies to define the MAPD problem as a lifelong version of the MAPF, is the work done by Ma et al. [88]. In this research, they present a selection of online MAPD algorithms, which will be discussed in more detail in the following section (subsection 7.2.3). Although the original algorithms were designed for online problems, they can also be applied to offline problems. In the research done by Liu et al. [149], one online algorithm in particular called CENTRAL (originally implemented by Ma et al. [88]), is altered to deal with the offline version of the problem and improved with respect to task planning, path planning and deadlock avoidance [149]. In its original version, CENTRAL iteratively assigns tasks using the Hungarian method and subsequently plans paths using Conflict-Based Search (CBS) (for more details on CBS, refer to subsection 6.1.3). Two variations are suggested by Liu et al. [149] and are Task Assignment - Prioritized (TA-P) and Task Assignment - Hybrid (TA-H).

For both TA-P and TA-H, the task assignment part is based on solving a special TSP on a directed weighted graph to find a suitable task sequence for each agent [149]. The idea of using a directed weighted graph and formulating the task allocation problem as an TSP is based on previous work [150, 151]. In this model, two objectives are used: the primary objective is to minimize the largest execution time of all task sequences and the secondary objectives focuses on minimizing the sum of all execution times of all task sequences. Rather than using travelling distance, minimizing for execution time also accounts for possible waiting for the release of tasks. Note that in the task assignment part, interaction with other agents and thus, collisions are not taken into account.

The solution of the TSP can be represented by a Hamiltonian cycle, containing each agent and

task vertex only once (Figure 7.7). Thus, for a set of N agents, the Hamiltonian cycle consists of N parts, where each part contains an agent vertex, a sequence of task vertices and another agent vertex. These N parts can be converted to N task sequences, where the sum of the edge weights of each part provides for a lower bound on the execution time. The reason that the sum only is a lower bound, can be explained by the fact that only the release time for the first task is taken into account. Thus, waiting for other tasks is not included. In addition, as mentioned before, rerouting to avoid collisions with other agents is not incorporated in the model as well [149].

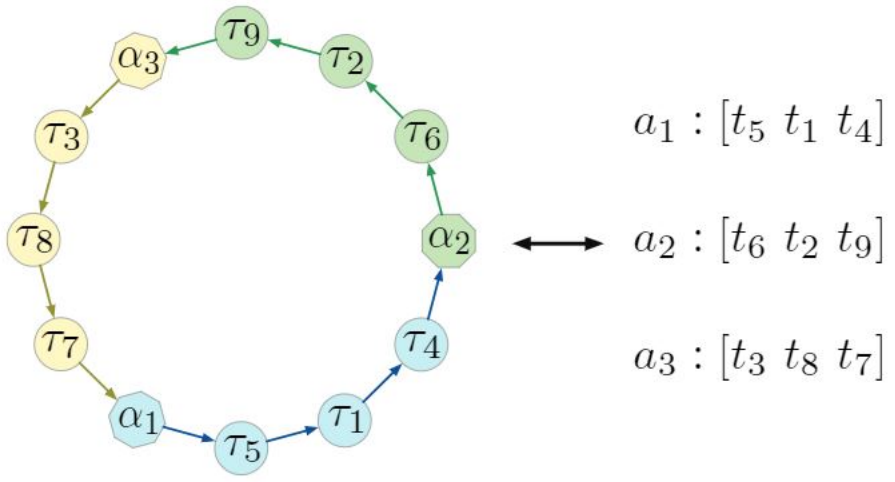


Figure 7.7: Representation of an example solution to the TSP model used in the work by Liu et al. [149] as a Hamiltonian cycle, containing three agent vertices and nine task vertices. On the right, three corresponding task sequences are shown for all three agents. Taken from [149].

With respect to path planning in the TA-P approach, this is based on the prioritized planning approach suggested by Van den Berg and Overmars [152]. In their work, they plan paths with the A* algorithm for agents using a priority order, where priority is based on estimated execution times in decreasing order (agents with larger execution times are given priority). After having planned a path for an agent, the paths for next agents are not allowed to collide with the already existing paths. Therefore, the planning of paths for agents with larger execution times is subjected to fewer constraints, which according to the authors, may result in a smaller makespan [152]. In TA-P, the authors suggest the following improvement with respect to the work of Van den Berg and Overmars: instead of using an estimate for the execution times to base the priority order on, the actual execution time is used. This actual execution time is based on the already existing paths in the model, instead of the estimate that does not take interaction with other agents into account [149].

In the TA-H approach, agents are divided into two groups for which paths are planned differently: new task agents (agents that are performing a task) for which paths are planned from their current location to the delivery location of their current task; and free agents (agents that have not yet arrived at the pickup location of their next task) for which paths are planned from their current location to the pickup location of their next task. For new task agents, paths are planned by modelling the problem as a MAPF instance and using an improved version of CBS (Improved Conflict-Based Search (ICBS)). Once free agents have arrived at the pickup location of their next task (and thus become a new task agent), the path is planned as described before and the resulting path remains unchanged until the new task agent reaches the delivery location of the task [149].

As soon as a new task agent has reached the delivery location of their previous task, the agent becomes a free agent. Then, by modelling the problem of finding a sub-path to the next pickup location as an AMAPF instance (thus, allowing for swapping of future tasks between agents), a polynomial-time min-cost max-flow algorithm is used to find the path for the agent to the pickup location of its next task. Replanning of paths for free agents is done at every time step in which the set of free agents changes [149].

Finally, both TA-P and TA-H approaches make use of the deadlock avoidance method ”reserving dummy paths”. A dummy path is ”a path with minimal travel time to the parking location of the agent” [149] and guarantees that there is at least one movement possible for the agent from its current location,

thus, avoiding deadlocks [149]. For this to hold, a dummy path should avoid collisions with the paths of all previous agents and in addition, a dummy path cannot contain parking locations of all other agents. In TA-P, a new dummy path is planned every time that a new sub-path is planned. Thus, the new dummy path replaces the non-final dummy path (the final dummy path is the only path that the agent will actually follow, since this path leads from the final delivery location back to its parking location). Note that when using TA-P, a new sub-path is immediately planned after a (non-final) dummy path has been found. However, for TA-H, this is not the case: subsequent sub-paths are planned only once agents switch from type (new task agent to free agent or v.v.). Therefore, all dummy paths are stored and collisions with this path should be avoided [149].

Table 7.1: Results of comparison of different MAPD algorithms performed on small warehouse grid (Figure 7.8), where "f" represents the task frequency per time step [149].

		CENTRAL		**TA-Prioritized**		**TA-Hybrid**		**TSP**
f	**agents**	makespan	runtime (sec)	makespan	runtime (sec)	makespan	runtime (sec)	makespan
1	10	1155	51	1094	10	1087	13	1062
	20	661	122	608	21	612	38	590
	30	553	180	546	35	528	118	525
	40	555	482	534	44	525	182	525
5	10	1117	110	1054	10	1039	10	1020
	20	603	218	551	19	549	19	519
	30	424	257	370	29	377	21	345
	40	332	345	289	41	285	31	268
10	10	1130	64	1036	10	1045	10	1017
	20	589	131	559	19	541	17	512
	30	422	221	369	19	373	21	343
	40	344	356	294	40	279	29	261
500	10	1101	50	1045	10	1037	11	1016
	20	580	124	535	19	539	14	508
	30	421	720	370	29	362	21	338
	40	357	976	275	39	280	22	254

In Table 7.1, the results of an experiment using the warehouse grid as visualized in Figure 7.8 are shown. For different task frequencies ranging between 1 and 500 (equivalent to releasing all tasks at the beginning) and a varying number of agents, both makespan and total run time (in seconds) of three different MAPD algorithms are compared with each other. Note that for TA-P and TA-H, the run time only reflects the run time of the path planning part. The task assignment is performed separately using the TSP solver, which was ran for 1,000 seconds after which the best Hamiltonian cycle has been chosen. Although both TA-Prioritzed and TA-Hybrid outperform CENTRAL in terms of makespan, the associated run times for the path planning cannot be implemented in a system that requires real-time operation.

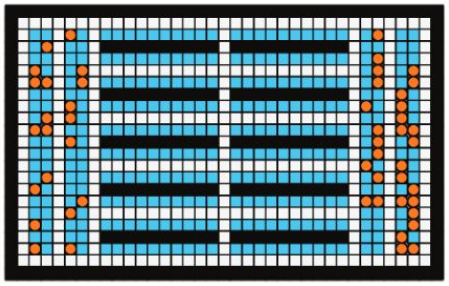


Figure 7.8: Representation of small warehouse on 21 x 35 grid. Black cells represent object, blue cells task endpoints and orange cells are non-task endpoints. Taken from [149].

7.2.3. Token Passing (TP) and Token Passing with Task Swaps (TPTS) (Online)

As mentioned earlier, Ma et al. [88] were among the first ones to define the MAPD problem as a lifelong MAPF instance. In their work, they present two novel online algorithms, both based on the idea of cooperative A*, where agents plan their path one after each other [153]. The first decoupled algorithm is called Token Passing (TP) and makes use of a so-called "token", that represents a shared block of memory containing information on the current paths of all agents, the task set (all tasks that have no agent assigned) and agent assignments. In each time step, any new tasks in the system are added to the task set in the token. If an agent has reached the end of its current path (that is, reached a destination location of a task), it requests the token. The token is passed to all agents one by one that request the token. From the token, each agent tries to find a task in the task set such that the path of any other agent does not end up in the pickup or delivery location of the task under consideration. If such a task is available in the task set, the agent assigns itself to the task with the minimal path costs (found using A*). Subsequently, a collision-free path from the current location of the agent is found through the pickup location to the delivery location of the task. However, if no task is available for which no other agent's path ends up in the pickup or delivery location of the task, the agent does not assign itself to a task in the time step. It either stays in its current location (if the current location is not a delivery location of a task) or finds a collision-free path from its current location to an endpoint that is neither a delivery location of a task nor an endpoint of any other path of the other agents [88].

A more effective version of TP is Token Passing with Task Swaps (TPTS) [88]. Instead of using a task set that contains all unassigned tasks, in TPTS the task set consists of all unexecuted tasks, allowing agents that swap tasks that are not yet executed. Similar to TP, agent r_i requests the token once it has reached the delivery location of its last task and tries to assign itself to a task t in the task set $\mathcal{T}$. If agent r_i finds a task that is already assigned to agent $r_{i'}$ but it reaches the pickup location in fewer time steps than agent $r_{i'}$, it assigns itself to the task and unassigns agent $r_{i'}$ from the task [88].

Table 7.2: Results of comparison of different MAPD algorithms performed on small warehouse grid (Figure 7.8), where "f" represent the task frequency per time step [88].

f	agents	CENTRAL		TP		TPTS	
		makespan	runtime (ms)	makespan	runtime (ms)	makespan	runtime (ms)
0.2	10	2,513	92.69	2,532	0.13	2,532	1.86
	20	2,513	493.83	2,540	0.26	2,520	9.82
	30	2,513	1,225.62	2,546	0.25	2,527	21.57
	40	2,511	2,246.66	2,540	38.88	2,524	27.49
1	10	1,143	141.32	1,198	0.20	1,182	0.37
	20	673	279.52	757	1.03	706	2.80
	30	557	446.38	607	2.81	561	8.45
	40	556	1,159.76	624	2.14	563	39.54
5	10	1,105	126.19	1,162	0.20	1,165	0.41
	20	594	350.13	655	1.02	645	1.68
	30	426	595.04	478	2.22	474	8.85
	40	334	864.56	418	4.15	396	12.31
10	10	1,090	125.55	1,163	0.22	1,172	0.40
	20	607	379.53	643	1.09	645	1.87
	30	414	593.89	526	1.98	491	10.82
	40	341	899.81	407	1.65	389	12.62

Table 7.2 show the results of experiments performed by Ma et al. [88] comparing the different MAPD algorithms on the warehouse presented in Figure 7.8 for a sequence of 500 tasks, released with different frequencies (f) and using a varying number of agents. Both makespan and run times per time step (in ms) are reported on. As can be seen, both TP and TPTS outperform CENTRAL when it comes to

scalability and computational efficiency, with run times per time step under 5 miliseconds for TP and under 40 miliseconds for TPTS [88]. Although CENTRAL outperforms both online MAPD algorithms in terms of makespan [88], the marginal increase in makespan for TP and TPTS might not be significant in the context of TaxiBotting, that require real-time operations. According to the authors, TP is the best choice when real-time computation is of primary concern, since this algorithm produces efficient outcomes for MAPD problems with hundreds of agents and tasks. However, since the scale of the TaxiBotting problem is expected to be less than hundreds of agents and tasks, implementation of the more efficient TPTS algorithm can also be considered.

Token Passing Approach for Multi-Item Delivery

Instead of assuming a single-item capacity for agents, Contini and Farinelli [154] develop a decentralized algorithm based on a token passing approach where robots can deliver multiple items in a single travel. Although TaxiBots will not have a multi-item capacity, the multi-item token passing approach is still of interest when clustering of tasks is considered (refer to subsection 5.4.3). In that case, the clustered tasks and corresponding pickup and delivery locations can be seen as a sequence of tasks (or locations) to be visited by the TaxiBot.

Contini and Farinelli [154] develop an iterative algorithm that is based on the same principles used in the token passing approach used by Ma et al. [88]. From the initial set of tasks $\mathcal{T}$, a set of multi-item partitions $\mathcal{P}$ is made. Since clustering of tasks is already elaborately described in subsection 5.4.3, focus in this section will be on the working principle of the iterative algorithm, which is visualized in Figure 7.9. As an input, a multi-item task set $\mathcal{P}$ is taken. Based on the current paths, a token containing the multi-item task set $\mathcal{P}$, is sent to the agent with the shortest current path. This agent chooses a task p_i from the set of multi-item tasks $\mathcal{P}$ and tries to find a path from its current location to all endpoints related with task p_i (pickup location s_i and delivery locations g_i^1, g_i^2, ...), back to the home endpoint of the agent. If a valid collision-free path is found using a variation of the A* algorithm, the path is stored in the token and the token is sent to the agent that now has the shortest path. If a valid path could not be found, the task is re-inserted in the token, which is sent to the next agent for finding a valid path. In the example, first the red agent chooses a task from the set $\mathcal{P}$. After having sent the token to the blue agent, the blue agent chooses a task and sends the task to the cyan agent. The cyan agent picks one of the remaining tasks and finds a collision-free path. Since it is the blue agent who has the shortest path after all agents have had their first pick, the cyan agent sends the token to the blue agent for picking the final task.

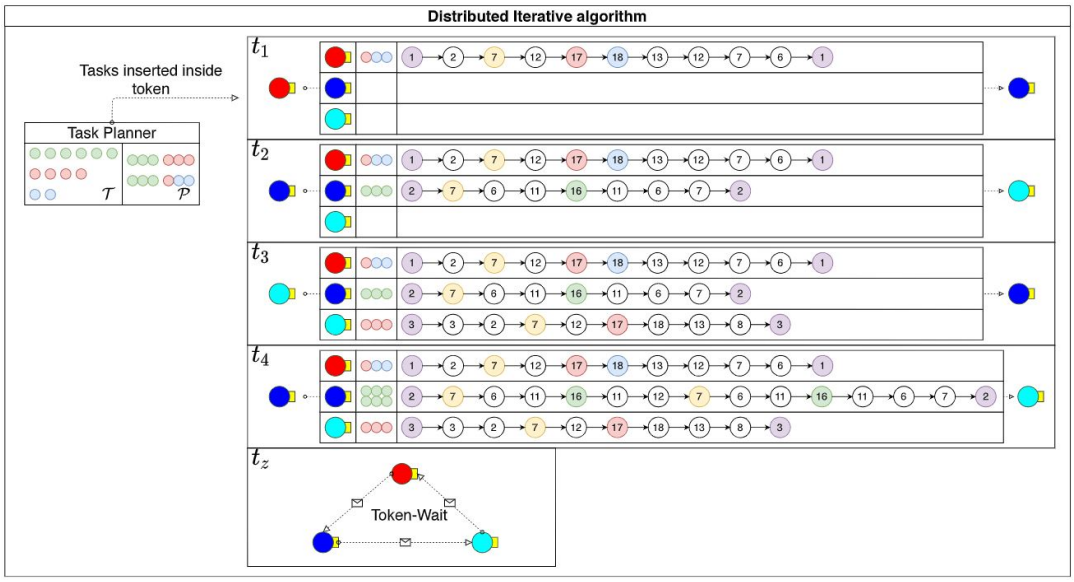


Figure 7.9: Visualization of the working principle of the iterative algorithm based on the token passing approach as described in the work of Contini and Farinelli [154].

7.2.4. Concluding Remarks on Decoupled MAPD Approaches

In the previous sections, three different frameworks have been evaluated that solve the MAPD problem in a decoupled manner, meaning that task allocation and path finding are performed separately. Based on the system requirements as formulated in subsection 4.6.2, a preferred framework for implementing in the remainder of this thesis will be chosen.

In this reasoning, one system requirement can be considered decisive: requirement item 6 related to real-time operations and modelling the problem in an online setting. With respect to real-time operations, it is of importance that run times per time step remain significantly below 1 second. Strictly speaking, the only approach that allow for this is Token Passing (TP), although research presented in section 7.3 will point out that Token Passing with Task Swaps (TPTS) is also capable of finding solutions with run times well below 1 second. In addition, both TP and TPTS are the only methods that are specifically designed for online problems.

Although it is shown that TP is more computationally efficient than TPTS, TPTS produces better results in terms of makespan. Therefore, it is proposed to use TPTS in the final model if computational times allow for it. Otherwise, TP is used to combine task allocation and path finding in a decoupled manner. Note that anyways, in the baseline model no reallocation of tasks (or task swaps) will be considered. Therefore, TP will be implemented anyways, possibly extended with task swaps if resources allow for it.

7.3. Coupled Approaches to Solve MAPD Problems

After having reviewed a selection of decoupled approaches to solving the MAPD problem, attention will be paid to solving techniques that integrate task allocation and path finding in a coupled manner. Similarly as for decoupled approaches, a distinction will be made between offline and online algorithms.

7.3.1. Multi-Agent Pickup and Delivery with Task Deadlines (MAPD-TD) (Offline)

In a very brief paper, Wu et al. [126] propose a priority-based framework that integrates task assignment with path planning for offline problems. In addition, they assume that a task t_i is characterized by a deadline d_i. The goal is to maximize the number of tasks completed before the associated deadline.

In section 5.3, a brief outline has been provided on how the next task to be assigned is selected. In summary, a measure flexibility f_i is used that represents the available time between the earliest time step any agent r_k can finish the task and the corresponding task deadline d_i. The most urgent task i^*, that is, the task with the lowest flexibility f_i, is selected to be assigned next.

Subsequently, the agent k^* is found to which task i^* is assigned. Let τ_k be the time step that agent r_k arrives at the delivery location of the last task in its assignment set a_k. For all agents that are capable of executing task i^* before its deadline, the agent is chosen for which Equation 7.1 holds. That is, the agent for which its current completion time of all tasks assigned τ_k is closest to the earliest time step c_{k,i^*} it can finish the most urgent task i^*, is assigned to the task [126].

$$k^* = \underset{c_{k,i^*} \leq d_i}{\operatorname{argmin}} (c_{k,i^*} - \tau_k) \tag{7.1}$$

After having assigned task i^* to robot k^*, this agent plans its collision-free path using an A* algorithm, taking into account the paths that are previously planned. Based on its new route, new values for $c_{k,i} \forall i \in \mathcal{T}$ have to be calculated.

In Table 7.3 [126], the average success rate (number of tasks completed before their deadline) is shown when testing the algorithm on a small warehouse (Figure 7.8) for a varying number of agents. The number of tasks equals ten times the number of agents. Furthermore, the deadline of each task is based on the parameter ϕ where the deadline of each task is set to be $(1 + \phi)$ times the completion time of the task with a hypothetical path (ignoring any potential collisions). As expected, the larger ϕ, the more tasks are completed before their deadline. Although more results on makespan, service time or run time are unfortunately not reported on, the idea of using flexibility as a measure to account for deadlines of tasks is expected to be suitable for the problem of assigning TaxiBots to aircraft before a certain time as well. However, the presented framework will have to be adjusted for an online setting.

Table 7.3: Results of the MAPD-TD algorithm representing the average success rate for 10 runs of an experiment in a warehouse environment (Figure 7.8) for a varying number of agents and tasks. Taken from [126].

		ϕ		
agents	**tasks**	0.0	0.1	0.25
10	100	0.8670	0.9380	0.9890
20	200	0.8090	0.8920	0.9810
30	300	0.7760	0.8580	0.9630
40	400	0.7378	0.8198	0.9402

7.3.2. Lifelong Enhanced Conflict-Based Search with Task Assignment (ECBS-TA) (Online)

In a study done by Hönig et al. [155], the CBS framework as explained in subsection 6.1.3 has been extended to include task assignment. The resulting algorithm, Conflict-Based Search with Task Assignment (CBS-TA), is capable of simultaneously assigning tasks and finding paths for agents in several MAPF instances. In their work, the authors show that their algorithm is both optimal and complete. In addition, the CBS framework allows for the extension to ECBS, resulting in a scalable, bounded suboptimal solver called Enhanced Conflict-Based Search with Task Assignment (ECBS-TA). In this section, the CBS-TA and ECBS-TA algorithms will be explained. Then, the extension suggested by Duchateau [138] to make the ECBS-TA suitable for online, lifelong MAPD problems will be elaborated on.

As explained in subsection 6.1.3, the CBS framework consists of a low-level and high-level search. In the low-level, the algorithm finds paths for each agent based on the constraints set in the high-level. The high-level detects conflicts and tries to resolve them. For the CBS-TA, two major adjustments are made in the high-level search with respect to the original CBS framework: (1) instead of using a search tree, a search forest is used; and (2) instead of using a single root node, CBS-TA starts with a root node representing the best possible assignment, but expands new root nodes only on demand (when the best possible assignment results in conflicts between agents) [155].

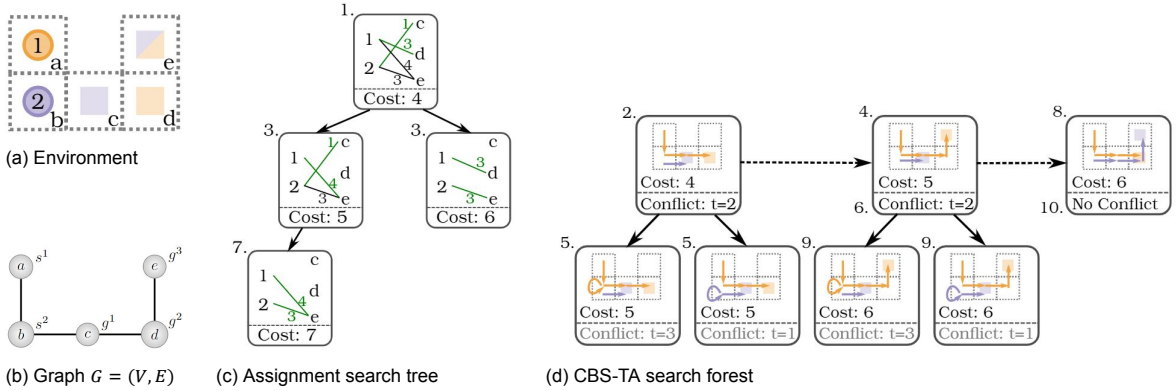


Figure 7.10: Visualization of working principle of CBS-TA. Taken from [155].

In Figure 7.10 [155], an example with two agents ($N = 2$) and three tasks ($M = 3$) is visualized to explain the working principle of CBS-TA (Figure 7.10a). The assignment matrix A and corresponding cost matrix C are shown in Equation 7.2 and Equation 7.3 respectively. Based on the cost matrix, the best assignment is found. In the work of Hönig et al. [155], the Hungarian method is used for this, but in principle, any task allocation approach can be used [155]. In the example, agent 1 is assigned to target d and agent 2 is assigned to target c. This assignment is added to the assignment tree (step 1, Figure 7.10c) and a root node is created in the search forest (step 2, Figure 7.10d).

$$A = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} c & d & e \end{array} \\ \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix} \end{array} \tag{7.2}$$

$$C = \begin{array}{c} \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} c & d & e \end{array} \\ \begin{pmatrix} \infty & 3 & 4 \\ 1 & \infty & 3 \end{pmatrix} \end{array} \tag{7.3}$$

After path construction for both agents based on the assignment, a conflict at $t = 2$ is found. Therefore, the next-best assignment is added to the search tree (step 3). This is done by making use of the

$constrainedAssignment(I, O, C)$ function, where I is the set of assignments that must be part of the solution, O the set of assignments that must be excluded from the solution and C the cost matrix. In the example, two possible successors are generated: in the first one, the assignment of agent 1 to target d is disallowed ($O = \{1 \mapsto d\}$); and in the second one, the assignment of agent 2 to target c is disallowed ($O = \{2 \mapsto c\}$), while enforcing the assignment of agent 1 to target d ($I = \{1 \mapsto d\}$). The option with the lowest cost ($[1 \mapsto e, 2 \mapsto c]$) is added as a new root node to the search forest (step 4) and paths are found again.

Then, we first try to resolve the conflicts found in for the first root node, by adding additional nodes to the search forest (step 5). Two child nodes are created: one in which agent 1 is restricted to be at node c at $t = 2$; and another one in which agent 2 is restricted to be at node c at $t = 2$. Again, conflicts are found and thus, the new root node is expanded (step 6) and the search tree is further extended with another possible assignment taking the constraints from previous steps into account (step 7). Whenever a root node is expanded, a new root node is added with the next-best assignment ($[1 \mapsto d, 2 \mapsto e]$, found in step 3). Similarly to before, we first try to resolve the conflicts found when constructing paths for the second root node (step 9). Since the child nodes also contain conflicts, paths are constructed for the third root node (step 10). Finally, no conflicts are found and thus, the third root node is selected as solution since it results in an optimal task assignment while guaranteeing collision-free paths for both agent 1 and agent 2.

When considering ECBS instead of CBS, a few adjustments are made to the framework to arrive at the ECBS-TA algorithm. As explained in subsection 6.2.2, the ECBS algorithm uses a focal search algorithm (instead of A*) for both the low- and high-level searches in the search. In a FOCAL list all nodes are stored that are within a bound ω of the lowest cost solution in the search forest. Then, using a secondary heuristic that estimates the number of conflicts, focal search is used to find the entry in FOCAL with the lowest estimated number of conflicts [155].

Due to the suboptimality bound ω, slack is generated in the search, allowing for more flexibility in when to add new root nodes. In the work of Hönig et al. [155], three variants are considered:

- **CBS-TA-style**: similarly as explained before, an additional root node is added every time that a root node is expanded.
- **MaxRoot**: using this approach, the idea is to add as many root nodes as possibly useful for a given bound ω. Based on a lower bound LB, which is the lowest cost among the already expanded nodes, all root nodes are added for which the cost is smaller than $\omega\mathrm{LB}$. However, this method is deemed impractical for problems with large N and M, due to the large increase of potential assignments.
- **MinRoot**: contrary to MaxRoot, this approach aims at adding the least amount of root nodes as possible. Based on the threshold $\omega\mathrm{LB}$, a root node is added when the lowest cost node in the assignment search tree exceeds the predefined threshold.

In his thesis, Duchateau implemented a lifelong version of ECBS-TA using a MinRoot approach (due to scalability reasons) to model the pickup and delivery of bags in an airport baggage hall using AGVs [138]. This was done by first invoking the ECBS-TA algorithm that provides for a coupled integration of task allocation and path finding as explained before. When at least one AGV received a new task, the path finding ECBS algorithm was invoked. However, since the allocation of new tasks is strictly not coupled to path finding, it is better to speak of a hybrid approach instead of an entirely coupled approach. In addition, tasks were allocated by making use of an improved auction algorithm instead of the original Hungarian method that was proposed by Hönig et al. [155].

The lifelong version of ECBS-TA is compared with two decoupled MAPD algorithms. Both MAPD algorithms make uses of auctions to allocate tasks: the first one follows a simple CNP model, while in the second algorithm, reselling and swapping of tasks is also taken into account using an IDMB auctioning approach [138]. For 30 minutes of real-time operations in a system with 60 AGVs and total capacity of 3600 bags/hour, the results as shown in Table 7.4 are obtained, including run time per time step (in seconds), service time (the time between a bag entering the system and being delivered, in time steps), the total number of bags handled and the delay (difference between calculated shortest path using A* and the actual length of the path, in time steps). As can be seen, the coupled MAPD algorithm performs consistently better on the three system KPIs. Although the run times are slightly higher per

time step when compared with the two decoupled algorithms, they remain well below 1 second per time step, thus, allowing for real-time operations [138].

Table 7.4: Results obtained by Duchateau [138] when comparing two decoupled MAPD algorithms with the coupled ECBS-TA algorithm in a lifelong setting where 60 AGVs need to deliver bags at a system capacity of 3600 bags/hour [138].

	Runtime (sec)	**Service time (time steps)**	**Number of bags handled**	**Delay (time steps)**
CNP + ECBS	**0.49**	222.7	1264	0.27
IDMB + ECBS	0.59	219.8	1274	**0.20**
ECBS-TA	0.67	**215.3**	**1282**	**0.20**

7.3.3. Lifelong Relative Regret-Based Marginal-Cost Based Task Assignment (RMCA(r)) (Online)

One of the first studies to develop a fully coupled MAPD algorithm that allows for scalability without losing solution quality, is the work done by Chen et al. [156]. They propose a MAPD algorithm that simultaneously performs task assignment and path planning by making use of a current assignment set $\mathcal{A}$ and a priority heap $\mathcal{H}$. In the current assignment set $\mathcal{A}$, the following items are stored: a set of assignments a_k characterized by an ordered sequence o_k of pick-up and drop-off actions assigned to each robot $k \in \mathcal{R}$; the current collision-free path of robot r_k; and the Total Travel Delay (TTD) associated with the assignments a_k to be performed by r_k. Note that the TTD is defined as the difference in the shortest path for robot r_k to pick-up the task as soon as it arrives in the system and drop it off at the intended location and the actual time between the task entering the system and being delivered.

The priority heap $\mathcal{H}$ contains a set of potential assignment heaps h_i for each unassigned task $i \in \mathcal{T}$. A potential assignment heap h_i contains for task i all possible assignments pa_k^i of task i to robot r_k for each $k \in \mathcal{R}$. The potential assignment pa_k^i stores information on an updated version of o_k, robot r_k's path and associated path cost if task i would be added to robot r_k's current assignment set a_k. Assignment of tasks from the set of unassigned tasks $\mathcal{P}^u$ initialized as $\mathcal{P}$ will continue until all tasks are assigned.

At the start of the algorithm, the set of assigned tasks $\mathcal{A}$ is empty. The priority heap $\mathcal{H}$ is initialized by including one potential assignment heap h_i for each task, where each heap h_i consists of the potential assignments pa_k^i for all $k \in \mathcal{R}$ (see also Figure 7.11). Within each potential assignment heap h_i, all potential assignments pa_k^i are sorted based on increasing order of marginal costs. The ordering of h_i within the priority heap $\mathcal{H}$ will be elaborated on below.

While the set of unassigned tasks $\mathcal{P}^u$ is not yet empty, the algorithm keeps assigning the top potential assignment pa_k^i of the top potential heap h_i, meaning that task i is assigned to robot r_k. Therefore, pa_k^i is added to assignment set a_k, h_i is deleted from heap $\mathcal{H}$ and task i is deleted from $\mathcal{P}^u$. Since the action sequence, path and associated path costs for robot r_k will change due to the addition of task i, all other potential assignments for robot r_k in any $h_j, j \in \mathcal{P}^u \setminus \{i\}$ will have to be recalculated. The potential assignments for all other agents $k', k' \neq k, k' \in \mathcal{R}$ in principle will not change. However, collisions may exist between their path and the new path of agent r_k. Therefore, for the first v elements in each potential assignment heap h_i, the paths are checked on collisions and updated if necessary using the function `updateHeapTop()`. Note that this is a faster method than checking all paths for agents other than k and saves considerable time, while only slightly influencing the task assignment outcome [156].

In their work, Chen et al. [156] propose different task selection methods to base the ordering of heaps h_i within the priority heap $\mathcal{H}$ on. For their lifelong experiment, a Regret-Based Marginal-Cost Based Task Assignment (RMCA) algorithm is used. This algorithm bases the next task i^* to be allocated on the difference in marginal cost of allocating task i in the route of the best robot and the second best robot. Task i is then assigned to the robot that has the lowest marginal cost.

The robot that has the lowest increase in marginal cost when inserting task i from the set $\mathcal{P}^u$ in its current route, is denoted $r_{k_1}^*$, which can be mathematically expressed using Equation 7.4, where q_1^* and q_2^* are the action sequences for picking up and delivering task i^* and $t(o_k)$ expresses the shortest path to all tasks in the ordered sequence o_k. The operator $\{t((o_k \oplus_{q_1} s_i) \oplus_{q_2} g_i)$ defines the insertion of s_i at the q_1th position of the current route and subsequently the insertion of g_i at the q_2th position of

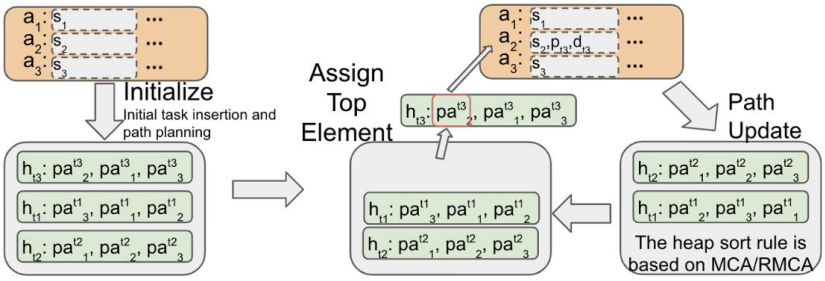


Figure 7.11: Visualization of the working principle of the algorithm that performs task assignment and path planning simultaneously as proposed by Chen et al. [156]. The assignment of three tasks $\{1, 2, 3\}$ to three agents $\{1, 2, 3\}$ is considered. The grey box represents the priority heap $\mathcal{H}$, the green box is the potential assignment heap h_i, the orange box is the current assignment set $\mathcal{A}$, the dashed box is the ordered action sequence o_k for each agent i, including its starting location s_i and pick-up and delivery location p_i and d_i respectively (if assigned any). Taken from [156].

the current route. In a similar fashion, the second-best robot $r^*_{k_2}$ can be defined using Equation 7.5.

$$(r^*_{k_1}, q^*_1, q^*_2) = \underset{\substack{r^*_{k_1} \in \mathcal{R} \\ 1<q_1\leq|o_k|, \\ q_1<q_2\leq|o_k|+1}}{\operatorname{argmin}} \{t((o_k \oplus_{q_1} s_i) \oplus_{q_2} g_i) - t(o_k)\} \tag{7.4}$$

$$(r^*_{k_2}, p^*_1, p^*_2) = \underset{\substack{r^*_{k_2} \in \mathcal{R}\setminus\{r^*_{k_1}\} \\ 1<p_1\leq|o_k|, \\ p_1<p_2\leq|o_k|+1}}{\operatorname{argmin}} \{t((o_k \oplus_{p_1} s_i) \oplus_{p_2} g_i) - t(o_k)\} \tag{7.5}$$

The ordering of potential assignment heaps $h_i, i \in \mathcal{P}^u$ is done based on relative regret in increasing order, where relative regret of task i^* is expressed using Equation 7.6.

$$i^* = \underset{i\in\mathcal{P}^u}{\operatorname{argmax}} \frac{t((o^*_{k_2} \oplus_{p^*_1} s_i) \oplus_{p^*_2} g_i)}{t((o^*_{k_1} \oplus_{q^*_1} s_i) \oplus_{q^*_2} g_i)} \tag{7.6}$$

After having defined the potential assignment pa^{i*}_k that is to be included in assignment set a_k, an `insert()` function is used to select the combination of q^*_1 and q^*_2 that minimizes the increase in marginal TTD (ignoring any potential collisions with other agents). Afterwards, the `planPath()` function is used to plan a collision-free path for the defined o^i_k (using A*) and calculates the actual marginal increase in terms of TTD. Finally, the function `updateHeapTop()` is called to update any potential assignments for agents with paths conflicting with the newly added path.

In order to improve the initial solution, the authors propose to make use of a solution improvement strategy by destroying already allocated tasks from the solution and reassigning them using RMCA. This is done until time out. The authors propose three different destroying strategies [156]:

- **Destroy Random**: a group of tasks from all assigned tasks is randomly selected, removed from the solution and re-assigned using RMCA.
- **Destroy Worst**: a group of tasks from all assigned tasks that have the worst TTD is selected, again removed and re-assigned using RMCA. By means of a tabu list, it is recorded which tasks have been selected to avoid selecting them again. Once all tasks are selected once, the tabu list is cleared and all tasks can be selected again.
- **Destroy Multiple**: in this strategy, a group of agents that have the worst sum of TTD is selected. For each agent, randomly one assigned task is removed from its current assignment set and re-assigned. Again, use is made of a tabu list to prevent selecting the same group of agents multiple times.

In the evaluation, Chen et al. [156] compared the performance of the proposed coupled MAPD algorithm with CENTRAL (a centralized algorithm developed by Ma et al. [88], refer to subsection 7.2.2) and TPTS (refer to subsection 7.2.3), which are two decoupled MAPD algorithms. Experiments are performed on the grid presentation of a warehouse (Figure 7.8) for a total of 500 tasks, that are released

with different task frequencies f and have to be executed by a varying number of agents. The improvement strategy "Destroy Random" is used, with group size of 5 and improvement time of 1 second for the entire run. In Table 7.5, the results on TTD, makespan and runtime per time step (in seconds) are shown.

As can be seen, RMCA(r) significantly improves TTD and makespan when compared with CENTRAL and TPTS for almost all configurations. Although TPTS in general has slightly better run time performance for higher task frequencies, RMCA(r) stays well below the limit of 1 second per time step that allows for real-time operations for every instance but one (task frequency of 2 with 50 agents) [156], making this approach very suitable to apply in the context of TaxiBotting.

Table 7.5: Results of comparison of different MAPD algorithms performed on small warehouse grid (Figure 7.8), where "f" represents the task frequency per time step. The values in bold represent the smallest values for the specific row. Taken from [156].

		Central			**TPTS**			**RMCA(r) with Random Improvement**		
f	**agents**	TTD	makespan	runtime (sec)	TTD	makespan	runtime (sec)	TTD	Makespan	runtime (sec)
0.2	20	4365	2528	0.364	3645	2528	**0.103**	**3138**	**2526**	0.205
	30	3864	2527	0.762	3002	2526	0.242	**2729**	**2525**	**0.208**
	40	3572	2527	1.300	2646	2525	0.442	**2297**	**2523**	**0.210**
	50	3394	2525	1.945	2456	2524	0.710	**2176**	**2523**	**0.214**
2	20	75294	**610**	0.125	82734	639	**0.022**	**65938**	626	0.489
	30	37327	446	0.284	47252	490	**0.099**	**30317**	**436**	0.705
	40	19930	376	0.426	30491	413	**0.273**	**13945**	**344**	0.884
	50	11185	328	**0.615**	21853	377	0.660	**6279**	**300**	1.022
10	20	116357	**587**	0.421	125374	626	**0.025**	**106290**	624	0.142
	30	76934	**419**	1.062	86267	462	**0.086**	**68166**	435	0.210
	40	56896	**337**	2.426	66171	383	**0.238**	**49140**	338	0.284
	50	45170	288	2.828	55409	339	0.559	**38050**	**280**	**0.362**

7.3.4. Concluding Remarks on Coupled MAPD Approaches

In order to determine the coupled approach to implement in the modelling part of this thesis, another look at the systems requirements as mentioned in subsection 4.6.2 is taken. Similarly as for decoupled approaches, it is of importance that the proposed framework is capable of dealing with the problem in an online setting (item 6). Although the first algorithm as proposed in subsection 7.3.1 is originally designed for modelling offline problems, the framework can be easily adapted to an online setting due to its similarities with the third (online) algorithm using priority heaps. Both algorithms integrate task allocation with path finding by first determining path costs for all agent-task combinations, ignoring all other agents. Then, based on some algorithm-specific ordering, the best agent-task combination is chosen, upon which all other agents update their path costs, avoiding collisions with the path already planned. In the first algorithm, this procedure is performed once at the beginning of the problem, but using some characteristics of the third algorithm, this can easily be adapted to be performed in an online setting.

However, the approach used by Wu et al. [126] is unfortunately not elaborated on in detail, possibly leading to additional time needed for implementation. In addition with the fact that results on solution quality are neither provided, this approach is not preferred. This leaves us with the hybrid version of ECBS-TA and the algorithm using priority heaps. Due to the fact that ECBS-TA is officially not a coupled approach, combined with the fact that it is specifically designed for using the ECBS approach in path planning, the framework using priority heaps (third algorithm) is proposed to implement as a coupled approach in the remainder of this thesis. It is possible to combine this approach with the preferred algorithms defined previously for task allocation chapter 5 and path planning chapter 6. However, it has to be noted that in the original implementation, the third algorithm does not consider kinematics of agents, possibly increasing run times. If computational times do not allow for implementing the coupled approach as proposed by Chen et al. [156], an online version of the algorithm as proposed by Wu et al. [126] will be implemented.

8

Research Proposal

In this thesis, an outline has been provided on general ground surface operations and more specifically, the ground surface operations at AAS in chapter 2. Based on the challenges that are currently encountered in this field of study, interesting developments related to the airport surface movement problem have been elaborated on in chapter 3, with special attention for the concept of engine-off taxi operations by the use of towing tugs such as the TaxiBot. A literature review on the consequences of implementing TaxiBots at airports has been performed, as well as an in-depth analysis on the possible concept of operations for implementation of TaxiBots at AAS, described in chapter 4. Based on the concept of operations, three different agent-based domains have been further explored, namely: Multi-Agent Task Allocation (MATA), Multi-Agent Motion Planning (MAMP) and the combination of both in Multi-Agent Pickup and Delivery (MAPD) problems in chapter 5, chapter 6 and chapter 7 respectively.

This final chapter concludes the literature study by formalizing a research proposal for the remainder of this thesis. First, a recap of the research gap as formulated earlier will be provided in section 3.3. Based on the research gap, the research objective and corresponding research questions will be discussed in section 8.2. Finally, an outline of the continuation of this thesis in terms of methodology will be outlined in section 8.3.

8.1. Recap Research Gap

In the past years, both the challenge of increasing demand for air transportation and the urge for aviation, including airport operations, to become more sustainable have received significant social and academic attention. One of the promising solutions that tackles the challenge related to sustainability is the use of external towing tugs, such as TaxiBots, that enable engine-off taxi operations at airports. However, due to the increasing demand for air transportation and the required throughput in the airport ground surface movement domain, it is of importance to evaluate the influence of implementing TaxiBots on the system performance. In this literature study, an extension of previous work is proposed that uses a hierarchical multi-agent control architecture to perform planning of taxi operations at AAS in a centralized manner and performs plan executions in a distributed manner. The existing architecture has been shown to provide for safe and efficient taxi operations [8, 9], however, it does not include the allocation of TaxiBots to flights that need to be towed.

In the academic world, the corresponding MAPD problem has received significant attention. However, limited research has been done on specifically integrating task allocation with path finding on real-world applications requiring real-time solutions. Based on these findings, a research objective and research questions have been defined, elaborated on in more detail in the following section.

8.2. Research Objective and Research Questions

A research objective represents the aim of a study. In our case, the research objective can be defined as follows:

"To design and evaluate an approach for combining task allocation with path finding for a cooperative fleet of TaxiBots to perform fully outbound towing at AAS, using a hierarchical multi-agent control

architecture."

Using this research objective as a starting point, the following set of research questions can be formulated:

1. **How can TaxiBot operations according to the defined set of assumptions and research scope be implemented in the existing hierarchical multi-agent system?**

   (a) What additional elements need to be implemented in the system to model the use of TaxiBots for fully outbound towing at AAS?

   (b) What communication structure will exist between TaxiBots themselves and between TaxiBots and the operator?

2. **How can greedy and auction-based allocation approaches be used in a lifelong MAPD problem to allocate TaxiBots to outbound flights?**

   (a) What are the objectives for allocating tasks and what allocation rules should be adopted by the TaxiBots?

   (b) What are the objectives and associated priorities for planning of paths?

   (c) How can replanning on a real-time basis improve the overall solution cost?

   (d) How can a good trade-off between computational efficiency and solution quality be maintained in the proposed framework?

   (e) How can the proposed framework be validated?

   As explained in section 5.3 and section 5.4, both a greedy algorithm based on flexibility with regards to the deadline of the task and TeSSI are among the most promising task allocation candidates. For path planning, the existing path finding algorithm based on a variant of the Safe Interval Path Planning (SIPP) algorithm will be used, in combination with either PBS or ECBS-CT (refer to subsection 6.2.3 for more details on the choice). Finally, both a decoupled approach based on Token Passing (TP) and a regret-based coupled approach based on priority heaps will be used to combine task allocation with path planning. For more details on the reasoning behind these choices, refer to subsection 7.2.4 and subsection 7.3.4 respectively.

3. **How do the different versions of the framework that combines task allocation and path finding perform when compared with each other and how do they compare with conventional operations in terms of system performance?**

   (a) What key performance indicators are relevant to measure system performance of the proposed framework (with respect to conventional operations)?

   (b) What are appropriate operational scenarios to use for evaluation of system performance when comparing different frameworks to combine task allocation with path finding?

   (c) What are appropriate operational scenarios to use for evaluation of system performance when comparing the different frameworks with conventional operations?

   (d) How does the solution cost of the proposed frameworks compare with optimal solutions?

   (e) What is the influence of structural assumptions on system performance and how can they be validated?

   (f) How does the performance of the proposed framework vary with changing parameters, such as fleet size?

   (g) How can the performance of the proposed framework provide insight in the implementation of engine-off tug enabled taxiing operations at AAS?

8.3. Research Methodology

In this section, a more detailed plan will be provided on how the aforementioned set of research questions will be answered. This will be done by elaborating on the methodological steps to be taken in subsection 8.3.1 and the corresponding time planning in subsection 8.3.2. Usually, a research methodology also includes the definition of the scope of the study and a corresponding set of assumptions. However, since this has been elaborated on extensively in chapter 4, the final concept of operations (including scope and assumptions) is not repeated here. The interested reader is referred to section 4.6 for more details on this section.

8.3.1. Methodological Steps

In this thesis, the hierarchical multi-agent control architecture developed by Soomers [8] and Kamphof [9] that provides for conflict-free routing of flights at AAS will be developed further to include allocation of TaxiBots to perform outbound towing (refer to subsection 4.6.3 for more details on this choice). The elaboration of the model and further research will consist of various steps that are outlined below.

1. **Model Orientation and Conceptualization**: first, the current existing model will have to be familiarized with and understood in order to implement the suggested allocation strategies. Then, the focus will be on answering research question 1 related to defining a conceptual model, including the development of additional agents and their interactions required for task allocation, as well as any additional changes to the existing environment. The proposed changes and additions will be formalized to be implemented in a baseline model.

2. **Baseline Model**: in the baseline model, two different approaches of combining task allocation with path finding will be implemented: a decoupled approach based on TP and a coupled approach based on priority heaps. Both approaches should allow for the arrival of new tasks to the system in an online setting and be able to allocate tasks/plan paths in real-time. For the allocation of tasks, the TeSSI algorithm will be implemented. If TeSSI turns out not be suitable in terms of computational efficiency, a greedy algorithm using the level of urgency of a task to determine the next task to be assigned will be implemented. With respect to path planning, the MAMP solver implemented in the existing model based on SIPP and PBS will be used.
   In the baseline model, the main focus is on how to combine task allocation with path finding for online problems in real-time, thus, focusing on research question 2. Therefore, the reallocation of tasks nor disruptions of the schedule due to unexpected events are not considered. Once the implementation of such a baseline model is achieved, possible model extensions can be considered. These extensions will be outlined below.

3. **Model Extensions**: after having completed an initial baseline model, several extensions to the baseline can be considered if available time and resources allow for this. A number of proposed extensions are listed below, in decreasing order of priority and expected relevance.

   (a) **Improvement of Task Allocation Strategy**: once an initial solution is found, several improvement strategies can be considered to improve the overall solution cost. Examples of this are the inclusion of reallocation of already assigned tasks and/or clustering of tasks before allocation.

   (b) **Disruptions Due to Unexpected Events**: since ground surface operations at airports are inherent to the existence of unexpected events, one step towards a more realistic simulation model would be to include a random variable representing delay. Possible operations in which delay could be included are e.g., the duration of TaxiBotting (include uncertainty in the velocity of TaxiBots), delay at the gate (include uncertainty in the issuing of TSAT) or the duration of decoupling (include uncertainty in the duration of decoupling). In order to cope with these types of delays, the model in general and the allocation of tasks specifically will probably have to be improved, moving back towards item 3a in this list. Specifically concerning allocation of tasks, it is suggested to extend the TeSSI algorithm towards an variant on the pTeSSI that is capable of coping with uncertainty.

4. **Evaluation and Sensitivity Analysis**: in the final evaluation, the system performance will be evaluated. This step mainly concerns research question 3 and concerns the definition of suitable

KPIs and operational scenarios to use for evaluation of the proposed frameworks. From an academic point of view, it would be of interest to determine how the solution cost of the algorithm compares with the optimal solution to the problem. From a practical point of view, it is interesting to focus on the implications on operations when allocation of TaxiBots is included in the problem. For example, think of the necessary amount of TaxiBots to perform fully outbound towing. A statistical analysis will be performed on the results to determine their significance.

8.3.2. Time Planning

In Table 8.1, the different methodological steps are presented, including the expected time needed to complete them.

Table 8.1: Research planning

High-Level	Low-Level	Duration (weeks)
Start-up	Familiarize with existing hierarchical control architecture	3
Model conceptualization	Develop additional agents to include for task allocation Design communication architecture between exisiting/added agents Design coordination architecture between existing/added agents	2
Implementation of baseline model	Implement conceptual model in existing hierarchical control architecture	8
Verification and validation of baseline model	Define and analyze testing scenarios Perform verification and validation of baseline model	2
Implementation of model extensions	Define and design relevant conceptual model extensions Implement conceptual model extensions in baseline model	5
Verification and validation of model extensions	Define and analyze testing scenarios Perform verification and validation of model extensions	1
Final evaluation and analysis	Define and analyze final operational scenarios to test model performance Perform sensitivity and statistical analysis Write draft	7
Conclude thesis work	Write final paper Prepare presentation and defence	4

III

Supporting Work

A

Model Elaboration

In this chapter, the simulation model is further elaborated on. First, all assumptions related to the concept of operations of conventional and tug-enabled taxiing at Amsterdam Airport Schiphol (AMS) are listed in section A.1. Then, in section A.2, a detailed description of both the environment and the agents is provided that are modelled in the Multi-Agent System (MAS). In section A.3, the pseudocode of the most important algorithms developed in this study is provided. Finally, the steps taken to verify the model are discussed in section A.4.

A.1. Assumptions Related to Concept of Operations

In this section, an overview is provided of all assumptions related to the concept of operations that is implemented in the simulation model. First, a general set of assumptions on tug-enabled taxiing operations is provided in subsection A.1.1. Next, specific modelling assumptions are elaborated on in subsection A.1.2.

A.1.1. General Assumptions on Tug-Enabled Taxiing Operations

The following set of operational assumptions is formalized concerning the implementation of tug-enabled taxiing operations in this study:

- Fully outbound towing is required, meaning that all outbound flights should be towed by a tug towards a decoupling location near the assigned runway.
- A finite fleet of identical tugs is available, assumed to be capable of towing all outbound aircraft, independent of size and weight.
- A fixed time duration for decoupling procedures of 120 seconds is assumed, based on TaxiBot trials performed by Royal Schiphol Group at AMS [6]. The same duration is used for coupling procedures at ramps, although no operational data is available to base this assumption on.
- Ramps are assumed to provide for sufficient space and capacity, allowing tugs to wait for an indefinite amount of time at the ramp location if assigned to a flight departing from that same ramp.
- Tugs return to the parking facility if not assigned any new tasks.
- Tugs driving on the service road network are not coordinated, meaning that separation among tugs driving in solo mode on the service road network is not assumed to be maintained. At locations where a service road crosses a taxiway, separation is ensured between tugs and any vehicle travelling on the taxiway.

A.1.2. Modelling Assumptions for Specific Case Study

For the simulation of conventional taxiing and tug-enabled taxiing operations at Amsterdam Airport Schiphol (AMS), the following set of modelling assumptions is formalized:

- All aircraft are mapped to an International Civil Aviation Organization (ICAO) Aerodrome Reference Code based on wingspan and aircraft length (refer to Appendix B for mapping of aircraft to an ICAO Aerodrome Reference Code), indicating size of the aircraft.
- Apron operations are not included in the simulation model and therefore, aircraft ramps are modelled as meta-ramps. Meta-ramps represent a group of aircraft ramps for a specific bay area, which do not require any pushback or push-pull procedures. Although meta-ramps are assumed to be conflict-free (allowing for multiple vehicles to be present at the same location at the same time), conflicts between arriving and departing flights at a ramp are detected and recorded. Since ramp assignment and scheduling is considered to be out of the scope of this research, ramp conflicts between arriving and departing flights are not resolved. For more details on the mapping of ramps to meta-ramps, refer to subsection A.2.1.
- Inbound flights spawn at a runway exit based on the ICAO Aerodrome Reference Code at the Actual Landing Time (ALDT) (refer to subsection A.2.1 for mapping of the ICAO Aerodrome Reference Code to designated entries and exits per runway). The goal location of arriving flights is a meta-ramp, corresponding with the assigned ramp in the flight schedule. Once the vehicle has arrived at its goal location, it is removed from the simulation.
- Outbound flights spawn at a meta-ramp corresponding with the assigned ramp in the flight schedule and are assigned the Target Startup Approval Time (TSAT) as a deadline to leave the meta-ramp. The goal location of departing flights is a set of possible runway entries, based on the ICAO Aerodrome Reference Code (refer to subsection A.2.1 for mapping of the ICAO Aerodrome Reference Code to designated entries and exits per runway). Once the vehicle has arrived at its goal location, it is removed from the simulation.
- Between the center of two vehicles moving on the airside roads, a separation of 150 meters has to be maintained at node locations that are traversed by both vehicles.
- Travel direction constraints that are currently in-place for some parts of the general taxiway network, do not have to be adhered to since a novel planning concept is put to test. For a selection of apron entries and exits, travel direction constraints do apply (refer to subsection A.2.1).
- A fixed time of 100 seconds is used to account for apron operations for all departing flights in MET scenarios. For all inbound flights, it is assumed that engine cool-down is performed entirely during taxiing from the runway to the assigned ramp.
- All vehicles move with constant speed over an edge and can change speed instantaneously at a node location (assuming infinite acceleration and deceleration rates). Furthermore, all vehicles have the intent to move to their goal location at maximum speed, unless constraints restrict this from happening. No uncertainty in kinematic characteristics of vehicles exists, both for planning of routes as for execution of the routing plan.

A.2. Detailed Description of Multi-Agent System

As explained in Section 3 of the scientific paper, the MAS consists of an environment and agents. In subsection A.2.1, the construction of a graph network based on the layout of Amsterdam Airport Schiphol (AMS) is provided. Next, the characteristics and properties of all agents is provided in subsection A.2.2.

A.2.1. Environment

In this section, the construction of the graph network of AMS is discussed. All its major elements will be elaborated on, including relevant assumptions and examples.

General Graph Construction of AMS

Based on geographical information extracted from QGis [157], the layout of AMS has been translated into a graph network consisting of 237 nodes and 390 bidirectional edges. Figure A.1 shows how satellite data is used to construct this network.

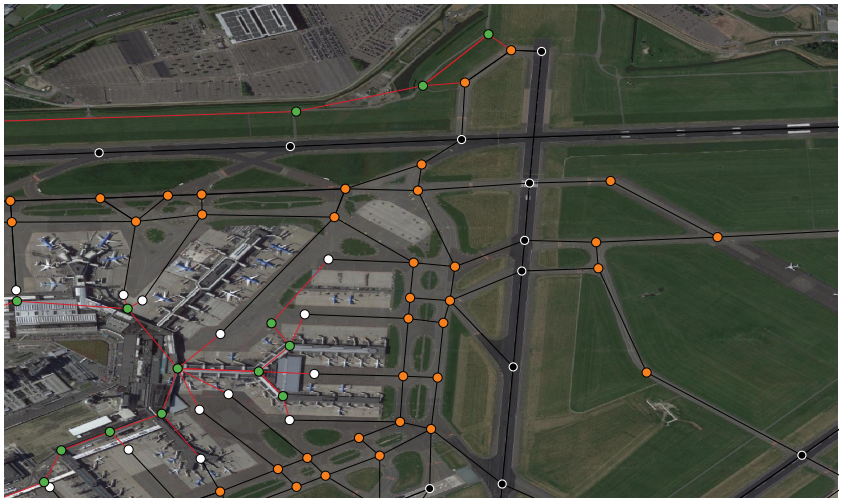


Figure A.1: Translation of taxiway network of Amsterdam Airport Schiphol using satellite images to a graph network, showing taxiway edges (black, thin line), service road edges (red, thin line), runway edges (black, thick line), runway nodes (black), taxiway nodes (orange), service road nodes (green) and meta-ramp nodes (white).

In the network, several types of nodes can be distinguished: taxiway nodes (orange), service road nodes (green), meta-ramp nodes (white), decoupling nodes (red), all-clear nodes (blue), tug parking node (purple) and runway nodes (black). In addition, three types of edges are used: taxiway edges (thin black line), runway edges (thick black line) and service road edges (thin red line). The entire network is shown in Figure A.2. Note that taxiway and service road nodes are excluded from this figure.

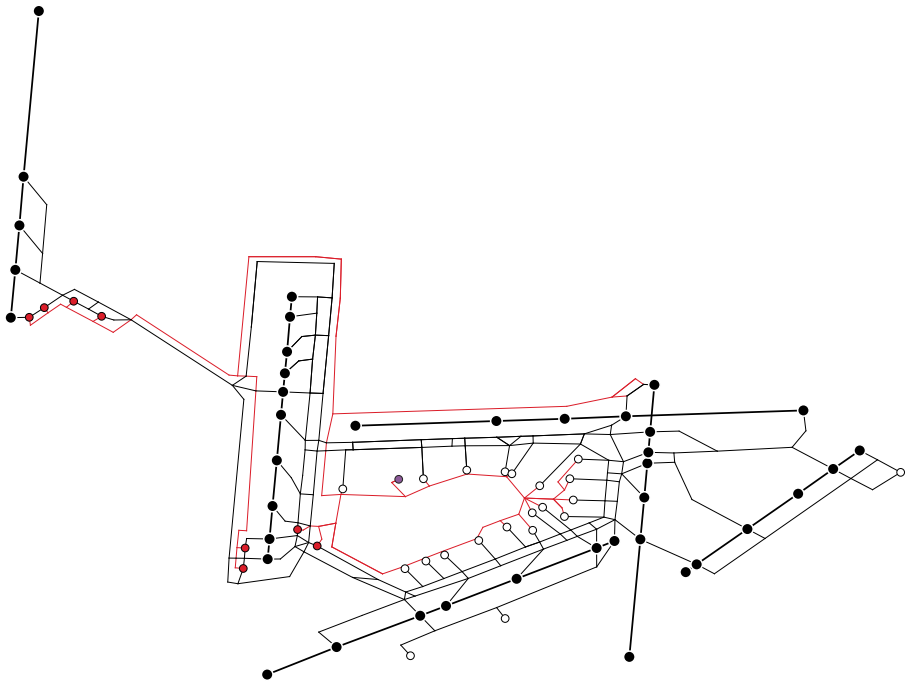


Figure A.2: Graph network of Amsterdam Airport Schiphol, showing taxiway edges (black, thin line), service road edges (red, thin line), runway edges (black, thick line), runway nodes (black), decouple nodes (red), meta-ramp nodes (white) and tug base node (purple).

Runway Placement and Configuration

In this study, Runway Mode of Operation (RMO) North is analyzed, meaning that the Polderbaan and Zwanenburgbaan are used as departing runways in northern direction. The Kaagbaan and Aalsmeer-

baan are used for arriving flights, coming from south-western or southern direction respectively. Runway usage for RMO North for different peaks is shown in Figure A.3.

(a) Arrival peak (A: 06 + 36R, D: 36L) (b) Transition (A: 06 + 36R, D: 36L + 36C) (c) Departure peak (A: 06, D: 36L + 36C)

Figure A.3: Runway usage in RMO North (arrival peak, transition and departure peak). Created by author.

Every aircraft is mapped to a category in the ICAO Aerodrome Reference Code system (refer to Appendix B). Based on the input of operational experts, a mapping of the ICAO categories to possible runway entries/exits is provided in Table A.1. Note that the names of specific runway entries and exits can be found in the aerodrome ground movement chart provided in Appendix D.

Table A.1: Mapping of ICAO Aerodrome Reference Code category to possible runway entry or exit.

Runway	Entry	Exit	ICAO Aerodrome Reference Code
36L	V1, V2, V3	-	A, B, C
	V4		A, B, C, D, E, F
36C	W9	-	A, B, C
	W10		A, B, C, D, E, F
06	-	S3	A, B
		S4	C
		S6	D, E, F
		S8	cargo flights only
36R	-	E1	A, B
		E2	C
		E3, E4	D, E
		E5	F

Apron Area

The majority of the aprons at AMS are located in the Schiphol Centre Area. In Figure A.4, the terminal and corresponding piers are indicated. In addition, the A-apron, J-apron, S-apron, R-apron and Y-apron are indicated. Three aprons that are present at AMS that are not indicated in the figure are the U-apron (located west of 18C/36C and north of 09/27), the K-apron and the M-apron (both located at Schiphol East). Note that flights departing from the R-apron, Y-apron, U-apron, K-apron and M-apron are not included in the simulation.

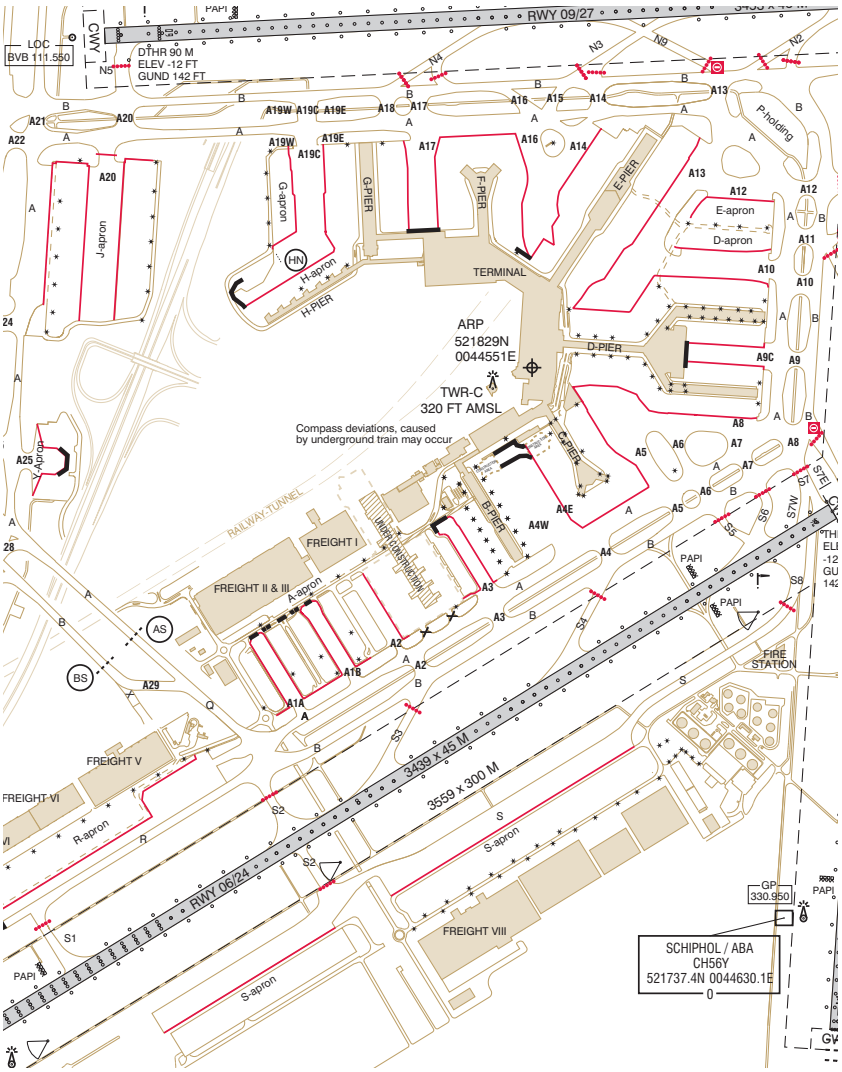
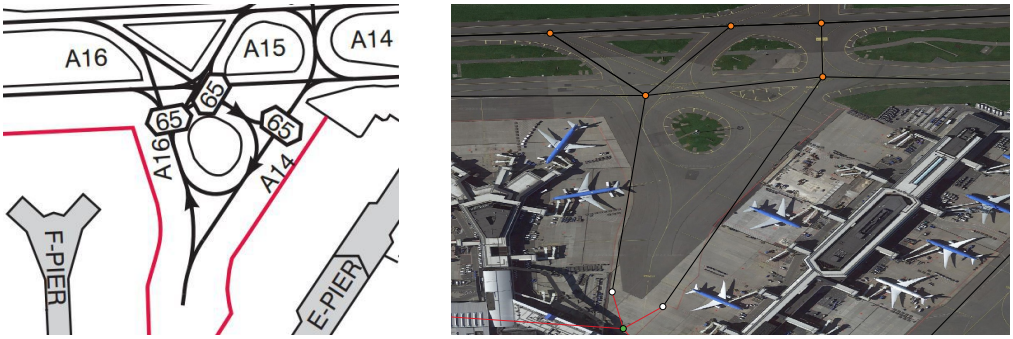


Figure A.4: Configuration of piers at AMS [158].

All ramps located in a specific apron are mapped to one or multiple meta-ramp nodes. In Figure A.6, all meta-ramps are shown, including labels. The labels of the meta-ramp nodes correspond with the name of the taxiway connecting the apron with the taxiway network and can be found in the aircraft docking chart in Appendix D. All ramps that can be reached through the specific apron taxiway are mapped to the meta-ramp node, independent of flight direction of the aircraft assigned to a ramp. The following exceptions are made regarding mapping of ramps to meta-ramps and labelling of meta-ramp nodes:

- **A4**: in reality, two taxiways exist for the apron between the B- and C-pier (A4W, A4E). Both taxiways are merged into a single taxiway and end up in the same meta-ramp node A4.
- **AC0**: the ramps C13, C15, C16 and C18 located at the head of the C-pier are mapped to a separate meta-ramp node AC0.
- **A5** and **A6**: all ramps assigned to arriving aircraft that are located between the C- and D-pier that are not mapped to meta-ramp A8, are mapped to meta-ramp node A5. All ramps assigned to departing aircraft located between the C- and D-pier that are not mapped to meta-ramp A8, are mapped to meta-ramp node A6. The differentiation on flight direction is done because of travel direction constraints for taxiway A5 and A6 (Figure A.5).



(a) Travel direction constraints for taxiway A14 and A16

(b) Translation of ramps between E- and F-pier to meta-ramps A14 and A16

Figure A.5: Example of the translation of travel direction constraints for apron entries/exits into multiple meta-ramp nodes.

- **A8**: the ramps D16, D18, D22, D24, D26 and D28 located at the South side of the D-pier are mapped to meta-ramp node A8.
- **A10**: all ramps assigned to arriving aircraft that are located between the D- and E-pier that are not mapped to meta-ramp node A12, are mapped to meta-ramp node A10.
- **A13**: all ramps assigned to departing aircraft located between the D- and E-pier that are not mapped to meta-ramp node A12, are mapped to meta-ramp node A13.
- **A14** and **A16**: all ramps assigned to arriving aircraft that are located between the E- and F-pier, are mapped to meta-ramp node A4. All ramps assigned to departing aircraft located between the E- and F-pier are mapped to meta-ramp node A16. The differentiation on flight direction is done because of travel direction constraints for taxiway A14 and A16 (refer to Appendix D).
- **A19C**: in reality, three taxiways exist for the G- and H-apron (A19W, A19C and A19E). All three taxiways are merged and end up in the same meta-ramp node A19C. Note that all ramps located in the G- and H-apron are mapped to this meta-ramp node.

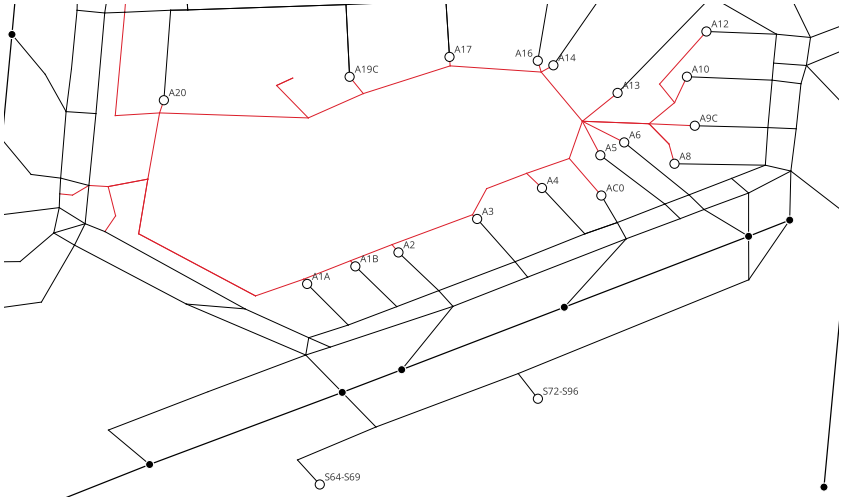
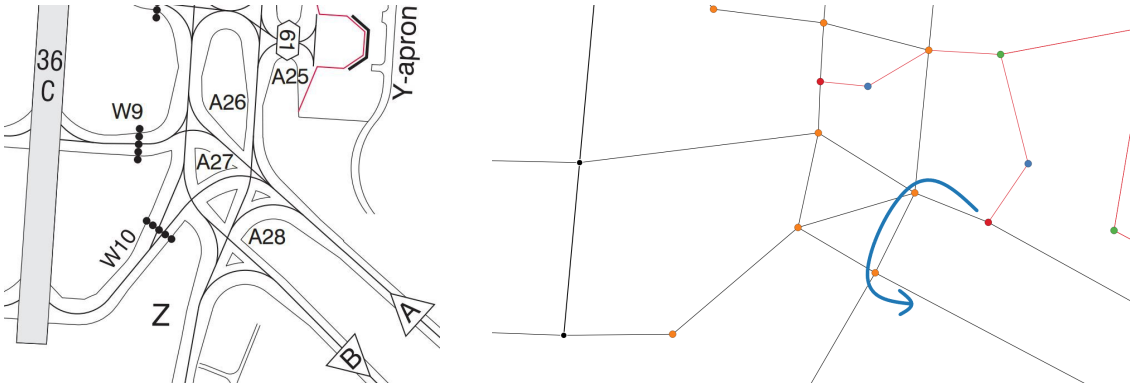


Figure A.6: Meta-ramp nodes used in the network, including labels.

Taxiway Network

The apron area and the runways are connected with each other by taxiways, which are accessible to aircraft. The details on the taxiway network for AMS are defined in the Aerodrome Ground Movement Chart provided for by Dutch ATC (LVNL) [158] (refer to Appendix D). This chart provides for the layout of all taxiways, including directions currently implemented on the taxiways and restricted turns on taxiway

crossings. An example of the translation of a crossing on the taxiway network to a simplified network structure is provided in Figure A.7.



(a) Snapshot of crossing near 36C from aerodrome ground movement chart [158].

(b) Translation of crossing into simplified network structure, indicating a turn on the crossing that is not allowed, based on Figure A.7a.

Figure A.7: Example of the translation of a crossing on the taxiway network at AMS to a simplified network structure, including restrictions on allowed turns.

Due to safety reasons, certain taxiways at AMS cannot be utilized when specific runways are operational. These taxiways include those that intersect with an active runway or are situated within the flight path of landing or departing aircraft. In Figure A.8, the taxiways that are blocked in RMO North are shown. Consequently, all aircraft taxiing towards runway 36L will have to pass the Zwanenburgbaan on its South side. Note that the service roads passing over the North side of the Zwanenburgbaan are not closed, allowing tug vehicles to make their return movement towards the terminals.

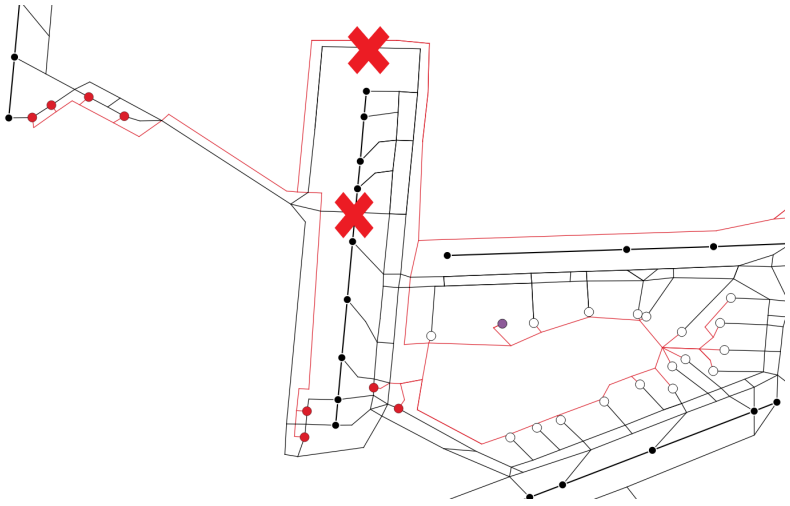


Figure A.8: When RMO North is in operation (36C used as departure runway in Northern direction), the taxiways crossing 36C and passing over its North side are closed due to safety reasons.

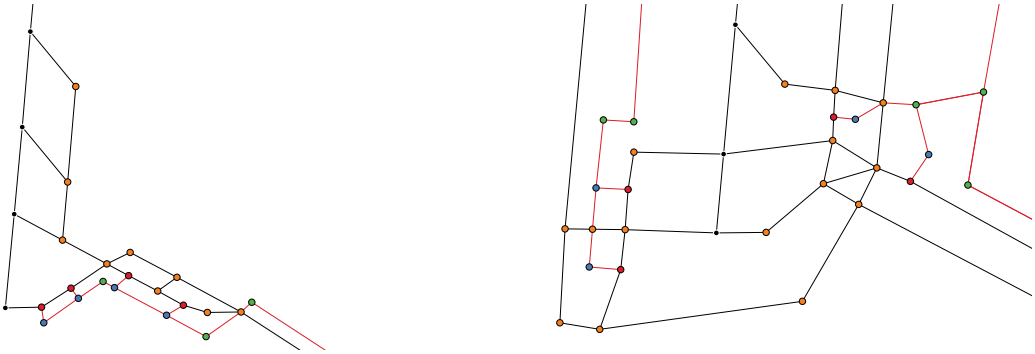
Decoupling Locations

For both departing runways, four decoupling locations are defined based on previous work of Soomers [8] and shown in Figure A.9. In his work, Soomers defined 15 possible decoupling locations in cooperation with Royal Schiphol Group for all runways at AMS. Although not all possible decoupling locations meet the criteria set by Schiphol [7], it is anticipated that future advancements in tug technology, protocols and airport infrastructure will make these locations workable.

For each decoupling location, an all-clear location is defined from which the tug driver provides an all-clear signal to the pilot indicating that the aircraft can safely resume its journey towards the runway. It is assumed that the tug can always reach the all-clear location from the decoupling location, independent of other traffic. In addition, it is assumed that no other traffic is blocked by tugs positioned at

all-clear locations. Therefore, all-clear locations are tried to be placed at existing service road infrastructure. This does not hold for the all-clear points related to the decoupling locations at the East side of runway 36C: currently, no infrastructure exists that allows for the return movement of tugs to the central service road without making use of taxiways or blocking other traffic.

Finally, it has to be noted that during trials using a TaxiBot at AMS, it became apparent that the current service road infrastructure does not allow for tug return movements in general, since these roads are not built for (more) heavy and wide traffic [6]. In addition, there are many users of these roads, including the fire department [6], that limit capacity of the service roads.



(a) Close-up of South end of runway 36L

(b) Close-up of South end of runway 36C

Figure A.9: Close-up of graph network of Amsterdam Airport Schiphol, showing taxiway edges (black, thin line), service road edges (red, thin line), runway edges (black, thick line), runway nodes (black), taxiway nodes (orange), service road nodes (green), decouple nodes (red) and all-clear nodes (blue).

A.2.2. Agent Specifications

In the current section, the characteristics and properties for all agents in the system (denoted by a C and P in the enumeration respectively) are described in more detail.

Airport Operations Agent

Role: define the set of of flights to be routed in the current planning window.

P1 **Define Aircraft to be Routed Property**: based on information obtained from the Routing Agent, the Airport Operations Agent checks for each aircraft that is planned to spawn before the end of the planning window if it has arrived at its goal location yet. If the route of the aircraft is not completed yet, the Maneuver-Sequencing for Non-Towed Aircraft Property is executed.

P2 **Maneuver-Sequencing for Non-Towed Aircraft Property**: for the aircraft that do not require any tug to tow them to their goal location, the Airport Operations Agent defines an activity sequence for the flight. This applies to all inbound flights and to outbound flights if we are considering a scenario with no towing at all. In the case of non-towed aircraft, the maneuver sequence consists of a single moving activity: taxiing.

- **Taxiing**: the maneuver taxiing entails the movement of an aircraft using its own engines from a starting position to a goal location. For outbound flights, it is assumed that the engines of the aircraft are fully warmed-up when decoupling is finished. For inbound flights, it is assumed that the entire engine cool-down can take place during the taxiing maneuver.

Routing Agent

Role: execute the PBS-TA algorithm, in cooperation with the Tug Allocation Agent, to find a conflict-free routing plan for all vehicles for the current planning window.

P1 **Provide for Current Status and Position of All Existing Individual Agents Property**: this property involves interaction between the Routing Agent and the Individual Agents (Tug and Aircraft Agents). At every time step t, the Routing Agent collects the position and status of all individual agents. If replanning is due, this information is shared with the Airport Operations Agent and the Tug Allocation Agent.

P2 **Create Individual Agents Property**: when replanning is triggered, the Routing Agent receives information on the set of vehicles to route in the current planning window from the Airport Operations Agent and the Tug Allocation Agent. For each vehicle, an individual agent is created that is characterized by an ID, origin, destination, relevant times and kinematic properties. In addition, the sequence of maneuvers to be performed by the vehicle is added as an attribute to the agent as well. If the vehicle has been routed before and has already started executing one of its activities, the agent also inherits the current status and position of the vehicle. Once an agent is created for every vehicle to route in the current planning window, the Get Routes Property is executed by the Routing Agent.

P3 **Get Routes Property**: once the Create Single Agents Property is performed, the Routing Agent tries to find a conflict-free routing plan for all vehicles. For each vehicle, the routing plan describes the route and time for every maneuver that is has to perform. The maneuvers can be grouped into two different activities: moving activities and waiting activities. Additional details on the working principle of the routing algorithm can be found in Section 4 of the scientific paper.

Tug Allocation Agent

Role: find an allocation of outbound flights to tugs.

P1 **Create STNs Property**: in order for the tug to make a bid on the tasks up for auctioning, it has to determine what is the optimal position to insert the task up for auctioning into its current schedule. In order to keep track of its schedule, the tug makes use of Simple Temporal Network (STN). More information on the STNs is provided in Section 4 of the scientific paper. At the beginning of every auction, the Tug Allocation Agent creates an STN for every tug in the fleet of available tugs. If the tug under consideration has already started executing a task at the moment of replanning, this task is inserted in its schedule as a fixed task. In addition, the Tug Allocation Agent shares the set of tasks up for action with every Tug Schedule Agent.

P2 **Get Allocation of Outbound Aircraft to Tugs Property**: for every aircraft in the remaining set of outbound flights, all Tug Agents submit a bid to the Tug Allocation Agent. Then, a winning tug is determined that gets assigned the task to tow the aircraft from the meta-ramp location to a decoupling location in vicinity of the designated runway. The winning tug executes the Insert Assigned Task into STN Property and adds the assigned task in its existing schedule. More details on the allocation framework are provided in Section 4 of the scientific paper.

P3 **Maneuver-Sequencing for Tugs Property**: based on the tasks assigned to each tug and the current status of the tug, a sequence of maneuvers is determined that the tug will have to execute.

- **Driving in solo mode**: the maneuver driving in solo mode entails the movement of the tug from its current position on the service road network to the pickup location of its next task (meta-ramp location). If the tug is driving in solo mode, the tug is controlled by the tug driver. The tug is not allowed to drive in solo mode on the taxiway network, except on a crossing of a service road with a taxiway.
- **Waiting**: if the tug arrives at the pickup location of its next task before its deadline (TSAT minus the time needed for coupling), the tug performs a waiting maneuver until the coupling activity starts.
- **Coupling**: the coupling maneuver entails the coupling of the tug to the assigned aircraft at the meta-ramp location. The coupling maneuver starts if the tug is present at the meta-ramp location and not before TSAT minus the coupling duration.
- **Driving in pilot mode**: the maneuver driving in pilot mode entails the movement of the tug-aircraft combination from the meta-ramp location to a decoupling location in the vicinity of the assigned runway. Although the pilot is in control of the tug-aircraft combination, the tug accounts for the propulsion of the vehicle combination. Note that the tug is not allowed to drive in pilot mode on the service road network.
- **Decoupling**: the decoupling maneuver entails the decoupling of the tug from the assigned aircraft at a decoupling location. The decoupling procedure includes the driver stepping out of the tug and unplugging the communication cable, driving of the tug from the decoupling

location to an all-clear position and the driver providing the pilot with an all-clear signal once arrived at that position [6]. The movement of the tug from the decoupling location to the all-clear location is not modelled: it is assumed that the tug spawns at the all-clear location at the moment that decoupling is finished. In addition, it is assumed that the engines of the aircraft are started and fully warmed up at the end of decoupling.

Tug Agents

Role: represent the individual tug in the auction process to find an allocation of flights to tugs. In addition, the Tug Agent executes its routing plan defined by the Routing Agent.

C1 **Kinematic Characteristics**: the Tug Agents are characterized from a kinematic perspective by the maximum allowable speed for each possible maneuver that they are able to perform. The details on vehicle kinematics can be found in Appendix B. Note that acceleration and deceleration are assumed to be infinite for all vehicles.

C2 **Flight Schedule Characteristics**: all information that is present in the flight schedule for the aircraft the tug has been assigned, is stored as an attribute for the agents. This concerns the flight ID, assigned ramp, assigned runway, ICAO Aerodrome Reference Code, the wake turbulence category and spawn time.

P1 **Insert Fixed Tasks into STN Property**: if the tug vehicle is coupled to an outbound flight at the moment of replanning, the corresponding aircraft cannot be reallocated to another tug anymore. Therefore, the task should be added to the schedule of the Tug Agent. Based on the remaining work time, a time slot is reserved for completion of this task in the STN of the tug.

P2 **Find Best Position to Insert Task into STN Property**: for a task in the set of unallocated tasks, the Tug Agent tries to insert the task in each available time segment in its STN. For every insertion position, the increase in cost with respect to its current schedule is recorded. After evaluation of every possible insertion position, the STN is reset to the current schedule of the tug to ensure that it only includes the tasks that are already allocated to the agent. The position for insertion of the task that results in the least increase in total delay of off-block time for all tasks in the schedule is considered to be the best insertion position.

P3 **Compute Bid Vector Property**: for every task in the set of unallocated tasks, the best insertion position of the task under consideration is determined using the item **P2** property. Based on the best insertion position, the bid of the agent for the task under consideration is computed and stored in a bid vector. Once the bids for all unallocated flights are calculated, the bid vector is shared with the Tug Allocation Agent to determine which task is being assigned next to which agent.

P4 **Insert Assigned Task into STN Property**: once a task has been assigned to a specific tug vehicle, the Tug Agent inserts this task into its STN.

P5 **Clear Existing Routing Information Property**: when replanning is triggered, the existing routing information for all Tug Agents is removed. Information on the path travelled up until the moment of replanning is preserved.

P6 **Retrieve Routing Plan Property**: once the Routing Agent has executed the Get Routes Property, it communicates the obtained solution with all Tug Agents. All individual agents store their own path to be executed over time.

P7 **Move Property**: for every time step t, every individual agent checks its routing plan and determines the position it should be at the end of the time step. If in the next time step the current maneuver is finished, not only the position is updated but the status of the tug as well. Once the Tug Agent has arrived at its goal destination, it is removed from the simulation.

Aircraft Agents

Role: execute its routing plan defined by the Routing Agent.

C1 **Kinematic Characteristics**: similarly as for Tug Agents, the Aircraft Agents are characterized from a kinematic perspective by the maximum allowable speed for each possible maneuver that they are able to perform. The details on vehicle kinematics can be found in Appendix B. Note that acceleration and deceleration are assumed to be infinite for all vehicles.

C2 **Flight Schedule Characteristics**: all information that is present in the flight schedule for the aircraft is stored as an attribute for the agents. This concerns the flight ID, assigned ramp, assigned runway, ICAO Aerodrome Reference Code, the wake turbulence category and spawn time (ALDT for inbound flights, TSAT for outbound flights).

P1 **Clear Existing Routing Information Property**: when replanning is triggered, the existing routing information for all Aircraft Agents is removed. Information on the path travelled up until the moment of replanning is preserved.

P2 **Retrieve Routing Plan Property**: once the Routing Agent has executed the Get Routes Property, it communicates the obtained solution with all Aircraft Agents. All individual agents store their own path to be executed over time.

P3 **Move Property**: for every time step t, every individual agent checks its routing plan and determines the position it should be at the end of the time step. If in the next time step the current maneuver is finished, not only the position is updated but the status of the aircraft as well. Once the Aircraft has arrived at its goal destination, it is removed from the simulation.

A.3. Algorithmic Implementation

In this section, pseudocode is presented for the most important algorithms developed in this study. First, the working principle of the PBS-TA framework is formalized in subsection A.3.1. Next, the algorithms related to the allocation of tasks are described in subsection A.3.2. Finally, the supporting functions related to path planning are mentioned in subsection A.3.3.

A.3.1. High-Level Framework Integrating Task Allocation with Path Planning

The pseudocode of Algorithm 1 shows the working principle of the high-level PBS-TA framework. The framework is based on the work done by Hönig et al. [155]. Changes with respect to the original algorithm are indicated in red. The supporting functions *findAssignment()* and *updatePlan()* are defined in subsection A.3.2 and subsection A.3.3 respectively.

Algorithm 1 High-Level of PBS-TA for every replanning moment. Adapted version of CBS-TA [155] with changes and additions marked in red.

Require: Graph, pickup and delivery locations of all tasks, current position of all agents, heuristics on shortest path, maximum number of trees to explore in forest $n_{iter,max}$
Ensure: Conflict-free path for each agent

Initialize the first root node
R.plan ← ∅
R.priorities ← ∅
R.constraints ← ∅
R.root ← True
R.iteration ← 0
R.assignment, R.constraints ← findAssignment(R.constraints)
for each vehicle a_i **do**
    success ← updatePlan(R, a_i) Find a path for vehicle a_i
    if not success **then return** no solution
    end if
end for
R.cost ← SIC(R.plan) SIC ← *sum of individual path costs*
successors ← {R.iteration: R} Initialize a dictionary containing all successors to consider per search tree
OPEN ← [R]

Start the expansion of the search forest
while OPEN not ∅ **do**
    P ← pick and remove best node from OPEN Lowest solution cost
    successors ← successors[P.iteration] \P Remove P from possible successors
    Find the first conflict in time for P
    if P has no conflict **then return** P.plan Solution in P is conflict-free
    end if

# **If a conflict is found, create successors to explore in future iterations**
    if P.root is True **and** P.cost ≠ 0 **and** P.iteration < $n_{iter,max}$ **then**
        R ← new root node
        R.plan ← ∅
        R.priorities ← ∅
        R.root ← True
        R.iteration ← P.iteration + 1
        R.assignment, R.constraints ← findAssignment(P.constraints)
        for each vehicle a_i **do**
            success ← updatePlan(R, a_i)
            if not success **then return** no solution
            end if
        end for
        R.cost ← SIC(R.plan)
        update successors dictionary with {R.iteration: R}
    end if

Continue on next page

```
36:     Conflict ← (a_i, a_j, location, time)                    ▯ First conflict in P
37:     for agent a_i in Conflict do
38:         Q ← new child node
39:         Q.priorities ← P.priorities + a_i > a_j
40:         Q.assignment ← P.assignment
41:         Q.root ← False
42:         Q.iteration ← P.iteration
43:         success ← updatePlan(Q, a_j)
44:         if not success then
45:             continue
46:         end if
47:         Q.cost ← SIC(Q.plan)
48:         successors[Q.iteration] ← successors[Q.iteration] + Q
49:     end for

50:     insert nodes in successors into OPEN in order of decreasing cost
51: end while
52: return no solution
```

A.3.2. Task Allocation

All algorithms related to the generation of an allocation of outbound flights to tugs are presented in the current section. First, Algorithm 2 shows how the solution pool of allocations is generated. Then, the pseudocode of the adapted TeSSI auction algorithm is shown in Algorithm 3. The changes with respect to the original implementation of TeSSI as described in the work of Nunes and Gini [117] are presented in red. Finally, the supporting functions *computeBid()*, *costSchedule()* and *calculateDOS()* are presented in Algorithm 4, Algorithm 5 and Algorithm 6 respectively.

Algorithm 2 findAssignment()

Require: List of all outbound AC $\mathcal{T} = \{AC_1, AC_2, ..., AC_m\}$, set of tug agents $\mathcal{R} = \{r_1, r_2, ..., r_n\}$, size of solution pool n_{alloc}, constraints
Ensure: Final assignment of departing flights to tugs

Remove the flights from $\mathcal{T}$ for which execution has already started
fixedAllocation ← {}
$\mathcal{T}' \leftarrow \mathcal{T}$
for each tug agent r_i in $\mathcal{R}$ **do**
    fixedAllocation ← $\{r_i : []\}$
    if r_i is coupling **or** r_i is towing **or** r_i is decoupling **then**
        $\mathcal{T}' \leftarrow \mathcal{T}' \setminus AC_j$ that r_i is currently executing
        fixedAllocation ← $\{r_i : [AC_j]\}$
    end if
end for
if len($\mathcal{T}'$) ← 0 **then return** fixedAllocation
end if

Create a solution pool containing allocations of flights to tugs
STACK ← ∅
CLOSED ← ∅
for i in range(n_{alloc}) **do**
    allocation, taskHighestMarginalCostIncrease, allocationCost ← auctionTeSSI($\mathcal{T}'$, $\mathcal{R}$, constraints, fixedAllocation)
    Insert (allocation, taskHighestMarginalCostIncrease) in CLOSED

    allocationCost ← cosineDOS(allocation, allocationCost, previously explored allocations)
    constraints.taskToAssignFirst ← taskHighestMarginalCostIncrease
    Add allocation to constraints.tabuDistribution
    Insert (allocation, allocationCost) into STACK
end for

allocationMinCost ← allocation with minimum allocationCost in STACK
constraints.taskToAssignFirst ← taskHighestMarginalCostIncrease for allocationMinCost in CLOSED
return allocationMinCost, constraints

Algorithm 3 auctionTeSSI(). Adapted version of the TeSSI auction algorithm [117], with changes and additions marked in red.

Require: List of outbound AC for which execution has not started yet $\mathcal{T}' = \{AC_1, AC_2, ..., AC_{m-j}\}$, set of tug agents $\mathcal{R} = \{r_1, r_2, ..., r_n\}$, constraints, assignment of outbound AC to tugs for which execution has already started (fixedAllocation)

Ensure: Possible assignment of flights to tugs

Initialize the auction by computing the bid vector of all agents in the initial state

for each tug agent r_i in $\mathcal{R}$ **do**
  Initialize an STN S_r to maintain the schedule of the tug
  if fixedAllocation for r_i is not $\emptyset$ **then**
    Insert fixed task AC_j for r_i in fixedAllocation into S_r
  end if
  bidVector$_{r_i}$ ← computeBid($\mathcal{T}'$, S_{r_i}, constraints)
  $Q_{\mathcal{T}'}^{r_i}$ ← bidVector$_{r_i}$
end for

Start the auctioning of tasks

allocationPossible ← fixedAllocation
costIncreasePerTask ← {}
totalAllocationCost ← 0
while $\mathcal{T}'$ is not $\emptyset$ **do**
  (winner, AC$_{min}$, minBidOverall) = (null, null, ∞)
  if constraints.taskToAssignFirst is not $\emptyset$ **then**
    # **Assign taskToAssignFirst to the tug with the minimum bid for this task**
    AC$_{min}$ ← constraints.taskToAssignFirst
    minBid$_r$ ← $\operatorname{argmin}_{r_i, AC_{first}} Q_{AC_{first}}^{r_i}$, winner ← r
    minBidOverall ← minBid$_r$
    constraints.taskToAssignFirst ← $\emptyset$
  else
    # **Assign the task that corresponds with the minimum bid in** $\mathcal{T}'$
    for each tug agent r_i in $\mathcal{R}$ **do**
      minBid$_r$, AC ← $\operatorname{argmin}_{r_i, AC} Q_{\mathcal{T}'}^{r_i}$
      if minBid$_r$ < minBidOverall **then**
        winner ← r, AC$_{min}$ ← AC
        minBidOverall ← minBid$_r$
      else if AC$_{min}$ is null **then**
        AC$_{min}$ ← AC
      end if
    end for
  end if
  if winner is null **then return** no solution
  else
    # **Insert task with minimum bid into schedule of winning agent**
    allocationPossible ← {r_{winner} : add AC$_{min}$to list}
    costIncreasePerTask ← {AC$_{min}$: minBidOverall}
    totalAllocationCost ← totalAllocationCost + minBidOverall
    Insert task AC$_{min}$ for r_{winner} into S_{winner}
    $\mathcal{T}' \leftarrow \mathcal{T}' - \{AC_{min}\}$
    bidVector$_{winner}$ ← computeBid($\mathcal{T}'$, S_{winner}, constraints)
    $Q_{\mathcal{T}'}^{winner}$ ← bidVector$_{winner}$

Continue on next page

```
        if S_winner in constraints.tabuDistribution then
            # Update bids of agents to prevent an already explored allocation from being generated
            for each tug agent r_i in R do
                if r_i is not r_winner then
                    bidVector_{r_i} ← computeBid(T', S_{r_i}, constraints)
                    Q_{T'}^{r_i} ← bidVector_{r_i}
                end if
            end for
        end if
    end if
end while
taskHighestMarginalCostIncrease ← AC in costIncreasePerTask for which minBidOverall is highest
return allocationPossible, taskHighestMarginalCostIncrease, totalAllocationCost
```

Algorithm 4 computeBid(r_{loc}, $\mathcal{T}'$, S_r, constraints). Adapted version of the *Task Scheduling Algorithm* that is part of the TeSSI auction algorithm [117]. Changes and additions are marked in red.

Require: Agent location r_loc, list of outbound AC for which execution has not started yet $\mathcal{T}' = \{AC_1, AC_2, ..., AC_{m-j}\}$, partial schedule of tasks S_r for agent r, constraints
Ensure: Vector containing the minimum bids of agent r for all AC in $\mathcal{T}'$

```
1:  bidVector ← {}
2:  for AC in T' do
3:      if S_r is empty then
4:          bid_AC ← costSchedule(S_r, AC, 0)
5:          bidVector ← {AC : bid_AC}
6:      else
7:          minBid_AC ← ∞
8:          for i = 0, ..., m where m is the number of tasks in S_r do
9:              Insert AC at position i in S_r
10:             Add task time points and all constraints to STN
11:             Propagate STN using Floyd-Warshall
12:             bid_AC ← costSchedule(S_r, AC, i)
13:             if bid_AC < minBid_AC then
14:                 Save i, bid_AC
15:             end if
16:             Reset STN to the copy prior to inserting task AC at position i
17:         end for
18:         bidVector ← {AC_i : bid_AC}
19:     end if
20: end for
21: return bidVector_{r_i}
```

Algorithm 5 costSchedule(S_r, AC, i)

Require: Partial schedule of tasks S_r for agent r, cost of current schedule cost_{S_r}, AC to insert in the schedule at position i, constraints
Ensure: Cost of schedule of tug r expressed in estimated delay with respect to pick-up time
  $S'_r \leftarrow S_r +$ insertion of AC at position i
  if S'_r in constraints.tabuDistribution **then**
    costIncrease $\leftarrow \infty$
  else
    cost $\leftarrow 0$
    for task t in S'_r **do**
      Compare the deadline of t with the pick-up timepoint for t in the STN
      if pick-up time $>$ deadline **then**
        cost $\leftarrow$ cost + (pick-up time - deadline)
      end if
    end for
    costIncrease $\leftarrow$ cost - cost_{S_r}
  end if
  return costIncrease

Algorithm 6 explains how the cosine degree of similarity is calculated for every newly generated allocation in the allocation solution pool. In order to be able to find the cosine similarity between two allocations, the allocations have to be converted to vectors of similar dimensions. An example of the translation of an allocation to a vector is shown in Figure A.10. The cosine degree of similarity between two vectors is then calculated according to Equation A.1 [159].

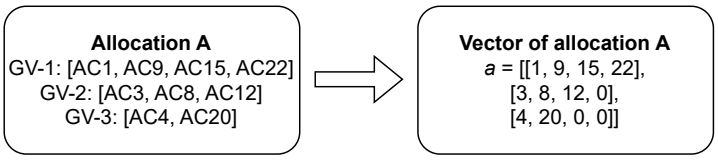


Figure A.10: Example of the conversion to a vector of an allocation of flights to three tugs in order to determine its cosine degree of similarity.

$$\text{DOS}(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} \tag{A.1}$$

Algorithm 6 calculateDOS()

Require: Allocation A to determine similarity of, cost of the allocation c_A and a set of previously explored allocations $\mathcal{A}$ in root nodes of existing search trees in path planning search forest.
Ensure: Cost of allocation corrected with the cosine degree of similarity
  if CPL-DIF used for finding a solution **then**
    Convert allocation A into a numerical vector
    storeDOS $\leftarrow$ []
    for previousAllocation in $\mathcal{A}$ **do**
      Convert previousAllocation into a numerical vector
      DOS $\leftarrow$ cosineDOS(previousAllocation, A)
      storeDOS $\leftarrow$ storeDOS + DOS
    end for
    $\text{DOS}_{mean} \leftarrow$ mean(storeDOS)
    allocationCost $\leftarrow \text{DOS}_{mean} \cdot$ allocationCost
  else
    allocationCost $\leftarrow$ allocationCost
  end if
  return allocationCost

A.3.3. Path Planning

In this section, the supporting function *updatePlan()* is shown in Algorithm 7. The function is based on the work of Ma et al. that discusses the Priority-Based Search (PBS) algorithm [140]. No changes are made to the supporting function with respect to the original implementation.

The pseudocode of the low-level search is presented in Algorithm 8 and Algorithm 9 respectively. Adaptations and changes with respect to the original implementation of the Safe-Interval Path Planning algorithm [145] are indicated in red.

Algorithm 7 updatePlan(N, a_i). Based on the work of Ma et al. [140]

LIST ← topological sorting on partially ordered set $(\{i\} \cup \{j | i \prec_N j\}, \prec_N)$
for each j in LIST **do**
  if j equals i or $\ni a_k : k \prec_N j, a_j$ collides with a_k in N.solution **then**
    Update N.plan by invoking a low-level search for a_j that avoids colliding with all agents a_k with higher priorities $(k \prec_N j)$
    if no path is returned by the low-level search **then return** False
    end if
  end if
end forreturn True

Algorithm 8 Safe-Interval Path Planning (SIPP). Adapted version of SIPP [145] with changes and additions marked in red.

Require: Set of goal configurations s_{goal}, USIs on network, agent kinematics

OPEN ← ∅
$g(s_{start}) \leftarrow 0$
$f(s_{start}) \leftarrow g(s_{start}) + h(s_{start})$
OPEN ← OPEN + s_{start}

while s_{goal} not expanded **do**
  Remove s with smallest f-value from OPEN
  successors ← getSuccessors(s)
  for each s' in successors **do**
    if s' was not visited before **then**
      $f(s') \leftarrow g(s') \leftarrow \infty$
    end if
    if $g(s') > g(s) + c(s, s')$ **then**
      $g(s') \leftarrow g(s) + c(s, s')$
      updateTime(s')
      $f(s') \leftarrow g(s') + h(s')$
      Insert s' into OPEN with $f(s')$
    end if
  end for
end while
return path

Algorithm 9 getSuccessors().

successors ← ∅
for each neighbor node n' of s **do**
  if n' not allowed to be reached from s **then**
    continue
  end if
  $m_{\text{time minimum}}$ ← time to reach n' from s with maximum velocity
  for safe interval SI_{edge} on edge to reach n' from s **do**
    if $SI_{edge,start}$ > end $time_{SI,\,s}$ or $SI_{edge,end}$ < start $time_{SI,\,s}$ **then**
      continue
    end if
    earliest leaving time from s ← max($SI_{edge,start}$, start $time_{SI,s}$)
    latest leaving time from s ← min($SI_{edge,end}$, end $time_{SI,s}$)
    if earliest leaving time > latest leaving time **then**
      continue
    end if
    earliest arrival time in n' ← max(earliest leaving time from s + $m_{\text{time minimum}}$, arrival time of previous (prioritized) vehicle in n')
    if earliest arrival time in n' > $SI_{edge,end}$ **then**
      continue
    else
      latest arrival time in n' ← min($SI_{edge,end}$, arrival time of next vehicle in n')
    end if
    for safe interval $SI_{n'}$ **do**
      if latest arrival time in n' < $SI_{n',start}$ or earliest arrival time in n' > $SI_{n',end}$ **then**
        continue
      end if
      earliest arrival time in n' ← max($SI_{n',start}$, earliest arrival time)
      latest arrival time in n' ← min($SI_{n',end}$, latest arrival time)
      if earliest arrival time in n' > latest arrival time in n' **then**
        continue
      end if
      s' ← configuration n' with interval $SI_{n'}$ and time t ← earliest arrival time
      Insert s' into successors
    end for
  end for
end for
return successors

A.4. Model Verification

This section elaborates on a selection of steps taken to verify the output of the simulation.

A.4.1. Visualization of Movements over Airport Network

Using PyGame, the output of the simulation model is visualized and shows the movement of all vehicles over the airport network. The visual tool served as means of continuous verification of the concept of operations. The movement of individual agents within the system and the sequence of their maneuvers are analyzed, and the interactions between multiple agents during conflicts are analyzed. Figure A.11 and Figure A.12 show multiple screenshots of the animation tool for (parts of) the airport network.

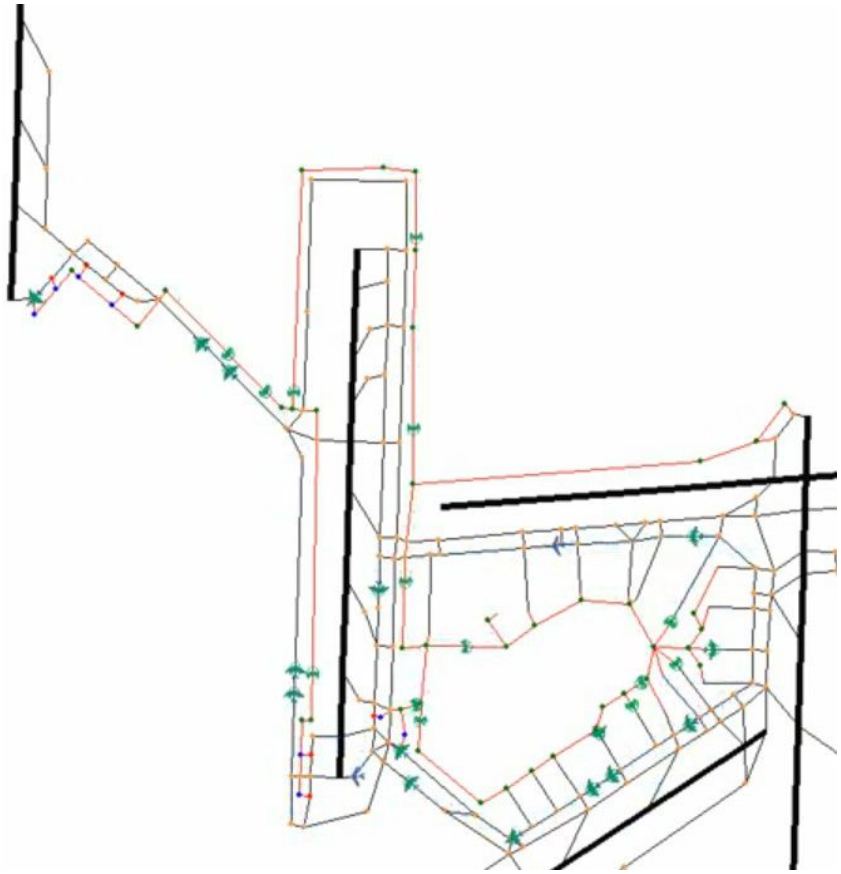
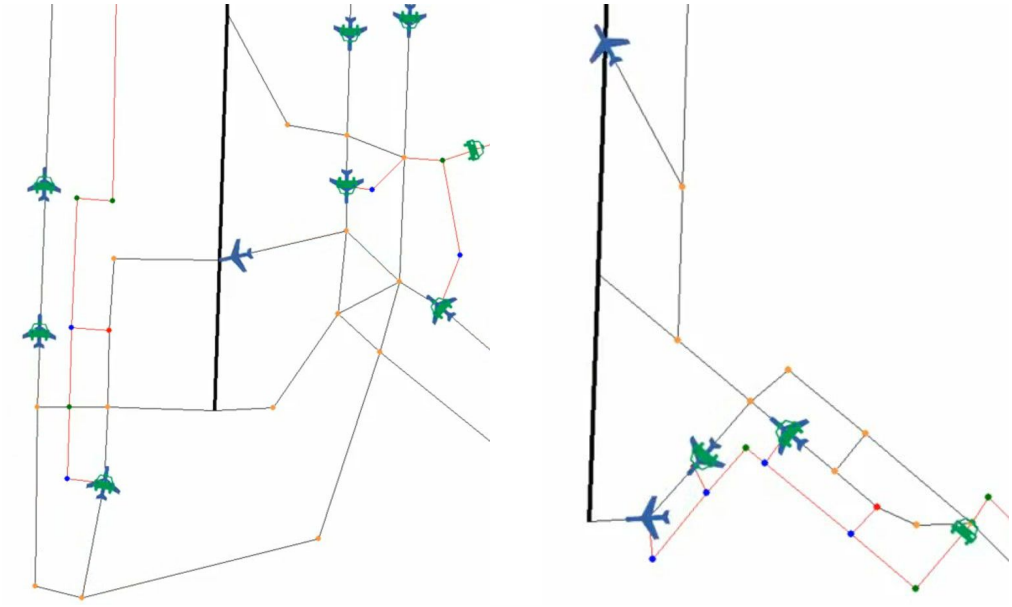


Figure A.11: Screenshot of the animation tool, showing the entire network and vehicles moving over it.



(a) South end of runway 36C (b) South end of runway 36L

Figure A.12: Screenshot of the animation tool, showing a zoomed-in section of the network near the entrance of departing runway 36C (a) and 36L (b).

A.4.2. Allocation of Outbound Flights to Tugs

Performance Analysis of TeSSI

To assess the performance of the TeSSI auction algorithm, a comparison has been made in terms of solution quality between the output of TeSSI and the MURDOCH algorithm [73]. The MURDOCH algorithm is a greedy allocation algorithm and is capable of dealing with an unbalanced assignment problem, in which the number of tasks to assign is larger than the number of agents. In addition, the MURDOCH algorithm is proven to be 3-competitive for the unbalanced assignment problem, meaning that the output of the algorithm is equal to at most three times the optimal solution. The general definition of a σ-competitive algorithm is shown in Equation A.2 [160].

When running the MURDOCH algorithm, all robots are initially assigned to their first task in an optimal way, using the Hungarian method. Once a robot finishes its task, it gets assigned one of the remaining tasks for which the costs are lowest. Similarly as for the TeSSI auction algorithm, the cost of an allocation is based on the resulting sum of the delay in off-block time of all aircraft.

For a time window between 21:00h and 21:30h in the outbound peak on July 17th in 2019 (refer to Appendix B for more info on the input data), a comparison is made between the allocation cost generated by TeSSI and by the MURDOCH algorithm for various number of tugs available. Based on the total number of outbound aircraft scheduled to depart in the time window (39), three different fleet sizes are considered: 39 tugs (balanced assignment problem), 19 and 13 tugs (unbalanced assignment problem). For every run, tugs are initialized at a random position at the service road network. In total, 100 runs are performed for every scenario to use as input for the comparison of the performance of TeSSI against the MURDOCH algorithm. The relative performance of TeSSI with respect to MURDOCH is calculated according to Equation A.3, where c_{TeSSI} and c_{MURDOCH} represent the allocation cost of TeSSI and MURDOCH respectively. For every fleet size, a number of different sizes of the allocation solution pool for TeSSI (refer to Section 4 in the scientific paper) has been evaluated as well.

$$c_{\text{algorithm i}} \leq \sigma \cdot c_{\text{optimal algorithm}} \qquad \text{(A.2)}$$

$$\text{performance} = \sigma \geq \frac{c_{\text{TeSSI}}}{c_{\text{MURDOCH}}} \qquad \text{(A.3)}$$

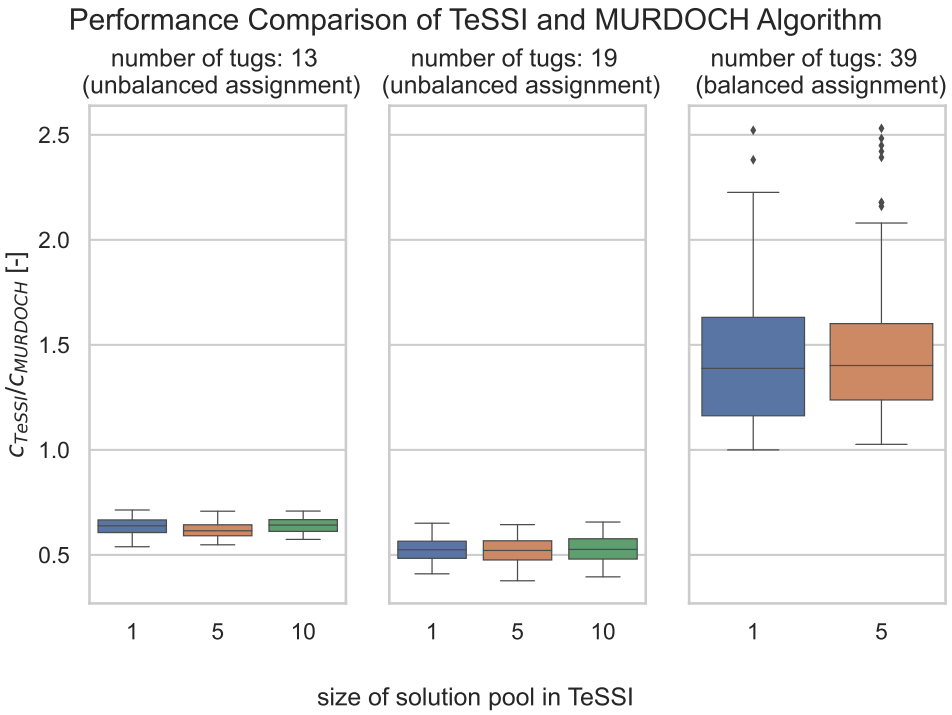


Figure A.13: Performance comparison of TeSSI and the MURDOCH algorithm for various fleet sizes and different sizes of allocation solution pools. For solution pools larger than 1, the allocation with minimum cost is chosen from the solution pool and used for comparison.

In Figure A.13, the performance of the TeSSI auction algorithm with respect to the MURDOCH algorithm is shown for various fleet sizes. We can see that for both unbalanced assignment problems (13 and 19 tugs available), TeSSI outperforms the MURDOCH algorithm and provides for a total sum of delay in off-block time that is between 30% and 50% lower. When evaluating the performance of TeSSI

for the balanced assignment problem, note that the MURDOCH algorithm generates the optimal solution using the Hungarian method. In general, we can see that TeSSI does not approach the optimal solution provided by the MURDOCH algorithm, which is in line with our expectations. When outliers are excluded, TeSSI is empirically shown to be almost 2-competitive with respect to the optimal solution for worst case solutions.

In order to evaluate the consequences of a larger allocation solution pool, the chances of finding an allocation with lower cost in a larger solution pool are investigated. This is done by evaluating the index of the allocation with minimum cost in the solution pool for various fleet sizes. Figure A.14 shows that for larger fleet sizes, the allocation with minimum cost is found in the first run of TeSSI for the majority of the cases. However, for smaller fleet sizes, the allocation with minimum cost is found only after several iterations of TeSSI in at least half of the cases. This result indicates that a larger allocation solution pool increases the chances of finding an allocation with minimum cost, especially when resources (robots) are limited with respect to the number of tasks.

However, Figure A.15 shows that a larger allocation solution pool significantly increases required run times, independent of fleet size. For larger fleet sizes, the required run times exponentially increase for larger solution pools. To limit computational complexity while still increasing the chance to find an allocation that minimizes delay in off-block time, an allocation solution pool of 5 is used in the scenarios evaluated.

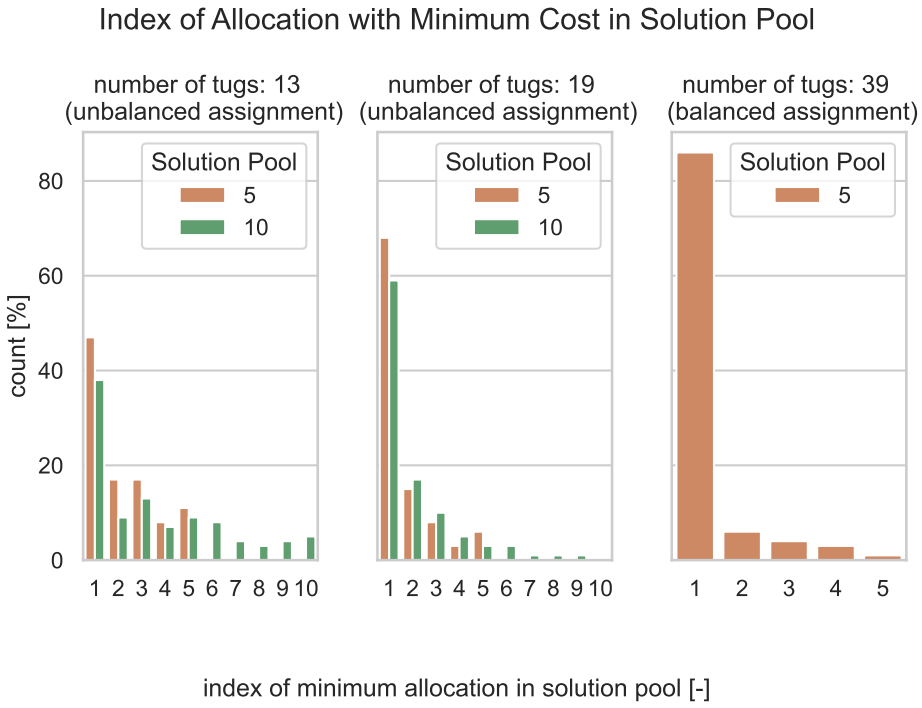


Figure A.14: Comparison of the index of the allocation with minimum cost in the allocation solution pool when running the adapted version of TeSSI for various fleet sizes.

Tug Schedule Consistency

Figure A.16 shows a snapshot of the sequence of different maneuvers for tugs over time during the outbound peak for a scenario in which unlimited tugs are available. The figure is used to verify the consistency of all schedules of the tugs and shows that all towing (driving pilot) activities are preceded by a coupling activity and followed up by a decoupling activity and tug return movement (driving solo) activity. In addition, no overlap between the towing activities is seen, indicating that the schedules are consistent.

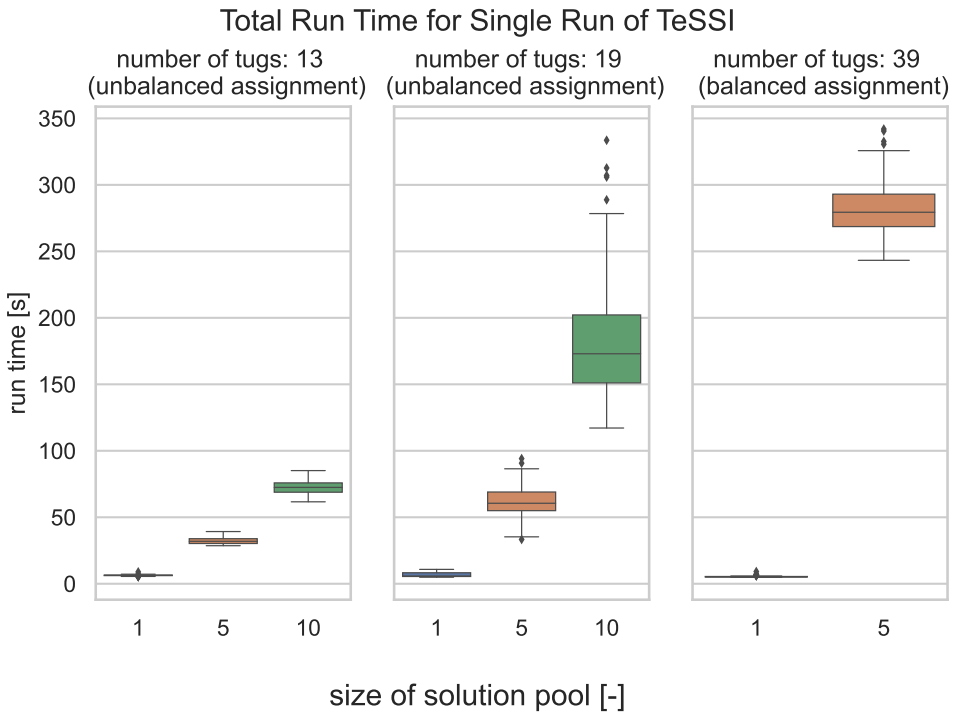


Figure A.15: Comparison of the run time performance of TeSSI for various fleet sizes and different sizes of the allocation solution pool.

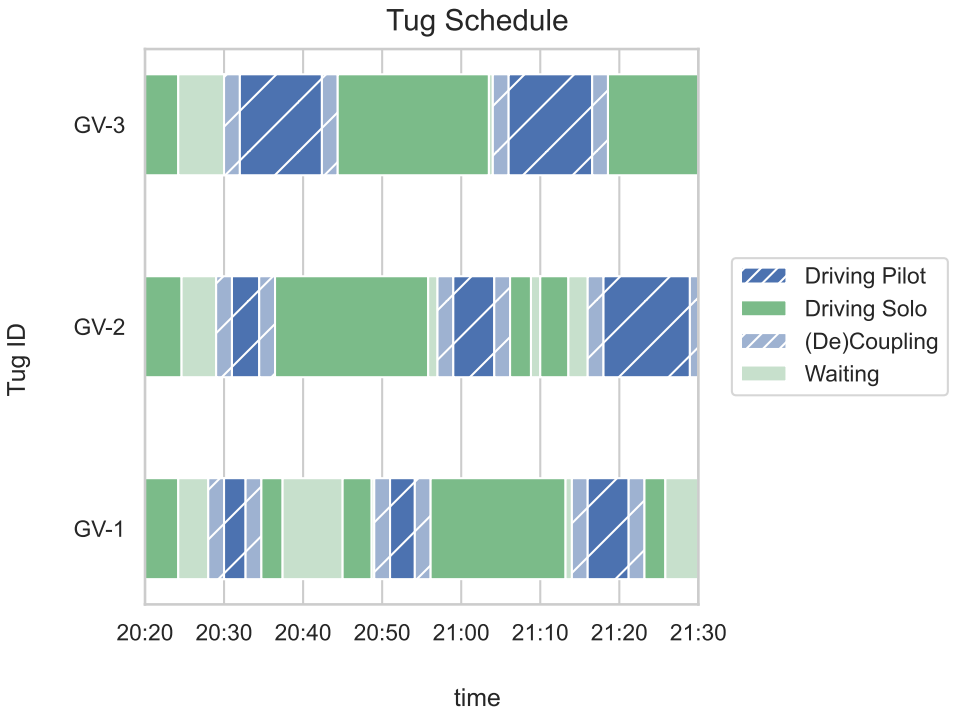
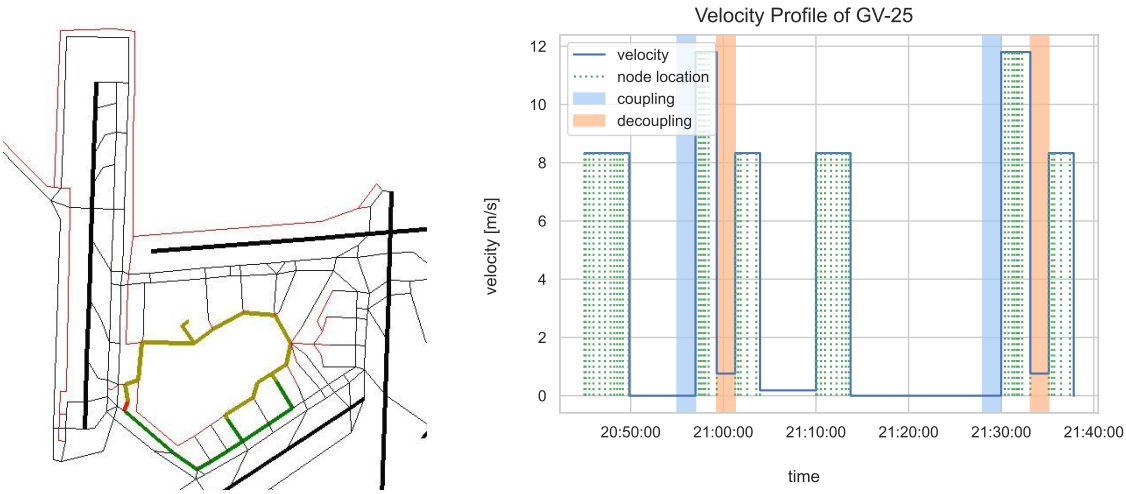


Figure A.16: Snapshot of the schedules of three tugs in the operational scenario with unlimited tugs are available in the outbound peak.

A.4.3. Route Planning of Individual Vehicles

The routes of individual vehicles are verified using velocity heatmaps and velocity profiles over time. Two examples are provided for two tug vehicles that are routed in the outbound peak and shown in Figure A.17 and Figure A.18 respectively. Both figures show that the vehicles travel along the shortest

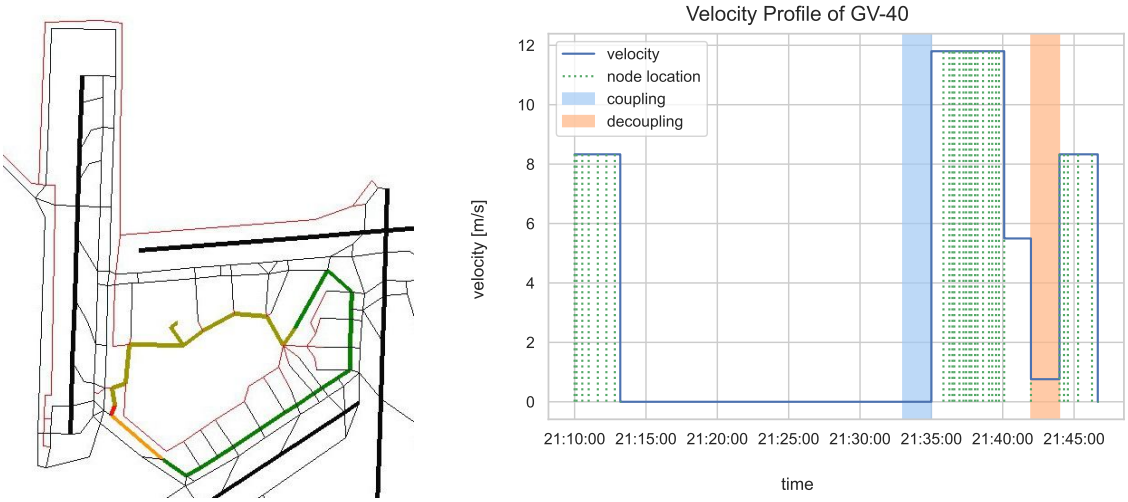
path in the network and that the velocity profiles match with the kinematics as presented in Appendix B. It can be clearly seen that the velocity is constant when travelling over an edge and only changes (instantaneously) when the vehicle is at a node location, which is in line with the assumptions.



(a) Velocity heatmap of the travelled path of tug with ID 25.

(b) Velocity profile of the travelled path of tug with ID 25.

Figure A.17: Figure showing the travelled path of tug with ID 25, including its velocity profile.



(a) Velocity heatmap of the travelled path of tug with ID 40.

(b) Velocity profile of the travelled path of tug with ID 40.

Figure A.18: Figure showing the travelled path of tug with ID 40, including its velocity profile.

B

Simulation Setup

The following chapter provides additional information on the setup of the simulation scenarios. In section B.1, the most important input parameters are listed that are independent of the scenario to evaluate. Then, in section B.2, all characteristics of the vehicles modelled in the simulation are provided. Finally, details on the flight schedule input are discussed in section B.3.

B.1. Simulation Input Parameters

For all scenarios evaluated, a number of input parameters are fixed, independent of fleet size or approach used. In Table B.1, an overview is provided of the most important fixed input parameters.

Table B.1: Overview of the most important fixed input parameters for the simulation model.

Parameter	Symbol	Value
planning window [min]	w_{plng}	30
replanning frequency [min]	h_{plng}	25
time step [s]	dt	1
separation margin [m]	d_{safety}	150
buffer in allocation [s]	$t_{\text{buffer allocation}}$	60

B.2. Vehicle Input Parameters

In this section, additional information is provided on the vehicles and their characteristics that are routed in the model. In subsection B.2.1, a mapping of all aircraft types present in the flight schedule to an ICAO and RECAT-EU category is provided. The specific kinematic characteristics of all vehicles are provided in subsection B.2.2. Finally, runway separation constraints are listed in subsection B.2.3.

B.2.1. Aircraft Characteristics

All aircraft in the flight schedule are mapped to a category in the ICAO Aerodrome Reference Codes and to a category that specifies the required runway separation based on wake turbulence constraints.

Table B.2: Mapping of all aircraft in the flight schedule to ICAO Aerodrome Reference Code and RECAT-EU category for wake turbulence constraints. Based on previous work [9].

Aircraft Name	ICAO Category	WTC Category
Cirrus Vision SF50	Cat A	Cat F
Diamond Katana (DA-40)	Cat A	Cat F
Bombardier Learjett (all types)	Cat B	Cat F
Embraer 120 Brasilia Adv.	Cat B	Cat F
Canadair CRJ-100/200ER	Cat B	Cat E
Cessna (all types)	Cat B	Cat E
Embraer EMB-135	Cat B	Cat E
Embraer EMB-505 Legacy 500	Cat B	Cat E
Dassault Falcon (all types)	Cat C	Cat E
Bombardier Q400	Cat C	Cat E
Embraer EMB-170/190	Cat C	Cat E
Boeing B737-300 (+ winglets)	Cat C	Cat D
Gulfstream (all types)	Cat C	Cat D
Airbus A318-100	Cat C	Cat D
Boeing B737-900	Cat C	Cat D
Boeing B737-600	Cat C	Cat D
Boeing B737-700	Cat C	Cat D
McDonnell Douglas MD-80	Cat C	Cat D
Boeing B737-800	Cat C	Cat D
Bombardier CS-300	Cat C	Cat D
Boeing B737-800 NG	Cat C	Cat D
Airbus A320-200 Sharklets	Cat C	Cat D
Boeing 737-700 NG	Cat C	Cat D
Airbus A319-100 LR/CJ	Cat C	Cat C
Airbus A320-200	Cat C	Cat C
Boeing B737-200F	Cat C	Cat C
Airbus A310-200/300F	Cat D	Cat B
Airbus A300F	Cat D	Cat B
Boeing B767-300	Cat D	Cat B
Boeing B757F	Cat D	Cat D
Airbus A330-200/300	Cat E	Cat B
Airbus A340-300E	Cat E	Cat B
Boeing B747-400 + 400ER	Cat E	Cat B
Boeing B747-400 Combi	Cat E	Cat A
Airbus A350-900/1000	Cat E	Cat B
Boeing B777F	Cat E	Cat E
Boeing B777/787	Cat E	Cat E
Boeing B747-800F	Cat F	Cat F
Airbus A380	Cat F	Cat E

B.2.2. Vehicle Kinematic Parameters

Table B.3 is displaying the maximum speed of aircraft, tug-aircraft combinations and tugs driving in solo mode.

Table B.3: Overview of the kinematic characteristics of all vehicles in the simulation.

Parameter	Symbol	Value
maximum velocity of aircraft [m/s]	$v_{\text{max, AC}}$	15
maximum velocity of tug in pilot mode [m/s]	$v_{\text{max, pilot}}$	11.8
maximum velocity of tug in solo mode [m/s]	$v_{\text{max, solo}}$	8.33

B.2.3. Runway Separation Constraints

All aircraft are mapped to a RECAT-EU category. This category is used to impose runway separation constraints, based on the values shown in Table B.4 [161]. All separation constraints are expressed in seconds.

Table B.4: Overview of the runway separation requirements based on RECAT-EU category, expressed in seconds [161].

Leader Follower	Cat A	Cat B	Cat C	Cat D	Cat E	Cat F
Cat A	60	100	120	140	160	180
Cat B	60	60	60	100	120	140
Cat C	60	60	60	80	100	120
Cat D	60	60	60	60	60	120
Cat E	60	60	60	60	60	100
Cat F	60	60	60	60	60	80

B.3. Flight Input Data

In Figure B.1, the flight movements over time (aggregated using an interval of 15 min) are shown for July 17th, 2019 at AMS. In addition, the number of flights departing per runway is shown in Figure B.2. The duration of the peaks is determined based on the number of departing and arrival runways that are in use, aggregated using intervals of 15 minutes. During an outbound peak, two departing and one arrival runway is in use; during an inbound peak, two arrival and one departing runway is in use; and in a transition period, two arrival and two departing runways are in use.

Based on both figures, it can be seen that the outbound peak with the highest amount of departures that lasts at least 1.5h takes place from 20:30-22:00h. For the inbound peak, two options can be considered: one from 07:30-09:00h and the other from 18:30-20:00h. Although the number of arrivals is larger in the inbound peak at the end of the day, the number of departures from 36L during that time slot is less consistent over time than in the time slot between 07:30-09:00h. Therefore, the latter time slot is used for evaluation of the inbound peak.

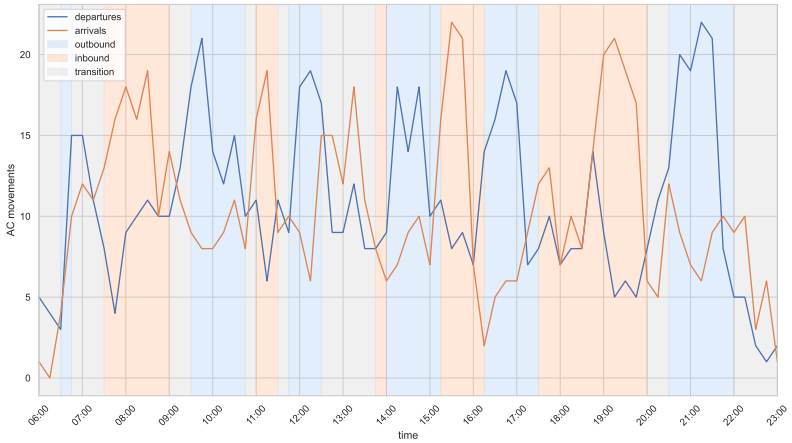


Figure B.1: Flight movements over time during July 17th, 2019.

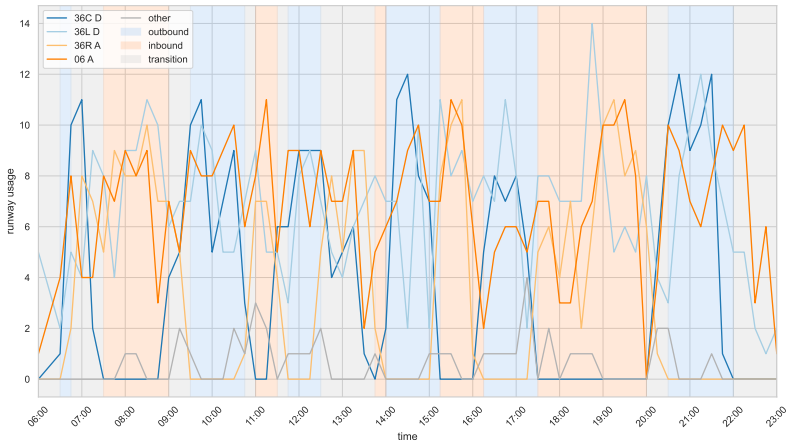


Figure B.2: Runway usage over time during July 17th, 2019.

Figure B.3 shows the breakdown of the various aircraft types that are scheduled to depart and arrive during each peak. It can be observed that the majority of the flights can be grouped under ICAO category C. The B737 and A320 family both belong to this category and are so far the only aircraft for which the TaxiBot is certified.

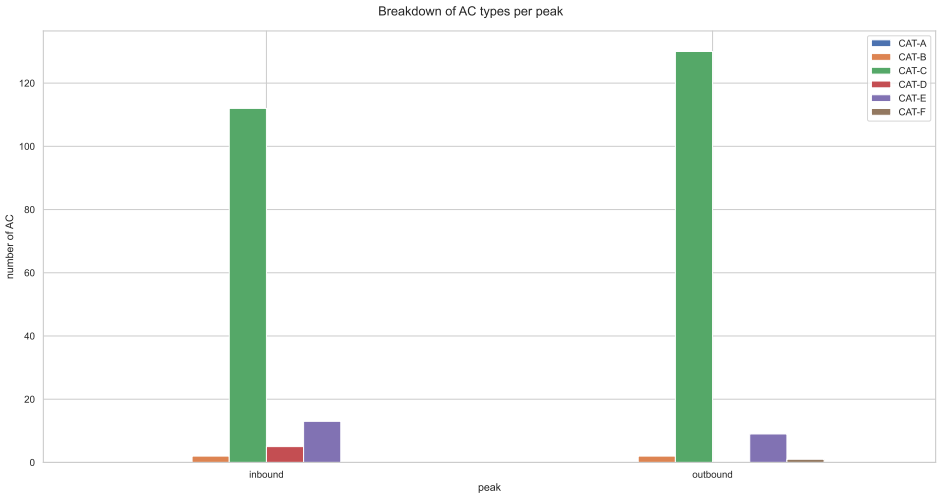


Figure B.3: Breakdown of ICAO aircraft types per peak.

B.3.1. Assumptions Related to Flight Input Data

In this section, a list of assumptions on the flight input data is stated.

- Flights that are present in the historic flight schedule departing or arriving at a runway that should strictly speaking not be in use during an outbound/inbound peak when operating RMO North, are deleted from the flight schedule. The active runways in RMO North can be found in Appendix A.
- Flights that are present in the historic flight schedule that are assigned a ramp outside of the Schiphol Centre parking area are deleted from the flight schedule. Similarly, flights that are assigned a HG-ramp are excluded from the flight schedule.

C

Additional Simulation Results

In this chapter, an elaboration is provided on the statistical analysis performed for hypothesis testing in section C.1. Furthermore, additional algorithmic and operational results are provided in section C.2 and section C.3 respectively. Finally, the coefficient of variation as a function of the number of simulation runs is shown for all scenarios evaluated in section C.4.

C.1. Elaboration on Statistical Analysis

In this study, hypothesis testing is performed using several statistical tests to analyze the data. Initially, the Shapiro-Wilk test is utilized to assess the normality of the data, leading to the decision to employ non-parametric tests since all data was found not to be normally distributed [162]. For comparing identical agents across different scenarios, the non-parametric Wilcoxon Signed-Rank test for paired populations is used [163]. Likewise, the non-parametric Mann-Whitney U test is applied for comparing results of different groups of agents [164]. To maintain a significance level of $\alpha = 0.05$ and minimize the risk of falsely rejecting the null hypothesis, the Bonferroni correction method was employed to adjust for family-wise error rates in multiple comparisons [165]. Based on the number of tests performed on a dependent variable (n), the alpha level is corrected using Equation C.1.

$$\alpha = \frac{0.05}{n} \tag{C.1}$$

In cases where a significant difference between two groups is detected, the Vargha-Delaney A-test is used as a measure of effect size [166]. This test provides a value ranging from 0 to 1, representing the probability of randomly selecting an observation from one group that is greater than an observation from the other group. A value of 0.50 indicates no difference between the groups, while values above 0.56, 0.64, and 0.71 are considered indicative of small, medium, and large differences, respectively. The same intervals apply when the value is below 0.5.

C.2. Additional Algorithmic Results

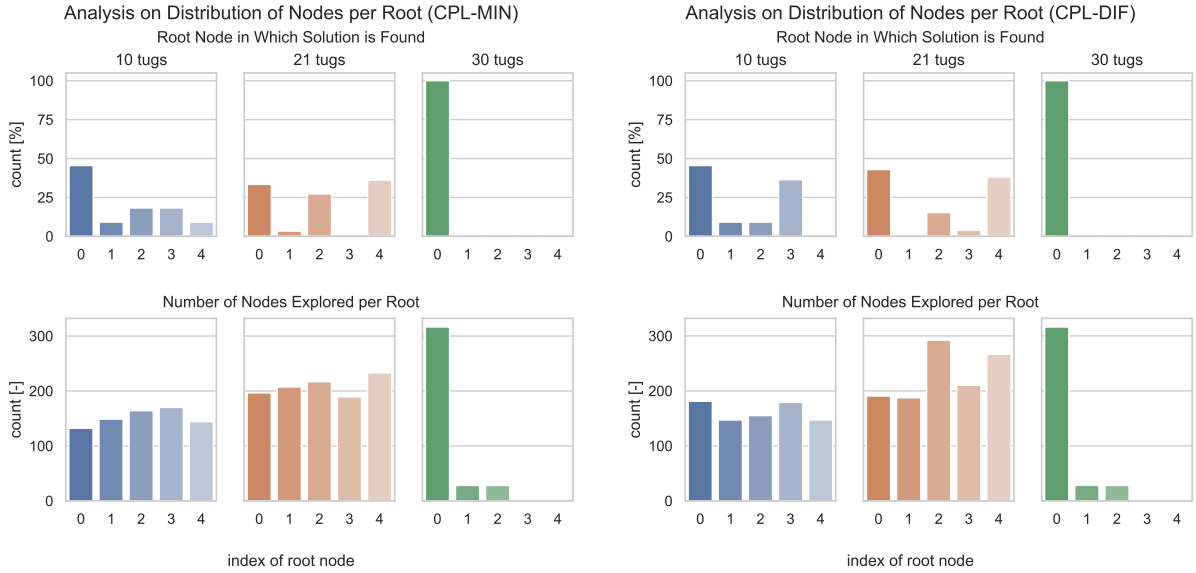
In this section, additional algorithmic results are provided, including an analysis of the allocation solution pool and the number of nodes explored for all algorithms evaluated.

C.2.1. Number of Nodes Explored per Algorithm

In Section 7 of the scientific paper, it is observed that large differences exist in the run times when comparing both coupled algorithms (CPL-MIN and CPL-DIF) with the decoupled algorithm for fleet sizes of 10 and 21 tugs available. It is stated that this difference is due to the increase in the number of nodes explored in the path planning search forest for the coupled algorithms when compared with the decoupled algorithm. In this section, results will be provided that support this conclusion.

Figure C.1 shows a comparison between the index of the root node in which the final solution is found and the number of nodes explored per root for the CPL-MIN and CPL-DIF algorithm. When comparing the number of nodes explored per simulation run for both coupled algorithms with the decoupled

algorithm (Table C.1), we can see that this number is significantly higher for a coupled algorithm compared to a decoupled algorithm, except for the scenario when having 30 tugs available. This can be explained due to the fact that the majority of the nodes in this scenario is explored in the first root node (Figure C.1). Therefore, the average number of nodes explored in one simulation is comparable for all three algorithms.



(a) CPL-MIN (b) CPL-DIF

Figure C.1: Graphs showing in which root node the solution is found and the number of nodes that are explored on average per root during one simulation, for CPL-MIN (a) and CPL-DIF (b).

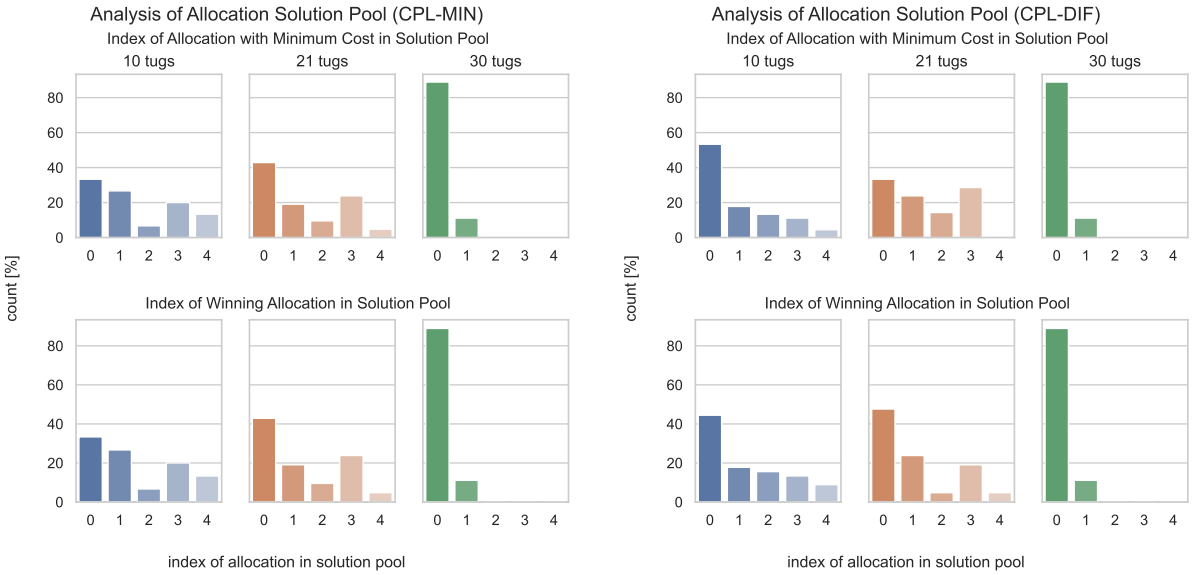
Table C.1: Comparison of the average number of nodes explored per simulation run for DCPL, CPL-MIN and CPL-DIF.

Fleet size	Average number of nodes explored per simulation run		
	DCPL	CPL-MIN	CPL-DIF
10 tugs	196	759	809
21 tugs	288	1042	1145
30 tugs	316	376	376

C.2.2. Allocation Solution Pool

In Section 4 of the scientific paper, the difference between the two coupled algorithms CPL-MIN and CPL-DIF is explained. The difference between both algorithms relates to the allocation that is picked from the allocation solution pool and used for the routing of vehicles. Whereas the allocation with minimum cost is picked in the CPL-MIN algorithm, a measure of similarity of all allocations in the solution pool with respect to previously explored allocations is included for CPL-DIF (for more details, refer to Section 4 of the scientific paper).

In this section, the influence of the measure of similarity on the cost calculation of allocations in the solution pool is elaborated on. In Figure C.2, two graphs are shown that compare the index of the allocation with minimum cost to the index of the winning allocation in the solution pool. As expected, no differences between both graphs are seen for CPL-MIN (Figure C.2a), since the allocation with minimum cost is picked as the winning allocation from the solution pool. From Figure C.2b, differences can be observed between the distributions for both types of indices for CPL-DIF. Thus, by including the cosine degree of similarity in the calculation of the overall cost of the allocations, the winning allocation is not in all cases the allocation with minimum schedule costs in the solution pool.



(a) CPL-MIN

(b) CPL-DIF

Figure C.2: Graphs showing the index of the allocation with minimum cost in the allocation solution pool and the index of the winning allocation in the allocation solution pool, for CPL-MIN (a) and CPL-DIF (b).

C.3. Additional Operational Results

In this section, additional operational results are provided, including the ratio of the waiting time of tugs over the total simulation time, as well as the number of ramp conflicts occurring for various fleet sizes.

C.3.1. Tug Waiting Time

In Figure C.3, the ratio of the waiting time over the total simulation time for all tugs is depicted for various fleet sizes. It can be seen that the ratio decreases as the fleet size decreases, which is according to our expectations. In addition, large differences can be observed for this ratio between tugs within the fleet for larger fleet sizes. This result indicates that the added value per tug decreases for an increasing fleet size.

In Table C.2, the number of conflicts at the ramp between arriving and departing flights is shown for various fleet sizes. In the case that a delayed departing aircraft still occupies a ramp that is assigned to an inbound flight that has reached the ramp, a conflict is detected (but not resolved, refer to subsection A.1.2). Note that we are recording the conflicts for the actual ramps, and not the meta-ramp nodes. We can see from the results in Table C.2 that a limited number of ramp conflicts occurs when having 10 or 21 tugs available in the outbound peak. Although the number of ramp conflicts is small in comparison to the total flight movements during the outbound peak (143), it still indicates that significant disruptions occur in terms of gate scheduling when having limited tugs available for outbound towing.

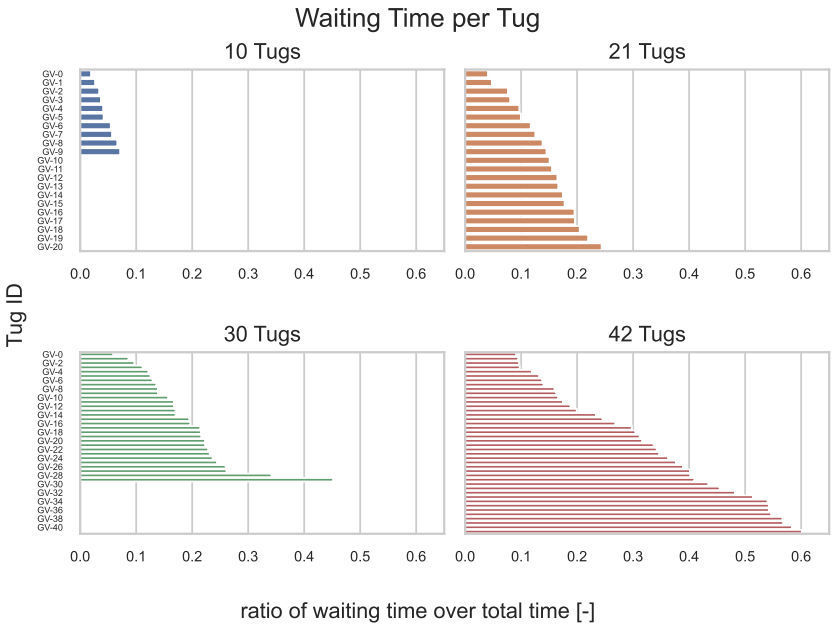


Figure C.3: Ratio of the waiting time per tug over the total simulation time for different fleet sizes. Based on simulations during the outbound peak.

Table C.2: Number of conflicts between arriving and departing flights at their assigned ramps (due to delay of departing aircraft) in the outbound peak.

Fleet size	10 tugs	21 tugs	30 tugs	42 tugs
Number of ramp conflicts	10	4	0	0

C.4. Analysis of Number Runs

In order to determine the number of runs per scenario, the coefficient of variation for the sum of total delay per flight for all aircraft is determined using Equation C.2. When plotted as function of the number of runs, a stabilization of the coefficient of variation indicates that sufficient runs are performed to perform analysis.

$$\mathrm{CV} = \frac{\sigma}{\mu} \tag{C.2}$$

In Figure C.4, Figure C.5 and Figure C.6, the coefficient of variation is shown for conventional taxiing operations in the outbound and inbound peak, for tug-enabled taxiing operations in the outbound peak and for tug-enabled taxiing operations in the inbound peak respectively. From the figures, it can be seen that 30 runs for all scenarios is definitely enough for stabilization of the coefficient of variation. Since hardly any variation exists as a function of the number of runs, one simulation per scenario would have sufficed as well. For the scenario of 21 tugs, slight variation exists. This can be explained by the fact that there are some situations in the TeSSI and PBS algorithm where tie-breaking is performed, which introduces variability in the generated solution.

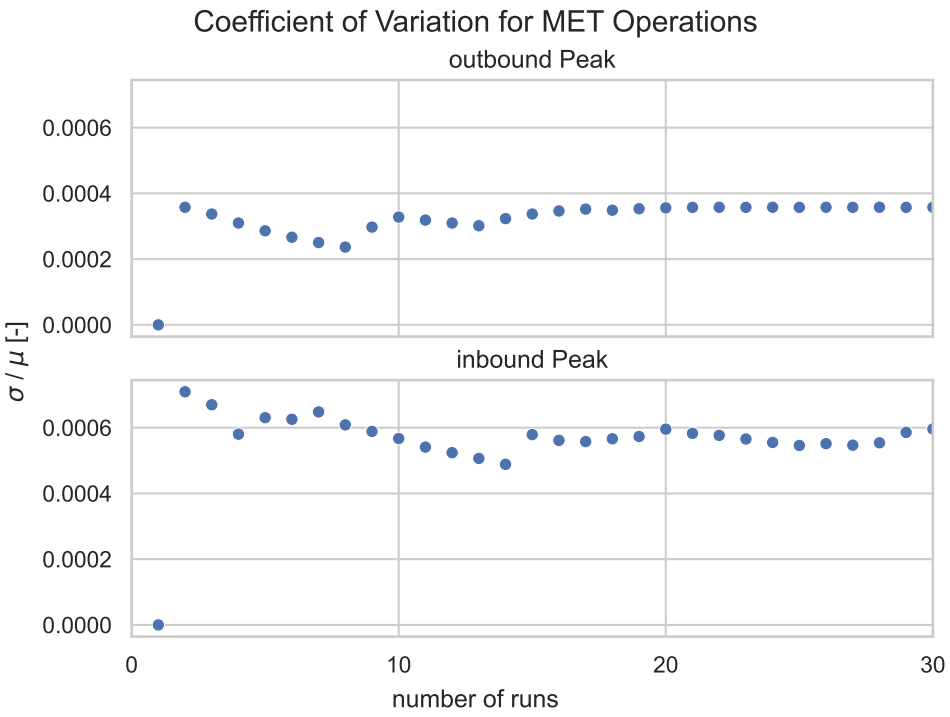


Figure C.4: Coefficient of variation for the sum of the total delay per flight as a function of the number of runs, for all conventional taxiing scenarios.

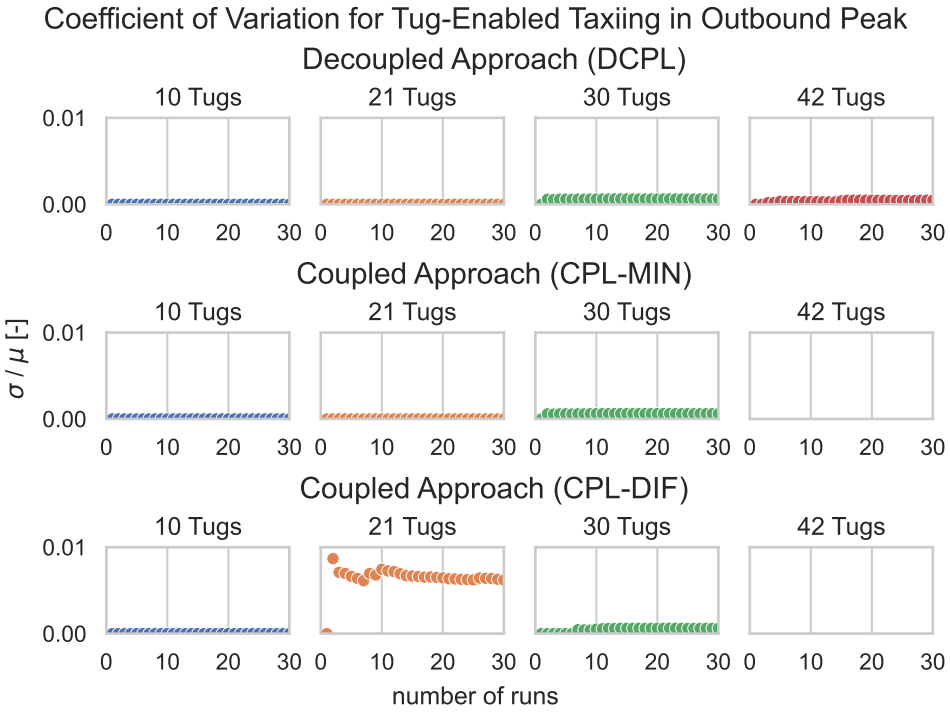


Figure C.5: Coefficient of variation for the sum of the total delay per flight as a function of the number of runs, for all tug-enabled taxiing scenarios in the outbound peak.

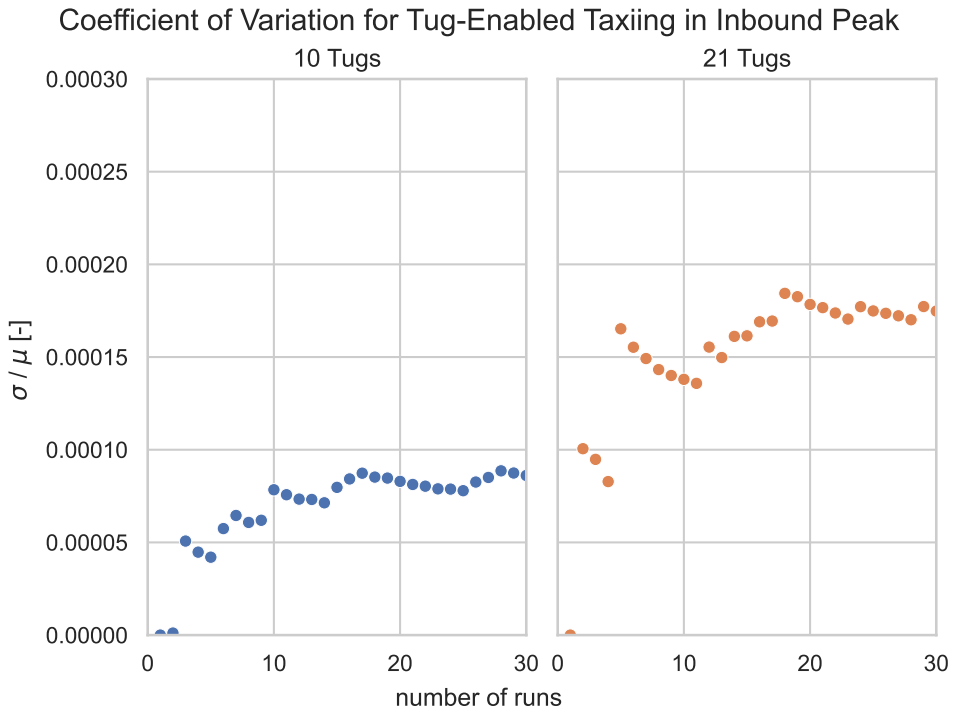


Figure C.6: Coefficient of variation for the sum of the total delay per flight as a function of the number of runs, for all tug-enabled taxiing scenarios in the inbound peak.

D

AIP Information

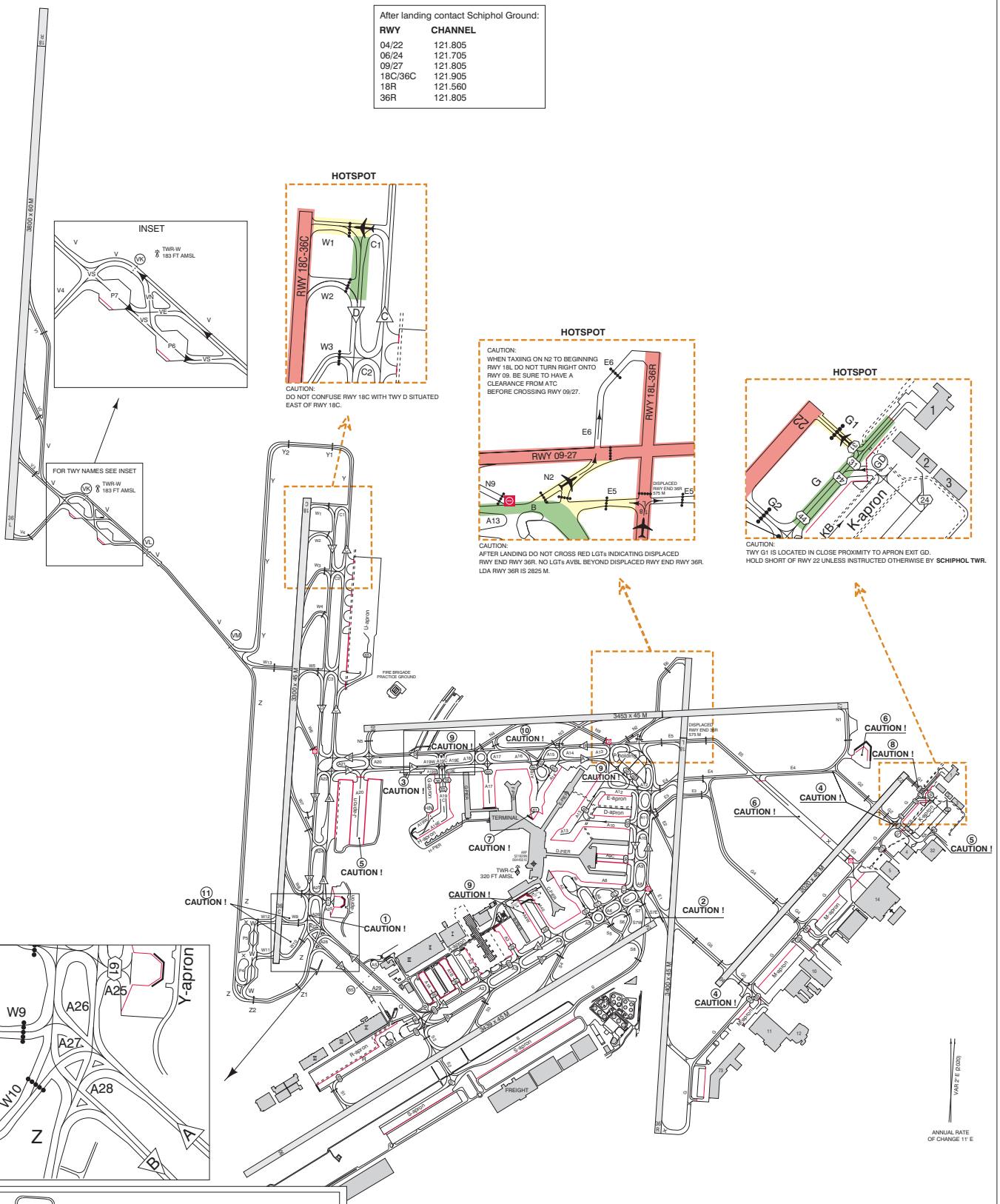
In this chapter, charts from the Aeronautical Information Package (LVNL) are provided that are relevant for the research [158].

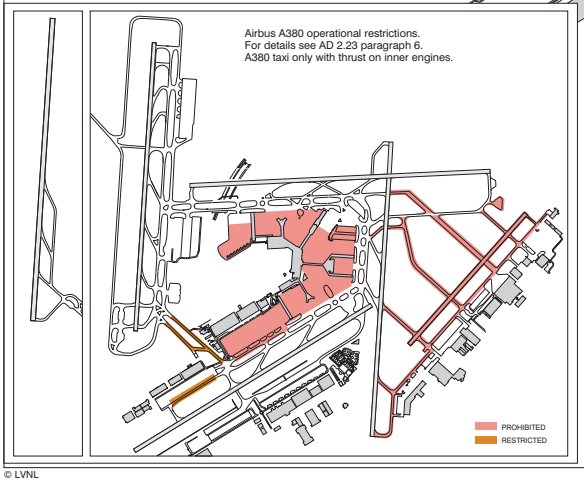
D.1. Aerodrome Ground Movement Chart

AD ELEV -11

After landing contact Schiphol Ground:

RWY	CHANNEL
04/22	121.805
06/24	121.705
09/27	121.805
18C/36C	121.905
18R	121.560
36R	121.805





DIRECTIONS ARE MAGNETIC
ELEVATIONS IN FEET AMSL
DIMENSION IN METERS

SCALE 1 : 20 000

M 500 0 500
FT 1000 0 1000 2000

General taxi instructions

RTF instruction inbound:
VIA NORTH: taxi via TWY A and north side of airport.
VIA SOUTH: taxi via TWY B and TWY Q.

RTF instruction outbound:
VIA NORTH: taxi via TWY B and north side of airport.
VIA SOUTH: taxi via TWY A and TWY Q.

CAUTION:

1. Do not confuse RWY 36C with TWY B situated east of RWY 36C.
2. TWY S7W designated for crossing RWY 06/24 only.
3. Avoid holding on the upslope between A19 and A20 to prevent backward movement of the aircraft.
4. Oversteering required for aircraft with wingspan >= 36 M.
5. J-apron and K-apron is not controlled by ATC.
6. Towing only.
7. MAX wingspan 61 M only applicable to aircraft taxiing to E3.
8. Vacating RWY 04 via TWY G1 is restricted to aircraft with a maximum wingspan of 31 M due to wingspan restriction on adjacent taxiways.
9. Standard taxi routing, unless otherwise instructed by ATC, for ACFT docking at ACFT stands specified below:
   ACFT stands B15 - B35: TWY A4W.
   ACFT stands C6 - C14: TWY A4E.
   ACFT stand E24: aircraft with wingspan greater than 65 M: TWY A12.
   ACFT stands G3 - G9 and H1 - H7, aircraft with wingspan 36 M or less: from TWY A/B via TWY A19E (orange line).
   ACFT stands G71 - G79, aircraft with wingspan 36 M or less: from TWY A/B via TWY A19W (blue line).
   ACFT stands G3 - G9 and G73 - G79, aircraft with wingspan greater than 36 M: from TWY A/B via TWY A19C.
10. When vacating RWY 27 N3 for TWY A, take the first left turn on N3 to cross TWY B.
11. After vacating RWY 18C via TWY W9 or TWY W10, taxiing is only possible in non-standard taxi routing either to TWY A southbound or TWY B northbound.

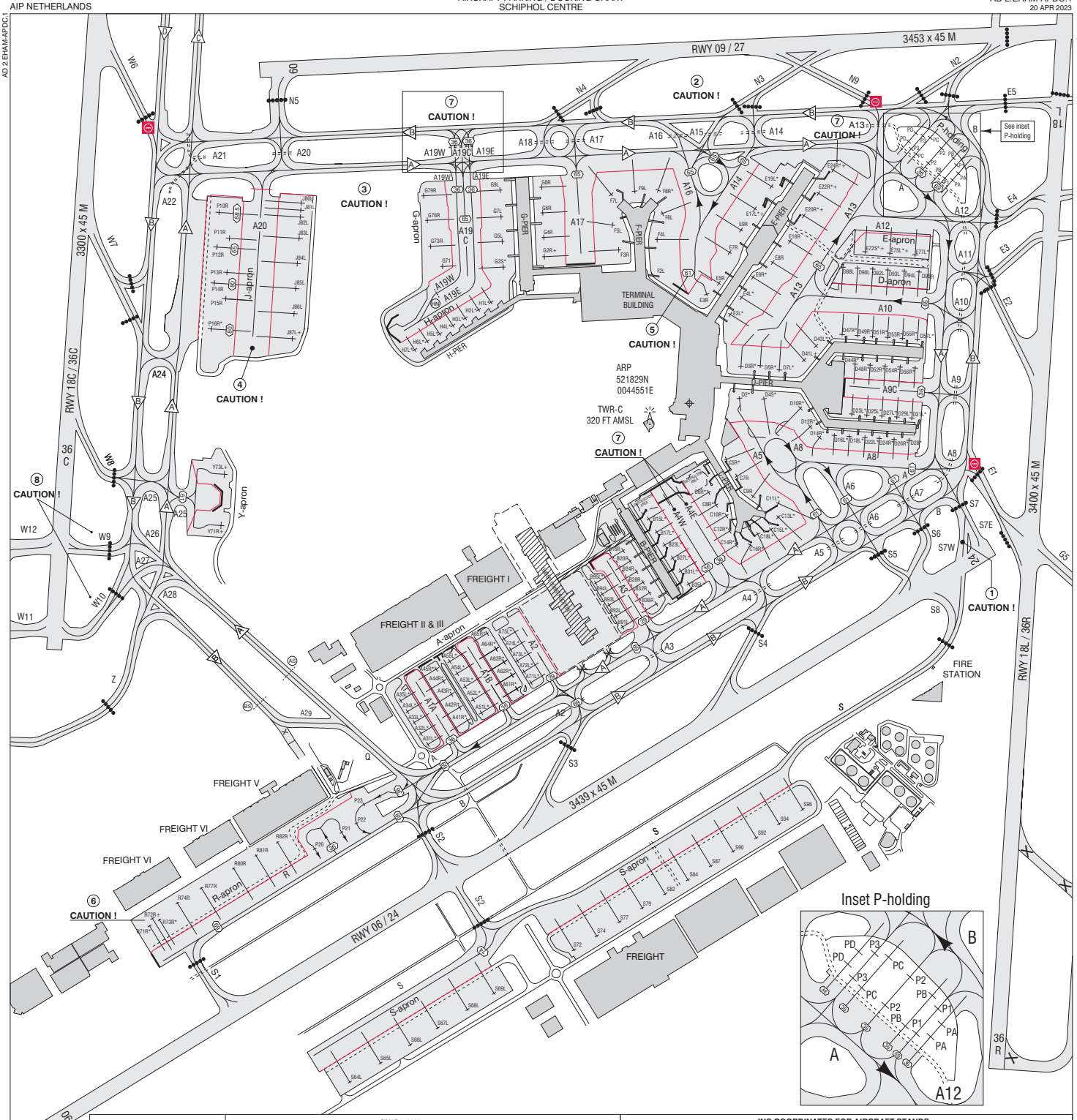
LEGEND

- No-entry: runway entry prohibited at this point. TWY E1 from TWY A, A8 and B. TWY G3 from TWY G. TWY N9 from TWY A and B. TWY W6 from TWY A, B and D.
- STOP BAR
- INTERMEDIATE HOLDING POSITION LIGHTED
- MAX WINGSPAN 29 M LIMITED ENTRY
- STANDARD TAXI ROUTING, UNLESS OTHERWISE INSTRUCTED BY ATC. ALL OTHER ROUTES MAY BE USED TWO-WAY ON ATC DISCRETION ONLY.
- ATC SERVICE BDRY
- BLAST FENCE
- HANGAR NO. 12
- FREIGHT STATION NO. II
- SERVICE ROAD
- NOT IN USE

HOTSPOT
- RUNWAY
- ENTRY / EXIT
- TAXIWAY

CHANGE: TWY W and W13; CAUTION; Taxi instructions; MAX WING SPAN added; editorial.

D.2. Aircraft Parking Chart Schiphol Centre



DIRECTIONS ARE MAGNETIC
ELEVATIONS IN FEET AMSL
DIMENSION IN METERS

SCALE 1 : 8 000

M 100 0 100 200 300 400 500
FT 500 0 500 1000 1500

VAR 2° E (2020)

NOTES

1. If the distance between the aircraft's engines and ground is in excess of 5 metres (16.4 FT) engine nr.2 may not be used at breakaway power at the gate and shall run idle until normal taxi speed has been reached due to blast problems.

2. At all aircraft stands except G71 nose-in parking and push-back procedures are applicable.

3. Remarks on push-pull / push-back procedures:
   C13: push-back into TWY A5.
   D2 & D4: B767 and larger push-back via TWY A5 next to C11.
   D3, D5: push-back into TWY A10.
   D7, D43: B757 and larger push-back into TWY A10.
   E2, E4: push-back into TWY A10.
   E3, E5, E7, E9: B757 and larger push-pull on TWY A16. Other aircraft push-back into TWY A14.
   E6: B757 and larger push-back into TWY A10.
   E8, E18: B757 and larger push-pull.
   E20, E22, E24: push-pull onto TWY A12.
   F2, F4, F6: B757 and larger push-pull on TWY A16. Other aircraft push-back into TWY A14.
   F3: push-back on TWY A opposite G9.
   G3: aircraft wingspan up to 36 M push-back on TWY A19W.
   G3, G5, G7, G73, G76: B757 and larger push-pull.
   G4: B767 and larger push-pull.
   G71: REF EHAM AD 2.20 paragraph 10 G-apron - push and hold procedure aircraft stand G71.
   H1, H2: push-back on TWY A19E.
   S-apron: push-back direction will be instructed by ATC.

4. Self-docking procedures apply to:
   All aircraft stands on the A-apron,
   aircraft stands B16, B20, B24, B28, B32 and B36,
   except during low visibility phase C and D.
   REF EHAM AD 2.22 paragraph 3.4 "Phase C and D".

   Self-docking procedure at aircraft stands:
   Stop aircraft when yellow STOP marking is in line
   with pilot's eye view at an angle of 90 DEG to the lead-in line.

5. Marshaller guidance is required for aircraft docking at the:
   G-apron
   J-apron(except P10, P12, P14 and P16 in case of de-icing, see EHAM AD 2.20 par. 9)
   R-apron
   Y-apron

6. P-holding:
   Either P1 available or PA and PB available.
   Either P3 available or PC and PD available.

8. Standard push-back directions:
   R=right turn (as seen from the push-back vehicle).
   L =left turn (as seen from the push-back vehicle).
   +=push-pull
   *=push back limit line
   S=straight backwards
   T=taxi-in / taxi-out

Schiphol Ground	121.705
	121.805
	121.905

CAUTION

1. TWY S7W is designated for crossing RWY 06/24 only.
2. Compass deviations, caused by underground train, may occur when an aircraft is parked at the stands of the E-pier, in the area between the E- and F-pier, or when following the TWYs in the vicinity of the E-pier.
3. Avoid holding on the upslope between A19 and A20 to prevent backward movement of the aircraft.
4. J-apron is not controlled by ATC.
5. MAX wingspan 61 M only applicable to aircraft taxiing to E3.
6. Access to aircraft stands R71and R73 is limited to towed aircraft only.
7. Standard taxi routing, unless otherwise instructed by ATC:
   ACFT stands B15 - B35: TWY A4W.
   ACFT stands C6 - C14: TWY A4E.
   ACFT stand E24: aircraft with wingspan greater than 65 M: TWY A12.
   ACFT stands G3 - G9 and H1 - H7, aircraft with wingspan 36 M or less: from TWY A/B via TWY A19E (orange line).
   ACFT stands G71 - G79, aircraft with wingspan 36 M or less: from TWY A/B via TWY A19W (blue line).
   ACFT stands G3 - G9 and G73 - G79, aircraft with wingspan greater than 36 M: from TWY A/B via TWY A19C.
8. After vacating RWY 18C via TWY W9 or TWY W10, taxiing is only possible in non-standard taxi routing either to TWY A southbound or TWY B northbound.

LEGEND

- No-entry: runway entry prohibited at this point. TWY E1 from TWY A, A8 and B. TWY N8 from TWY A and B. TWY W6 from TWY A, B and D.
- (35) MAX WINGSPAN 35 M, LIMITED ENTRY
- CROSSING VEHICLES
- STOP BAR
- BLAST FENCE
- ATC SERVICE BDRY
- STANDARD TAXI ROUTING, UNLESS OTHERWISE INSTRUCTED BY ATC. ALL OTHER ROUTES MAY BE USED TWO-WAY AT ATC DISCRETION ONLY.
- (HN) INTERMEDIATE HOLDING POSITION

INS COORDINATES FOR AIRCRAFT STANDS

STAND	LAT	LONG
A31	521756.91N	0044511.30E
A32	521757.93N	0044510.26E
A33	521758.94N	0044509.22E
A34	521759.96N	0044508.17E
A35	521800.98N	0044507.13E
A41	to be surveyed	
A42	to be surveyed	
A43	to be surveyed	
A44	to be surveyed	
A45	to be surveyed	
A51	to be surveyed	
A52	to be surveyed	
A53	to be surveyed	
A54	to be surveyed	
A55	to be surveyed	
A61	to be surveyed	
A62	to be surveyed	
A63	to be surveyed	
A64	to be surveyed	
A65	to be surveyed	
A71	521802.78N	0044526.98E
A72	521803.80N	0044525.94E
A73	521804.82N	0044524.90E
A74	521805.84N	0044523.85E
A75	521806.86N	0044522.81E
B15	521818.22N	0044544.87E
B16	521815.20N	0044540.84E
B17	521817.02N	0044546.10E
B20	521814.24N	0044542.01E
B23	521815.81N	0044547.34E
B24	521813.26N	0044543.01E
B27	521814.60N	0044548.58E
B28	521812.28N	0044544.01E
B31	521813.39N	0044549.81E
B32	521811.30N	0044545.01E
B35	521812.18N	0044551.05E
B36	521810.27N	0044546.06E
B91	to be surveyed	
B92	to be surveyed	
B93	to be surveyed	
B94	to be surveyed	
B95	to be surveyed	

STAND	LAT	LONG
C5	521823.85N	0044556.56E
C6	521820.74N	0044553.88E
C7	521822.26N	0044558.13E
C8	521819.54N	0044554.97E
C9	521820.42N	0044600.67E
C10	521818.54N	0044556.56E
C11	521819.74N	0044604.57E
C12	521817.08N	0044557.00E
C13	521818.46N	0044604.54E
C14	521815.82N	0044558.18E
C15	521817.31N	0044603.37E
C16	521815.76N	0044600.19E
C18	521816.84N	0044601.41E
D2	to be surveyed	
D3	521832.60N	0044558.63E
D4	521829.50N	0044601.24E
D5	521832.62N	0044601.57E
D7	521832.34N	0044604.48E
D10	521828.19N	0044607.91E
D12	521826.96N	0044609.28E
D14	521825.94N	0044610.71E
D16	521826.12N	0044612.95E
D18	521826.05N	0044615.24E
D22	521825.96N	0044617.63E
D23	521828.16N	0044615.60E
D24	521825.89N	0044619.80E
D25	521828.08N	0044617.90E
D26	521825.81N	0044622.08E
D27	521828.00N	0044620.19E
D28	521825.73N	0044624.36E
D29	521827.92N	0044622.48E
D31	521827.84N	0044624.77E
D41	521833.29N	0044608.11E
D43	521834.71N	0044610.14E
D44	521832.86N	0044616.04E
D47	521835.74N	0044612.71E
D48	521832.78N	0044618.33E
D49	521835.66N	0044616.21E
D51	521835.77N	0044618.40E
D52	521832.70N	0044620.62E
D53	521835.64N	0044620.53E
D54	521832.62N	0044622.92E
D55	521835.48N	0044622.83E
D56	521832.53N	0044625.21E
D57	521835.34N	0044624.97E
D88	521841.81N	0044614.46E
D90	521841.74N	0044616.78E
D92	521841.66N	0044619.10E
D93	521841.58N	0044621.42E
D94	521841.55N	0044623.74E
D95	521841.49N	0044626.07E

STAND	LAT	LONG
E2	521837.40N	0044556.96E
E3	521839.10N	0044553.63E
E4	521838.98N	0044558.73E
E5	521840.91N	0044555.51E
E6	521840.64N	0044600.41E
E7	521843.07N	0044557.12E
E8	521842.25N	0044603.38E
E9	521845.21N	0044558.82E
E17	521847.18N	0044600.92E
E18	521844.32N	0044606.03E
E19	521849.28N	0044603.43E
E20	521846.73N	0044608.10E
E22	521848.76N	0044610.28E
E24	521851.01N	0044612.50E
E72	521843.14N	0044617.82E
E75	521843.01N	0044621.65E
E77	521842.88N	0044625.47E
F2	to be surveyed	
F3	521843.33N	0044540.19E
F4	521844.26N	0044545.73E
F5	521845.64N	0044539.22E
F6	521846.70N	0044546.49E
F7	521848.43N	0044538.92E
F8	521848.81N	0044545.86E
F9	521848.72N	0044542.36E
G2	521842.61N	0044528.47E
G3	521841.64N	0044522.63E
G4	521844.39N	0044528.43E
G5	521843.98N	0044522.42E
G6	521846.74N	0044528.22E
G7	521846.33N	0044522.20E
G8	521849.08N	0044528.00E
G9	521848.80N	0044521.95E
G71	521841.57N	0044513.04E
G73	521843.52N	0044510.88E
G76	521845.88N	0044510.55E
G79	521848.22N	0044509.95E
H1	521838.24N	0044520.71E
H2	521837.49N	0044518.76E
H3	521836.74N	0044516.82E
H4	521835.99N	0044514.88E
H5	521835.24N	0044512.94E
H6	521834.49N	0044510.99E
H7	521833.74N	0044509.05E
J80	521847.40N	0044451.48E
J81	521846.74N	0044451.65E
J82	521846.10N	0044451.36E
J83	521844.08N	0044451.41E
J84	521841.74N	0044451.19E
J85	521839.39N	0044450.97E
J86	521837.05N	0044450.76E
J87	521834.71N	0044450.54E

STAND	LAT	LONG
P10	521846.54N	0044438.46E
P11	521843.61N	0044438.20E
P12	521843.27N	0044438.16E
P13	521840.78N	0044438.28E
P14	521839.87N	0044437.85E
P15	521838.45N	0044437.72E
P16	521835.57N	0044437.45E
R71	521739.70N	0044429.11E
R72	521740.40N	0044429.56E
R73	521740.28N	0044430.61E
R74	521741.87N	0044433.36E
R77	521743.32N	0044437.18E
R80	521745.19N	0044442.00E
R81	521746.41N	0044445.37E
R82	521747.65N	0044448.62E
S64	to be surveyed	
S65	to be surveyed	
S66	to be surveyed	
S67	to be surveyed	
S68	to be surveyed	
S69	to be surveyed	
S72	521737.85N	0044533.84E
S74	521739.18N	0044537.30E
S77	521740.51N	0044540.77E
S79	521741.84N	0044544.23E
S82	521743.17N	0044547.70E
S84	521744.50N	0044551.17E
S87	521745.83N	0044554.63E
S90	521747.16N	0044558.10E
S92	521748.49N	0044601.57E
S94	521749.88N	0044604.98E
S96	521751.22N	0044608.43E
Y71	521817.23N	0044439.12E
Y73	521822.02N	0044439.90E

CHANGE: MAX WING SPAN added.odt/xml.

Bibliography

[1] H. Udluft. “Decentralization in Air Transportation”. Delft University of Technology, 2017. DOI: 10.4233/uuid:2af25aa3-be84-4d37-9332-fe2af319db1e.

[2] Rijksoverheid. *CO_2 uitstoot luchtvaart verminderen*. Accessed on: June 28th, 2022. [Online]. Available: https://www.rijksoverheid.nl/onderwerpen/luchtvaart/co2-uitstoot-luchtvaart.

[3] Airport Operators Association. *Aircraft on the Ground CO2 Reduction Programme*. Tech. rep. 2010.

[4] Royal Schiphol Group. *Sustaining Your World*. Accessed on: June 28th, 2022. [Online]. Available: https://www.schiphol.nl/en/schiphol-group/page/road-to-the-most-sustainable-airports/. 2022.

[5] Advanced Engine Off Navigation (AEON). *Engine-off taxiing techniques*. Accessed on: April 4th, 2022. [Online]. Available: https://www.aeon-project.eu/engine-off-taxiing-operations/.

[6] Royal Schiphol Group. *Final Results Taxibot Proof of Concept*. Tech. rep. 2021.

[7] Royal Schiphol Group. *Feasibility Study Sustainable Taxiing at Schiphol*. Tech. rep. 2021.

[8] J. Soomers. “Hierarchical Multi-Agent Path Planning For Delay Mitigation in Airport Engine-Off Towing System”. Delft University of Technology, May 2022.

[9] J.G. Kamphof. “Design and Analysis of Tug-Enabled Engine-Off Taxiing Operation Using Hierarchical Multi-Agent Planning”. Delft University of Technology, 2022.

[10] J.A.D. Atkin, E.K. Burke, and S. Ravizza. “The Airport Ground Movement Problem: Past and Current Research and Future Directions”. In: *Proceedings of the 4th International Conference on Research in Air Transportation (ICRAT)* (2010), pp. 131–138.

[11] Royal Schiphol Group. *Amsterdam Airport Schiphol Facts: What You Would Like To Know*. Accessed on: June 21th, 2022. [Online]. Available: https://www.schiphol.nl/nl/route-development/pagina/amsterdam-airport-schiphol-airport-facts/. 2022.

[12] Y. Jiang, Z. Liao, and H. Zhang. “A Collaborative Optimization Model for Ground Taxi Based on Aircraft Priority”. In: *Mathematical Problems in Engineering* 2013 (2013), pp. 1–9. DOI: 10.1155/2013/854364.

[13] EUROCONTROL Airport CDM Team. *Airport CDM Implementation Manual*. 5.0. EUROCONTROL. Mar. 2017.

[14] EUROCONTROL. *Specification for Advanced-Surface Movement Guidance and Control System (A-SMGCS) Services*. 2.0. Apr. 2020.

[15] Schiphol. *Schiphol Airport CDM Operations Manual*. 1.3. Sept. 2019.

[16] T. Noortman. “Agent-Based Modelling of an Airport’s Ground Surface Movement Operation: Understanding the Principles and Mechanisms of Decentralized Control”. Delft University of Technology, May 2018.

[17] K. Fines. “Agent-Based Distributed Planning and Coordination for Resilient Airport Surface Movement Operations”. Delft University of Technology, Nov. 2019.

[18] B. Benda. “Agent-Based Modelling and Analysis of Non-Autonomous Airport Ground Surface Operations”. Delft University of Technology, Nov. 2020.

[19] S. Polydorou. “A Learning-Based Approach for Distributed Planning and Coordination of Airport Surface Movement Operations”. Delft University of Technology, Sept. 2021.

[20] International Civil Aviation Organization (ICAO). *Annex 2: Rules of the Air*. 10.0. July 2005.

[21] EUROCONTROL. *Network Operations*. [Online]. Accessed on: March 28th, 2022. Available: https://www.eurocontrol.int/network-operations.

[22] V.A. Bijsterbosch and W.H. Dalmeijer. *Gebruiksprognose 2022*. Royal Schiphol Group. Oct. 2021.

[23] Redactie Luchtvaartnieuws. *Schiphol neemt nieuw avioduct over A4 in gebruik*. Accessed on: April 14th, 2022. [Online]. Available: https://www.luchtvaartnieuws.nl/nieuws/categorie/3/airports/schiphol-neemt-nieuw-avioduct-over-a4-in-gebruik. Dec. 2021.

[24] HSE Office. *Schipholregels*. Royal Schiphol Group. Apr. 2022.

[25] Royal Schiphol Group. *RASAS: Regulation Aircraft Stand Allocation Schiphol*. Tech. rep. 2020.

[26] Royal Schiphol Group. *Bedrijfshandboek Amsterdam Airport Schiphol: Deel 1 Aerodrome Manual (version 7.0)*. Tech. rep. 2020.

[27] Royal Schiphol Group. *Handboek Landingsterrein: Deel 1.2.5 Uitvoeren Pushbackbewwegingen (version 5.0)*. Tech. rep. 2020.

[28] International Civil Aviation Organization (ICAO). *Post-COVID-19 Forecasts Scenarios*. June 2021. Accessed on: March 30th, 2022. [Online]. Available: https://www.icao.int/sustainability/Pages/Post-Covid-Forecasts-Scenarios.aspx.

[29] International Air Transport Association (IATA). *Air Passenger Numbers to Recover in 2024*. Mar. 2022. Accessed on: March 30th, 2022. [Online]. Available: https://www.iata.org/en/pressroom/2022-releases/2022-03-01-01/.

[30] R. Morris et al. "Self-Driving Aircraft Towing Vehicles: A Preliminary Report". In: *AAAI Workshop: Artificial Intelligence for Transportation*. Ed. by A. Botea and S.A. Meijer. Vol. WS-15-05. AAAI Technical Report. AAAI Press, 2015, pp. 41–48.

[31] R. Morris et al. "Planning, Scheduling and Monitoring for Airport Surface Operations". In: *AAAI Workshop: Planning for Hybrid Systems*. Ed. by S. Thiébaux D. Magazzeni S. Sanner. Vol. WS-16-12. AAAI Technical Report. AAAI Press, 2016, pp. 608–614.

[32] I. Edem et al. "The Future of Conventional Aircraft Ground Propulsion Systems in Relation to Fuel Consumption and CO2 Emission". In: *International Journal of Thermal & Environmental Engineering* 13.2 (2016), pp. 91–100. DOI: 10.5383/ijtee.13.02.003.

[33] G. Sirigu et al. "A Fleet Management Algorithm for Automatic Taxi Operations". In: *International Conference on Research in Air Transportation (ICRAT)*. June 2016.

[34] Single European Sky ATM Research (SESAR). *About Single European Sky*. Accessed on: March 31st, 2022. [Online]. Available: https://www.sesarju.eu/background-ses.

[35] Federal Aviation Administration (FAA). *This is NextGen*. Accessed on: March 31st, 2022. [Online]. Available: https://www.faa.gov/nextgen/this_is_nextgen/. Jan. 2022.

[36] SESAR Joint Undertaking. *Automation in air traffic management: Long-term vision and initial research roadmap*. Tech. rep. 2020. DOI: 10.2829/968328.

[37] Royal Schiphol Group. *An autonomous airport in 2050*. Accessed on: April 1st, 2022. [Online]. Available: https://www.schiphol.nl/en/innovation/page/an-autonomous-airport-in-2050/.

[38] G. Sirigu. "Planning and Reconfigurable Control of a Fleet of Unmanned Vehicles for Taxi Operations in Airport Environment". Politecnico di Torino, 2017. DOI: 10.6092/polito/porto/2675463.

[39] S. Ahmadi. "Green Airport Operations: Conflict And Collision Free Taxiing Using Electric Powered Towing Alternatives". Concordia University Montreal, Dec. 2019.

[40] I. Azorín Gonzalez et al. "Optimization of Future Ground Operations for Aircraft". In: (2013).

[41] N. Okuniek and D. Beckmann. "Towards Higher Level of A-SMGCS: Handshake of Electric Taxi and Trajectory-Based Taxi Operations". In: *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. 2017, pp. 1–10. DOI: 10.1109/DASC.2017.8102047.

[42] *TaxiBot: Concept*. Accessed on: April 4th, 2022. [Online]. Available: https://www.taxibot-international.com/concept. 2013.

[43] M. Lukic et al. “State of the Art of Electric Taxiing Systems”. In: *International Conference on Electrical Systems for Aircraft, Railway, Ship Propulsion and Road Vehicles (ESARS) and International Transportation Electrification Conference (ITEC)*. Nov. 2018. DOI: 10.1109/ESARS-ITEC.2018.8607786.

[44] M. Lukic et al. “Review, Challenges, and Future Developments of Electric Taxiing Systems”. In: *IEEE Transactions on Transportation Electrification* 5.4 (2019), pp. 1441–1457. DOI: 10.1109/TTE.2019.2956862.

[45] Jakub Hospodka. “Cost-benefit Analysis of Electric Taxi Systems for Aircraft”. In: *Journal of Air Transport Management* 39 (2014), pp. 81–88. ISSN: 0969-6997. DOI: 10.1016/j.jairtraman.2014.05.002.

[46] International Air Transport Association (IATA). *WheelTug: Preparing Airports for E-Taxi*. Accessed on: April 5th, 2022. [Online]. Available: https://www.iata.org/en/about/sp/newsletter/wheeltug-preparing-airports-for-e-taxi/. Mar. 2021.

[47] E. Van Baaren and P.C. Roling. “Design of a Zero Emission Aircraft Towing System”. In: *AIAA Aviation 2019 Forum*. DOI: 10.2514/6.2019-2932.

[48] *Narrow-Body TaxiBot*. Accessed on: April 4th, 2022. [Online]. Available: https://www.taxibot-international.com/narrow-body-taxibot. 2013.

[49] *Wide-Body TaxiBot*. Accessed on: April 4th, 2022. [Online]. Available: https://www.taxibot-international.com/wide-body-taxibot. 2013.

[50] J. Hospodka. “Electric Taxiing - Taxibot System”. In: *MAD 10* 02.10 (2014), pp. 17–20. DOI: 10.14311/MAD.2014.10.03.

[51] To70 Aviation. *Sustainable Taxi at Schiphol Airport - FTS Study*. Tech. rep. 2020.

[52] Z. Chua et al. “Initial Assessment Of The Impact Of Modern Taxiing Techniques on Airport Ground Control”. In: *HCI-Aero 2016, International Conference on Human-Computer Interaction in Aerospace*. HCI-Aero ’16 Proceedings of the International Conference on Human-Computer Interaction in Aerospace. Paris, France: ACM, Sept. 2016.

[53] I.F.A. Vis. “Survey of research in the design and control of automated guided vehicle systems”. In: *European Journal of Operational Research* 170.3 (2006), pp. 677–709. DOI: 10.1016/j.ejor.2004.09.020.

[54] M. Soltani et al. “An eco-friendly aircraft taxiing approach with collision and conflict avoidance”. In: *Transportation Research Part C: Emerging Technologies* 121 (2020). DOI: 10.1016/j.trc.2020.102872.

[55] B.J.L. Tindemans. “A Greedy Approach To The Minimisation Of Deviations Of The Dynamic Vehicle Routing Problem With Electric Taxiing Systems”. Delft University of Technology, Oct. 2021.

[56] J. Kroese. “Electric Taxiing: An Optimisation Study on the Feature of Airport Operations”. Delft University of Technology, Jan. 2021.

[57] A.L. Salihu, S. M. Lloyd, and A. Akgunduz. “Electrification of airport taxiway operations: A simulation framework for analyzing congestion and cost”. In: *Transportation Research Part D: Transport and Environment* 97 (2021). DOI: 10.1016/j.trd.2021.102962.

[58] L. Mantecchini, M.N. Postorino, and E. Gualandi. “Integration Between Aircraft And Handling Vehicles During Taxiing Procedures To Improve Airport Sustainability”. In: *International Journal of Transport Development and Integration* 1 (May 2016), pp. 28–42. DOI: 10.2495/TDI-V1-N1-28-42.

[59] L. Khammash, L. Mantecchini, and V. Reis. “Micro-Simulation Of Airport Taxiing Procedures To Improve Operation Sustainability: Application of Semi-Robotic Towing Tractor”. In: *2017 5th IEEE International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. 2017, pp. 616–621. DOI: 10.1109/MTITS.2017.8005587.

[60] E.V.M. Van Baaren. “The Feasibility Of A Fully Electric Aircraft Towing System”. Delft University of Technology, May 2019.

[61] N.J.F.P. Guillame. “Finding The Viability Of Using An Automated Guided Vehicle Taxiing System For Aircraft”. Delft University of Technology, Apr. 2018.

[62] J.W. Smeltink et al. “An Optimisation Model for Airport Taxi Scheduling”. In: *Informs Journal on Computing - INFORMS* (Jan. 2004).

[63] Ferway & MovingDot. *RECAT-EU For Departures At Schiphol*. 1.0. Knowledge and Development Center (KDC). Nov. 2020.

[64] A. Cook and G. Tanner. *European Airline Delay Cost Reference Values*. 4.1. University of Westminster. Dec. 2015.

[65] P. Vaishnav. “Costs and Benefits of Reducing Fuel Burn and Emissions from Taxiing Aircraft: Low-Hanging Fruit?” In: *Transportation Research Record* 2400.1 (2014), pp. 65–77. DOI: 10.3141/2400-08.

[66] A. Sharpanskykh. *Lecture 1: Introduction to Agents and Multi-Agent Systems. Emergence.* [PowerPoint Slides]. Delft University of Technology. 2020.

[67] A. Sharpanskykh. *Lecture 2: Specification of Models of Multi-Agent Systems. Languages for Multi-Agent Systems.* [PowerPoint Slides]. Delft University of Technology. 2020.

[68] A. Sharpanskykh. *Lecture 6: Multi-Agent Planning and Scheduling.* [PowerPoint Slides]. Delft University of Technology. 2020.

[69] B. Chen and H.H. Cheng. “A Review of the Applications of Agent Technology in Traffic and Transportation Systems”. In: *IEEE Transactions on Intelligent Transportation Systems* 11.2 (2010), pp. 485–497. DOI: 10.1109/TITS.2010.2048313.

[70] J.L. Adler and V.J. Blue. “A Cooperative Multi-Agent Transportation Management and Route Guidance System”. In: *Transportation Research Part C: Emerging Technologies* 10.5 (2002), pp. 433–454. DOI: 10.1016/S0968-090X(02)00030-X.

[71] P. Baran. “On Distributed Communications Networks”. In: *IEEE Transactions on Communications Systems* 12.1 (1964), pp. 1–9. DOI: 10.1109/TCOM.1964.1088883.

[72] S.T. Chen. “Multi-Agent Planning and Coordination for Automated Aircraft Ground Handling: Literature Study”. Delft University of Technology, 2022.

[73] B.P. Gerkey and M.G. Mataric. “A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems”. In: *The International Journal of Robotics Research* 23 (09 Sept. 2004), pp. 939–954. DOI: 10.1177/0278364904045564.

[74] G. Ayorkor Korsah, A. Stentz, and M. Bernardine Dias. “A Comprehensive Taxonomy for Multi-Robot Task Allocation”. In: *The International Journal of Robotics Research* 32 (12 Oct. 2013), pp. 1495–1512. DOI: 10.1177/0278364913496484.

[75] F.S. Hillier and G.J. Lieberman. *Introduction to Operations Research*. 10th ed. McGraw-Hill Education, 2015.

[76] A. Khamis ad A. Hussein and A. Elmogy. “Multi-robot Task Allocation: A Review of the State-of-the-Art”. In: *Cooperative Robots and Sensor Networks 2015*. Ed. by A. Koubâa and J.R. Martínez-de Dios. Springer International Publishing, 2015, pp. 31–51. DOI: 10.1007/978-3-319-18299-5_2.

[77] J.G. Martin et al. “Multi-Robot Task Allocation Problem with Multiple Nonlinear Criteria Using Branch and Bound and Genetic Algorithms”. In: *Intelligent Service Robotics* 14 (05 Nov. 2021), pp. 707–727. DOI: 10.1007/s11370-021-00393-4.

[78] M.H. Lin, T.F. Tsai, and C.S. Yu. “A Review of Deterministic Optimization Methods in Engineering and Management”. In: *Mathematical Problems in Engineering* 2012 (June 2012). DOI: 10.1155/2012/756023.

[79] K. Garapati et al. “A Game of Drones: Game Theoretic Approaches for Multi-robot Task Allocation in Security Missions”. In: *ROBOT 2017: Third Iberian Robotics Conference*. Ed. by A. Ollero et al. Springer International Publishing, 2018, pp. 855–866.

[80] H.W. Kuhn. “The Hungarian Method for the Assignment Problem”. In: *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*. Ed. by M. Jünger et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 29–47. DOI: 10.1007/978-3-540-68279-0_2.

[81] P. Rizzo. “Auction-Based Task Allocation for Online Pickup and Delivery Problems with Time Constraints and Uncertainty”. Delft University of Technology, Feb. 2021.

[82] G.A. Mills-Tettey, A. Stent, and M.B. Dias. “The Dynamic Hungarian Algorithm for the Assignment Problem with Changing Costs”. In: (July 2007).

[83] M. Badreldin, A. Hussein, and A. Khamis. “A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation”. In: *Advances in Artificial Intelligence* 2013 (Nov. 2013). DOI: 10.1155/2013/256524.

[84] N. Atay and B. Bayazit. *Mixed-Integer Linear Programming Solution to Multi-Robot Task Allocation Problem*. Tech. rep. Washington University in St. Louis, 2006. DOI: 10.7936/K7R49P04.

[85] M. Darrah, W. Niland, and B. Stolarik. “Multiple UAV Dynamic Task Allocation Using Mixed Integer Linear Programming in a SEAD Mission”. In: *American Institute of Aeronautics and Astronautics* (Sept. 2005). DOI: 10.2514/6.2005-7164.

[86] S.S. Sonda. “Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams”. Massachusetts Institute of Technology, 2012.

[87] B.B. Werger and M.J. Matarić. “Broadcast of Local Eligibility for Multi-Target Observation”. In: *Distributed Autonomous Robotic Systems 4*. Ed. by L.E. Parker, G. Bekey, and J. Barhen. Tokyo: Springer Japan, 2000, pp. 347–356. DOI: 10.1007/978-4-431-67919-6_33.

[88] H. Ma et al. “Lifelong Multi-Agent Path Finding for Online Pickup and Delivery Tasks”. In: *Autonomous Agents and Multi-Agent Systems* (May 2017), pp. 837–845.

[89] D. Juedes et al. “Heuristic Resource Allocation Algorithms for Maximizing Allowable Workload in Dynamic, Distributed Real-Time Systems”. In: *Proceedings of the 18th International Conference on Parallel and Distributed Processing Symposium*. May 2004. DOI: 10.1109/IPDPS.2004.1303072.

[90] K. Zhang, E.G. Collins, and D. Shi. “Centralized and Distributed Task Allocation in Multi-Robot Teams via a Stochastic Clustering Auction”. In: *ACM Transactions on Autonomous and Adaptive Systems* 7.2 (July 2012). DOI: 10.1145/2240166.2240171.

[91] A. Hussein and A. Khamis. “Market-Based Approach to Multi-Robot Task Allocation”. In: *2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR)*. Dec. 2013, pp. 69–74. DOI: 10.1109/ICBR.2013.6729278.

[92] Y. Zhu and L. Wu. “Structure Study of Multiple Traveling Salesman Problem using Genetic Algorithm”. In: *2019 34rd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. 2019, pp. 323–328. DOI: 10.1109/YAC.2019.8787633.

[93] V. Arya, A. Goyal, and V. Jaiswal. “An Optimal Solution to Multiple Travelling Salesperson Problem Using Modified Genetic Algorithm”. In: *International Journal of Application or Innovation in Engineering and Management (IJAIEM)* 3.1 (Jan. 2014), pp. 425–430.

[94] S.M. Sampaio, A. Dantas, and C.G. Camilo-Junior. “Routing Sales Territory by Solving a TSP Variant with Genetic Algorithm in a Multi-constraint and Complicated Real-world Application”. In: *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. 2019, pp. 1692–1699. DOI: 10.1109/SSCI44817.2019.9002685.

[95] M.U. Arif and S. Haider. “An Evolutionary Traveling Salesman Approach for Multi-Robot Task Allocation”. In: *9th International Conference on Agents and Artifical Intelligence*. Jan. 2017, pp. 567–574. DOI: 10.5220/0006197305670574.

[96] R.J. Alitappeh and K. Jeddisaravi. “Multi-Robot Exploration in Task Allocation Problem”. In: *Applied Intelligence* 52.2 (Jan. 2022), pp. 2189–2211. DOI: 10.1007/s10489-021-02483-3.

[97] M. Shelkamy et al. “Comparative Analysis of Various Optimization Techniques for Solving Multi-Robot Task Allocation Problem”. In: *2020 2nd Novel Intelligent and Leading Emerging Sciences Conference (NILES)*. 2020, pp. 538–543. DOI: 10.1109/NILES50944.2020.9257967.

[98] Q.M. Ha et al. “A Hybrid Genetic Algorithm for the Traveling Salesman Problem with Drone”. In: *Journal of Heuristics* 26.2 (Nov. 2019), pp. 219–247. DOI: 10.1007/s10732-019-09431-y.

[99] S. Mirjalili. “Genetic Algorithm”. In: *Evolutionary Algorithms and Neural Networks: Theory and Applications*. Springer International Publishing, 2019, pp. 43–55. DOI: 10.1007/978-3-319-93025-1_4.

[100] E. Edison and T. Shima. “Integrated Task Assignment and Path Optimization for Cooperating Uninhabited Aerial Vehicles Using Genetic Algorithms”. In: *Computers & Operations Research* 38.1 (2011), pp. 340–356. DOI: 10.1016/j.cor.2010.06.001.

[101] R. Patel et al. “Decentralized Task Allocation in Multi-Agent Systems Using a Decentralized Genetic Algorithm”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020, pp. 3770–3776. DOI: 10.1109/ICRA40945.2020.9197314.

[102] J. Du et al. “Task Allocation in Multi-Agent Systems with Swarm Intelligence of Social Insects”. In: *2010 Sixth International Conference on Natural Computation*. Vol. 8. 2010, pp. 4322–4326. DOI: 10.1109/ICNC.2010.5583611.

[103] H. Zhao and L. Cheng. “Ant Colony Algorithm and Simulation for Robust Airport Gate Assignment”. In: *Mathematical Problems in Engineering* 2014 (Sept. 2014). DOI: 10.1155/2014/804310.

[104] G. Oh et al. “PSO-based Optimal Task Allocation for Cooperative Timing Missions”. In: vol. 49. 17. 2016, pp. 314–319. DOI: 10.1016/j.ifacol.2016.09.054.

[105] G. Sirigu et al. “Hybrid Particle Swarm Optimization with Parameter Fixing: Application to Automatic Taxi Management”. In: *Journal of Air Transportation* 28.2 (2020), pp. 36–48. DOI: 10.2514/1.D0172.

[106] Y. Chen, D. Yang, and J. Yu. “Multi-UAV Task Assignment With Parameter and Time-Sensitive Uncertainties Using Modified Two-Part Wolf Pack Search Algorithm”. In: *IEEE Transactions on Aerospace and Electronic Systems* 54.6 (2018), pp. 2853–2872. DOI: 10.1109/TAES.2018.2831138.

[107] H. Kurdi et al. “Adaptive Task Allocation for Multi-UAV Systems Based on Bacteria Foraging Behaviour”. In: *Applied Soft Computing Journal* 83 (2019). DOI: 10.1016/j.asoc.2019.105643.

[108] S. Saravanan et al. “Review on State-Of-The-Art Dynamic Task Allocation Strategies for Multiple-Robot Systems”. In: *Industrial Robot* 47.06 (Sept. 2020), pp. 929–942. DOI: 10.1108/IR-04-2020-0073.

[109] B.P. Gerkey and M.J. Mataric. “Sold!: Auction Methods for Multirobot Coordination”. In: *IEEE Transactions on Robotics and Automation* 18.5 (Oct. 2002), pp. 758–768. DOI: 10.1109/TRA.2002.803462.

[110] A.P. Enríquez Gómez. *Comparison of Multi-Robot Task Allocation Algorithms*. Tech. rep. Hochschule Bonn-Rhein-Sieg: University of Applied Sciences, 2019.

[111] A. Hussein and A. Khamis. “Market-based approach to Multi-robot Task Allocation”. In: *2013 International Conference on Individual and Collective Behaviors in Robotics (ICBR)*. 2013, pp. 69–74. DOI: 10.1109/ICBR.2013.6729278.

[112] E. Schneider, E.I. Sklar, and S. Parsons. “Mechanism Selection for Multi-Robot Task Allocation”. In: *Towards Autonomous Robotic Systems*. Ed. by Y. Gao et al. Springer International Publishing, 2017, pp. 421–435. DOI: 10.1007/978-3-319-64107-2_33.

[113] M.B. Dias et al. “Market-Based Multirobot Coordination: A Survey and Analysis”. In: *Proceedings of the IEEE* 94.7 (2006), pp. 1257–1270. DOI: 10.1109/JPROC.2006.876939.

[114] S. Koenig, P. Keskinocak, and C. Tovey. “Progress on Agent Coordination with Cooperative Auctions”. In: 2010, pp. 1713–1717.

[115] S. Koenig et al. “The Power of Sequential Single-Item Auctions for Agent Coordination”. In: July 2006.

[116] G.P. Das et al. “A Distributed Task Allocation Algorithm for a Multi-Robot System in Healthcare Facilities”. In: *Journal of Intelligent & Robotic Systems* 80.1 (Oct. 2015), pp. 33–58. DOI: 10.1007/s10846-014-0154-2.

[117] E. Nunes and M. Gini. “Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints”. In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI’15. Austin, Texas: AAAI Press, 2015, pp. 2110–2216. DOI: 10.5555/2886521.2886614.

[118] B. Heap and M. Pagnucco. “Repeated Sequential Single-Cluster Auctions with Dynamic Tasks for Multi-Robot Task Allocation with Pickup and Delivery”. In: *Multiagent System Technologies*. Ed. by M. Klusch, M. Thimm, and M. Paprzycki. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 87–100. DOI: 10.1007/978-3-642-40776-5_10.

[119] H.L. Choi, L. Brunet, and J.P. How. “Consensus-Based Decentralized Auctions for Robust Task Allocation”. In: *IEEE Transactions on Robotics* 25.4 (2009), pp. 912–926. DOI: 10.1109/TRO.2009.2022423.

[120] N. Buckman, H.L. Choi, and J.P. How. “Partial Replanning for Decentralized Dynamic Task Allocation”. In: *Proceedings of the AIAA Scitech 2019 Forum Conference*. Oct. 2018. DOI: 10.2514/6.2019-0915.

[121] G. Arslan, J. Marden, and J. Shamma. “Autonomous Vehicle-Target Assignment: A Game-Theoretical Formulation”. In: *Journal of Dynamic Systems, Measurement and Control* 129 (Sept. 2007), pp. 584–596. DOI: 10.1115/1.2766722.

[122] R. Cui, J. Guo, and B. Gao. “Game Theory-Based Negotiation for Multiple Robots Task Allocation”. In: *Robotica* 31.6 (2013), pp. 923–934. DOI: 10.1017/S0263574713000192.

[123] M. Smyrnakis, H. Qu, and S. Veres. “Improving Multi-robot Coordination by Game-Theoretic Learning Algorithms”. In: *2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI)*. 2017, pp. 417–424. DOI: 10.1109/ICTAI.2017.00071.

[124] H. Wu and H. Shang. “Potential Game for Dynamic Task Allocation in Multi-Agent System”. In: *ISA Transactions* 102 (2020), pp. 208–220. DOI: 10.1016/j.isatra.2020.03.004.

[125] S. Park, Y.D. Zhong, and N.E. Leonard. “Multi-Robot Task Allocation Games in Dynamically Changing Environments”. In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 8678–8684. DOI: 10.1109/ICRA48506.2021.9561809.

[126] X. Wu et al. “Multi-Agent Pickup and Delivery with Task Deadlines”. In: *Proceedings of the 14th International Symposium on Combinatorial Search (SoCS 2021)*. Vol. 12. 01. Association for the Advancement of Artificial Intelligence (AAAI), 2021, pp. 206–208.

[127] M. Lagoudakis et al. “Auction-Based Multi-Robot Routing”. In: June 2005, pp. 343–350. DOI: 10.15607/RSS.2005.I.045.

[128] X. Zheng, S. Koenig, and C. Tovey. “Improving Sequential Single-Item Auctions”. In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Nov. 2006, pp. 2238–2244. DOI: 10.1109/IROS.2006.282567.

[129] X. Zheng and S. Koenig. “K-Swaps: Cooperative Negotiation for Solving Task-Allocation Problems”. In: *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence*. Ed. by C. Boutilier. July 2009, pp. 373–379. DOI: 10.5555/1661445.1661505.

[130] S. Koenig et al. “Sequential Bundle-Bid Single-Sale Auction Algorithms for Decentralized Control”. In: *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence*. Ed. by Manuela M. Veloso. 2007, pp. 1359–1365.

[131] G. Sharon et al. “Conflict-Based Search for Optimal Multi-Agent Path Finding”. In: *Artificial Intelligence* 219 (2015), pp. 40–66. DOI: 10.1016/j.artint.2014.11.006.

[132] E.W. Dijkstra. “A Note on Two Problems in Connexion With Graphs”. In: *Numerische Mathematike* 01.01 (1959), pp. 269–271. DOI: 10.1007/BF01386390.

[133] P.E. Hart, N.J. Nilsson, and B. Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. DOI: 10.1109/TSSC.1968.300136.

[134] R. Stern et al. “Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks”. In: *Proceedings of the 12th International Symposium on Combinatorial Search*. Vol. 10. 01. Association for the Advancement of Artificial Intelligence (AAAI), 2019, pp. 151–158. DOI: 10.48550/ARXIV.1906.08291.

[135] H. Ma. “Target Assignment and Path Planning for Navigation Tasks with Teams of Agents”. Department of Computer Science, University of Southern California, Aug. 2020.

[136] D. Atzmon et al. “Robust Multi-Agent Path Finding”. In: *Journal of Artificial Intelligence Research* 67 (2020), pp. 549–579.

[137] A. Andreychuk et al. “Multi-Agent Pathfinding with Continuous Time”. In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI-19)*. International Joint Conferences on Artificial Intelligence Organization, July 2019, pp. 39–45. DOI: 10.24963/ijcai.2019/6.

[138] W. Duchateau. “Multi-Agent Pickup and Delivery in a Distributed Baggage Handling System”. Delft University of Technology, Mar. 2021.

[139] M. Barer et al. “Suboptimal Variants of the Conflict-Based Search Algorithm for the Multi-Agent Path Finding Problem”. In: *Proceedings of the 7th Annual Symposium on Combinatorial Search (SoCS 2014)*. Vol. 5. 01. Association for the Advancement of Artificial Intelligence (AAAI), 2014, pp. 19–27.

[140] H. Ma et al. “Searching with Consistent Prioritization for Multi-Agent Path Finding”. In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*. AAAI’19. AAAI Press, 2019, pp. 7643–7650. DOI: 10.1609/aaai.v33i01.33017643.

[141] H. Ma et al. *Overview: Generalizations of Multi-Agent Path Finding to Real-World Scenarios*. 2017. DOI: 10.48550/ARXIV.1702.05515.

[142] W. Hoenig et al. “Multi-Agent Path Finding with Kinematic Constraints”. In: *Proceedings of 26th International Conference on Automated Planning and Scheduling (ICAPS 2016)*. 2016, pp. 477–485.

[143] J. Li et al. “Multi-Agent Path Finding for Large Agents”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 33.01 (July 2019), pp. 7627–7634. DOI: 10.1609/aaai.v33i01.33017627.

[144] L. Cohen et al. “Optimal and Bounded-Suboptimal Multi-Agent Motion Planning”. In: *Proceedings of the 12th Annual Symposium on Combinatorial Search (SoCS 2019)*. Vol. 10. 01. Association for the Advancement of Artificial Intelligence (AAAI), 2019, pp. 44–51.

[145] M. Phillips and M. Likhachev. “SIPP: Safe Interval Path Planning for Dynamic Environments”. In: *2011 IEEE International Conference on Robotics and Automation*. 2011, pp. 5628–5635. DOI: 10.1109/ICRA.2011.5980306.

[146] H. Ma et al. “Lifelong Path Planning with Kinematic Constraints for Multi-Agent Pickup and Delivery”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. July 2019, pp. 7651–7658. DOI: 10.1609/aaai.v33i01.33017651.

[147] Moving AI Lab. *Pathfinding Benchmarks*. [Online]. Accessed on: June 22th, 2022. Available: https://www.movingai.com/benchmarks/.

[148] K. Yakovlev et al. “On the Application of Safe-Interval Path Planning to a Variant of the Pickup and Delivery Problem”. In: *Proceedings of the 17th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2020)*. 2020, pp. 521–528. DOI: 10.5220/0009888905210528.

[149] M. Liu et al. “Task and Path Planning for Multi-Agent Pickup and Delivery”. In: *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*. Ed. by N. Agmon et al. May 2019.

[150] E. Osaba et al. “A Parallel Meta-heuristic for Solving a Multiple Asymmetric Traveling Salesman Problem with Simultaneous Pickup and Delivery Modeling Demand Responsive Transport Problems”. In: *Hybrid Artificial Intelligent Systems*. Ed. by E. Onieva et al. Springer International Publishing, 2015, pp. 557–567.

[151] S. Yoon and J. Kim. “Efficient Multi-Agent Task Allocation for Collaborative Route Planning with Multiple Unmanned Vehicles”. In: *IFAC-PapersOnLine* 50.1 (2017), pp. 3580–3585. DOI: 10.1016/j.ifacol.2017.08.686.

[152] J.P. Van den Berg and M.H. Overmars. “Prioritized Motion Planning for Multiple Robots”. In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 430–435. DOI: 10.1109/IROS.2005.1545306.

[153] D. Silver. “Cooperative Pathfinding”. In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment* 1.1 (June 2005), pp. 117–122.

[154] A. Contini and A. Farinelli. “Coordination Approaches for Multi-Item Pickup and Delivery in Logistic Scenarios”. In: *Robotics and Autonomous Systems* 146 (2021). DOI: 10.1016/j.robot.2021.103871.

[155] W. Hönig et al. “Conflict-Based Search with Optimal Task Assignment”. In: *Proceedings of the 17th International Conference for Autonomous Agents and Multiagent Systems (AAMAS 2018)*. Ed. by M. Dastani et al. International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 757–765. DOI: 10.5555/3237383.3237495.

[156] Z. Chen et al. “Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery”. In: *IEEE Robotics and Automation Letters* 6.03 (2021), pp. 5816–5823. DOI: 10.1109/LRA.2021.3074883.

[157] QGIS Development Team. *QGIS Geographic Information System*. QGIS Association. 2023. URL: https://www.qgis.org.

[158] LVNL. *Integrated Aeronautical Information Package (eAIP) for EHAM*. [PDF charts]. Accessed on: March 5th, 2023. [Online]. Available: https://eaip.lvnl.nl/web/2023-04-06-AIRAC/html/index-en-GB.html. 2023.

[159] J. Han, M. Kamber, and J. Pei. “2 - Getting to Know Your Data”. In: *Data Mining (Third Edition)*. Ed. by J. Han, M. Kamber, and J. Pei. Third Edition. The Morgan Kaufmann Series in Data Management Systems. Boston: Morgan Kaufmann, 2012, pp. 39–82. DOI: 10.1016/B978-0-12-381479-1.00002-2.

[160] Mark S. Manasse, Lyle A. McGeoch, and Daniel Dominic Sleator. “Competitive algorithms for on-line problems”. In: *Symposium on the Theory of Computing*. 1988.

[161] F. Rooseleer and V. Treve. *RECAT-EU: European Wake Turbulence Categorisation and Separation Minima on Approach and Departure*. 1.2. Eurocontrol. Feb. 2018.

[162] S. S. Shapiro and M. B. Wilk. “An Analysis of Variance Test for Normality (Complete Samples)”. In: *Biometrika* 52.3/4 (1965), pp. 591–611.

[163] R.F. Woolson. “Wilcoxon Signed-Rank Test”. In: *Wiley Encyclopedia of Clinical Trials*. John Wiley & Sons, Ltd, 2008, pp. 1–3. DOI: 10.1002/9780471462422.eoct979.

[164] “Mann–Whitney Test”. In: *The Concise Encyclopedia of Statistics*. Springer New York, 2008, pp. 327–329. DOI: 10.1007/978-0-387-32833-1_243.

[165] J.M. Bland and D.G. Altman. “Multiple significance tests: the Bonferroni method”. In: *BMJ* 310.6973 (1995), p. 170. DOI: 10.1136/bmj.310.6973.170.

[166] A. Vargha and H. D. Delaney. “A Critique and Improvement of the CL Common Language Effect Size Statistics of McGraw and Wong”. In: *Journal of Educational and Behavioral Statistics* 25.2 (2000), pp. 101–132. DOI: 10.3102/10769986025002101.