Transportation Research Part C 186 (2026) 105583

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Towards next-generation airport surface movement operations using hierarchical-distributed multi-agent planning

Malte von der Burg *, Jorick Kamphof, Joost Soomers, Alexei Sharpanskykh

Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, 2629 HS, Delft, the Netherlands

ARTICLE INFO

Keywords:
Multi-agent system
Multi-agent motion planning
Airport surface movement operations
Concept of operations
Hierarchical-distributed control
Engine-off taxiing

ABSTRACT

Coordinating the movements of aircraft along the surface of busy airports is a complex task, involving both humans and machines. In response to the interrelated challenges of ever-increasing demand, emission reduction, and sustaining safety levels, technological advances are emerging which, in turn, necessitate novel concepts of operations (ConOps). To this end, diverse operational concepts with varying technological aspects, level of automation, and degree of control centralisation could potentially offer a solution. However, before such innovative ConOps are matured to be deployable at real-world airports, extensive evaluation based on computational modelling is necessary. To enable such detailed modelling and analysis of the large variety of concepts for next-generation airport surface movement operations, this paper proposes a generalised architecture based on the hierarchical-distributed multi-agent system modelling paradigm. We illustrate the generalised architecture by providing a specific model instance for a ConOps based on centralised planning and distributed, fully-automated control. As essential part of this model, we introduce the novel Multi-Agent Motion Planning on Airport Surfaces (AS-MAMP) algorithm to represent the decision-logic for coordinating all ground movements. The two-level solver builds on prominent prioritised planners such as Priority-Based Search (PBS) and its variant Greedy PBS. As current low-level solvers were insufficient to plan realistic 4D trajectories for ground movements, we developed the new Safe Interval Motion Planning (SIMP) algorithm. By defining activity sequences per agent, SIMP plans trajectories across the operational processes during taxiing such as pushback, tug coupling/decoupling, and engine-start. The motions of aircraft and ground vehicles are based on finite acceleration, and avoid dynamic obstacles in continuous space and time, necessitating to define states with *feasible motion intervals*. We benchmark AS-MAMP by varying its high-level prioritisation scheme, and exchanging SIMP with two existing low-level solvers, namely the Safe Interval Path Planning (SIPP) and A* algorithms. Based on scenarios on both a synthetic and a real-world airport layout, we demonstrate the efficacy of AS-MAMP to plan safe and efficient 4D trajectories. Moreover, we show that the model is able to handle runway throughput levels that match or exceed those of large European airports.

1. Introduction

The air transport sector faces three connected challenges. First, the demand is predicted to increase to around 10 billion annual passengers by 2050 (IATA, 2021), more than doubling the amount of 2019. Second, the sector strives to achieve net-zero emissions

* Corresponding author.
E-mail addresses: m.f.vonderBurg@tudelft.nl (M. von der Burg), o.a.sharpanskykh@tudelft.nl (A. Sharpanskykh).

https://doi.org/10.1016/j.trc.2026.105583
Received 8 January 2025; Received in revised form 22 November 2025; Accepted 13 February 2026
Available online 5 March 2026

by the same time (IATA, 2021) to meet the Paris Agreement (Paris Agreement, 2015). As third challenge, the level of safety must be sustained or further increased. Zooming in on airports as key points in the air transport network, recent incidents during taxiing and at runways (SKYbrary, 2024b,a) underscore the importance of safety of the operations.

The safe and efficient planning and control of all ground movements at an airport is referred to as airport surface movement operations (ASM Ops). Especially at large airports, it is a challenging task, with complex interactions between diverse stakeholders as well as machines (Morris et al., 2016). Air Traffic Control Officers (ATCOs) provide clearances and guidance for aircraft and vehicles across the taxiway and runway network. Their decision-making is informed by other stakeholders, such as airport operators, airlines, and ground handlers. While pilots and vehicle drivers are responsible for the safe operation of their vehicles and determine their taxi speed accordingly, they rely on the overall coordination by and communication through ATCOs.

For ASM Ops, the three challenges of meeting the predicted demand, reducing emissions, and maintaining safety are highly interrelated. To address the demand, either larger or more aircraft are needed, and airports must facilitate this growth with additional capacity. However, infrastructural expansions of airports are expected to be insufficient to achieve higher airport capacity (EUROCONTROL, 2018). Thus, the congestion on the airport surface will increase. In turn, this will amplify the workload of ATCOs (Chua et al., 2017) which increases the risk of human errors and may induce additional delays and congestion. Furthermore, the taxiing times of aircraft, i.e. the duration of moving over the airport surface, become harder to predict. In addition to having a direct influence on the respective flights, this may trigger knock-on effects on the entire network, resulting in even higher inefficiencies (EUROCONTROL, 2021).

Regarding emissions, the taxi times can reach a significant share of the total flight time especially for short-haul flights, with combined taxi-in and taxi-out times reaching up to 30 min for European flights (EUROCONTROL, 2020). Furthermore, an analysis using a fuel-burn model showed that between 2 % (long-haul flight, 9630 km) and 17 % (short-haul flight, 350 km) of total fuel is burnt during taxiing (Rowland, 2023). The corresponding emissions can be effectively reduced when the aircraft do not have to use their engines for taxiing. In recent years, various new engine-off techniques have been explored and potentially yield significant decreases in carbon and noxious emissions (Lukic et al., 2019; Zoutendijk et al., 2023). On the other hand, solutions such as the TaxiBot system (TaxiBot, 2023), which can tow aircraft to runways and back, add additional vehicles to the traffic, again amplifying the issues of airport capacity, ATCO workload, and increased probability for human errors. Likewise, alternative propulsion technologies in development such as hydrogen or battery-powered aircraft require new concepts of operations (ConOps) that need to be thoroughly evaluated (ACAAF-TF, 2025).

Besides introducing new technology for taxiing, another possible direction to address the current challenges is to increase the level of automation. Artificial intelligence systems could provide extensive decision-support for ATCOs, pilots, and vehicle drivers, and could even take over certain repetitive tasks. In previous research, both NASA and DLR developed future ConOps for airports in the US and Europe based on surface trajectory-based operations (STBO), providing four-dimensional trajectories (4DTs) to coordinate taxi routes (Hooey et al., 2024; Okuniek et al., 2016, 2018). Other research assessed the achievable positioning tolerances of pilots and auto-pilot systems when following the 4DTs, and revealed that trajectory errors below 22 m are feasible to achieve, with mean errors even below 7 m for (partially) automated aircraft taxiing (Bernatzky, 2019). However, high levels of automation are deemed to be reachable only in the far-term (Hooey et al., 2024). Indeed, such human-automation teaming does not yet exist in real-world applications since today's technology is not sophisticated enough (Rieth and Hagemann, 2022). To gain the trust of ATCOs, the automation must be proven to be capable of handling tasks autonomously in a safe and efficient way.

On the other hand, instead of the mostly centralised control exerted in current-day operations, future ConOps may rely on alternative control strategies. For instance, previous work studied concepts for air traffic management with decentralised control (Câmara et al., 2014; Udluft, 2017). Nonetheless, these may substantially change the work of ATCOs, and must thus be evaluated in detail with all involved stakeholders. In turn, any evaluation of future ConOps necessitates a control system model that represents the underlying operations with sufficient fidelity. For instance, the interrelated processes of ASM Ops as well as the new engine-off taxiing techniques must be incorporated to demonstrate the efficacy to safely coordinate also congested traffic situations. To compute realistic 4D trajectories, not only the various operational processes, but also the heterogeneous fleet of differently sized aircraft and ground vehicles with different kinematics must be represented in detail.

However, most existing models of ASM Ops are geared towards studying either an isolated part of the system or one specific operational concept, based mainly on current-day operations. Furthermore, previous studies often relied on multiple simplifying assumptions to handle the inherent complexity, e.g. pre-defining taxiing routes and focusing on a single bay area (Okuniek et al., 2018). In turn, such models are inept to study the potentially wide range of ConOps to address new technological solutions, future operational approaches, as well as environmental requirements, let alone to compare the efficacy of different approaches with each other on a detailed level.

To address this research gap, we first define the problem of ASM Ops and derive a set of modelling requirements. As one of the main contributions, we then propose a generalised multi-agent architecture with a hierarchical-distributed structure to facilitate studying concepts for next-generation ASM Ops across, for example, varying degrees of technological maturity, automation, and control centralisation. We use the paradigm of multi-agent systems due to its inherent modularity, flexibility in modelling agent properties, interactions, and control structures (Helbing, 2012), and suitability to represent real-world systems (Dorri et al., 2018). While some of the modelling requirements are addressed by the generalised architecture, others depend on the specific ConOps. To this end, the framework provides design freedom to instantiate tailored models that reflect particular operational concepts, and enables to reuse model components, explore the implications of different coordination mechanisms, and compare system-wide performance across alternative ConOps.

Next, we illustrate the proposed generalised architecture by focusing on one model instance tailored to centralised planning for fully automated control: despite posing significant challenges for its implementation in real-world settings, it is expected to offer the highest operational efficiency, and can thus be used as baseline for future comparisons with other model instances. In the derived multi-agent system (MAS) model, the movements of agents are coordinated by assigning priorities to them. We represent the underlying decision-logic through the newly developed algorithm for multi-agent motion planning on airport surfaces (AS-MAMP). As its backbone, 4D trajectories are computed that satisfy both the operational processes as well as constraints necessary to coordinate with movements higher in priority, formalised as novel single-agent algorithm called Safe Interval Motion Planning (SIMP). It is inspired by the Safe Interval Path Planning (SIPP) algorithm (Phillips and Likhachev, 2011) and its variants. SIMP further extends the capabilities of SIPP-based low-level planners to deal with all important characteristics of ASM Ops such as the agent kinodynamics and sizes, safety zones around agents, and the different operations like pushback, engine-start, tug coupling, and decoupling.

In the AS-MAMP algorithm, we conduct multi-agent coordination based on the Priority-Based Search (PBS) algorithm (Ma et al., 2019) and its greedy variant GPBS (Chan et al., 2023). Nonetheless, combining SIMP with other high-level solvers is in general possible. To benchmark the AS-MAMP algorithm, we assess the performance of multiple PBS-variants as well as sequential planning based on the first-come-first-served principle often used in literature. Moreover, to showcase the algorithmic properties of SIMP, we benchmark it against SIPP and A* requiring additional assumptions to calculate trajectories in the AS-MAMP algorithm.

In a nutshell, the main contributions of this paper are:

- the generalised architecture for hierarchical-distributed multi-agent system models to study new concepts of airport surface movement operations, allowing to instantiate models with differing operational/procedural concepts, levels of automation, and degrees of centralisation;
- the elaborated MAS model for a novel concept of airport surface movement operations based on centralised planning and distributed, fully automated control;
- the AS-MAMP algorithm as two-level solver for conflict-free multi-agent motion planning that accounts for the heterogeneous fleet of agents, their shapes and kinodynamic constraints;
- the SIMP algorithm as SIPP-based solver accounting for finite acceleration as well as upper and lower speed bounds, thus enabling to plan kinodynamically feasible trajectories in complex, graph-based environments.

In the following, we first describe the problem of airport surface movement operations, and define a list of requirements that we deem indispensable to model the operations realistically (Section 2). Based on these, we outline previous research related to modelling ASM Ops and multi-agent path planning algorithms in Section 3. We then describe the generalised MAS model architecture for studying new ASM Ops concepts (Section 4). As the core element of an instantiation of the generalised MAS model focused on centralised planning, we formalise the AS-MAMP algorithm (Section 5). We benchmark the AS-MAMP components against A*, SIPP, and sequential prioritisation often used in literature (Section 6). Furthermore, the influence of important algorithmic parameters of and operational aspects on the AS-MAMP algorithm are discussed. In Section 7, we reflect on our modelling approach, list its limitations as well as real-world challenges, and point out directions for future work. Finally, we summarise the main findings and contributions of this work in Section 8.

2. Problem definition for airport surface movement planning

In this section, we first describe the operational background of current and potential future airport surface movement operations. We then formulate modelling requirements for creating high-fidelity models to study next-generation operational concepts. Moreover, the modelling requirements will guide the assessment of previous work carried out in Section 3 as well as for proposing a generalised system modelling architecture in Section 4.

2.1. Operational background

In its current form, ASM Ops are controlled and coordinated by air traffic control officers (ATCOs) who control the movements on the airport surface from the centrally located control tower. To ensure safe and efficient operations, the ATCOs provide clearances and guidance for all aircraft and ground vehicles moving along the taxiway network. ATCOs rely on other stakeholders to aid their decision-making, such as airport operators, ground handlers, and airlines that manage e.g. the flight schedules, stand allocation, and turn-around processes. As pilots and drivers are responsible for the safety of their individual vehicle, they determine the taxiing speed themselves. However, as they are unaware of the entire traffic situation and pertaining information, they rely on the overall coordination by and communication through the ATCOs, and are obliged to follow the ATCOs' instructions. As consequence, ASM Ops are an inherently cooperative system.

Currently, aircraft mainly use their own engines to move over the airport surface, referred to as multi-engine taxiing (MET) in this paper. After landing, inbound flights taxi to the stand, during which their engines cool down. Then, the turn-around processes commence, some of which are to unload the baggage, refuel, clean, and let new passengers board the aircraft. Once ready for pushback

with a pushback-truck (PBT) coupled to the aircraft, the existing taxi procedures can be split into three parts: 1) the pushback and (possibly) push-pull manoeuvre, 2) PBT decoupling and engine-start, and 3) taxiing to the runway. Before step 3) can be performed, the aircraft engines must be running. The engine-start procedure usually begins once the pushback commences. Only in the case of a push-pull manoeuvre, engine-start must be performed entirely after finishing the pull manoeuvre as it is carried out to avoid engine blast.

In next-generation ASM Ops, new means of taxiing as well as novel aircraft propulsion systems are envisioned. As such, different engine-off taxiing techniques have matured beyond the research stage, with the TaxiBot system (TaxiBot, 2023) being closest to operational deployment. We thus pick the latter as example of a future ConOps, referred to as tug-enabled taxiing (TET) in this work. Based on the TaxiBot specification and interviews with operational experts, outbound TET operations can be divided into four stages: 1) pushback and direction switch, 2) tug-enabled taxiing to a decoupling location close to the runway, 3) decoupling and all-clear signal between tug and aircraft, and 4) taxiing onto the runway. Depending on airport regulations, engine-start can potentially be performed during stage 2). Nonetheless, holding at the decoupling location may be necessary when the duration of pushback, taxiing, and decoupling is shorter than the time needed to start the engines. For arriving aircraft, TET comprises three relevant stages: 1) exiting the runway and taxiing to a coupling location close to the runway, 2) coupling to the waiting tug, and 3) tug-enabled taxiing to the ramp.

For both flight directions, the tugs have to drive to the coupling location, and once decoupled travel to the next assignment or the tug-base. Airports may provide extra service-roads, or require them to use the taxiway network managed by ground control. Eventually, the tugs are expected to be equipped with electric engines (Zoutendijk et al., 2023). In turn, this requires recharging over the day of operations, either at dedicated charging stations or at the stand, affecting the task assignment and route planning of tugs after decoupling.

2.2. Modelling requirements

To study new concepts for ASM Ops in detail, the real-world operations must be modelled, simulated, and thoroughly evaluated, requiring a high-fidelity model. In turn, it must address the requirements derived from the operational background. With Fig. 1 providing an overview, the contemplated modelling requirements are discussed below.

Requirement 1: system model shall represent multiple interacting actors

Various stakeholders partake in ASM Ops, sharing the general objectives of safe, efficient, and predictable operations besides their individual tasks, which may be conflicting between actors. Aircraft and additional vehicles such as tugs share limited resources, i.e. runways, taxiways, and bay areas. Differences in dimensions across aircraft types and between aircraft and tugs lead to a heterogeneous vehicle fleet. Moreover, their kinodynamics or non-holonomic properties must be accounted for, i.e. finite acceleration and deceleration, minimal speed to avoid stop-and-go, and speed restrictions on both straight as well as curved taxiway segments. Due to the strict safety standards, all actors are part of a *cooperative system*, making coordination between actors and control of their actions essential.

Requirement 2: system model shall accommodate the interrelated tasks

In general, ASM Ops comprise different operational tasks such as regular taxiing, pushback-manoeuvres, engine-start, holding, and tug operations. In parts, these tasks depend on and form interactions between each other, e.g. between aircraft in queues, landing aircraft vacating the runway, or in the bay areas. Moreover, some tasks have associated time constraints that must either be surpassed or reached in time (i.e. deadlines). Examples include an arriving flight whose assigned stand is still occupied, a departing flight that must take off within a departure slot due to airspace restrictions (i.e. CTOT-slots assigned by Eurocontrol), or aircraft in need of a tug or towing vehicle.

Requirement 3: system model shall comply with safety-related aspects

The operations must meet the strict safety standards at all times. As such, all vehicles must keep sufficient safety clearances to each other while moving over the airport surface. Furthermore, any engine-blast risk must be minimised and sufficient wake-turbulence separation during takeoff and landing must be kept. Moreover, the system model should exhibit predictable and consistent behaviour under both nominal and non-nominal operational conditions, which is essential for the identification and mitigation of potential hazards.

Requirement 4: system model shall represent airport environments in detail

To represent diverse ASM Ops across different airport sizes and use cases, the system model must be capable of accommodating variations in airport topology and complexity. Especially for large airports, such as Amsterdam Airport Schiphol (AAS), the taxiway system is rather complex and contains not only straight and curved segments, but also compounded intersections, runway crossings and go-arounds, long single-lane taxiways, and designated holding areas. Other infrastructural constraints include the inability of aircraft to turn around on a taxiway segment, temporarily unavailable taxiways, limited stand space, and pushback procedures. Additionally, the model should incorporate operational aspects such as arrival and departure schedules, and the runway mode of operation (RMO).

Requirement 5: system model shall fulfil operational goals

The system model shall optimise the operations to meet key operational objectives such as high operational efficiency through the reduction of taxi times and delays, and maintaining throughput levels that align with current and projected traffic demand. Furthermore, the model should account for additional performance metrics such as the environmental impact, e.g. fuel consumption and emissions. This allows to assess the trade-offs between different operational goals as well as compare alternative operational strategies under varying scenarios.

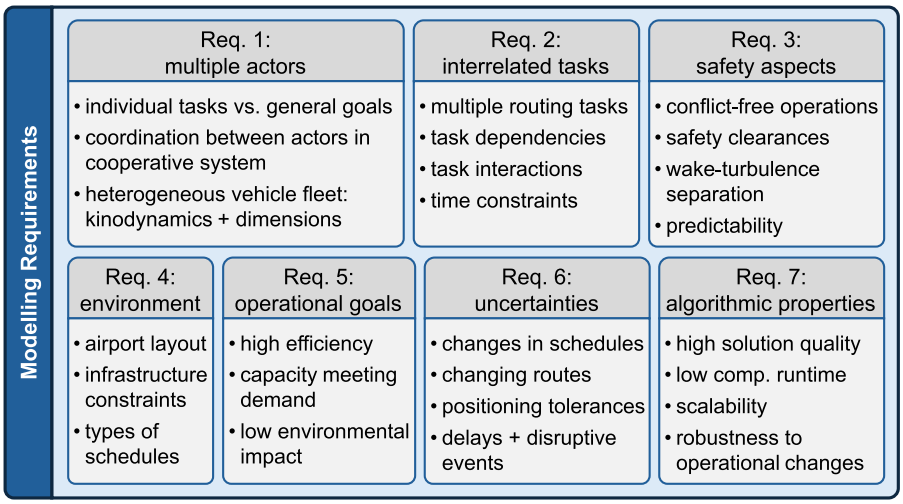

Fig. 1. Overview of modelling requirements to study new concepts of operations for airport surface movements in detail.

Requirement 6: system model shall accommodate operational uncertainties

In actual operations, changes to planned actions or anticipated effects arise that must be handled accordingly, also during the routing of vehicles (Lee, 2024). Examples include changes to flight schedules, task assignments of tugs, or changing runway configurations. Such changes often have short lead times (e.g. RMO changes may be known less than 30 min in advance). During taxiing, vehicle routes may change due to either operational reasons or errors in following the instructions provided by ATC. Furthermore, positioning tolerances must be accounted for, due to sensor noise or technological limits in precisely executing the assigned vehicle trajectories. Delays in procedures and actions, communication, and various other forms (Morris et al., 2016), up to (confined) disruptive events such as sudden vehicle breakdowns, may occur throughout the operations.

Requirement 7: decision-support algorithms shall be effective and efficient

The decision-support algorithms deployed in the system model shall deliver high-quality solutions, generating conflict-free and efficient trajectories that are operationally feasible and that realistically reflect operational constraints. To enable fast-time simulation and iterative optimisation, the algorithms shall achieve low computational runtimes and scale with airport layout complexity and traffic density. Additionally, they shall remain robust to dynamic operational changes such as those mentioned in Requirement 6, thereby ensuring reliable performance across a range of scenarios.

3. Related work and research gaps

In line with its current form of control, most previous work modelled ASM Ops as a centralised system, and studied related sub-problems. Other research has focused on developing new concepts of operations of ASM Ops as a centralised, but also as a distributed system. We will describe these in the next subsection, and thereafter outline the suitability of multi-agent path planning algorithms for planning conflict-free trajectories in ASM Ops.

3.1. Modelling of airport surface movement operations

In previous research, ASM Ops were divided into specific optimisation problems: the Ground Routing Problem (GRP) seeks conflict-free taxi schedules, typically framed as MILP or GA formulations (Atkin et al., 2010), while the Runway Scheduling Problem (RSP) maximised runway throughput via MIP, DP, branch-and-bound, or heuristics (Bennell et al., 2011; Lieder et al., 2015). Guépet et al. (2017) reviewed studies that fully or partially couple GRP and RSP, and argued that their better integration significantly reduces the taxi times. Additional sub-problems tackled runway-induced taxi congestion through gate-holding (Balakrishnan and Jung, 2007; Ravizza et al., 2014; Guépet et al., 2016), or pushback control and derated takeoffs to lower emissions (Ashok et al., 2017).

Other research focused on approaches for predicting taxi times and anticipating delays under uncertainty (Yin et al., 2024; Li et al., 2019). To this end, different approaches were used, e.g. using a stochastic model (Lee, 2024), chance-constraint programming (Wang et al., 2021), or a fuzzy rule-based approach (Brownlee et al., 2018). Other studies used data-driven approaches to predict the landing times of aircraft (Wang et al., 2018; Dhief et al., 2020), to visually show the runway exit prediction (Woo et al., 2022), or to provide recommended gate-hold times of outbound aircraft (using historical trajectories to fix their paths) (Ali et al., 2022).

Collectively, these studies have shown tangible gains in operational efficiency and predictability by optimising the taxi times as well as narrowing the forecast errors. However, they still treat ground, runway, and turnaround processes as weakly coupled subsystems. As a consequence, system-wide objectives influenced by dependencies across process boundaries are not optimised. This motivates integrated, concept-driven frameworks for assessing new concepts of operations (ConOps), as discussed in the next section.

3.2. Studying new concepts of operations

As approach towards fully automated tower operations, Morris et al. (2016) proposed a centralised automation system using discretised time and a grid-based layout with obstacles to represent taxiway networks. Their prototype of a surface management system focuses on data analysis for behaviour modelling, as well as continuous route planning, scheduling, and monitoring.

NASA's ConOps for far-term Surface Trajectory-Based Operations (STBO) envisions that pilots actively participate in precisely meeting time-based goals (Hooey et al., 2024). In two phases, their approach uses time-based 4D taxi clearances first provided as coarse targets at specific locations, and eventually as high-precision trajectories. However, for the second phase, the authors highlighted that more research into advanced flight deck equipment is needed. Focusing on the first phase, various fast-time and human-in-the-loop trials using their developed decision-support tools showed promising results for more efficient and predictable operations (Jung et al., 2011; Hayashi et al., 2024; Bakowski et al., 2015, 2017).

Using a simulation platform of similar scope, DLR tested prototypes of the Airport Collaborative Decision Making (A-CDM; EUROCONTROL, 2016) and the Advanced Surface Movement Guidance and Control System (A-SMGCS; ICAO, 2004) approaches for European airports. Likewise, their human-in-the-loop simulations showed similar operational benefits when pilots adhere to the conflict-free 4D trajectory clearances (Gerdes and Schaper, 2015). In a joint effort, NASA and DLR also developed a shared concept of operations and evaluated it using fast-time simulations (Okuniek et al., 2018). However, the taxi routes were pre-defined and were planned on a first-come-first-served basis. Both aspects likely lead to sub-optimal solutions and may not be applicable to complex traffic situations.

In general, centralised systems may lack knowledge of local conditions affecting operations, and communication might be impeded by capacity and bandwidth issues (Udluft, 2017). Alternatively, ASM Ops can also be modelled with decentralised control, as studied by Udluft (2017) for a basic airport layout. In such a system, the communication and control mechanisms are shifted to multiple local entities. While these approaches enhance robustness, resilience, and the quality of local solutions, they do not guarantee a global optimum and may lead to inefficient use of global resources.

Collectively, simulation platforms that focus on evaluating novel concepts for ASM Ops commonly generate realistic 4D trajectories but rather use simple optimisation techniques to coordinate the movements, often following the first-come-first-served principle. As already outlined in the previous section, paths can also be planned without such stringent order with potentially better solutions. To this end, the domain of multi-agent path planning (MAPP) provides a multitude of algorithms to solve both simplified and complex problems, as discussed next.

3.3. Multi-agent path planning algorithms for conflict-free routing

Interestingly, many MAPP-related papers motivate their work with ASM Ops (e.g. Ma et al., 2016; Cohen et al., 2019; Walker, 2022), while such algorithms are often neglected in operations-focused research (e.g. Zou et al., 2018; Jiang et al., 2021; Zaninotto et al., 2021; Yin et al., 2024). To close this gap, we first provide an overview of important MAPP concepts and algorithms, from solving classical problems towards using them for real-world applications. As summarised below, two-level solvers stand out in their wide range of applicability, with Priority-Based Search (PBS) being especially suitable for ASM Ops. In a separate subsection, we then outline the Safe Interval Path Planning (SIPP) algorithms as state-of-the-art single-agent planner used in two-level solvers. Finally, we provide a summary of the suitability of different path planning algorithms for ASM Ops.

3.3.1. Overview of algorithms to solve MAPP problems

Much research has concentrated on the classical multi-agent path finding (MAPF) problem in which several assumptions are utilised to simplify the search: the environment is defined as static, both time and space are discretised, agents are depicted as points that only occupy a single location per timestep, and move between these within unit time (Stern, 2019). Moreover, it is typically assumed that the generated plans are perfectly executed (Ma et al., 2017). Comprehensive overviews of the MAPF problem and its algorithmic landscape are provided in Stern and Sturtevant (2019, with MAPF definitions, variants, benchmarks), Felner et al. (2017, with MAPF variants focused on search-based optimal solvers), and Gao et al. (2024, with classic but also beyond classic MAPF solvers).

For solving classical MAPF problems, two-level solvers have proven to be most suited: they combine the strengths of different algorithms and yield better solution quality and lower computational times than other approaches such as reduction-based, rule-based, or other one-level search-based solvers. Prominent examples of optimal two-level MAPF solvers are M* (Wagner and Choset, 2011), the Increasing Cost Tree Search (ICTS) (Sharon et al., 2013), and Conflict-Based Search (CBS) (Sharon et al., 2012).

The latter has found wide use to generate optimal solutions to classical MAPF instances. CBS is based on the ideas of building a binary conflict-tree during the search, which is expanded using a best-first search to sequentially resolve all conflicts between agents. To improve its computational runtime, several sub-optimal variants were proposed (Barer et al., 2014; Cohen et al., 2016; Li et al., 2021). While these algorithmic improvements showed promising results in trading off solution quality for runtime, any CBS-variant may still be too slow for applications requiring close to real-time path planning computations, especially when relaxing the assumptions of classical MAPF, as done by e.g. Li et al. (2019), Walker et al. (2020).

As alternative two-level solver, Ma et al. (2019) introduced the Priority-Based Search (PBS) algorithm. PBS generates a priority-tree to explore the state space by applying different priority orderings for the agents, requiring a lower prioritised agent to avoid all paths of those higher in priority. The priority-tree is expanded using a depth-first search, resulting in fast execution times. While PBS is unbounded suboptimal and incomplete, the authors claim that it provides near-optimal solutions, also on an airport-like map

(DAO-map brc202d). Recently, Chan et al. (2023) introduced Greedy PBS (GPBS) and multiple algorithmic extensions for PBS to lower the computational runtime especially for MAPF instances with both many agents and obstacles. Overall, the PBS-family is regarded as one of the leading MAPF algorithms, and was thus used in several recent studies (Li et al., 2023; Zheng et al., 2023). Its algorithmic properties and the concept of reasoning with priorities make it particularly well suited for generating conflict-free trajectories in ASM Ops.

As general approach to decrease execution times of MAPF algorithms, Silver (2005) introduced the idea of windowing the search, which is also referred to as bounded-horizon planning in other articles (Li et al., 2021). Using a windowed approach, conflict resolution is done up to a pre-defined time horizon w, while conflicts beyond that limit are ignored. This decreases the search space and thus the required runtime of the path planning algorithm.

In previous research, some of the assumptions of classical MAPF mentioned above were relaxed. As examples, studies explored the online MAPF variant requiring frequent replanning (Švancara et al., 2019), the MAPF_R problem for continuous, non-uniform edge weights (Walker et al., 2018), MAPF instances with delay probabilities (Ma et al., 2017), or robustness in planning (Ma et al., 2018). Li et al. (2019) explored MAPF instances with Large Agents (LA-MAPF), in which agents occupy not one but a set of points in a graph at any timestep. New conflict types are necessary to coordinate their movements, increasing the number of possible collisions. Since MAPF plans can usually not be directly executed by real-world robots (Yakovlev et al., 2019; Hoenig et al., 2016) proposed MAPF-POST which uses a temporal network to post-process MAPF solutions so that non-holonomic robots can execute the plans with guaranteed safety distances between them.

Nonetheless, the strong assumptions to solve such MAPF instances do often not apply to real-world problems (Ma et al., 2016). Dependent on the application, continuous time and space dimensions as well as non-holonomic, heterogeneous agent properties may need to be planned for. The combination of these is often referred to as multi-agent or multi-robot motion planning (MAMP or MRMP). The latter usually addresses open spaces requiring roadmaps to be generated through e.g. sampling-based techniques (González et al., 2016). However, the taxiway layouts of airports do not resemble open spaces, but rather consist of sparsely located intersections connected by roads in a continuous, two-dimensional Euclidean environment. Sampling-based techniques are thus not applicable to coordinate the agents in ASM Ops. Instead, the taxiway network can be transformed into a graph using the physical coordinates of intersections as nodes and the length of the interconnecting taxiways as edge weights, leading to a representation in continuous space. Alternatively, one study transformed the taxiway network using a grid-based layout with obstacles (Morris et al., 2016). For both layout transformations, the above mentioned two-level search-based solvers can be applied when coupled with a low-level algorithm that can deal with the continuous dimensions and non-holonomic agent properties.

3.3.2. Safe interval path planning (SIPP) algorithms

As key advancement for single-agent path planning, Phillips and Likhachev (2011) introduced the Safe Interval Path Planning (SIPP) algorithm to deal with dynamic obstacles. SIPP is based on the idea of aggregating timesteps into time intervals: the duration that a moving obstacle occupies a graph location is defined as unsafe interval (USI), while the time between two USIs as safe interval (SI). This representation of the time dimension reduces the amount of potential states at every graph location and therefore the search space. Besides decreasing the runtime significantly, this also enables planning with continuous time and space, arbitrary motion durations, and different agent speeds.

For this reason, SIPP was subsequently used and further extended as single-agent planner (Yakovlev and Andreychuk, 2017, 2021; Zou and Borst, 2024), and as low-level solver for solving multi-agent instances (Yakovlev et al., 2019; Andreychuk et al., 2019, 2021). To generalise SIPP to continuous time with large agents moving on 4-neighbour grids, Ma et al. (2019) used SIPP with a reservation table (SIPPwRT), in which the shape of agents is expressed by a safety radius. Together with the velocities of two agents, but without accounting for acceleration and deceleration rates, they then compute time offsets to set safe intervals for lower prioritised agents. Li et al. (2022) generalised SIPP to handle soft obstacles, called SIPP with Soft Constraints (SIPPS). Soft obstacles mark the pending conflicts with the plans of other agents that are not yet set as hard constraints. These are tracked in SIPPS to minimise the number of unresolved conflicts when creating a new plan. The authors used SIPPS in their proposed algorithm MAPF-LNS2 that starts with an invalid MAPF solution still containing unresolved conflicts. The remaining count of collisions is then gradually minimised by repeatedly replanning for a subset of agents. MAPF-LNS2 thus lowers the remaining collisions within a defined runtime limit, or improves the initial solution to be conflict-free.

Nonetheless, Ali and Yakovlev (2023) argued that all previous work on SIPP does not account for the kinodynamic constraints of agents prohibiting them to stop instantaneously. They reasoned that these agent properties make vanilla SIPP and its variants incomplete. To address this issue, they proposed SIPP with (Wait) Interval Projection (SIPP-IP) and proved that it is both complete and optimal. In SIPP-IP, each state S has an associated non-overlapping, projected safe interval representing the time interval that can be safely reached from the safe interval at a previous wait-location in the path to S. However, their idea only holds for states that originated from a wait-location, limiting its applicability to real-world problems.

3.3.3. Summarising the applicability of the existing path planning algorithms for ASM Ops

Table 1 summarises the properties of the algorithms reviewed above, in line with the requirements defined in Section 2.2. While MILP and CBS solvers provide optimal solutions, they are intractable to plan for many agents in a complex operational environment within a reasonable time. Machine-learning/reinforcement-learning (ML/RL) approaches are deemed not yet capable to handle the intrinsic complexities of realistic representations of ASM Ops. MAPF-LNS2 aims to iteratively improve invalid solutions with remaining conflicts, is typically applied to problem instances involving thousands of agents, and was so far only tested on uniform grid

Table 1
Algorithmic properties of route optimisation techniques as backbone for a high-fidelity system model of ASM Ops.

Criterion	MILP	FCFS	ML/RL	CBS	PBS	MAPF-LNS2
planning scheme	single-stage	sequential	varies	best-first search	depth-first search	iterative refinement
low-level solver	MILP solver	A*, SIPP	learned policy	A*, SIPP	A*, SIPP	SIPPS
conflict handling	predefined constraints	sequential	learned avoidance	constraint-tree	priority-tree	via sub-groups with priorities
search space	very large	small	small after (extensive) learning	very large	medium	medium
solution quality	optimal	suboptimal (mostly low)	depends on learning	optimal	near-optimal	anytime improvement
safety guarantees	yes	yes	?	yes	yes	conflicts may remain
scalability	low	high	high (post-training)	low	medium	high
real-time use	no	yes	yes	no	potentially	potentially
typical environment	graph-based	varying	open spaces	varying	varying	uniform grid

environments. All these properties of MAPF-LNS2 make it less suitable for the operations considered in this paper. In contrast, prioritised planning algorithms such as PBS and FCFS are well aligned with conflict resolution strategies in ASM Ops, in which priorities are commonly assigned. While FCFS likely provides inefficient solutions, PBS-based algorithms are regarded to deliver near-optimal solutions, and to scale well also in complex environments. We thus deem PBS-like algorithms to be most suitable to carry out multi-agent coordination in ASM Ops models.

In addition to prioritised planning, the efficient generation of conflict-free, 4D agent trajectories is essential for obtaining high-quality solutions within reasonable runtimes. To this end, the SIPP-family offers a strong basis for single-agent planning. However, SIPP must be tailored to the operational context of ASM Ops and extended to support motions with finite acceleration and non-zero initial speeds.

3.4. Addressing the research gaps

The reviewed literature showed that most existing models of ASM Ops are geared towards studying either an isolated part of the system or one specific operational concept, mainly based on current-day operations. Such models are thus inept to study the potentially wide range of ConOps that result from new technological solutions as well as future operational and environmental requirements, let alone to compare the efficacy of different approaches with each other on a detailed level.

To address the research gap, we propose a generalised multi-agent system architecture for studying next-generation concepts of airport surface movement operations. The architecture has a flexible, hierarchical-distributed structure, taking the defined requirements from Section 2.2 as well as the outlined advantages and limitations of both centralised and distributed systems into account. Although highly distributed systems are conceivable, the stringent safety standards (Requirement 3) necessitate a form of operational oversight, thereby motivating the adoption of an underlying hierarchy in the generalised architecture to coordinate the operations. We use the paradigm of multi-agent systems due to their demonstrated suitability to represent real-world systems, as shown across various applications (Dorri et al., 2018), and their inherent modularity, flexibility, and expressiveness. Both heterogeneous agent properties and behaviour as well as randomness can be integrated into an agent-based model (Helbing, 2012), facilitating the design of detailed models to address the identified modelling requirements. In the context of ASM Ops, multi-agent system modelling thus allows to organise the complex and intertwined operational tasks into multiple smaller tasks and assign these to various agents with different roles.

Based on the generalised architecture, specific model instances can be defined to study new concepts of operations characterised across various dimensions such as procedural aspects (e.g. selected taxiing techniques, effect of and compliance with existing/new procedures), degree of centralisation (from centralised control to distributed control with enacted system-wide constraints), and level of automation per task (allocated to a human vs. an automated system), requiring different coordination schemas. As a consequence, the generalised architecture offers design freedom to develop more elaborate and tailored models, while allowing to reuse the overall simulation environment as well as model components. Moreover, it enables comparing system-level performance (see Requirement 5) across varying ConOps, and to iteratively refine the operational concepts in fast-time simulations and human-in-the-loop experiments with stakeholders. So far, several model instances were developed and used in research projects with both low (AEON, 2023) and high levels of automation (ASTAIR, 2025), and in studies exploring fully automated operations (von der Burg and Sharpanskykh, 2025), engine-off taxiing (von der Burg and Sharpanskykh, 2024), and human-automation interactions (von der Burg and Sharpanskykh, 2024).

In the following section, we first describe the proposed generalised model architecture, and one model instance designed for fully automated ASM Ops with centralised path planning. To this end, we introduce the novel Multi-Agent Motion Planning on Airport Surfaces (AS-MAMP) algorithm for planning conflict-free, efficient, 4D trajectories for all ground movements. While we use PBS as primary high-level search in AS-MAMP, other prioritised planning approaches such as GPBS or FCFS can be deployed alternatively. As low-level solver in AS-MAMP, we formulate the novel Safe Interval Motion Planning (SIMP) algorithm to address the outlined shortcomings of SIPP, described as part of the overarching AS-MAMP algorithm in Section 5. In Section 6, we benchmark the PBS variants, and compare SIMP to SIPP and A* with kinodynamic constraints. Furthermore, we evaluate the performance of the AS-MAMP algorithm on both a synthetic as well as a real-world airport layout.

4. Multi-agent system models to study future airport surface movement operations

ASM Ops tasks can be divided in several categories: planning (strategic and long-term decisions), scheduling (tactical decisions), routing (tactical path planning), providing guidance (operational path planning), and steering an aircraft or ground vehicle (operational plan execution). The progression from strategic planning to operational control reflects a hierarchical structure among the tasks. At the same time, the strict safety standards of ASM Ops (Requirement 3) necessitate coordinated oversight across all operational levels. Together, these aspects motivate the choice of a hierarchical-distributed system architecture to manage the complexity and to ensure safety as well as efficiency.

4.1. Generalised hierarchical-distributed control architecture

To enable studying diverse concepts of next-generation ASM Ops, we thus propose a generalised multi-agent system architecture with a hierarchical-distributed structure. It comprises multiple levels of control through distinctive agent categories with potentially limited system knowledge, scope of control, and access to information. We define the agent categories in analogy to the hierarchy of ASM Ops tasks mentioned above. Illustrated by Fig. 2, we outline the different agent categories and environmental objects constituting the generalised architecture in the following.

In the generalised architecture, agents are associated with ASM Ops tasks to exert control and exchange data in line with the hierarchical placement of the agent categories. Within each category, multiple agents may operate and engage in centralised and/or distributed control mechanisms, informed by communication within their level as well as across hierarchical layers. Furthermore, all agents can potentially interact with environmental objects as part of their perception and decision-making processes.

The agent category for strategic planning tasks such as defining the flight plans is deliberately disregarded, since such tasks are carried out long before the actual ASM Ops take place. They are placed in the environmental object of **External Data & Events** along with other information from stakeholders outside the scope of ASM Ops. These include dynamic system inputs such as weather forecasts and weather events, as well as CTOT-slots assigned by the network manager, among others. The environmental object called **Airport Infrastructure** constitutes the taxiway network as well as other infrastructure such as remote parking locations, recharging stations, or ground vehicle depots to enact the specific ConOps. It is represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, where edges $e \in \mathcal{E}$ represent taxiway centrelines and nodes $v \in \mathcal{V}$ correspond to merging points between taxiways as well as special points such as stands or tug coupling/decoupling locations. The edges are constructed using Bézier curves with non-uniform lengths l_e,

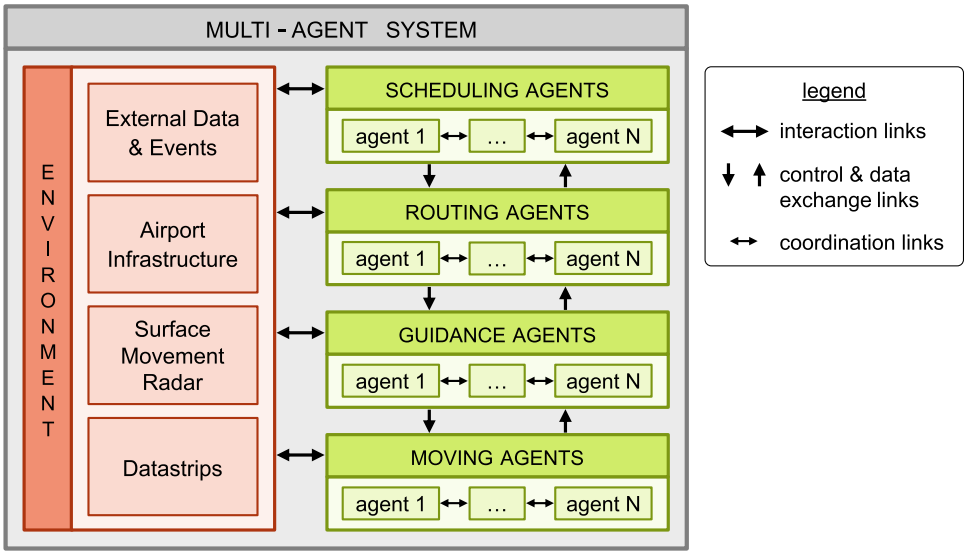

Fig. 2. Generalised hierarchical-distributed multi-agent system modelling architecture to study diverse concepts for next-generation airport surface movement operations.

enabling a realistic and flexible representation of airport layouts. To reflect operational constraints, additional edge attributes such as speed limits $v_{\text{lim},e}$ based on the radii of curvature can be assigned to the edges. Moreover, the environment provides the **Surface Movement Radar** as means of surface surveillance to provide information on e.g. cooperative ground movements (i.e. those that allow to be detected). In analogy to current-day flight-strips, the **Datastrips** store any information pertaining to ground movements such as schedule or route data. Dependent on the modelled ConOps, different agents can access, supplement, or store data in the environmental objects.

The **Scheduling Agents** $a_i \in \mathcal{A}^{\text{SKL}}$ are responsible to manage all airport schedules such as flight schedules, runway configurations, or task allocations of ground vehicles. Furthermore, they issue constraints pertaining to the Airport Infrastructure, e.g. due to the runways in use or temporarily unavailable taxiway segments. Both schedules and layout constraints are shared with the **Routing Agents** $a_i \in \mathcal{A}^{\text{RTN}}$. Their task is to conduct path planning for individual ground movements, and may include coordinating plans of multiple movements, dependent on the underlying ConOps. Based on the available route information, the **Guidance Agents** $a_i \in \mathcal{A}^{\text{GDS}}$ can further refine the routing plans, and exert their control by providing e.g. clearances or route instructions to the **Moving Agents** $a_i \in \mathcal{A}^{\text{MOV}}$ who are responsible to execute the operational plans by safely steering the aircraft or ground vehicles.

Besides regular taxiing, other operational processes such as inbound/outbound holding, pushback/push-pull manoeuvres, or coupling/decoupling of tugs (see Section 2.1) must be executed, accounting for dependencies between tasks as well as time constraints (Requirement 2). To this end, we represent the tasks as basic activities and their interdependencies as activity sequences per Moving Agent, defined as:

Definition 1 (Activities and activity sequence of Moving Agents)**.** Each activity is represented as a tuple $\alpha = (\tau, \theta)$, with the activity type $\tau \in \mathcal{T} = \{\texttt{GoTo}, \texttt{Follow}, \texttt{Wait}\}$ and set of parameters θ:

- for `GoTo` activities, θ is defined by a set of potential goal locations $G \subset V$;
- for `Follow` activities, θ is defined by a predefined path $P = \langle e_1, e_2, \dots, e_m \rangle$ to be followed;
- for `Wait` activities, θ is defined by a wait duration $\Delta t \in \mathbb{R}_{>0}$.

Each Moving Agent $a_i \in \mathcal{A}^{\text{MOV}}$ is assigned an *activity sequence*:

$$S_i = \langle \alpha_1, \alpha_2, \dots, \alpha_n \rangle_i, \quad \alpha_{k,i} = (\tau, \theta)_{k,i} \tag{1}$$

which must be executed in the given order. Each agent's first activity $\alpha_{1,i}$ must begin at a designated start location $v_{s,i} \in \mathcal{V}$, specified by the agent's initial position within the airport layout.

As example, the activity sequence of an outbound aircraft requiring a push-pull manoeuvre can be expressed as sequence of activity types $\langle \texttt{Follow}, \texttt{Wait}, \texttt{Follow}, \texttt{Wait}, \texttt{GoTo} \rangle$ with the corresponding parameters to follow a standard pushback-and-pull path with an intermittent stop to switch movement direction towards the location at which the pushback-truck is decoupled, and taxiing towards one of the runway entry points. Moreover, the definition of activities allows to express activity sequences for any technological concept: both multi-engine taxiing as well as tug-enabled taxiing including inbound/outbound holding as well as adherence to CTOT-slots can be modelled (von der Burg and Sharpanskykh, 2024, 2025). Likewise, movements of aircraft with novel, more sustainable propulsion systems such as hydrogen or battery-powered aircraft can be represented as a combination of `GoTo`, `Follow`, and `Wait` activities.

Based on the generalised architecture, a large variety of models can be instantiated that are tailored to a specific ConOps, while retaining the overall system structure. To this end, the operational tasks can be re-allocated to different agents, and their decision-making abilities can be customised with different coordination mechanisms. For example, the degree of control centralisation could differ between model instances: in a centralised setting, all movements could be coordinated up front by one or multiple Routing Agents, based on schedules issued by the Scheduling Agents. This would minimise the guidance and control tasks carried out by Guidance Agents and Moving Agents during plan execution. In contrast, in a model with distributed path planning, the Moving Agents could reason themselves about each required action in their activity sequence. They could then negotiate with other agents to resolve conflicting situations based on their individual circumstances and operational needs. In such a setting, the Routing Agents and Guidance Agents could act as intermediaries, set traffic rules, or issue special priorities, in accordance with the schedules provided by the Scheduling Agents. Likewise, the generalised architecture allows to define ConOps with differing levels of automation: each ASM Ops task can either be allocated to a human, an automated system, or split into a collaborative effort between humans and automation. In line with these examples, we elaborate one model instance representing a ConOps for fully automated ASM Ops with centralised path planning in the following subsection.

To summarise, the proposed generalised modelling architecture fulfils Requirement 1 and 2 through its hierarchical-distributed architecture as well as the allocation of tasks to the corresponding agent categories, and Requirement 4 with the environment definitions. The remaining requirements comprising safety aspects, operational goals, dealing with uncertainties, and the algorithmic properties of the utilised coordination mechanisms must be ensured when instantiating a MAS model based on a specific ConOps.

4.2. Multi-agent system model for autonomous operations with centralised planning

For the remainder of this paper, we consider an operational concept for autonomous operations with centralised planning which was modelled based on the generalised architecture presented in the previous subsection. Although this ConOps poses significant implementation challenges in real-world settings, it is expected to offer the highest operational efficiency. In turn, it can serve as a suitable baseline for future comparisons with alternative operational concepts. In the following, we present an overview of the multi-agent model structure and its important properties. The complete specification of the model components are provided in Appendix A, with Fig. A.1 illustrating the MAS model.

As sole Scheduling Agent, the centralised **Airport Ops Agent** a_{APO} schedules the flights and tasks of ground vehicles, and issues layout constraints on the **Airport Infrastructure**. It shares both schedules and constraints with the single, centralised **Routing Agent** a_{RTN}. Its task is to conduct centralised path planning to provide conflict-free 4D trajectories for all ground movements, formalised as routing plans:

Definition 2 (Routing plans). Based on an agent's activity sequence $\mathcal{S}_i$, path $\mathcal{P}_i = \langle v_1, v_2, \cdots, v_m \rangle_i$ for agent $a_i \in \mathcal{A}^{\mathrm{MOV}}$ is defined. The path forms the basis for the agent's routing plan Π_i defined as:

$$\Pi_i = \langle \pi_i(v_1), \pi_i(v_2), \cdots, \pi_i(v_m) \rangle, \quad \pi_i(v_k) = (t_{v_k}, d_{v_k}, v_{v_k}, \alpha_{v_k})_i \tag{2}$$

where each tuple specifies per node v_k the respective time t_{v_k}, travelled distance d_{v_k}, speed v_{v_k}, and activity α_{v_k}.

The Routing Agent shares the routing plans with the **Guidance Agents** $a_i \in \mathcal{A}^{\mathrm{GDS}}$ that are positioned at every junction in the taxiway network. Each Guidance Agent controls those agents that are travelling towards its location by sending them instructions based on the received trajectories. To ensure safe operations, the Guidance Agents monitor the ground movements, and locally adjust the planned trajectories, if necessary (Requirement 3). To this end, the Guidance Agents use the **Surface Movement Radar** (SMR) as one of the environmental services to obtain the position and speed of an agent. Once an agent a_i has passed the Guidance Agent's location, it hands over the control to the next Guidance Agent along the agent's path by transferring the **Datastrip** D_i associated with that agent. D_i stores all data pertaining to the agent's routing plan, its activity sequence S_i, and the current SMR data of a_i. The **Moving Agents** $a_i \in \mathcal{A}^{\mathrm{MOV}}$ represent the aircraft pilots and tug drivers, and are modelled to be fully cooperative. We thus assume that they execute the received instructions to the best of their possibilities when taxiing over the airport surface. To address Requirement 3, all Moving Agents must keep a minimum safety distance to each other, expressed as a circular safety zone $\mathcal{Z}_i$ around the agents. Since pilots and auto-pilots can only follow planned trajectories with limited precision (see Requirement 6), the safety zones include a positioning tolerance δ_{pos} as allowable deviation from the intended trajectories.

In this MAS instance, while any perturbations can in general be communicated upstream, the Moving Agents as well as the Guidance Agents rely strongly on the centralised routing carried out by the Routing Agent. Therefore, to meet the operational goals outlined in Requirement 5, the Routing Agent shall determine conflict-free 4D trajectories that minimise the total taxi time of all Moving Agents. As connected objectives, the resulting trajectories shall also reduce the total taxi distance and mitigate any stop-and-go movements. Moreover, to keep the centralised planning tractable (Requirement 7) and to account for operational uncertainties (Requirement 6), we base the decision-logic of the Routing Agent on the concept of windowed prioritised planning:

Definition 3 (Windowed prioritised planning). Centralised path planning is done in planning rounds. Each planning round k starts at time $t_{\mathrm{p},k} \geq 0$. In each round, the Routing Agent computes or updates the routing plans Π_i of those agents $a_i \in \mathcal{A}^{\mathrm{MOV}}$ that are taxiing within the planning window $[t_{\mathrm{p},k}, t_{\mathrm{p},k} + w_{\mathrm{p}}]$ where $w_{\mathrm{p}} > 0$ denotes the fixed span of the planning window. In turn, the Routing Agent plans periodically within the maximum replanning period $h_{\mathrm{p}} > 0$, such that $t_{\mathrm{p},k+1} \leq t_{\mathrm{p},k} + h_{\mathrm{p}}$.
To deconflict concurrent agent trajectories, the Routing Agent assigns priorities between Moving Agents $a_i, a_j \in \mathcal{A}^{\mathrm{MOV}}$, with $a_i \prec a_j$ indicating that agent a_i has higher priority than agent a_j. To ensure that a_j gives way to a_i, the routing plan Π_j of agent a_j must not violate the safety zone $\mathcal{Z}(\Pi_i)$ around the routing plan Π_i of agent a_i within the planning window. Conflicts outside the current planning window can be safely disregarded as these are accounted for in subsequent planning rounds.

As representation of the Routing Agent's decision-logic to carry out centralised path planning and address Requirement 7, we propose the Multi-Agent Motion Planning on Airport Surfaces (AS-MAMP) algorithm in the following section, with an in-depth description of its technical details provided in Appendix B.

5. Multi-agent motion planning on airport surfaces (AS-MAMP) algorithm

The AS-MAMP algorithm is a two-level solver that uses a high-level search for multi-agent coordination by iteratively updating the individual agent trajectories in its low-level search. Fig. 3 illustrates the core structure of the AS-MAMP algorithm: it combines PBS variants (see Section 3.3) as high-level with the newly developed Safe Interval Motion Planning (SIMP) algorithm as low-level.

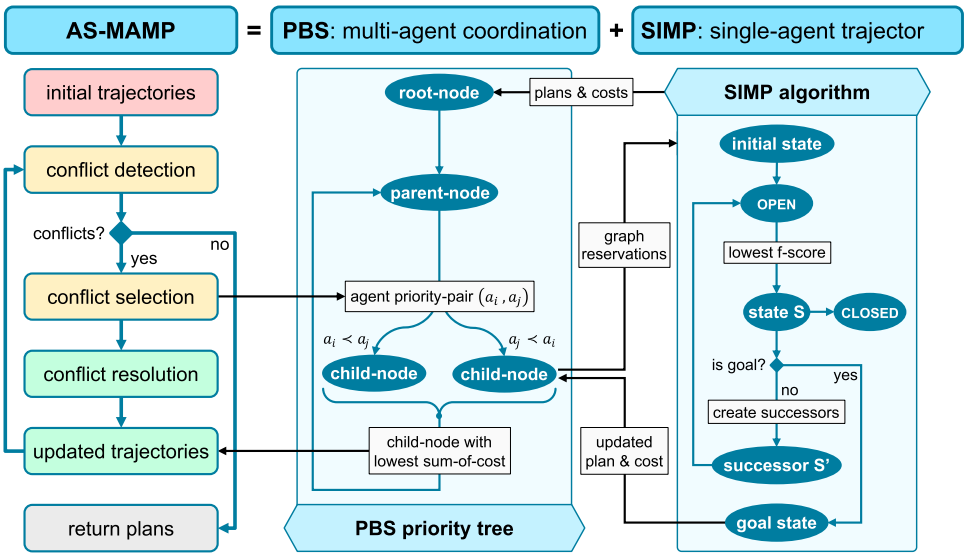

Fig. 3. Overview of the Multi-Agent Motion Planning on Airport Surfaces (AS-MAMP) algorithm. The representation of the SIMP algorithm is simplified, see the pseudocode in Algorithms B.3, B.4 for details.

Based on an initial set of agent trajectories, the loop comprising conflict detection, selection, and resolution is initiated. The search continues until a conflict-free solution is found, returning the coordinated agent trajectories as routing plans. We summarise the core algorithmic concepts of AS-MAMP and define key terms in the following. An in-depth description of AS-MAMP is provided in Appendix B.

5.1. High-level search in AS-MAMP

The PBS algorithm de-conflicts agents whose 4D trajectories intersect by constructing a priority ordering: it maintains a priority tree with each parent-node having up to two child-nodes to establish a priority-relation between two conflicting agents. Per child-node, a new priority-pair is added that constrains one of the two agents to avoid the other agent's path. Based on the sum-of-cost of the paths of all agents in each child node, PBS picks the priority order that results in the lowest overall cost, and expands this node next. In contrast to the original PBS algorithm, we define the cost of an agent trajectory as a linear combination of its taxi time and taxi distance:

Definition 4 (Cost function). The routing plan cost $c(\Pi_i)$ of agents $a_i \in \mathcal{A}^{\text{MOV}}$ is defined as sum of taxi time $t_{\text{taxi},i}$ and travelled distance $d_{\text{taxi},i}$, with a pre-defined constant $c_{d,i}$ [s/m] translating d_{taxi} into a cost with unit time:

$$c(\Pi_i) = (t_{\text{taxi},i} + c_{d,i} \cdot d_{\text{taxi},i}) \tag{3}$$

As defined in Definition 3, routing plans are deconflicted by accounting for the safety zones around agents. During replanning in AS-MAMP, higher-priority agents are treated as dynamic obstacles, with their respective shapes (see Appendix B.3) temporarily blocking parts of the graph $\mathcal{G}$, added to the graph reservations set for an agent:

Definition 5 (Graph reservations). The 4D trajectory of an agent $a_i \in \mathcal{A}^{\text{MOV}}$ must satisfy all graph reservations $\mathcal{R}_i$ imposed on that agent. Each graph reservation $\varrho \in \mathcal{R}_i$ is defined as a tuple:

$$\varrho = (\lambda, [t_s, t_e], \kappa) \tag{4}$$

- where $\lambda \in \mathcal{L}$ is a graph location with $\mathcal{L} \subset \mathcal{V} \cup \mathcal{E} \cup \{(\epsilon, \delta_1, \delta_2) \mid \epsilon \in \mathcal{E}, 0 \leq \delta_1 < \delta_2 \leq 1\}$ denoting either a node, an edge, or a segment of an edge between relative distances δ_1 and δ_2;
- $[t_s, t_e] \subset \mathbb{R}_{\geq 0}$ is the time interval during which the location λ must not be traversed by agent a_i;
- $\kappa \in \mathcal{K}$ indicates the reservation type, e.g. dynamic obstacle, wake-turbulence separation, CTOT-slot, etc.

The graph reservations are also used to detect conflicts between agents:

Definition 6 (Conflicting agent pair and priority relation). Let $a_i, a_j \in \mathcal{A}^{\mathrm{MOV}}$ be two agents with respective sets of graph reservations $\mathcal{R}_i$ and $\mathcal{R}_j$. The pair (a_i, a_j) is called a *conflicting agent pair* if there exist reservations $\varrho_k \in \mathcal{R}_i$ and $\varrho_l \in \mathcal{R}_j$ for which locations and time intervals overlap, i.e.

$$\lambda_k \cap \lambda_l \neq \emptyset \quad \wedge \quad [t_\mathrm{s}, t_\mathrm{e}]_k \cap [t_\mathrm{s}, t_\mathrm{e}]_l \neq \emptyset \tag{5}$$

Following from Definition 3, the 4D trajectories of a_i and a_j are deconflicted by creating a new PBS child-node with an additional *priority relation* $\prec \in \mathcal{A}^{\mathrm{MOV}} \times \mathcal{A}^{\mathrm{MOV}}$ such that either $a_i \prec a_j$ or $a_j \prec a_i$. A new routing plan of the lower-priority agent is calculated that satisfies the updated set of graph reservations by invoking the low-level solver, which is further specified in the following subsection. The high-level search continues until a child-node is expanded without any collisions. The changes to the original PBS algorithm are outlined in Appendix B.1, with the corresponding pseudocode provided in Algorithm 1. In Section 6.2, we benchmark the standard depth-first search in PBS against multiple variants to expand the priority-tree, including the first-come-first-served (FCFS) method.

5.2. Low-level search in AS-MAMP

As low-level solver in AS-MAMP for calculating a single-agent 4D trajectory that respects all graph reservations defined for that agent, we have developed the Safe Interval Motion Planning (SIMP) algorithm. Similar to SIPP (see Section 3.3), the SIMP algorithm reasons with safe time intervals, iteratively explores the search space with states defined on nodes, and uses motion principles for the mapping between states. However, to deal with the graph reservations in SIMP, safe intervals are not only defined on nodes but at any graph location $\lambda \in \mathcal{L}$ from Definition 5:

Definition 7 (Unsafe and safe intervals). For an agent $a_i \in \mathcal{A}^{\mathrm{MOV}}$, the time intervals of the graph reservations set for a location $\lambda \in \mathcal{L}$ are merged and ordered in time to form the *unsafe intervals* for a_i at λ:

$$\mathcal{U}_{i,\lambda} = \langle\, [t_\mathrm{s}, t_\mathrm{e}]_1, [t_\mathrm{s}, t_\mathrm{e}]_2, \cdots, [t_\mathrm{s}, t_\mathrm{e}]_n \,\rangle_{i,\lambda} \tag{6}$$

Then, the *safe intervals* denote the complement of $\mathcal{U}_{i,\lambda}$ as the sequence of time intervals accessible to a_i at λ:

$$\mathcal{I}_{i,\lambda} = \left([0, \infty] \setminus \bigcup \mathcal{U}_{i,\lambda} \right)^{\max} = \left\langle\, [0, t_{1,\mathrm{s}}], [t_{\mathrm{e},1}, t_{\mathrm{s},2}], \cdots, [t_{\mathrm{e},n}, \infty] \,\right\rangle_{i,\lambda} \tag{7}$$

Moreover, in contrast to the instantaneously accelerated motions assumed in SIPP, the low-level search shall account for the kinodynamic agent constraints, requiring to define the following motion principles:

Definition 8 (Motion principles). Motions are approximated with piecewise-constant longitudinal acceleration, i.e. we apply the equations for rectilinear motions:

$$\text{final speed} \quad v_\mathrm{f} = v_\mathrm{i} + a\,t = \sqrt{v_\mathrm{i}^2 + 2\,a\,d} \tag{8}$$

$$\text{traversal time} \quad t = \frac{2d}{v_\mathrm{i} + v_\mathrm{f}} = \frac{v_\mathrm{f} - v_\mathrm{i}}{a} \tag{9}$$

$$\text{travelled distance} \quad d = v_\mathrm{i}\,t + \frac{1}{2}\,a\,t^2 = \frac{v_\mathrm{i} + v_\mathrm{f}}{2}\,t \tag{10}$$

with initial speed v_i and acceleration/deceleration a. In SIMP, feasible motions respecting the kinodynamic agent constraints as well as the graph reservations are calculated on-the-fly during the search, with the cost of a motion defined in analogy to Eq. (3).

Per vehicle type, a maximum speed limit is set for both straight and curved edges. Furthermore, to avoid stop-and-go movements, the agents need to have a minimum speed with the exception of special locations in the taxiway network to fulfil `Wait` activities.

As consequence of the finite acceleration underlying the motion principles, agents are also not able to stop instantaneously. In turn, this violates a core assumption of SIPP and makes it incomplete, as noted by Ali and Yakovlev (2023). The minimum speed requirement at most edges and nodes further complicates this issue, illustrated in Fig. 4: the designated stand of a landing aircraft is still occupied for a while, i.e. a long goal-constraint is placed on its goal-node, indicated by the red shade at 0 m in the upper right plot. Since the aircraft is neither allowed to stop along the direct route to its stand, nor travel slower than the minimum speed, it

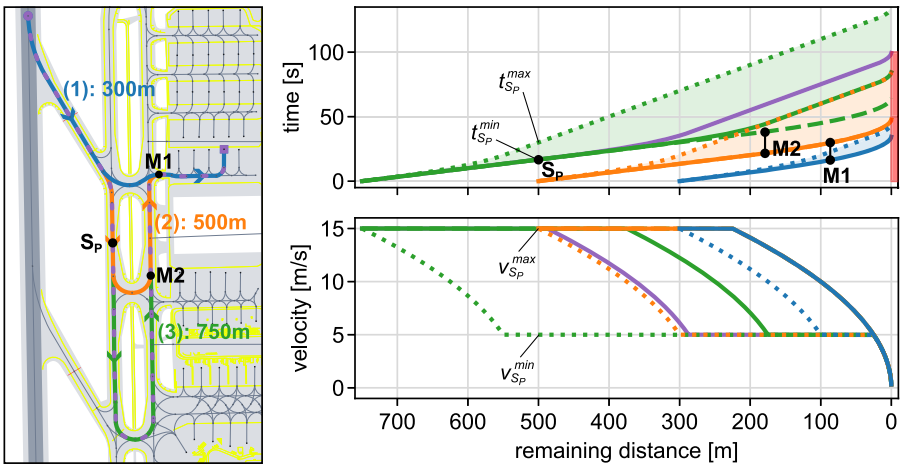

Fig. 4. Exemplary trajectories through layout (left) and state space (right) to a stand that is occupied till $t = 100\,\mathrm{s}$ resulting in a long goal constraint (red shade at 0 m).

must take a detour. In the low-level search, states along path (1) in blue, then along path (2) in orange, and finally along path (3) in green would be created. However, in vanilla SIPP each node would have a single safe time interval. Thus, the state of the orange path at the merge-location M1 would be discarded as its f-score is higher than the already existent state of the blue path at M1. Likewise, the state at M2 of the green path would not be added to its OPEN-list. As a consequence, the low-level search would fail to find a solution. To address this issue, we take the approach in Ali and Yakovlev (2023) further, and combine an agent configuration with a feasible motion interval (FMI) to distinguish states in SIMP:

Definition 9 (State definition in SIMP). A state S is defined by an agent configuration S_{cfg} and a *feasible motion interval* (FMI). The configuration comprises the current node v, next node v' (as direction), speed limit v_{lim} (depending on the previous and next edge), and the current activity α_k from the agent's activity sequence $\mathcal{S}$:

$$S_{\mathrm{cfg}} = (v,\, v',\, v_{\mathrm{lim}},\, \alpha_k) \tag{11}$$

The corresponding FMI of state S is defined as:

$$S_{\mathrm{FMI}} = (t_{\mathrm{S,min}},\, t_{\mathrm{S,max}},\, v_{\mathrm{S,max}},\, v_{\mathrm{S,min}}) \tag{12}$$

with the quickest feasible motion arriving at $t_{\mathrm{S,min}}$ with speed $v_{\mathrm{S,max}}$, and the slowest feasible motion arriving at $t_{\mathrm{S,max}}$ with $v_{\mathrm{S,min}}$. Given the minimally allowed speed v_{min},

$$v_{\mathrm{min}} \le v_{\mathrm{S,min}} \le v_{\mathrm{S,max}} \le v_{\mathrm{S,lim}} \tag{13}$$

When another state with identical configuration exists, and their FMIs overlap, the FMIs are adjusted to be temporally disjunct: for two states S and S^* with the f-scores $f_S < f_{S^*}$, the FMI of S^* is adjusted so that $t_{S^{*,min}} \ge t_{\mathrm{S,max}}$ and $v_{S^{*,max}} \le v_{\mathrm{S,min}}$. In case $t_{S^{*,max}} < t_{\mathrm{S,max}}$, the new state is discarded. If $f_S = f_{S^*}$, the state with the earlier arrival time is kept, and the FMI of the other state is adjusted accordingly.

In the following, we illustrate the use of FMIs with the example in Fig. 4. In its upper right plot, the arrival time interval of state S_{P} is visualised by the shaded green area between the solid and dotted lines representing the earliest arrival time $t_{\mathrm{S_P,min}}$ and latest arrival time $t_{\mathrm{S_P,max}}$, respectively. Likewise, the lower right plot shows the speed range of S_{P} between $v_{\mathrm{S_P,max}} = 15\,\mathrm{m/s}$ (solid green line) and $v_{\mathrm{S_P,min}} = 5\,\mathrm{m/s}$ (dotted green line).

At the merge location M1, the FMI of the orange state does not overlap with that of the blue state, and is directly added to OPEN. At M2, the FMIs of the orange state $S_{\mathrm{o,M2}}$ and green state $S_{\mathrm{g,M2}}$ are respectively defined as

$$S_{\mathrm{o,FMI}} = (t_{\mathrm{S,min}},\, t_{\mathrm{S,max}},\, v_{\mathrm{S,max}},\, v_{\mathrm{S,min}})_{\mathrm{o}} = (23\,\mathrm{s},\, 45\,\mathrm{s},\, 13\,\mathrm{m/s},\, 5\,\mathrm{m/s})$$
$$S_{\mathrm{g,FMI}} = (t_{\mathrm{S,min}},\, t_{\mathrm{S,max}},\, v_{\mathrm{S,max}},\, v_{\mathrm{S,min}})_{\mathrm{g}} = (38\,\mathrm{s},\, 95\,\mathrm{s},\, 13\,\mathrm{m/s},\, 5\,\mathrm{m/s})$$

which overlap on the interval between (38 s, 45 s). The g-score of the green state must be higher since its path is longer. Therefore, $f_{\mathrm{S_o,t=38s}} < f_{\mathrm{S_g,t=38s}}$. As a result, the motion of the green state must be adjusted to create a temporally disjunct FMI: its adjusted earliest arrival time $t_{\mathrm{S_g,min}} \cong t_{\mathrm{S_o,max}} = 38\,\mathrm{s}$ with a speed of $v_{\mathrm{S_g,max}} = v_{\mathrm{S_o,min}} = 5\,\mathrm{m/s}$. Subsequent states are created in analogy to keep the FMIs temporally disjunct. In Fig. 4, this is visualised by the solid green line matching the dotted orange line.

However, when following the resulting green trajectory (solid line), the aircraft still arrives too early at the goal. Thus, the motions along the previous edges must be adjusted so that the goal is reached just after the end of the goal constraint, as illustrated by the purple trajectory along path (3). We use this case to introduce two schemes in SIMP to anticipate restrictions as well as to repair infeasible motions:

Definition 10 (Anticipation and backtracking schemes in SIMP). Let S denote the currently explored state.
The *anticipation scheme* in SIMP sets auxiliary constraints for the motion over the current edge $\epsilon = (v, v') \in \mathcal{E}$ to ensure that time and speed restrictions at upcoming edges can still be fulfilled. These constraints are state-dependent, i.e. only valid for state S.
The purpose of the *backtracking scheme* is to repair an infeasible motion over the current edge ϵ necessitating a lower speed or later arrival time at S. To this end, it recursively backtracks to a preceding state S_{pre} from which a feasible motion over the sub-path $\mathcal{P}_{\text{sub}} = \langle \epsilon_{\text{pre}}, \cdots, \epsilon \rangle$ exists, allowing to create a feasible successor S^* of S.

To illustrate the backtracking scheme, let us assume that a state at the beginning of the final edge towards the goal is explored, leading to an infeasible motion since the required time $t_S = 100\,\text{s}$ and speed $v_S = 0\,\text{m/s}$ cannot be satisfied. The backtracking scheme then iteratively backtracks to the parent state S_P at approximately 500 m remaining distance to the goal to create the purple trajectory (solid line). However, the anticipation scheme allows SIMP to create the purple trajectory directly from S_P, which avoids the recursive calculation of motion feasibility that is necessary in the backtracking scheme. To this end, it checks the upcoming edges for restrictions, finding the goal constraints at the end of paths (2) and (3). By checking whether the motion along the paths can satisfy all restrictions, SIMP then creates a successor at the goal, connected by intermediate states to S_P. Using the anticipation scheme reduces the runtime of SIMP by limiting the creation of states that would lead to an infeasible trajectory. Although SIMP primarily uses this scheme, it can only anticipate on those constraints that are valid for a node or an entire edge. We thus need the backtracking scheme to resolve infeasible motions that result from graph reservations defined only for a segment of an edge (see Appendix B.4 for further details).

We further minimise the amount of generated states in SIMP by defining a heuristic function that uses domain-specific circumstances to estimate the remaining cost to reach the goal location:

Definition 11 (Heuristic function in SIMP). Similar to A* and SIPP, the SIMP algorithm uses an admissible heuristic function $h(S)$ to estimate the remaining cost from a given search state S to one of the goal locations $G \in \mathcal{V}$ associated with the final activity α_f in the agent's activity sequence $\mathcal{S}$. $h(S)$ accounts for the current edge ϵ as travel direction, current speed v_S, speed restriction at the goal $v_{\text{lim,goal}}$, the set of remaining activities $\mathcal{S}_{\text{rem}} \subseteq \mathcal{S}$, and the vehicle category `cat` associated with the agent:

$$h(S) = h\left(\epsilon, v_S, v_{\text{lim,goal}}, \mathcal{S}_{\text{rem}}, \texttt{cat}\right) \tag{14}$$

The SIMP algorithm terminates if either a new plan is found, or no unexplored states are left, and the high-level search continues as described above. For in-depth descriptions as well as the respective pseudocodes of PBS and SIMP, the reader is referred to Appendix B.

6. Model evaluation

In this section, we evaluate the performance of the multi-agent system model for autonomous operations with centralised planning, with the main focus on its most important centralised planning component – the AS-MAMP algorithm. To this end, we use multiple scenarios on both a synthetic and a real-world airport layout (Section 6.1). First, we benchmark AS-MAMP with PBS variants and FCFS as high-level solvers (Section 6.2), combined with SIMP, SIPP, and A* with kinodynamic constraints as low-level solvers in AS-MAMP (Section 6.3). Next, we study the influence of key operational parameters (Section 6.4), analyse the effect of the planning window (Section 6.5), and assess its computational efficiency and scalability (Section 6.6). We then evaluate AS-MAMP using the flight schedule of two operational days at Amsterdam Airport Schiphol in Section 6.7, simulating both multi-engine taxiing (MET, see Section 2) and tug-enabled taxiing (TET). Finally, to stress-test the algorithm, we examine the maximal attainable runway throughput per hour for MET and TET operations on the synthetic layout, and link them to those reported for large European airports (Section 6.8).

6.1. Definition of test cases

In taxiway networks, potential bottlenecks may arise in different locations. To assess how AS-MAMP deals with them, we analyse the bottlenecks separately. To this end, we created the synthetic layout visualised in Fig. 5 which accentuates the bottlenecks, and define two schedules each with 7 flights in total as listed in Table 2.

In the scenario *Queuing*, the 7 aircraft depart from all available gates in the layout (see Fig. 5). To isolate the runway as bottleneck, **test case A** considers MET without pushback-operations and with engines being already started. To optimise the trajectory costs, the AS-MAMP algorithm must determine a suitable runway sequence so that the WTC-separation (see Table A.2) leads to minimal queuing delay before entering the runway (see Fig. 6). For **test case B**, pushback-operations and engine-start are included, congesting the bay

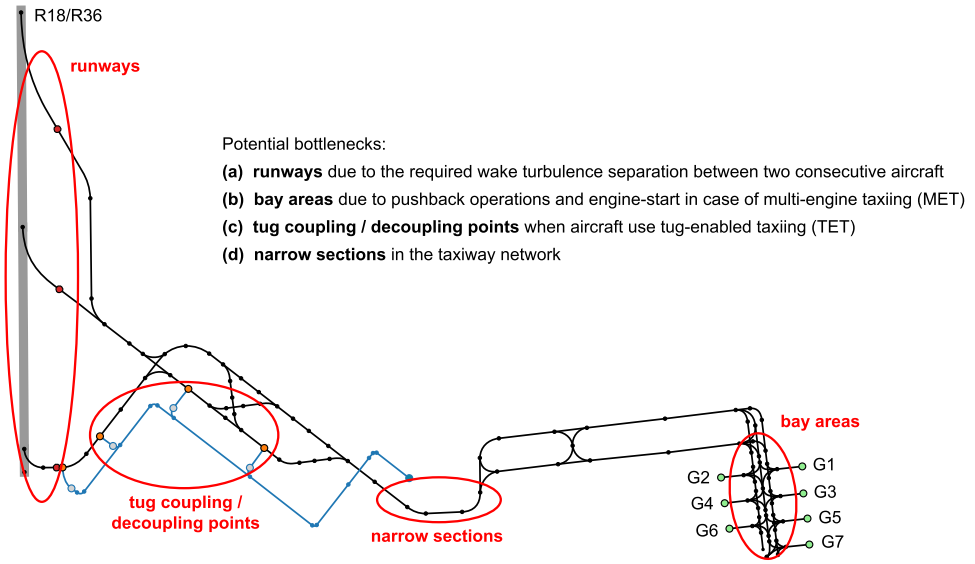

Fig. 5. Synthetic layout with potential bottlenecks marked by red circles that form the basis of the test cases.

Table 2
Flight schedules for the sensitivity analyses on the synthetic layout, with T as fictive reference time.

(a) Queuing scenario				
ID	Type	At	From	To
1	ICAO-C	$T + 0$ s	G1	R36
2	ICAO-E	$T + 20$ s	G2	R36
3	ICAO-C	$T + 40$ s	G3	R36
4	ICAO-E	$T + 60$ s	G4	R36
5	ICAO-C	$T + 80$ s	G5	R36
6	ICAO-E	$T + 100$ s	G6	R36
7	ICAO-C	$T + 120$ s	G7	R36
(b) Crossflow scenario				
ID	Type	At	From	To
1	ICAO-E	$T + 0$ s	R18	G2
2	ICAO-E	$T + 60$ s	R18	G4
3	ICAO-E	$T + 120$ s	R18	G6
4	ICAO-C	$T + 0$ s	G1	R36
5	ICAO-C	$T + 40$ s	G3	R36
6	ICAO-C	$T + 80$ s	G5	R36
7	ICAO-C	$T + 120$ s	G7	R36

area: decoupling from the pushback-truck takes 60 s, and engine-start either 180 s (ICAO-C) or 360 s (ICAO-E). Thus, AS-MAMP must find a suitable pushback-sequence. Note that agents can either hold at their gate, or on the bay area after pushback. To investigate the bottleneck of tug-decoupling locations, **test case C** uses the same setup as case A but with tug-enabled taxiing. AS-MAMP has to choose one of the four decoupling locations and must account for the decoupling duration of 120 s. In **test case D**, we use the *Crossflow* scenario with three large arriving and four medium-sized departing aircraft, without considering pushback-operations, to isolate the narrow section in the layout as bottleneck. Therefore, the AS-MAMP algorithm must optimise the sequence in which the departing and arriving aircraft pass through it.

First, AS-MAMP is benchmarked against PBS variants as well as alternative low-level solvers based on these synthetic test cases using a *single* planning round. However, to evaluate the influence of its windowed planning (Definition 3), continuous schedules are needed. The schedule generation is outlined next to the respective assessment in Section 6.5. Moreover, in Section 6.7, we examine the performance of high-level variants in AS-MAMP on the real-world layout of Amsterdam Airport Schiphol using the flight schedules of two of the busiest days at the airport to date. Our previous studies provide more information as well as the required data comprising the layout, runway configurations, and flight schedules to test both MET and TET scenarios (von der Burg and Sharpanskykh, 2025, 2024). In Section 6.8, we stress-test the AS-MAMP algorithm by evaluating the potential runway throughput on the synthetic layout with continuous schedules for departing flights of MET (test case A) and TET operations (test case C).

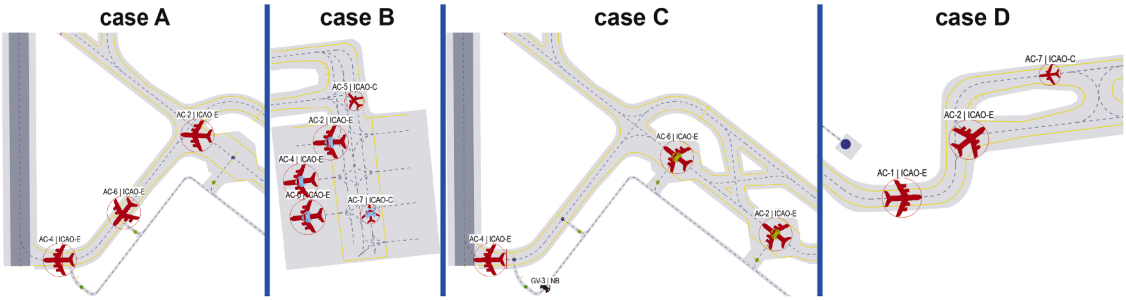

Fig. 6. Simulation snapshots illustrating the bottleneck of each test case on the synthetic layout.

Table 3
Default values for the key parameters of AS-MAMP.

parameter	aircraft	tugs	unit
maximum speed $v_{\max}$	15	11.8*	m/s
maximum turn speed $v_{\text{turn,max}}$	5	5	m/s
minimum speed $v_{\min}$	1.5	1.5	m/s
acceleration acc	0.25	0.25	m/s^2
deceleration dec	-0.75	-0.75	m/s^2
cost factor c_d	0.1	0.1	s/m
cost factor c_{prio}	1	1	–
safety factor for path edges σ_{path}	3	3	–
safety factor for neighbouring edges σ_{NE}	1	1	–
positioning tolerance δ_{pos}	0	0	m

*: as current limit of the reference TaxiBot system (TaxiBot, 2023)

The values for key AS-MAMP parameters such as the kinodynamic properties, cost factors, safety factors, and positioning tolerance are provided in Table 3. The maximum speeds for straight and curved segments were determined through expert interviews and current technological limitations of tugs. To evaluate the capabilities of the AS-MAMP algorithm, we chose rather low and differing absolute values for the acceleration and deceleration of the agents, which have a large influence on the KPIs (see Section 6.4). As specified in Appendix B.3, the safety radii along an agent's path and the neighbouring edges are defined as factors in relation to the shape-radius (see Table A.1) of the leading aircraft. Their respective values listed in Table 3 resulted from expert interviews.

As operational key performance indicators (KPIs), we mainly use the sum of taxi times t_{taxi} and taxi distances d_{taxi} of all flights. However, to show the relative increase in taxi time or distance between different parameter settings, the KPIs are normalised by the lowest value of the tested set. When assessing the real-world layout, we use the taxi time distributions per runway. Furthermore, we track the number of collisions between vehicles during the execution of the planned routes, and ensure that none exist. As search-related KPIs, we use the cost, computational runtime, assigned priority pairs, as well as the amount of explored high-level nodes and low-level states. Additionally, we use the number of operationally obsolete stops exceeding 10 s to assess the secondary planning objective of mitigating stop-and-go movements (see Section 4.2).

6.2. Benchmarking the high-level search

To benchmark the high-level search of the AS-MAMP algorithm, we compare its *depth-first* search against PBS variants such as Greedy PBS (Chan et al., 2023) without and with the use of induced constraints (IC), as well as the first-come-first-served (FCFS) scheme often used in more application-oriented studies. To establish an optimal priority ordering, we use a *best-first* search in PBS. Table 4 provides a comparison of these when applied to the four test cases. First, we analyse the results when running PBS without IC, followed by searching with IC.

As expected, while the best-first search yields the highest solution quality, it requires a large number of explored PBS nodes, total sum of SIMP states as well as a long runtime. Substantially less time is required for the depth-first and greedy searches to find a solution, with only a fraction of PBS nodes being explored. Since the runtime correlates with the total sum of SIMP states, the greedy search terminates faster (cases C and D) or slightly faster (cases A and B). Furthermore, due to picking the child-node with the lowest number of remaining conflict pairs, it finds solutions requiring less priority relations. However, in comparison to the solution cost of the best-first search, its solution quality varies more between the cases than with a depth-first search. With a pre-defined priority sequence, the FCFS approach results in the fewest explored SIMP states, and thus the lowest runtimes for all test cases. However, in contrast to the depth-first and greedy searches, it also leads for all cases to suboptimal solutions, ranging between 3 % (case C) to 30 % (case B).

When running PBS with IC, recall that conflict detection returns the conflict pair with the highest number of induced constraints, likely selecting a different pair than running PBS without IC. Despite returning the optimal priority ordering among those explored, also the best-first search does not explore all possible priority pair sequences. Therefore, when using IC, the number of PBS nodes,

Table 4
AS-MAMP benchmark using high-level variants and low-level alternatives, with cost ratios listed w.r.t. "best-first".

	case	cost	approx. runtime	prio pairs	high-level nodes	low-level states	extra stops (> 10 s)
AS-MAMP: high-level variants							
best-first	A	4718.1	200	11	571	357,103	0
	B	7601.5	43	13	247	78,847	0
	C	6089.4	580	11	1529	1,083,529	0
	D	5172.9	110	11	452	206,106	0
depth-first	A	100 %	3	12	25	10,686	0
	B	100 %	3	13	27	8067	0
	C	102 %	7	15	31	16,658	0
	D	105 %	4	14	24	9386	0
greedy	A	100 %	3	9	19	8875	0
	B	126 %	3	8	17	6496	0
	C	113 %	4	9	19	8356	0
	D	100 %	3	10	18	6217	0
FCFS	A	110 %	1	21	1	2508	0
	B	130 %	1	21	1	1818	0
	C	103 %	1	21	1	2016	0
	D	113 %	1	21	1	1895	0
AS-MAMP: high-level variants with induced constraints (IC)							
best-first with IC	A	100 %	240	13	627	416,354	0
	B	100 %	87	13	415	145,138	0
	C	100 %	750	11	1757	1,348,195	0
	D	100 %	110	12	468	198,606	0
depth-first with IC	A	100 %	5	13	27	11,829	0
	B	100 %	7	13	27	7930	0
	C	114 %	14	18	37	20,479	0
	D	107 %	19	13	53	24,077	0
greedy with IC	A	101 %	5	10	21	10,875	0
	B	102 %	9	10	21	10,400	0
	C	112 %	6	10	21	11,595	0
	D	100 %	4	10	18	6829	0
AS-MAMP: alternative low-level solvers							
best-first + SIPP	A	90 %	110	15	1293	445,462	14
	B	91 %	15	12	217	42,103	5
	C	90 %	250	16	2245	1,025,266	12
	D	86 %	32	12	455	141,674	7
depth-first + SIPP	A	90 %	2	16	33	7336	14
	B	100 %	1	12	25	4949	6
	C	94 %	2	15	31	9607	15
	D	87 %	1	10	18	4564	8
greedy + SIPP	A	93 %	1	7	15	3161	16
	B	109 %	2	9	19	5257	5
	C	101 %	2	9	19	5662	24
	D	87 %	2	10	18	4948	7
FCFS + SIPP	A	97 %	< 1	21	1	1107	19
	B	118 %	< 1	21	1	1094	7
	C	93 %	< 1	21	1	1316	19
	D	96 %	< 1	21	1	1188	12
depth-first + Spacetime-A*	A	105 %	220	12	25	307,244	0
	B	105 %	140	13	27	218,002	0
	C	118 %	470	16	33	590,362	0
	D	107 %	110	13	24	225,988	0
greedy + Spacetime-A*	A	110 %	180	10	21	317,768	0
	B	119 %	86	8	17	187,079	0
	C	110 %	440	11	23	542,321	0
	D	107 %	79	11	21	185,523	0
FCFS + Spacetime-A*	A	116 %	35	21	1	69,083	0
	B	128 %	25	21	1	52,343	0
	C	109 %	52	21	1	69,815	0
	D	122 %	28	21	1	60,374	0

total sum of SIMP states, and runtime change in general. Furthermore, the conflict free solutions contain a different sequence of priority pairs, and the final count of pairs may differ. While "best-first with IC" returns identical solution costs for all test cases, the resulting priority sequences indeed differ w.r.t. "best-first". Likewise, using induced constraints in the depth-first and greedy searches leads to different solution costs. For test cases C and D, the solution quality degrades using "depth-first with IC", while it improves for cases B and C when IC is used in the greedy search. Nonetheless, the returned solution quality of "greedy with IC" continues to fluctuate more than that of the depth-first search. Regarding runtime, using IC seems to increase the number of explored PBS nodes and SIMP states, leading to higher computational times.

Overall, the comparison among the different high-level search strategies without and with IC suggests that the conflict selection procedure can be further optimised to combine the advantages of a fast greedy search with a more consistent solution quality returned by a depth-first search. The runtime could also be reduces with effective partial priority relations handed to PBS, as confirmed by the low runtime of the FCFS approach. In Section 6.7, we complement the evaluation of the high-level variants by testing AS-MAMP on a real-world airport layout.

6.3. Benchmarking the low-level search

To benchmark the low-level search of the AS-MAMP algorithm, we run AS-MAMP with two alternative low-level solvers, each necessitating different additional assumptions:

- SIPP: extends the regular SIPP states to accommodate activity-IDs; motions are instantaneously accelerated and assume constant speed according to the edge's speed limit; agents can hold on nodes between the motions.
- Spacetime-A*: extends the regular A* states to accommodate the arrival time, arrival speed, and activity-IDs; arrival speeds are discretised, with an increment of $1.5\,\mathrm{m/s}$; agents can only hold at nodes after finishing a `Wait` activity, with the hold-time discretised into $10\,\mathrm{s}$ increments, and another hold-state only created upon exploring the previous one.

Table 4 lists their results when applied to the four test cases in combination with the high-level variants from the previous section. In comparison to SIMP, the adapted SIPP solver requires less low-level states to find a solution, but often needs to explore more high-level nodes. Nonetheless, it has a lower runtime for all test cases across all examined high-level variants. Furthermore, SIPP appears to return a better solution quality in all cases. However, this is a result of the instantaneously accelerated motions, leading to trajectories that are impossible to be executed in real-world airport operations. Moreover, the computed trajectories require frequent stop-and-go movements with hold-times exceeding $10\,\mathrm{s}$. In contrast, the Spacetime-A* solver produces more realistic trajectories due to the same kinodynamic agent properties applied as in SIMP. However, its search is highly inefficient, requiring a large amount of low-level states to find solutions due to its large search space. To conclude, the low-level benchmark emphasises that SIMP combines the advantages of returning realistic and operationally efficient 4D trajectories from a large search-space with the low computational runtimes of SIPP-based search.

6.4. Influence of key operational parameters

In this subsection we analyse how the acceleration and deceleration, minimum speed, cost-factor c_d in the cost function in SIMP, and positioning tolerance δ_pos influence the KPIs. To this end, we simulate the four synthetic test cases with changing values for each of these parameters. All simulations are run with a best-first search without IC to yield the highest solution quality.

The choice of acceleration and deceleration has a strong impact on the taxiing duration, as shown exemplary for case A in Fig. 7: the higher the acceleration and/or deceleration, the faster the aircraft are able to complete their taxiing. In contrast, the influence on the travelled distance is minimal. This trend holds for the other cases as well. Thus, with the values of $acc = 0.25\,\mathrm{m/s^2}$ and $dec = -0.75\,\mathrm{m/s^2}$, approximately $10\,\%$ higher taxi times result in comparison to the highest considered absolute value of $2\,\mathrm{m/s^2}$ for both parameters.

Since the maximum speeds are operational and/or technological limits, we did not vary them. Nonetheless, it is self-evident that changing these values will affect the taxiing duration. When varying the minimum speed v_min between $0.5\,\mathrm{m/s}$ to $5.0\,\mathrm{m/s}$, the taxi time increased by more than $7\,\%$ for $v_\mathrm{min} \geq 2.0\,\mathrm{m/s}$ in test case D. While case C showed a small increase up to $1\,\%$ at $v_\mathrm{min} = 5.0\,\mathrm{m/s}$, test cases A and B were unaffected by the different minimum speeds. This suggests that the AS-MAMP algorithm is capable of creating trajectories without the need for agents to stop at arbitrary locations to resolve conflicts, as long as $v_\mathrm{min} \leq 1.5\,\mathrm{m/s}$. The operational benefit is that pilots have to apply breakaway-thrust less often, resulting in fuel savings and reduced engine-blasts that lower the associated risks.

Regarding the cost function in SIMP (see Eq. (3)) our hypothesis was that small values of c_d have a positive impact on the KPIs: while for $c_\mathrm{d} = 0$ the algorithm may choose long detours to save a few seconds in taxi time, slightly including the travelled distance in the objective function should mitigate these. In contrast, high values for c_d should result in shorter paths but increasingly long taxi times, since the secondary objective of optimising for distance may outweigh the primary one of low taxi times. When varying c_d between $0\,\mathrm{s/m}$ to $0.5\,\mathrm{s/m}$, the hypothesis is supported: setting $c_\mathrm{d} = 0$, the taxi distance increases by $1.7\,\%$ for test case A and $3.1\,\%$ for test case C, while cases B and D are unaffected. For $c_\mathrm{d} = 0.5$, especially case D is affected: both taxi time and taxi distance increased by $3.9\,\%$ and $0.8\,\%$, respectively. For cases A and C, the taxi distances were unaffected, and the taxi times showed a small increase of $1.7\,\%$ (case A) and $3.1\,\%$ (case C).

Increasing the positioning tolerance δ_pos of each agent from $0\,\mathrm{m}$ to $50\,\mathrm{m}$, showed a negligible influence on test cases A and B. In contrast, the taxi time abruptly increased to $> 6\,\%$ for $\delta_\mathrm{pos} >= 20\,\mathrm{m}$ in test case C, and $> 4\,\%$ for $\delta_\mathrm{pos} >= 10\,\mathrm{m}$ in test case D. In

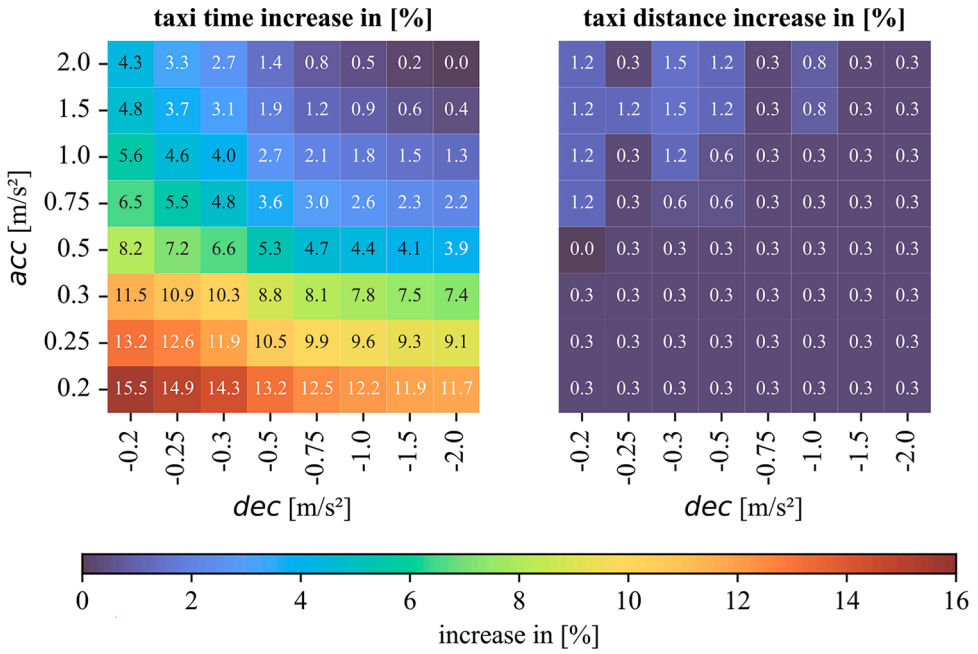

Fig. 7. Sensitivity analysis of case A depicting how acceleration *acc* and deceleration *dec* influence the taxi time and distance, shown as increase in [%] w.r.t. the lowest sum over all flights.

Table 5
Extending the flight schedules from Table 2: each base-schedule is repeatedly appended with a time shift of t_{cycle}.

t_{cycle}	replications of schedule in Table 2(a)	resulting flights
15 min	6	42
12 min	8	56
10 min	9	63
9 min	10	70
8 min	12	84

both cases, the taxi time further increased with rising positioning tolerances adding around 2 % at $\delta_{\text{pos}} = 50$ m, while the increase in taxi distance remained < 0.8 %. This suggests that higher positioning tolerances likely impact confined passages in airport layouts. Nonetheless, the overall impact on the KPIs should be reasonably small for $\delta_{\text{pos}} < 20$ m.

6.5. Influence of planning window

As the flight schedules of the test cases described in Section 6.1 contain merely 7 flights, a single planning round sufficed to route all aircraft in the simulations conducted up to this point. To assess how the AS-MAMP algorithm deals with continuous schedules on the synthetic layout and thus the necessity to re-plan, we extended the flight schedules in Table 2 to over 90 minutes by repeating the seven flights: for each replicate, the parameter t_{cycle} is added to the fictive reference time T. Table 5 provides an overview of the number of replications and the resulting flights of the flight schedules created for different values of t_{cycle}.

For the analysis, we varied w_{p} between 10 min to 120 min and h_{p} between 20 % to 80 % in relation to w_p. With the number of agents exceeding the feasibility of a best-first search, we used a *best-of* depth-first search as slight modification in PBS: after finding the first solution, we store it, let PBS pick the lowest-cost search-node, and continue the depth-first search for an alternative solution. This procedure is repeated until a maximum number of solutions (in this case 24) is found, returning the one with the lowest cost.

For test case A, the average taxi time remained unaffected. Likewise, for test cases C and D, the average taxi time did not change significantly for all considered values of w_{p} and h_{p} when $t_{\text{cycle}} \leq 10$ min. With $t_{\text{cycle}} > 10$ min, the simulations showed stronger taxi time variations across varying w_{p} and h_{p} while increasing in general for lower t_{cycle}. This is further discussed in Section 6.8. Based on the chosen t_{cycle} values, all simulations of test case B failed: subsequent aircraft were spawned at stands still occupied with holding aircraft, due to the confined bay area of the synthetic layout combined with the long engine-start times of up to 6 min. However, a separate analysis with $t_{\text{cycle}} \geq 22$ min showed results similar to those of the other test cases.

Regarding the total planning time measured as sum over all planning-rounds, the expected trend is observed for all test cases, aligning with the observations from our previous study (von der Burg and Sharpanskykh, 2025). For smaller w_{p}, AS-MAMP has to

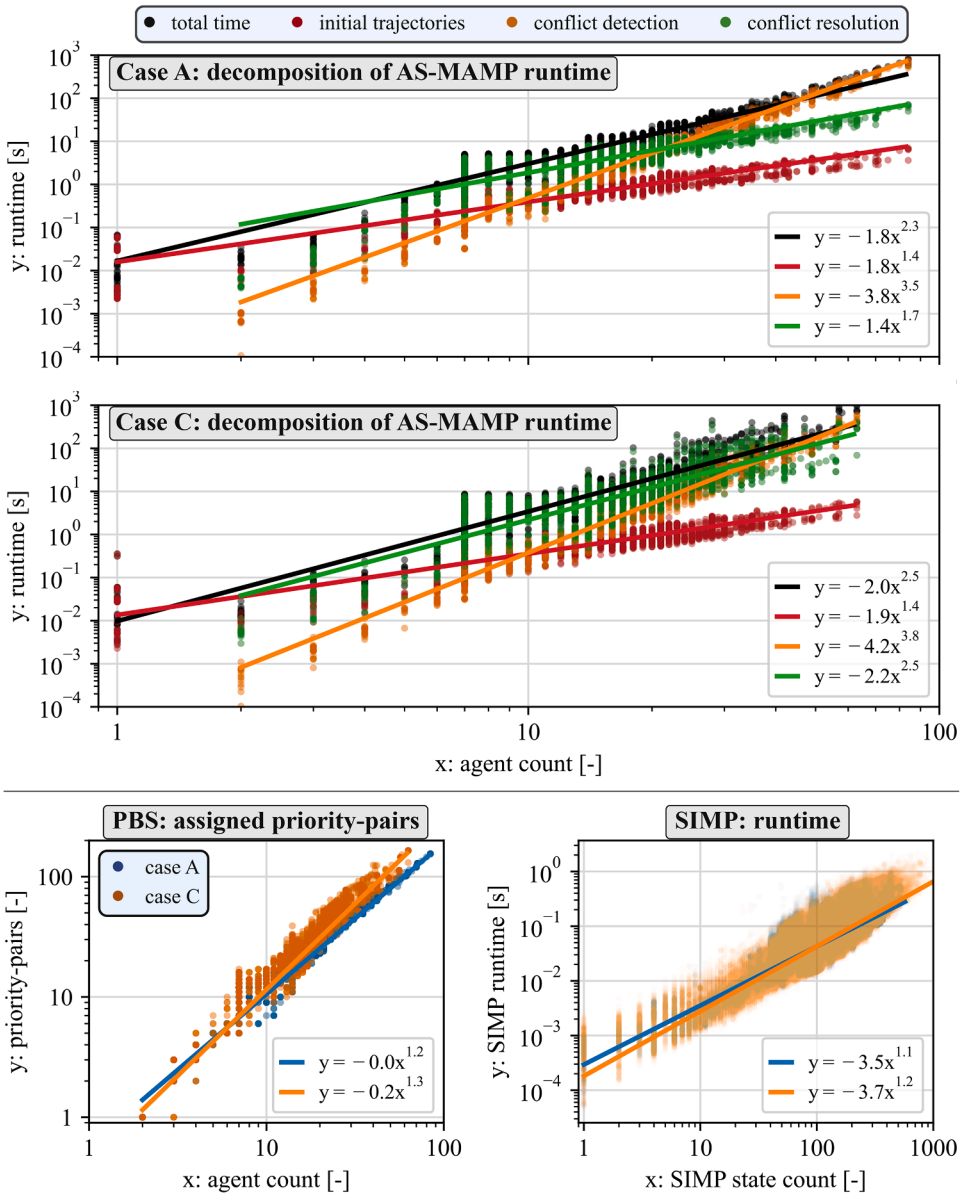

Fig. 8. AS-MAMP computational performance based on 200 simulations per test cases A and C, resulting in total in 3854 planning rounds with approx. 695 k SIMP-calls.

assign fewer priority-relations since conflicts occurring beyond w_p are ignored, reducing the overall runtime. Regarding h_p, more frequent replanning leads to an increase in total planning time. In summary, the windowed planning approach greatly improves the runtime performance of the AS-MAMP algorithm without sacrificing its solution quality.

6.6. *Computational efficiency and scalability of AS-MAMP algorithm*

Using the 200 simulations conducted per test cases A and C for the sensitivity analysis of the continuous schedules presented above, we study the computational efficiency and scalability of the AS-MAMP algorithm in this subsection. The simulations result in 3854 planning rounds in total. Per planning round, the traffic load as well as the traffic situation vary, depending on the time point of replanning determined by w_p and h_p of the respective simulation. For the following analysis, we use the performance data of the first conflict-free solution that PBS found in each of the planning rounds, visualised through multiple log-log plots in Fig. 8.

We can decompose the total time that the AS-MAMP algorithm requires to find a conflict-free solution into three component runtimes: for creating the initial trajectories in the PBS root-node, for detecting conflicting agent-pairs throughout the PBS search, and for subsequently resolving each conflict by invoking SIMP. The upper part of Fig. 8 shows this decomposition in dependence

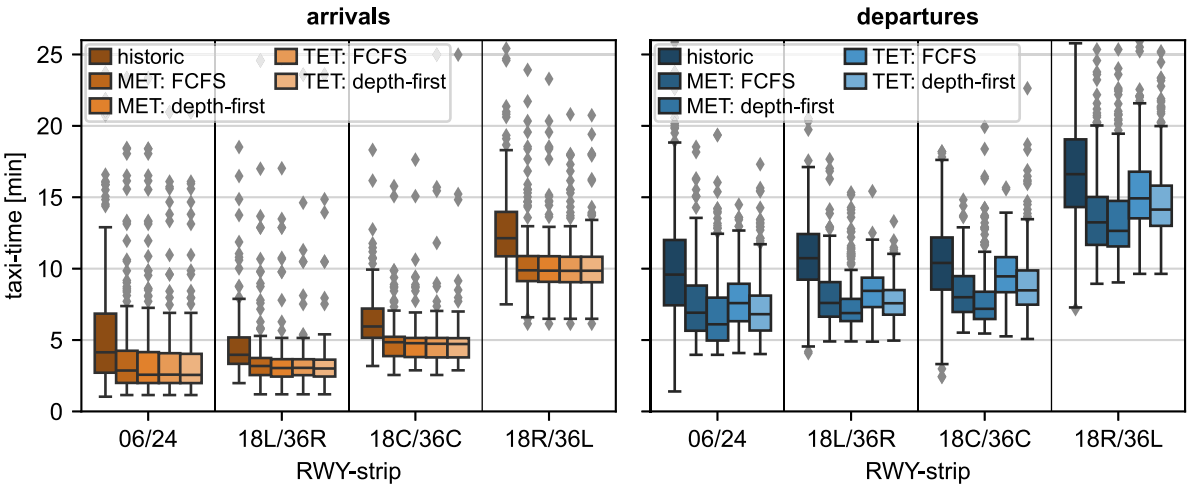

Fig. 9. Box-and-whisker plots per runway for both inbound and outbound flights, showing the taxi time distributions of historical data, as well as simulations of multi-engine taxiing (MET) and tug-enabled taxiing (TET) operations using either FCFS or depth-first search as high-level in AS-MAMP.

of the size of the agent population across all planning rounds for case A and case C, respectively. For both test cases, finding the initial trajectories has almost a linear time complexity, matching the expectation following from calling SIMP n-times to find n agent trajectories. The effort to detect conflicts between the trajectories increases with the number of agents to be routed concurrently. For conflict detection, we use an exhaustive search as noticeable in its pseudocode provided in Algorithm 2 in Appendix B.5. This approach leads to an almost quartic time complexity, resulting in the largest increase in the AS-MAMP runtime for growing agent populations. Thus, for a large number of agents, the time required by the conflict detection mechanism predominantly drives the total runtime to find a conflict-free solution.

While the conflict resolution for test case A has an almost linear time complexity, it increases beyond a quadratic time complexity for case C, and varies more between planning rounds with equal agent population. As shown on the bottom left in Fig. 8, a comparison of the necessary number of assigned priority-pairs in PBS to find a conflict-free solution reveals a similar trend: for equal agent populations, test case C shows higher values that vary more between planning rounds. In general, this means that more PBS nodes need to be explored with SIMP being called more often, leading to an increase in total time.

To generate initial as well as updated agent trajectories across the 400 simulations, the SIMP algorithm is called approx. 695 k times in total. As visualised in the bottom right plot in Fig. 8, it has an almost linear time complexity over the number of generated states, regardless of the test case. Furthermore, it requires approx. 120 states and 0.07 s to calculate an agent trajectory that adheres to the issued activity sequence and constraints set for the agent's route on the exemplary layout.

6.7. Testing AS-MAMP on real-world airport layout

Turning to a real-world airport layout, we examine the scalability of the AS-MAMP algorithm based on previous operational studies as mentioned in Section 6.1. In the following, we evaluate the different high-level variants benchmarked in Section 5.1 for both MET and TET operations, based on simulations with $w_p = 20$ min and $h_p = 10$ min. Figs. C.1 and C.2 provide the resulting taxi time distributions as boxplots per runway for both arriving and departing flights, respectively for MET and TET operations. Furthermore, the taxi time distributions of the high-level searches "depth-first", "greedy with IC", and "FCFS" are compared against the historical operations in Fig. 9.

For inbound flights, the different PBS variants show almost identical taxi time distributions across all runways as well as between MET and TET (used only by outbound flights), while FCFS leads to slightly longer taxiing durations. Nonetheless, in comparison to the historical operations, the centralised path planning in AS-MAMP produces consistently lower and less varying taxi times. For outbound flights, the depth-first search outperforms the other variants. Except for FCFS, the alternative high-level variants produce similar distributions, yet with slightly elevated median or interquartile range marked by the boxes in the three figures. In contrast, the FCFS variant results in higher average taxi times with larger variations between agents for both MET and TET operations. However, similar to inbound flights, any of the tested high-level variants shows an overall reduction in taxi time in comparison to the historical operations.

To examine the computational runtime and scalability of AS-MAMP on real-world layouts, we use the data of multiple simulations with differing values of the planning window. The three component runtimes resulting from these simulations have a similar time complexity than those visualised in Fig. 8 (the respective log-log plots are enclosed in Fig. C.3 for reference). The resulting overall runtime is shown in Fig. 10 for both MET and TET operations simulated with the depth-first search and FCFS approach as high-level variants. The latter has almost a linear time complexity with increasing agent populations, and thus scales better than any of the tested

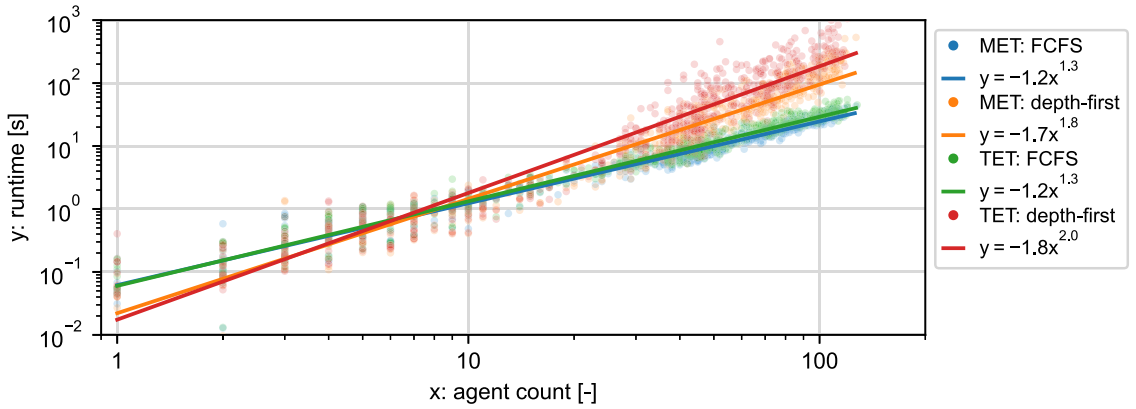

Fig. 10. Computational runtimes of planning rounds of simulations of multi-engine taxiing (MET) and tug-enabled taxiing (TET) operations using either FCFS or depth-first search as high-level in AS-MAMP.

PBS variants. Nonetheless, the approximately quadratic time complexity of the depth-first search indicates that AS-MAMP scales in general well with respect to both the size of the agent population as well as the complexity of the airport environment.

To summarise, the depth-first without induced constraints showed the best performance across the simulations carried out on the synthetic as well as the real-world layout. In comparison, the FCFS approach often used in application-oriented studies resulted in longer average taxi times of up to 30 % for the synthetic test cases, and 10 % for the real-world scenarios. Moreover, AS-MAMP is more robust than a FCFS approach which more frequently failed to return a solution, especially in high traffic demands. Nonetheless, based on the conducted runtime analyses, the current computational efficiency of AS-MAMP leaves room for improvements, which we further discuss in Section 7.

6.8. Stress-test: potential runway throughput

In this part, we focus on the potential maximal runway throughput per hour to demonstrate the applicability of AS-MAMP for busy airports. To this end, we fix $w_p = 30\,\text{min}$ and $h_p = 15\,\text{min}$, vary t_{cycle} from 15 min to 7.5 min, and compare multi-engine taxiing (MET, test case A) to tug-enabled taxiing (TET, test case C). Fig. 11 shows the average taxi times, the maximal runway throughput, and maximal occupancy rate for increasing traffic loads (i.e. lower t_{cycle}) for both test cases as well as either FCFS or depth-first search used as high-level in AS-MAMP.

For each of the four scenarios, the taxi distance remains unaffected across the tested traffic loads, while the taxi time increases significantly for smaller t_{cycle}, indicating a rising congestion. However, this also maxes out the achievable runway throughput within 60 min as well as the occupancy rate of the four scenarios. While the simulations of MET operations reach the limit of the occupancy rate of 100 % irrespective of either a 15 min or 60 min frame, the depth-first search results in a higher maximal hourly throughput of 54 flights, with FCFS maxing out at 49 flights.

For TET operations (test case C), the occupancy rate for a 15 min frame shows that its highest value of around 98 % is approached for $t_{\text{cycle}} \geq 10\,\text{min}$, matching the increase in taxi time. For lower traffic loads, the TET case consistently shows higher runway occupancy rates, while having identical runway throughput compared to the MET case. The hourly occupancy rates are in general higher for FCFS, limiting its achievable runway throughput to 43 flights in comparison to 47 flights with the depth-first search. Overall, this means that the AS-MAMP algorithm cannot optimise the runway sequence of TET as much as in the MET case.

This is also reflected in the most congested TET scenario with $t_{\text{cycle}} = 7.5\,\text{min}$: as visualised in Fig. 12 for a depth-first search, aircraft queue almost until the bay area when the simulation crashes, around one hour into the 90 min flight schedule. Also the choice of simulation parameters plays a role: with $v_{\text{min}} = 1.5\,\text{m/s}$, the AS-MAMP algorithm eventually runs out of route options for the agents, and thus fails to find a solution. In contrast, without the necessity to decouple, minimal queuing occurs in test case A.

As mentioned above, using a depth-first search, the chosen set of parameters results in a maximal runway throughput within 60 min of 54 and 47 takeoffs for MET and TET, respectively. To put this into perspective, Table 6 lists the maximal runway throughput of all possible runway configurations for the five largest airports in Europe (EUROCONTROL, 2023): the highest throughput per runway is reached by London Heathrow with 54 departing flights per hour. Therefore, when aircraft use MET, the multi-agent system is capable of matching or exceeding the maximal runway throughput seen at these large European airports. For TET, the runway throughput in the simulated setting still exceeds those of most European airports. Note that the potential throughput can be increased when the taxiway network is adapted: with either better accessible or more decoupling locations, the AS-MAMP algorithm would have more options to optimise the takeoff sequence and thus the potential runway capacity.

Regarding the runway throughputs and occupancy rates of simulations on the real-world layout, both MET and TET operations reached similar maximal values of around 46 hourly takeoffs with up to ≈ 80 % occupancy rates, irrespective of the high-level search in AS-MAMP as well as the runway configuration of the scenarios. In turn, we believe that this validates that the two cases analysed above are suitable stress-tests for AS-MAMP: all scenarios with $t_{\text{cycle}} \geq 10\,\text{min}$ resulted in higher occupancy rates than those found in

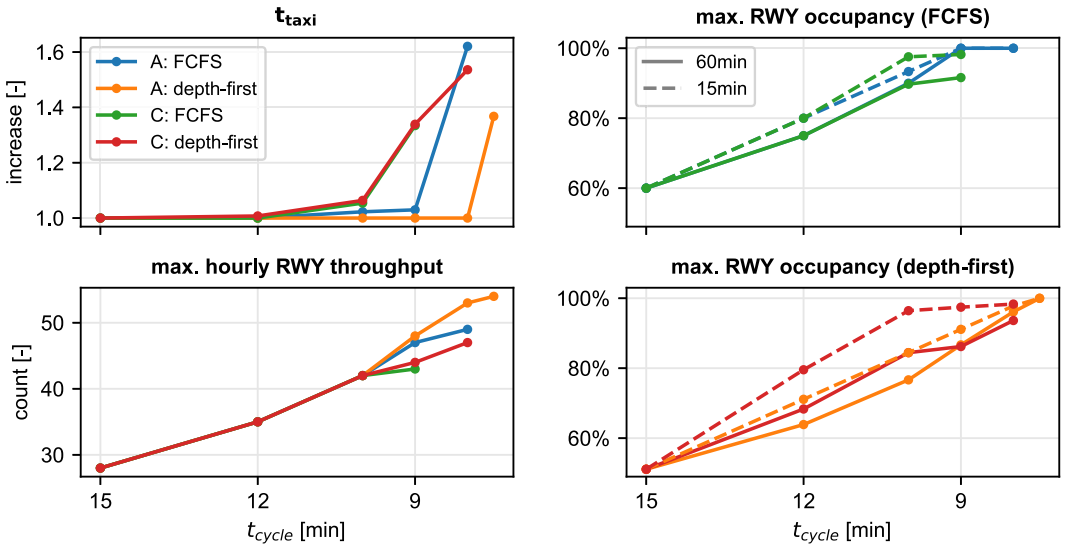

Fig. 11. KPIs w.r.t. t_{cycle} for continuous schedules of test cases A and C, comparing the high-level variants FCFS and depth-first.

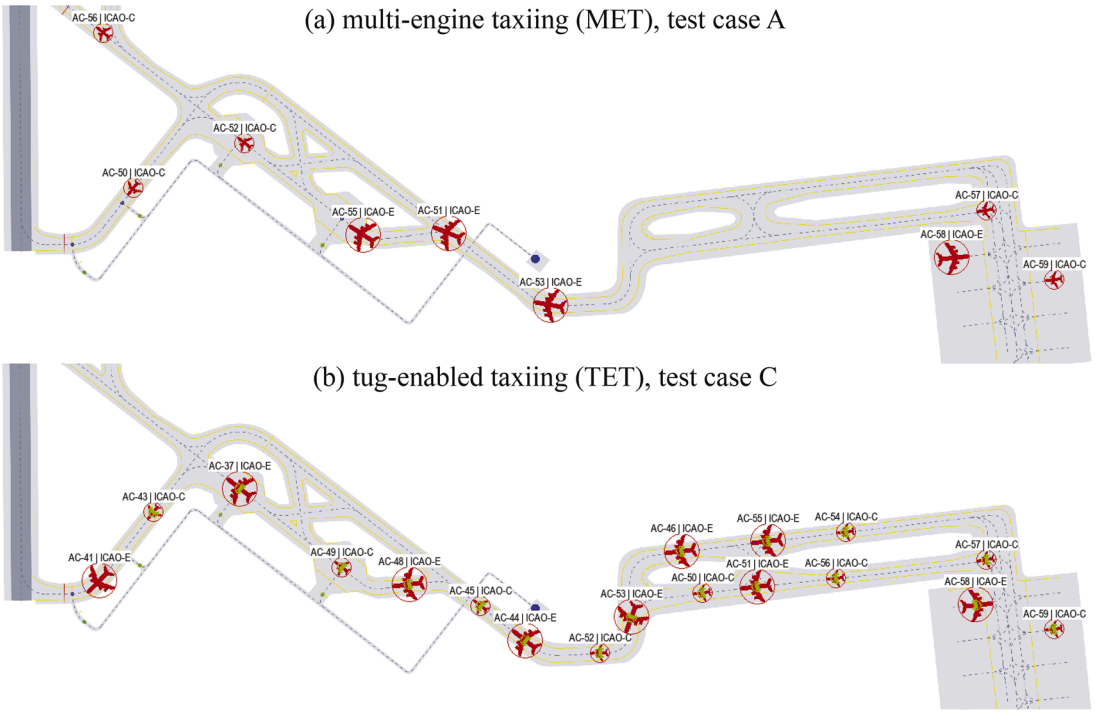

Fig. 12. Simulation snapshots at ≈ 60 min into continuous schedule with $t_{\text{cycle}} = 7.5$ min and depth-first search.

the historical flight schedules. This underscores that the AS-MAMP algorithm is able to coordinate ASM Ops in an efficient way, and only returns planned trajectories when these are indeed conflict-free.

7. Discussion

The results obtained in the simulation experiments presented above demonstrated the ability of the AS-MAMP algorithm to coordinate agent trajectories safely and efficiently, as backbone of a MAS model to study a next-generation concept for fully-automated surface movement operations with centralised planning. Despite the promising results, many real-world challenges and limitations

Table 6
Maximal runway throughputs and occupancy rates for departures within 60 min of the 5 largest European airports in (a), as well as scenarios simulated with the AS-MAMP high-level variants FCFS in (b) and depth-first search in (c).

(a) historical data (runway occupancy rate not reported).					
airport code	EDDF (FRA)	EHAM (AMS)	LFPG (CDG)	EGLL (LHR)	LEMD (MAD)
departure runways	2	1	2	1	2
max. throughput	68	46	75	54	70

(b) simulated data: FCFS				
layout	EHAM		case A	case C
operations	MET	TET	MET	TET
max. throughput	45	46	49	43
max. occupancy	76 %	79 %	100 %	92 %

(c) simulated data: depth-first			
EHAM		case A	case C
MET	TET	MET	TET
45	46	54	47
77 %	80 %	100 %	94 %

remain before the ConOps underlying this MAS model could be deployed in live airport operations. In turn, numerous research questions remain to be explored in future work, of which we consider only a few in the following subsections: we first reflect on the aspects related to the AS-MAMP algorithm in Section 7.1, and then discuss the limitations of the MAS model in Section 7.2. Broadening the perspective, we discuss open research directions regarding the proposed generalised architecture for studying diverse concepts for next-generation airport surface movement operations in Section 7.3.

7.1. Reflection on the AS-MAMP algorithm

We formulated the AS-MAMP algorithm as centralised decision-logic to conduct multi-agent coordination for all concurrent ground movements. A fast response time of the AS-MAMP algorithm is crucial for enabling human-in-the-loop experiments with real-time adjustments from human operators, probabilistic delays, and disruptions, and is ultimately necessary for deployment in live operations. Therefore, the AS-MAMP algorithm should provide conflict-free solutions within mere seconds regardless of traffic density, necessitating further improvements to its current computational complexity. The exhaustive search currently deployed as conflict detection mechanism turned out to drive the required time when coordinating the trajectories among large agent populations representative of busy traffic scenarios, and thus needs refactoring or replacement. Parallelisation of the algorithm, e.g. to compute the initial trajectories or to recalculate them during conflict resolution, could further reduce its computational complexity. In turn, the total runtime of AS-MAMP shown in Figs. 8 and C.3 would likely remain below the green line, i.e. < 1 s to deconflict the routes for up to 20 agents and < 10 s for up to 50 agents.

As highlighted by the comparison to a first-come-first-served approach, the conflict selection procedure could be optimised to minimise runtime while preserving a high-quality solution. Despite SIMP having an almost linear time complexity, the number of explored states could be minimised by further improving the heuristics as well as introducing suitable pruning strategies. To this end, future research should also investigate the potential of enhancing both high-level and low-level in AS-MAMP for example with machine learning or reinforcement learning (ML/RL) techniques: while still in its infancy, a growing body of literature explores how these can be combined with and integrated into MAPF algorithms to augment their path planning capabilities as well as help in the execution of the generated routing plans (Alkazzi and Okumura, 2024). Given the almost linear computational complexity of SIMP in relation to the number of explored states observed in Figs. 8 and C.3, a considerable improvement in runtime is to be expected. In turn, deconflicting up to 50 agents might be achievable within 1 s.

Regarding the evaluation of AS-MAMP, the stepwise approach of first using a simplified, synthetic layout and then shifting to a real-world airport layout proved helpful to benchmark both high-level and low-level of AS-MAMP. To further investigate the efficacy of AS-MAMP, simulations could be conducted on airports with differing layouts across varying traffic densities. Moreover, other ground movements such as towing traffic as well as other ground vehicles using the taxiways could be included, which is closely linked to the overall model instance and thus further discussed in the next subsection.

In summary, potential directions for future work on the AS-MAMP algorithm include:

- to minimise the runtime of AS-MAMP with enhancements to the conflict detection scheme, the conflict selection and prioritisation mechanism, and to SIMP using e.g. focal search, pruning strategies, or improved heuristics;
- to explore how ML/RL techniques could be utilised in AS-MAMP;
- to extend the evaluation of AS-MAMP to other real-world airport layouts.

7.2. Reflection on the MAS model

In general, developing realistic models for real-world applications such as ASM Ops is a challenging and labour-intensive task. While we elaborated one model instance, simulating all its underlying conceptual aspects is beyond the scope of this paper. Thus, the ability of the MAS model to generalise to the large variability in airport sizes, their taxiway structures, and specific operational constraints remains untested and requires further validation. Furthermore, additional operations such as towing movements could be included in this MAS model instance.

Despite the promising simulation results, the feasibility of deploying this model instance with the AS-MAMP algorithm as a centralised path planner into live operations remains limited at this stage. The reliance on a single coordination entity introduces potential scalability and fault-tolerance concerns, especially under high traffic density or degraded system conditions. Moreover, the simulations presented in this paper relied on precise localisation and reliable communication between agents to ensure an execution according to the conflict-free routing plans. However, deviations during execution can occur for example due to mechanical issues, human interventions, or obstacles on the taxiways unknown to the agents in the MAS model. Furthermore, highly dynamic, unforeseen events such as sudden weather changes or emergency reroutings may challenge the resilience of the MAS model, requiring quick actions of agents. As first steps towards a potential real-world integration of the underlying ConOps, both adaptive replanning capabilities as well as empowering the agents to carry out local adjustments need to be thoroughly studied.

In summary, potential directions for future work on the MAS model instance include:

- to conduct further simulations of and interviews with operational experts from different airports that face varying operational challenges and bottlenecks;
- to include additional operations such as towing movements into the model instance;
- to evaluate its real-time adaptability by implementing the conceived local adjustment mechanisms based on distributed control elements.

7.3. Reflections on generalised hierarchical-distributed modelling architecture

In general, an important future research direction is to study various decision-making, coordination, and control mechanisms to explore the large design space for next-generation airport surface movement operations. To name one direction, ML/RL techniques could boost the abilities of MAS model instances, for example by providing predictions based on historical airport data that the different agents can use in their coordination, communication, and decision making. While ML/RL techniques could prove helpful in addressing real-world challenges, the implications of using such techniques must be thoroughly studied: for example, as key difference to AS-MAMP that anticipates on conflicts and mitigates deadlocks through its centralised coordination mechanism, implicit approaches such as ML/RL techniques must provably meet the stringent safety requirements of ASM Ops.

Besides, real-world ASM Ops are a complex socio-technical system, and current regulations as well as liability concerns pose constraints on introducing autonomy into the operations. While we recently explored how interactions between human operators and the MAS can be incorporated into such model instances (von der Burg and Sharpanskykh, 2024), many fundamental questions remain, e.g. the desirable level of automation, the allocation of tasks between humans and automated systems, and questions related to shared situation awareness. In general, human-automation teaming must be carefully designed to ensure effectiveness, trust, and accountability, particularly when air traffic controllers remain responsible for safety-critical decisions.

Moreover, real-world airport environments are subject to uncertainties such as weather disruptions, equipment failures, or ad-hoc ground vehicle movements, which demand robust fallback mechanisms and real-time adaptability. On the other hand, integrating automated planning systems into existing air traffic management workflows requires alignment with current operational procedures and regulatory constraints, as well as certification to be provably safe. Overall, we strongly agree with the view expressed in e.g. (Rieth and Hagemann, 2022; Ali et al., 2025) that human-automation teaming is required in an operational setting. As outlined in this paper, we believe that this should be realised as a hierarchical-distributed system. Connecting to the discussion on the algorithmic runtime in Section 7.1, distributed coordination schemes could keep the number of agents per coordination task well below 20. Since the envisioned runtime improvements would allow for sub-second calculations of deconflicted trajectories, many of the above mentioned real-world challenges as well as the algorithmic scalability issues could be accounted for when shifting to distributed coordination.

Collectively, addressing these points will require a joint effort between researchers, industry stakeholders, and policymakers. Likely, an iterative approach is necessary to design, test, and refine operational concepts, and to contrast the efficacy of operational ideas. We hope that the generalised architecture proposed in this paper will support these efforts as well as provide a basis for discussion between the parties involved.

In summary, potential directions for future work regarding the generalised MAS architecture include:

- to instantiate other models that satisfy the modelling requirements provided in Section 2, across ConOps addressing e.g. divergent decision-making mechanisms or degrees of control centralisation;
- to incorporate stochastic aspects representing e.g. sensor noise, communication delays, or operational deviations and disruptions into model instances, and evaluate how these affect agent decision-making;
- to explore approaches for adaptive human-automation teaming, tested in human-in-the-loop experiments;
- to compare and contrast the operational performances of various model instances to evaluate their potential implications for being introduced into real-world operations, and to synthesise these findings and lessons learned for improvement of the generalised modelling architecture.

8. Conclusions

As one of the key contributions of this paper, we proposed a generalised multi-agent modelling architecture with a hierarchical-distributed structure. Four agent categories (Scheduling Agents, Routing Agents, Guidance Agents, and Moving Agents) are organised in hierarchical layers. Within each category, multiple agents may operate and engage in centralised and/or distributed control mechanisms, informed by communication and coordination within and across the hierarchical levels. Through their perception, all agents can potentially receive relevant data from environmental objects to inform their decision-making processes. Based on the generalised architecture, a large variety of MAS models can be instantiated that are tailored to a specific ConOps, while retaining the overall system structure. Nonetheless, significant gaps remain in demonstrating how such control system models could be adapted to the constraints of live operations, as pointed out in Section 7. Therefore, the generalised architecture should be seen as a research tool to structure and analyse models of future operational concepts, rather than a deployable solution in its present form.

We elaborated one model instance based on a ConOps with centralised planning and a fully-automated control system. The centralised Routing Agent issues conflict-free trajectories by assigning priorities between the agents that move concurrently over the airport surface. It shares the routing plans with the distributed Guidance Agents that subsequently instruct the Moving Agents which in turn execute the commands accordingly. As backbone of this model instance, we formalised the decision-logic of the Routing Agent as the Multi-Agent Motion Planning on Airport Surfaces (AS-MAMP) algorithm. This two-level solver uses windowed prioritised planning to compute conflict-free 4D agent trajectories. As ASM Ops are a safety-critical system, we designed the AS-MAMP algorithm to return a solution only when it is certainly conflict-free. Moreover, as key contribution to the MAPF-field, we showed how the SIPP algorithm (Phillips and Likhachev, 2011) can be enhanced to work with finite accelerations as well as upper and lower speed restrictions of agents. To this end, we introduced the novel Safe Interval Motion Planning (SIMP) algorithm that is used as low-level solver in AS-MAMP.

We evaluated the AS-MAMP algorithm using multiple scenarios on both synthetic and real-world airport layouts: we benchmarked its high-level search using PBS variants and the first-come-first-served (FCFS) approach often used in application-oriented studies, as well as SIMP as its low-level search against SIPP and A* with kinodynamic constraints. The conducted simulations highlighted that AS-MAMP scales with approximately quadratic time complexity, with SIMP scaling alomst linearly in time, also for the tested traffic scenarios at Amsterdam Airport Schiphol with up to 120 hourly movements. Finally, we showed that the attainable maximum hourly runway throughputs match or exceed those at the largest European airports. Furthermore, contrasting PBS with FCFS underscored that the latter trades fast solutions for up to $30\,\%$ loss in efficiency as well as $10\,\%$ attainable runway throughput.

Overall, the obtained simulation results demonstrate the effectiveness of the proposed MAS model with centralised path planning in enabling safe, efficient, and scalable autonomous surface movement operations under realistic airport conditions, making it a viable benchmark model for studying next-generation airport surface movement operations.

CRediT authorship contribution statement

Malte von der Burg: Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Project administration, Methodology, Investigation, Data curation, Conceptualization; **Jorick Kamphof:** Software, Methodology, Data curation, Conceptualization; **Joost Soomers:** Software, Methodology, Data curation, Conceptualization; **Alexei Sharpanskykh:** Writing – review & editing, Validation, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Data availability

Data will be made available on request.

Acknowledgements

This article is part of the ASTAIR - Auto Steer Taxi at AIRport project which has received funding from SESAR 3 JU under grant agreement no. 101114684 under European Union's Horizon Europe research and innovation programme.

Appendix A. Detailed specifications of multi-agent system model

Fig. A.1 shows the multi-agent system (MAS) model for fully-automated airport surface movement operations (ASM Ops) with centralised planning and distributed plan execution. It is derived from the generalised architecture presented in Fig. 2. The environment services and agents comprising this system model instantiation are specified in the following.

A.1. Specification of environment

The Environment comprises a Global Clock to synchronise the time between all agents, the Airport Layout representing the taxiway network, the Surface Movement Radar to track Moving Agents on the airport surface, and the Datastrips associated with every Moving Agent. The latter is used to pass control between Guidance Agents and to store the necessary operational as well as analysis data.

The taxiway network is represented by a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with nodes $v \in \mathcal{V}$ and directional edges $\epsilon \in \mathcal{E}$. An exemplary Airport Layout is shown in Fig. A.2. Tug decoupling locations (orange nodes) are specific to each runway (grey edges) and are used for

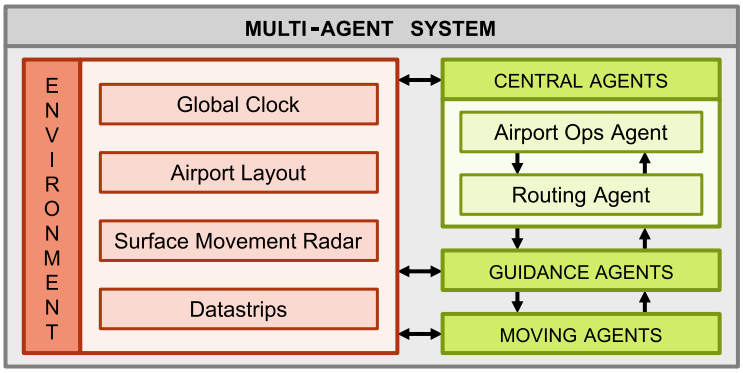

Fig. A.1. Instantiation of multi-agent system model for fully-automated airport surface movement operations (ASM Ops) with centralised planning and distributed plan execution.

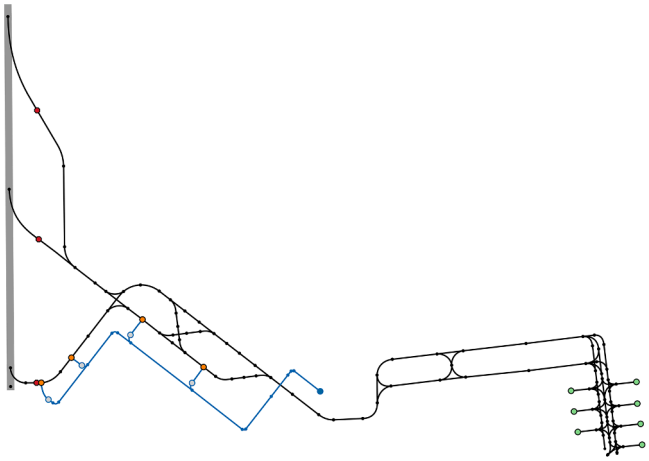

Fig. A.2. Exemplary layout of taxiway network. Taxiway centrelines as well as pushback paths are shown as black, runways as grey, and service roads as blue edges. Small nodes represent taxiway intersections (black) or service road intersections (blue). Large nodes indicate aircraft ramps (green), stopbars (red), decoupling locations (orange), or their associated all-clear points (grey).

tugs to couple to or decouple from aircraft that taxi with tug-enabled taxiing. Each bidirectional road segment between two nodes is constructed from two unidirectional edges that connect the nodes. Taxiway edges (black) as well as service road edges (blue) can be obtained from the actual locations of these taxiways at an airport, such as Amsterdam Airport Schiphol. Ramp nodes (green) represent the aircraft parking positions. The airport layout graph is fully accessible to the Routing Agent and remains static throughout the simulation. However, edges can be constrained to prohibit vehicles to be routed over these. Such layout constraints are set by the Airport Ops Agent and described below.

A.2. Specification of Airport Ops Agent

The Airport Ops Agent manages all information that influences the airport surface movement operations as a whole: it schedules the flights and tasks of ground vehicles, handles necessary changes of these, and determines the active runways, i.e. the runway mode of operation (RMO), for the next 15 min. Furthermore, it notifies the Routing Agent of any temporarily unavailable taxiway segments. The Airport Ops Agent has the following properties:

- **P1 Receive Request Property:** both Routing Agent and Guidance Agents can send information that affect the operations to the Airport Ops Agent, which then acts accordingly. For instance, a Guidance Agent could report a blocked taxiway segment, or the Routing Agent may request a change in the schedules to allow for conflict-free routing.
- **P2 Update Schedule Entry Property:** the Airport Ops Agent schedules all flights and tasks of tugs and updates the entry in the respective schedule accordingly, including any necessary changes that arise during the operations (see **P1**). The updated schedules as well as the time point of the earliest change are shared with the Routing Agent.
- **P3 Update RMO Property:** at each time point t, the Airport Operations Agent checks whether the active runways have to change. If so, it adjusts the blockage intervals from $t + 15$ min onwards for all edges affected by the RMO change. It then shares the new set of layout constraints with the Routing Agent.

P4 **Request Replanning Property:** if the time point of earliest change in either schedules or RMO falls within the planning window w_p, the Airport Ops Agent sends a request for immediate re-planning of the routes to the Routing Agent.

A.3. Specification of Routing Agent

The role of the Routing Agent is to perform the central route planning. To do so, it gathers the updated schedule data, layout constraints, along with local information from the Guidance Agents, and converts this into route parameters per agent. Then, it computes a plan for all vehicles within the planning window w_p using the AS-MAMP algorithm specified in Section 5, and shares the updated routing plans with the Guidance Agents. The properties of the Routing Agent are as follows:

P1 **Receive Request Property:** the Routing Agent can receive a re-planning request from both Airport Ops Agent and Guidance Agents. The former triggers re-planning if changes to the schedules or the layout constraints become effective within the planning window, while the Guidance Agents do so when the deviation between planned and executed route of a Moving Agent exceeds time thresholds. The request is stored internally and further handled in **P2**.

P2 **Check Replanning Property:** at each time point t the Routing Agent checks whether a re-planning request was made or the internal re-planning timer elapsed. If so, the timer is reset, and the routes are re-planned by executing **P3**, **P6**, and **P7**.

P3 **Update Constraints for Route Planning Property:** the layout constraints from the Airport Ops Agent are gathered and stored internally. These change foremost when the RMO switches, and allow the Routing Agent to anticipate on upcoming taxiway closures or re-openings corresponding to the altered RMO.

P4 **Handle Time Constraints of Moving Agent Property:** to address potential time constraints of a Moving Agent (pertaining to Requirement 2), the Routing Agent assesses whether these are achievable and whether remote and/or stand holding is necessary, which influences the activity sequence of the Moving Agent that it defines through **P5**. Furthermore, it sets operational constraints pertaining to the time constraints, to account for these during path planning done by its **P7**.

P5 **Define Activity Sequence of Moving Agent Property:** to include the various parts of the route of a Moving Agent such as push-pull manoeuvres, coupling / decoupling of tugs, or holding in path planning, the Routing Agent converts these into one of the following three activities, in accordance with Definition 1:

- `GoTo` activities comprise a start location and a set of goal locations, so that in path planning, two degrees of freedom exist: time and route. This activity is used as input to taxi from one point at the airport to another point at the airport, for example for regular taxiing.
- `Follow` activities consist of a predefined list of path segments that must sequentially be part of the route. Therefore, time is the only remaining degree of freedom in planning, and the path cannot be changed. This activity is used for instance for pushback and push-pull manoeuvres.
- `Wait` activities prescribe a waiting location and waiting duration that have to be accounted for in path planning. Operations such as coupling to or decoupling from a tug are examples of such.

Using a combination of these, the Routing Agent defines an activity sequence according to the schedule of a Moving Agent, which has to be executed in the respective order.

P6 **Update Route Parameters of Moving Agents Property:** per Moving Agent that is scheduled to taxi within the planning window, the Routing Agent collects all necessary parameters to plan its route. This encompasses the kinodynamic representation according to the vehicle type of the Moving Agent, the activity sequence derived from its schedule (shared by the Airport Ops Agent), and in case the Moving Agent is already taxiing also the latest execution data (shared by the controlling Guidance Agent).

P7 **Do Route Planning Property:** based on the collected information, the Routing Agent updates the routing plans of all agents. Following Definition 3, it generates trajectories that are conflict-free within its planning horizon by assigning priorities between agents. Its underlying decision logic is expressed by the AS-MAMP algorithm (see Section 5). The Routing Agent shares the resulting routing plans with the Guidance Agents.

A.4. Specification of Guidance Agents

Each Guidance Agent controls the Moving Agents that are travelling towards its location and hands over the control to the next Guidance Agent once a Moving Agent has passed its location. Based on the generated routing plans, it gives instructions to the Moving Agents, and monitors their execution. It acquires the necessary position and speed data from the radar. When the executed route deviates from the plan, a Guidance Agent can adjust the route locally to minimise the deviations. If the deviation to the planned route exceeds time thresholds, it can request re-planning of the routes. The Guidance Agents thus act as link between the centralised planning and the local operations. They have the following properties:

P1 **Receive Updated Routing Plan Property:** once a new routing plan is issued by the Routing Agent, the Guidance Agent updates the planned routes of the Moving Agent under its control and generates new corresponding instructions by executing **P2**.

P2 **Send Instructions to Moving Agent Property:** as one-way means of communication, the Guidance Agents send instructions for executing the next part of the planned routes to the Moving Agents. This property is thus executed whenever a Moving Agent has to change its speed or heading within the pre-defined execution time window w_{exec}. Furthermore, coupling/decoupling as well as engine-start instructions are sent.

P3 **Monitor Execution of Planned Routes Property:** at each time point t, each Guidance Agent obtains the positions, headings, and speeds of all Moving Agents under its control from the radar. The agent then estimates the time and speed at which each

Table A.1
Categorisation of aircraft sizes into ICAO-types.

ICAO-type	shape diameter [m]	exemplary aircraft type
ICAO-A	12	Cirrus Vision SF50
ICAO-B	25	Cessna, Learjet
ICAO-C	40	A320, B737, EMB 170/190
ICAO-D	54	A300F, B767
ICAO-E	72	A350, B747, B787
ICAO-F	80	A380, B747-800F

Table A.2
Minimum separation times in [s] for departing aircraft to mitigate wake turbulence. The aircraft types correspond to those defined in Table A.1.

leading \ following	ICAO-A	ICAO-B	ICAO-C	ICAO-D	ICAO-E	ICAO-F
ICAO-A	80	60	60	60	60	60
ICAO-B	100	60	60	60	60	60
ICAO-C	120	60	60	60	60	60
ICAO-D	120	100	80	60	60	60
ICAO-E	140	120	100	60	60	60
ICAO-F	180	160	140	120	100	60

Moving Agent will pass its location, and shares these with the Routing Agent. The Guidance Agents also use the radar data to monitor that their instructions are executed according to the planned routes. If deviations arise, they can primarily re-instruct the Moving Agents to close the gap. As last resort, they can execute **P4** to trigger centralised re-planning of the routes.

P4 Request Replanning Property: with this property, the Guidance Agents request re-planning from the Routing Agent iff the deviation between a planned and executed route of a Moving Agent exceeds pre-defined thresholds.

P5 Communicate with Moving Agent Property: the Guidance Agents communicate with the Moving Agents under their control whenever required. Thus, they can pass on information from or about a Moving Agent to the centralised agents.

P6 Handover Control Property: whenever a Moving Agent has passed the location of a Guidance Agent, it transfers the respective Datastrip and thus the control responsibility to the closest next Guidance Agent on the planned route of the Moving Agent. Once a Moving Agent has reached its goal, the DS is placed in storage, and can be utilised for post-operation analyses.

A.5. Specification of Moving Agents

Moving Agents represent either an aircraft or a tug. They are modelled to have a circular size with a shape diameter corresponding to their respective type. All aircraft are categorised according to the six size-types of the ICAO aerodrome reference codes (ICAO, 2016), as listed in Table A.1. Tugs are modelled with a diameter of 12 m based on the reference TaxiBot system (TaxiBot, 2023).

When consecutive aircraft take off from a runway, the Moving Agents have to keep a time-based separation minima to ensure sufficient wake turbulence separation. The times are dependent on the type-combination of leading and following aircraft, as listed in Table A.2. The values were mapped to the ICAO-types from Table A.1 on the basis of the RECAT-EU time-based separation minima (Rooseleer and Treve, 2023).

The role of the Moving Agents is to execute the instructions that they received from the Guidance Agents. They thus have the following properties:

P1 Execute Instructions Property: the Moving Agents execute the received instructions to the best of their capabilities to limit the amount of necessary re-instructions. The trajectories provided to the Moving Agents include an allowable positioning tolerance around the speed profile in analogy to the Ownship-display used in Bakowski et al. (2017).

P2 Communicate with Guidance Agent Property: in any case, the Moving Agents can communicate with the controlling Guidance Agent to exchange status updates and relevant operational information. They thus notify the Guidance Agent whenever a process such as tug decoupling or engine-start is estimated to finish. As further example, they could notify the Guidance Agent that their preferred maximal speed is lower than expected, so that the routing plans can be adapted accordingly.

Appendix B. Detailed description of AS-MAMP algorithm

In the following sections, we outline, discuss, and illustrate the technical details of the AS-MAMP algorithm and the definitions from Section 5.

B.1. Adaptations to PBS as high-level search of AS-MAMP

The pseudocode of the adapted version of PBS as high-level solver of AS-MAMP is provided in Algorithm 1 for reference. Changes to the original algorithm from Ma et al. (2019) are marked by red line-numbers.

Algorithm 1: High-Level Search of AS-MAMP as adaptation of PBS from Ma et al. (2019) (changes marked by red line-numbers).

```
Input: MAMP instance, searchMode, rootConstraints, rootPriorities (= ∅ by default)
## root-node:  SAMP-plan for all agents, adhere to root-constraints
1:  root ← SearchNode(type="root")                          # create root-node as class-instance
2:  root.plan ← ∅
3:  root.rootConstraints ← rootConstraints
4:  root.priorities ← rootPriorities
5:  foreach agent a_i do
6:      success ← root.updatePlan(a_i)                      # invokes SIMP, updates plan in-place
7:      if not success then
8:          return "no solution"
9:  root.cost ← c_N                                                          # see Eq. (B.1)
## child-nodes:  detect and resolve conflicts between agents
10: STACK ← root
11: while STACK ≠ ∅ do
12:     N ← pick and remove node from STACK according to searchMode
13:     conflict ← N.detectConflict()                       # get conflict pair, see Algorithm B.2
14:     if no conflict then
15:         break                                           # N contains conflict-free solution
16:     N.children ← ∅                                  # conflict exists:  resolve it for both agents
17:     foreach a_i in conflict do
18:         N' ← SearchNode(type="child")
19:         N'.plan ← N.plan
20:         N'.rootConstraints ← N.rootConstraints
21:         N'.priorities ← N.priorities ∪ {a_j ≺ a_i}              # agent a_j has priority over a_i
22:         success ← N'.updatePlan(a_i)                # updates a_i and all a_k for which a_i ≺ a_k
23:         if success then
24:             N'.cost ← c_N                                                # see Eq. (B.1)
25:             N.children ← N.children ∪ {N'}
26:     STACK ← STACK ∪ {N.children in order of decreasing cost}
27: return N.plan of solution-node N OR "no solution"
Output: conflict-free solution to MAMP-instance if existent and found
```

PBS optimises the sum-of-cost of the agent trajectories. Thus, we define the cost of a PBS search-node N as:

$$c_N = \sum_{i=0}^{A} c_{\text{prio},i} \cdot c(\Pi_i) \tag{B.1}$$

with the number of agents A, the routing plan cost $c(\Pi_i)$ from Definition 4, and $c_{\text{prio},i}$ as dimensionless cost to weigh the importance of the route of agent i for the overall optimisation goal. For instance, aircraft can be assigned a higher value for c_{prio} than tugs to increase their prioritisation: an increase in plan cost of the aircraft is penalised more than that of the tug, potentially influencing the resulting priority order.

In the root-node of PBS (Line 1 to 1), the trajectories of all agents are calculated taking merely the constraints set by the Routing Agent (see its **P3** and **P4**) into account. This represents the single-agent motion planning (SAMP) solution of the AS-MAMP search for each Moving Agent. Note that a partial priority ordering can be set in the root-node of the adapted version of PBS as well.

The next search-node is picked based on the chosen *search-mode*, for which we distinguish between either a depth-first, best-first, or greedy search. The **depth-first search** uses the top node of the STACK as new parent-node. Newly created child-nodes are thus placed on top of the STACK in order of decreasing cost. The **greedy search** follows the approach of Chan et al. (2023) to greedily place the child-node with the lowest number of remaining conflict-pairs on top of the STACK irrespective of its sum-of-cost. However, in contrast to the original Greedy PBS (GPBS) implementation, we do not alter the low-level solver. With these two search-modes, PBS only backtracks when it cannot find a solution in the explored branch. While these quickly lead to a solution, the resulting priority orderings may be sub-optimal and thus lead to higher taxi times of the agents. In contrast, the **best-first search** returns the optimal solution within the priority-tree as it continues to pick the node with the lowest cost in the priority-tree until the solution is conflict-free. However, this search mode is limited by both computational time and memory, making it feasible only for small agent populations. In Section 6, we use this search-mode to assess the suboptimality of the depth-first and greedy searches for the analysed case studies. As discussed in Section 7.1, implementing other search-modes may lead to faster as well as higher quality solutions, and should thus be explored in future work.

The authors of GPBS also introduced the concept of using induced constraints (IC) to further speed up the high-level search. While their use does not necessarily sacrifice solution quality, it does affect the chosen conflict to be resolved (see Appendix B.5) and thus the resulting priority order. We adopt this technique as variation of all three search-modes when benchmarking AS-MAMP as presented in Section 6.

B.2. Motion principles

Definition 8 is based on the following assumptions for modelling vehicle motions:

- the motions are approximated with piecewise-constant acceleration, i.e. $a(t) = \text{const.}$, forming a *piecewise motion segment* over the time interval t;

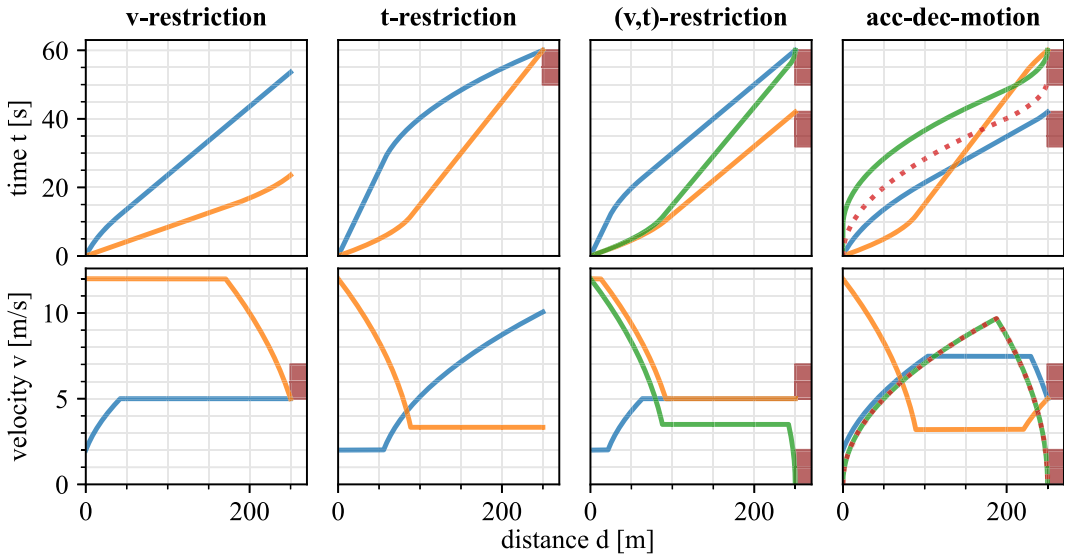

Fig. B.1. Exemplary motions visualised per column both in (d,t)-graph (top) and (d,v)-graph (bottom). The speed and time restrictions for the different examples are marked by red shades.

- since the radius of curvature r of all taxiway segments is large in comparison to the travelled distance and the change in velocity within that distance, we neglect the normal component of the acceleration;
- despite that, speed limits have to be met to traverse curved taxiway segments that have a radius of curvature below a pre-defined threshold value.

Differentiating $a(t) = \text{const.}$ twice and assuming relative distances per motion segment yields Eqs. 8 to 10.

We apply these equations to calculate on-the-fly motions along the graph edges during single-agent path planning. Since the equations are only valid for constant acceleration, we split motions into piecewise segments of constant acceleration if needed, and allow for kinks both at nodes and anywhere on edges. In general, the speed profile is optimised for smallest traversal time and highest final speed. The speed is bounded by an agent-dependent maximum speed $v_{\max}$ as well as a minimum speed $v_{\min}$ when in motion. During planning, reaching a full stop is only allowed at the end-nodes of edges. The agents can wait and hold arbitrarily long at these, as long as no reservations from other agents exist. While waiting is bound to nodes, holding may also occur at the beginning (but not at the end) of a motion along an edge.

By convention, a motion always connects the state at the beginning of an edge with the state at the beginning of a subsequent edge. A trajectory over multiple edges is thus split into one motion per edge before generating the resulting states on the respective nodes (see Appendix B.7). Note that the duration of motions can have arbitrary values, allowing for execution in continuous time.

Fig. B.1 illustrates different motions over one or multiple edges with a path length of $d = 250\,\text{m}$. The exemplary motions are plotted in both a (d,t)-graph (top) and a (d,v)-graph (bottom). Speed limits and minimal traversal times as restrictions are depicted by red shades. In the first column, a speed limit of $v_{\text{f,lim}} = 5\,\text{m/s}$ must be reached at the end of the path. Thus, to minimise the traversal time, acceleration from $v_{\text{i}} < v_{\text{f,lim}}$ to $v_{\text{f}} = v_{\text{f,lim}}$ takes place immediately (blue line), while deceleration from $v_{\text{i}} > v_{\text{f,lim}}$ to $v_{\text{f}} = v_{\text{f,lim}}$ occurs towards the end of the path (orange line).

In contrast, when the minimal traversal time must be exceeded due to time restrictions (second column), the motion is optimised for maximum final speed: acceleration occurs towards the end of the segment (blue line), while deceleration happens immediately (orange line). Although both motions take the same time, the blue trajectory yields a substantially shorter traversal time on subsequent edges because of its higher final speed. This illustrates that the fastest overall motion depends not only on past but also on upcoming restrictions. Reaching each node as quickly as possible is thus not always ideal.

The two remaining columns show combinations of speed and time restrictions. In the regular case (third column), acceleration (blue line) and deceleration (orange line) take place in between the two segments with constant speeds v_{i} and $v_{\text{f}} = v_{\text{f,lim}}$, ensuring both constraints are satisfied. In contrast, when $v_{\text{f,lim}} = 0\,\text{m/s}$ (green line), a constant speed v_{const} with $v_{\text{f}} > v_{\text{const}} \geq v_{\min}$ is inserted between two deceleration segments to meet the minimal traversal time. Any set of constraints requiring $v_{\text{const}} < v_{\min}$ leads to an infeasible motion by definition.

The regular motions with monotonic profiles described so far do not cover special cases such as $v_{\text{i}} = v_{\text{f,lim}} = 0\,\text{m/s}$. Furthermore, these profiles are too restrictive when motions span multiple edges, as required for backtracking and anticipation (see Definition 10). We therefore define **acc-dec-motions**, which include an acceleration and a deceleration phase to and from an intermediate speed v_{mid}. This speed results from the restrictions at the beginning and end of the travelled distance, and is bounded by $v_{\max}$ and $v_{\min}$. To cover the full distance, an additional segment with constant speed $v_{\text{const}} = v_{\text{mid}}$ may be inserted in between. By convention, except for holding at the start of an acc–dec motion, a constant speed is neither allowed at its start nor its end.

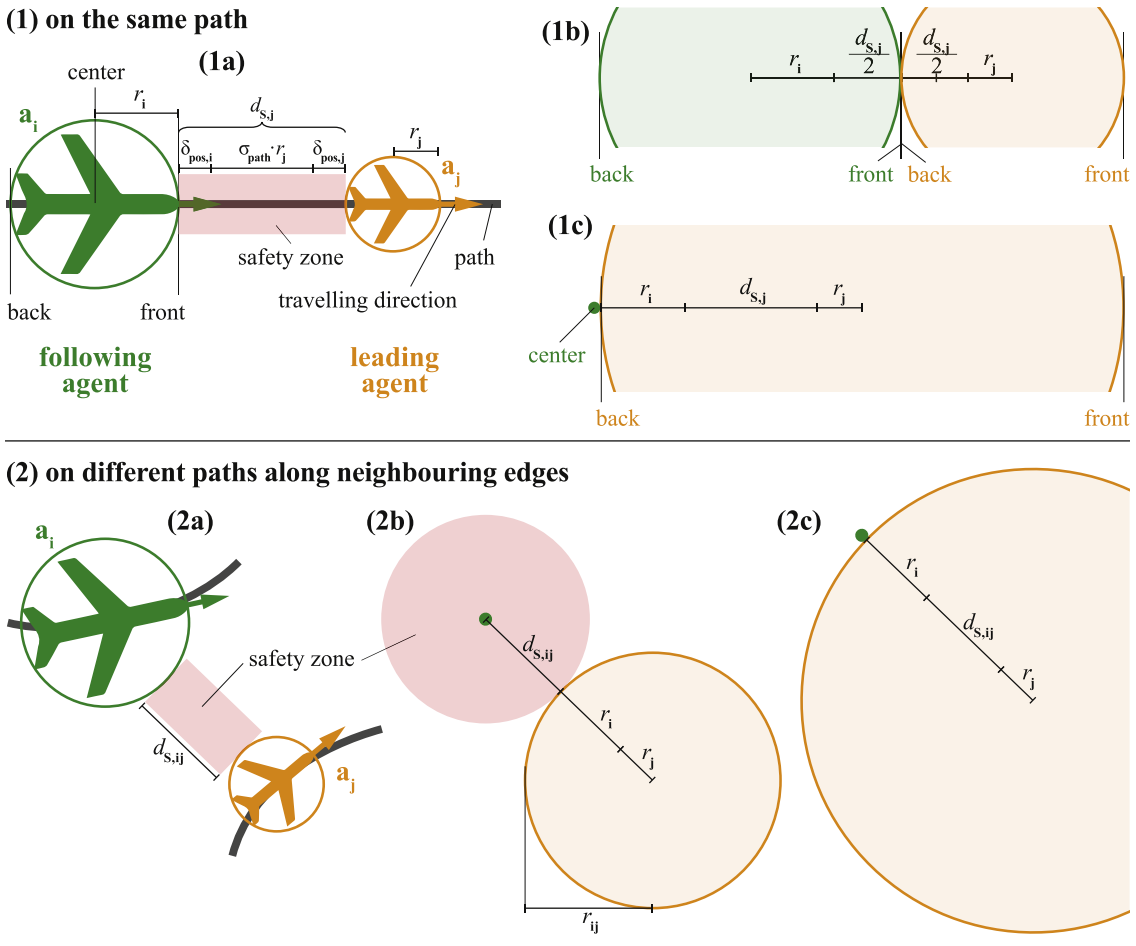

Fig. B.2. Definition of shapes and safety distances when two agents a_i and a_j travel (1) on the same path and (2) on different paths.

The last column in Fig. B.1 shows examples of such motions. In comparison to the third column, the acc-dec-motion allows the blue and orange motions to reach the end of the respective other time restriction depicted by the red shades. The green and dotted red lines depict the special case mentioned above: while the dotted red motion ends in the red shaded area and is therefore infeasible, the green motion includes holding at its start and thus exceeds the minimal traversal time. For these cases, a maximum holding time $t_{\text{hold,max}}$ can be defined as additional constraint if needed. Potentially, this results in a constant intermediate segment with a smaller v_{mid}.

B.3. Definition of motion profiles, shapes, and safety distances

In general, agents have a circular shape with a radius according to their type as listed in Table A.1. Each shape has a centre, front, and back, as illustrated in Fig. B.2. The centre represents an agent's reference point as defined in Li et al. (2019). A motion thus specifies the movement of the centre. We refer to the centre's collective motion over multiple edges as a **motion profile**, i.e. the agent's four-dimensional trajectory (4DT). The motion profile specifies the time points and speeds at the absolute distances at which the acceleration changes. Therefore, for each segment in the motion profile with constant acceleration, we can use Eqs. 8 to 10 to determine any time point t, travelled distance d as absolute or relative value, and speed v on the continuous scale. As the front and the back of a shape move in unity with the centre, their respective motion profile is obtained by shifting the absolute distances of the central motion profile by the radius of the shape. Then, the corresponding motion over an edge in the path of the agent can be determined in analogy to obtaining a single point in the profile as described above.

In real-world operations, taxiing vehicles have to keep a minimum safety distance. For aircraft, this is mainly due to the engine blast that poses a safety risk to any object or person in their wake. We know from expert interviews that other vehicles should keep a safety distance of around twice the length of the aircraft in front of them to mitigate the risk of engine blast. The experts highlight that the actual safety distance maintained by pilots and ground vehicle drivers often falls short of the recommended length. Since pilots and auto-pilots can only follow planned trajectories with limited precision (see Requirement 6), we include an allowable deviation as positioning tolerance within the safety zone.

In AS-MAMP, we hence define that all agents have to keep a minimum safety distance d_S between each other in dependence of their shape-radii as well as the positioning tolerance δ_{pos} for each agent. Since the taxiway network is represented as a graph, we distinguish between the minimum safety distance between agents on the same path (see Fig. B.2.1), and that between agents on different paths (see Fig. B.2.2). When travelling *on the same path*, we define the minimum safety distance between two agents in relation to the shape-radius of the leading agent. As illustrated in Fig. B.2.1a, for an agent a_i following an agent a_j along the same path, their minimum safety distance $d_{S,j}$ is defined as

$$d_{S,j} = \sigma_{path} \cdot r_j + \delta_{pos,i} + \delta_{pos,j} \tag{B.2}$$

with the safety factor σ_{path} multiplying the shape-radius r_j of agent a_j, and δ_{pos} for each of the two agents. We make use of the safety factor to provide an operational parameter that can be adjusted when studying airport surface movement operations with the multi-agent system. Nonetheless, we recommend to use values for σ_{path} between 2 to 6 to match the practical considerations of the experts. We will evaluate the effect of different positioning tolerances with a sensitivity analysis of δ_{pos} in Section 6.

When travelling *on different paths*, we define the minimum safety distance between two agents a_i and a_j in relation to their average shape-radius $\frac{r_i+r_j}{2}$ as:

$$d_{S,ij} = \sigma_{NE} \cdot \frac{r_i + r_j}{2} + \delta_{pos,i} + \delta_{pos,j} \tag{B.3}$$

with the safety factor σ_{NE}. Since the risk of engine blast is lower when agents do not directly trail each other, we recommend to use values for σ_{NE} between 1 to 3.

When determining the remaining gap between two moving shapes, different representations of their shapes are possible, which is illustrated in the sub-figures (b) and dummyTXdummy– (c) in Fig. B.2. For two agents on the same path, half of the minimum safety distance can be attributed to each agent, respectively (1b). This representation is useful in conflict detection, as outlined in Appendix B.5. Since SIMP plans a new trajectory for the centre of a shape, (1c) depicts a representation in which one of the agents is represented as point while both shape-radii as well as the minimum safety distance are associated with the other agent. The same approach is feasible for shapes on neighbouring edges (see 2c). Additionally, when setting graph reservations as outlined in Appendix B.4, the representation in (2b) is needed: it attributes both shape-radii as combined shape to a single agent while the other agent keeps the minimum safety distance as safety zone with radius $d_{S,ij}$. To this end, we define the radius of the combined shape as $r_{ij} = r_i + r_j$.

B.4. Graph reservations

To account for dynamic obstacles during single-agent path planning (i.e. the low-level search), the motions of other agents must be known to the planner. While the original SIPP implementation defines collision intervals to do so, these do not account for the shape and kinematics of other agents. As noted by Walker and Sturtevant (2019), an algebraic equation to detect collisions between accelerated motions must still be derived. The alternative approaches reviewed in Section 3.3 draw on assumptions that are invalid for graphs with arbitrarily long, curved, and oriented edges. Moreover, we must also account for layout constraints such as temporarily blocked edges due to changing RMOs during planning. These considerations motivate a new approach to handle dynamic obstacles and time constraints as graph reservations, as defined in Definition 5 and described in the following.

Once the SIMP algorithm has terminated for an agent, the resulting plan is converted into graph reservations. More specifically, we distinguish between three types of graph reservations:

- **path-reservations** on edges along the path of an agent;
- **neighbourhood-reservations** on all edges not on the path of an agent but swept by its shape, which includes edges on the path in reverse direction;
- **WTC-reservations** on runway entries to ensure sufficient temporal separation according to the wake turbulence category (WTC) of an agent with respect to other agents.

Path-reservations are necessary to let an agent follow – but not overtake – a higher prioritised agent that is travelling in front of it, while keeping the minimum safety distance as defined in Appendix B.3. Such reservations are illustrated in the top part of Fig. B.3: the orange agent *AC-2* (ICAO-C) traverses the edges (1,2), (2,3) and (3,4) towards the runway entry (4,10). It is followed by the green agent *AC-1* that has to use the runway entry (8,11) due to its larger size (ICAO-E). They thus share the edges (1,2) and (2,3). To reserve the traversed path, the motion of an agent's centre as well as the radius of its shape are stored separately per edge as path reservation.

In contrast, **neighbourhood-reservations** are set from an agent traversing an edge onto neighbouring edges. In the illustrated example in Fig. B.3, the orange agent *AC-2* cannot traverse edge (3,4) as long as the blue agent *AC-3* still occupies edge (6,5), since their shapes would collide. A neighbourhood-reservation comprises two parts to represent the space-time collision interval: the reservation is set for the *edge-interval* as reserved segment of the other edge which must not be traversed by other agents during the *time-interval* as its second part.

An edge-interval marks the start and end positions normalised by the length of the edge for which the reservation is set. These positions result from the geometric overlap based on the combined shape of two agents as defined in Fig. B.2.2b. For the illustrated example, the bottom row of Fig. B.3 depicts the radius r_{CE} as combined shape of AC-2 with type ICAO-C and AC-3 with type ICAO-E. With the combined shape, we calculate the geometric overlap between a pair of edges: when the combined shape traverses edge

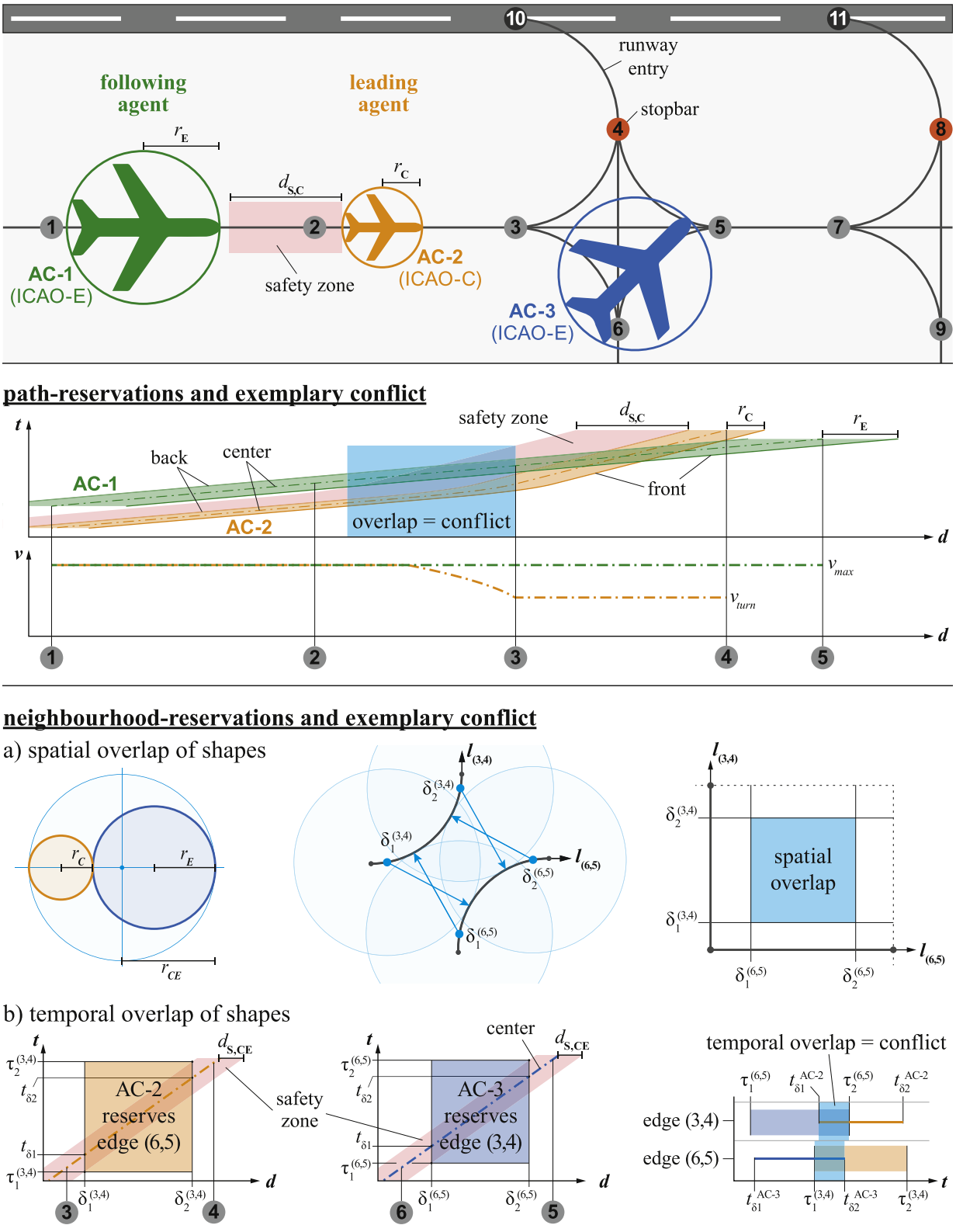

Fig. B.3. Graph reservations and exemplary conflicts (illustrated in top part), with path-reservations on edges along an agent's path (middle part) and neighbourhood-reservations on all other edges that are swept by an agent's shape (bottom part).

(3,4) it overlaps with edge (6,5) between $\delta_1^{(3,4)}$ and $\delta_2^{(3,4)}$. The overlap-interval of edge (6,5) with edge (3,4) is likewise calculated as $\delta_1^{(6,5)}$ and $\delta_2^{(6,5)}$. These normalised positions on the edge-pair form the edge-intervals of each edge when setting a neighbourhood-reservation from the other edge. Since the spatial overlap between an edge-pair depends on the combined shape of two agents, each combination of two agent types yields one edge-interval-pair per edge-pair. The intervals are pre-calculated for all edge-pairs and shape-combinations.

The time-interval as second part of a neighbourhood-reservation is determined with the motion profile over an edge: as outlined in Appendix B.3, we determine the time points at which the safety zone around the point-shape of an agent (green agent a_i in shape-representation 2b in Fig. B.2) enters and leaves the normalised positions δ_1 and δ_2 on the edge from which the reservation is set. In the example, agent *AC-2* reserves the edge (6,5) between $\tau_1^{(3,4)}$ and $\tau_2^{(3,4)}$, given the safety distance $d_{S,ij}$ around it. This results in the neighbourhood-reservation on edge (6,5) between the normalised positions $\left(\delta_1^{(6,5)}, \delta_2^{(6,5)}\right)$ as edge-interval during $\left(\tau_1^{(3,4)}, \tau_2^{(3,4)}\right)$ as time-interval. Likewise, agent *AC-3* reserves the edge-interval $\left(\delta_1^{(3,4)}, \delta_2^{(3,4)}\right)$ of edge (3,4) for the time-interval $\left(\tau_1^{(6,5)}, \tau_2^{(6,5)}\right)$.

We account for the required takeoff-separation according to the wake turbulence category (WTC) of a departing aircraft by defining **WTC-reservations**. These are set on the directional edges of all entries to the respective runway as time intervals per type-combination of the agent with any other aircraft type. A time interval consists of the start time point t_s and end time point t_e:

$$t_s = t_i^{StB} - t_{j \to i}^{WTC} \qquad \text{(i departs after j)} \tag{B.4}$$

$$t_e = t_i^{StB} + t_{i \to j}^{WTC} \qquad \text{(i departs prior to j)} \tag{B.5}$$

with t_i^{StB} denoting the time at which the centre of agent i is at the stopbar before entering the runway, and t^{WTC} being the durations required for the WTC-separations between agents a_i and a_j. In the example in Fig. B.3, both agents *AC-3* and *AC-1* use edge (8,11) as runway entry. Presuming that AC-3 will be in front, it reserves the runway entries (8,11) and (4,10) between t_s and t_e with $t^{StB} = t_8$ as the time of passing the stopbar at node (8) and $t_{j \to i}^{WTC} = t_{i \to j}^{WTC} = 60\,\text{s}$ from Table A.2.

Before updating the plan of a de-prioritised agent, all reservations from agents higher in priority are gathered. The illustrated collisions in Fig. B.3 are thus resolved when a priority order between the three agents is set. The next subsection outlines how the necessary conflict detection is carried out.

B.5. Conflict detection

Following Definition 6, we execute a conflict detection scheme to find all conflicts between each pair of agents. The corresponding pseudocode is outlined in Algorithm 2 for reference. We use an approach based on an exhaustive search: the reservations of each agent a_i are compared against those of every other agent a_j. Conflict detection is skipped if a priority order already exists between the two agents, or their plans are disconnected in time. Otherwise, we compare the path-, neighbourhood-, and WTC-reservations of agent a_i with those of agent a_j.

Algorithm 2: Conflict detection: without or with induced constraints (IC).

```
Input: graph reservations of all agents
conflicts ← ∅
foreach agent a_i do
    foreach agent a_j do
        ## check if conflict detection is needed between a_i and a_j
        skipDetection ← checkConflictDetectionNeeded(a_i, a_j)
        if skipDetection then
            continue                                   # conflict detection not required
        ## detect path-conflicts between a_i and a_j
        foreach edge in plan of a_i do
            if edge in plan of a_j then
                conflict ← checkMotionOverlap(motions of a_i, a_j along edge)
                conflicts ← conflicts ∪ {conflict}
        ## detect neighbourhood- and WTC-conflicts between a_i and a_j
        foreach edge in plan of a_i do
            foreach reservation of a_j on edge do
                if motion of a_i overlaps with reservation then
                    conflict ← store conflict-information
                    conflicts ← conflicts ∪ {conflict}
if without IC then
    conflict ← conflict in conflicts with min(t_start)
else
    conflict ← conflict with max(IC); breaking ties with min(IC_lower), then min(t_start)
return conflict
Output: agent pair in conflict
```

The agents are in a **path-conflict** if their path-reservations overlap on an edge that both of them traverse. To detect such a conflict, we first determine the sequence of the agents passing the edge. In the example in Fig. B.3, agent *AC-2* is in front of agent *AC-1* on the edges (1,2) and (2,3). Using the shape-representation 1b in Fig. B.2, we then compute the minimal gap between the front-line of the shape of the trailing agent (AC-1) with the back-line of the shape of the leading agent (AC-2). Note that half of

the minimum safety distance is included in each shape as depicted in Fig. B.2.1b. The minimal gap between the two shifted motion-profiles (see Appendix B.3) is analytically calculated by determining the minimum distance between the underlying motion-functions. If the distance is negative, the motions overlap, and thus form a path-conflict. As marked in the example, the motions of AC-1 and AC-2 overlap on edge (2,3), since AC-2 slows down for the curved edge (3,4) while AC-1 can keep its maximum speed to go straight ahead along edge (3,5).

In contrast, agents are in a **neighbourhood-conflict** if the neighbourhood-reservations of one agent overlap with the motion of the other agent's centre on the same edge. Recall that a neighbourhood-reservation consists of a time-interval during which another agent must not traverse the corresponding edge-interval of the edge for which the reservation is set. To detect such conflicts, we iterate over each edge in the plan of agent a_i. For each neighbourhood-reservation of agent a_j, we determine the time-interval at which the centre of a_i enters and leaves the normalised edge-positions according to the edge-interval of the reservation. The agents are in a neighbourhood-conflict if the resulting time-interval of a_i overlaps with the time-interval of the neighbourhood-reservation set by a_j. In Fig. B.3, such a conflict occurs between agent *AC-2* and agent *AC-3* between the edges (3,4) and (6,5): the motion of the safety zone around the centre of AC-2 traversing edge (3,4) overlaps with the neighbourhood-reservation of AC-3 that are set for that edge. Likewise, the reservation of AC-2 on edge (6,5) overlaps with the motion of AC-3 over that edge.

In analogy, agents are in a **WTC-conflict** if the WTC-reservation of one agent overlap with the motion of another agent on any edge that is an entry to the same runway. Using Eq. (B.5), the time interval of agent a_i occupying the runway is determined. The agents are in a WTC-conflict if it overlaps with the time interval that agent a_j reserved on all runway entries.

The conflict detection scheme is applied to every combination of two agents and yields all existing conflicts. In general, the first one in time is selected. However, when using a PBS search-mode with induced constraints (IC), we follow the approach of GPBS with IC (Chan et al., 2023): the agent pair with the largest IC-set is selected, breaking ties by choosing the pair with the smallest set of agents lower in priority. This is complemented by breaking ties in favour of the earliest conflict in time. To resolve the selected conflict, recall that two new PBS search-nodes are created with each conflicting agent being added to the others priority list. Per search-node, a new list of conditions for the de-prioritised agent is formed and SIMP is invoked.

B.6. SIMP: objective function and heuristic search

Similar to A* and SIPP (see Section 3.3), the Safe Interval Motion Planning (SIMP) algorithm searches for a path from a start to a goal location by minimising the associated cost of the resulting trajectory. The f-score $f(S)$ of a state S estimates the path cost:

$$f(S) = g(S) + h(S) \tag{B.6}$$

with the g-score $g(S)$ as accrued cost between the start location and S, and the h-score $h(S)$ as estimation of the remaining cost to reach the goal. Recall from Definition 11 that the heuristic function $h(S)$ also accounts for the vehicle category cat. This determines which edges of the graph can be traversed in general, as some edges are not accessible to all agents: for example, service roads cannot be used by aircraft, and some taxiways are wingspan-restricted.

To force SIMP to follow the agent's activity sequence, we calculate $h(S)$ as sum of the heuristic per activity $h(\alpha_k)$:

$$h(S) = \sum h(\alpha_k) \tag{B.7}$$

For `GoTo` activities, multiple end nodes may be defined (see **P5** of Routing Agent). In that case, the minimum of each $h(S)$ per combination j of end nodes within the sequence is chosen so that $h(S)$ remains consistent:

$$h(S) = \min(h_j(S)) \tag{B.8}$$

Each $h(\alpha_k)$ is calculated by using Eq. (3) with $d_{\mathrm{taxi}} = d_{\mathrm{h}}(\alpha_k)$ and $t_{\mathrm{taxi}} = t_{\mathrm{h}}(\alpha_k)$ as well as the c_{d} defined for the respective agent. For $d_{\mathrm{h}}(\alpha_k)$, the node-to-node shortest distance along the graph edges between the start and end node of the respective activity is used in general. However, since aircraft cannot turn on the spot at a state S in their current activity α_{S}, we use an edge-to-node shortest distance for $d_{\mathrm{h}}(\alpha_{\mathrm{S}})$ based on the directional edge in the travelling direction of the agent. These distances are pre-calculated for all edge-node-combinations in the layout. Per $d_{\mathrm{h}}(\alpha_k)$, and depending on the speeds at the start and end nodes of α_k, we apply Eqs. 8to 10 to calculate the corresponding minimal traversal time $t_{\mathrm{h}}(\alpha_k)$. Note that for `Wait` activities, $d_{\mathrm{h}} = 0$ and t_{h} equals the waiting duration.

Throughout the search, to ensure that the heuristic is admissible and consistent, we verify that:

$$h(S) \leq c_{\mathrm{m}}(S \rightarrow S') + h(S') \tag{B.9}$$

with $c_{\mathrm{m}}(S \rightarrow S')$ representing the motion cost between state S and its successor S' and $h(S')$ denoting the h-score from S' to the goal. c_{m} is calculated by using Eq. (B.1) with $t_{\mathrm{taxi}} = t_{\mathrm{m}}$ as the duration of motion m and $d_{\mathrm{taxi}} = d_{\mathrm{m}}$ as the traversed distance.

B.7. SIMP: state space

In the following, we provide additional details on the state definition in SIMP from Definition 9, and the core part of the SIMP algorithm is provided in Algorithm 3. The algorithm keeps track of its state space with an OPEN-list and a CLOSED-list, akin to A* and SIPP. However, the lists serve a different purpose in SIMP, and states also store additional information, such as a reference to their parent-state, the g-, h-, and f-scores, the arrival distance, and speed v_{mid} to indicate an acc-dec-motion (see Appendix B.2).

The purpose of the OPEN-list is to mark unexplored states. To follow the logic behind SIPP, OPEN shall contain states that cluster the motions along the following edge towards the subsequent node into intervals to reduce the number of similar states in the search

Algorithm 3: Safe Interval Motion Planning (SIMP) Algorithm.

```
   Input: route parameters of agent, reservations of higher prioritised agents
1: CLOSED ← ∅
2: OPEN ← getInitStates()                                   # creates start states S_init
3: while OPEN ≠ ∅ do
4:     S ← state in OPEN with lowest f-score
5:     if S fulfils goal-conditions then
6:         plan, cost ← getPlan(S)                          # backtracks plan from S to S_init
7:         return plan, cost                                # solution found: SIMP can terminate
8:     if S not in CLOSED then
9:         add S to CLOSED
10:    else
11:        chooseStateToKeep(S)   # same state-ID in CLOSED: keeps the state with lower f-score
            and updates child-states
12:        continue                          # successors of this state were already created before
13:    successors ← getSuccessors(S)
14:    foreach S* in successors do
15:        cfg, FMI ← extract from S*
16:        S* ← make FMI of S* temporally disjunct
17:        OPEN ← OPEN ∪ S*                                 # indexed by (cfg, FMI)
18: return ∅, ∞                                             # no solution found
   Output: plan from start to goal node and its cost, if existent
```

space, motivating the definition of states through a combination of agent configuration S_{cfg} and the state's feasible motion interval S_{FMI}, as defined in Definition 9.

To create a trajectory with minimal cost (see Appendix B.6), the state S in OPEN with the lowest f-score is picked and removed. Successors of S are generated by executing the function **getSuccessors**, which is further outlined in the next subsection. S is then moved to the CLOSED-list (Lines 8 to 12). Once a state in OPEN is picked that meets the goal conditions, SIMP terminates and returns the resulting plan and cost (Lines 5 to 7).

In contrast to clustering states, the purpose of the CLOSED-list is to quickly obtain a parent-state during the search as well as generate the final plan once the search has terminated without recalculating the motions between two states. We thus define the unique state-ID S_{ID} for states in CLOSED as:

$$S_{\text{ID}} = (v,\ v',\ t_{\text{S}},\ v_{\text{S}},\ \alpha_k) \tag{B.10}$$

with the arrival time t_{S} and arrival speed v_{S} at node v as well as the activity α_k that is valid from the state onwards. If a state with identical state-ID is added to CLOSED, SIMP keeps the one with the lower f-score, and updates its state-values as well as those of all its successors accordingly. Correspondingly, all child-states are recursively updated.

B.8. SIMP: state generation

In this subsection, we outline the generation of successors based on the pseudocode of the function **getSuccessors** and an illustrating example provided in Fig. B.4. Due to our state definition, multiple successor states S' may be defined for traversing the edge on which state S is defined. To limit the amount of repetitive computations for the motion along an edge, we first gather the speed restrictions and safe time intervals that must be satisfied to enter the next edges. These form multiple sets of boundary conditions for the motion along an edge. We cluster them along with the next edges that require an identical set of these motion boundaries (lines 5 in Algorithm 4). Furthermore, we sort them based on earliest arrival time within the safe interval, breaking ties in favour of highest speed limit v_{lim}. Then, following the sorted order, a motion is computed that satisfies the set of conditions, and the respective successors are created (Lines 8 to 22).

In the example in Fig. B.4, three operations are needed to create five regular successors (ID1 to ID5): first, the motion to enter edge (3,5) is created taking 12 s with $v_{\text{i}} = v_{\text{f}} = 15\,\text{m/s}$ (ID1). Then, meeting the speed limit of $v_{\text{lim}} = 5\,\text{m/s}$ for edges (3,4) and (3,6), the fastest motion takes 16.4 s, arriving within the first safe interval of each edge (ID2 and ID3). Finally, the successors ID4 and ID5 are created to arrive at the beginning of the second SIs on edges (3,5) and (3,6). To form a motion that spans 40 s, it is necessary to immediately start to decelerate from v_{i} to $v_{\text{f}} = 1.4\,\text{m/s}$, and keep a constant speed for the remaining length of the edge (2,3). In contrast, a successor of S that enters edge (3,4) at the beginning of its second safe interval is infeasible: its motion would require a duration of 110 s, but the safe interval on edge (2,3) already ends at $t = 100\,\text{s}$, thus requiring the motion to last less than 90 s.

The two successors created for the second SIs on edges (3,5) and (3,6) enter these edges with a lower speed than the speed limit of 5 m/s: to reach the required traversal time, the agent has to slow down significantly. In this case, arriving later and/or with a smaller speed at node (2) would yield a higher final speed at node (3). Therefore, for the sake of this example, we use the backtracking scheme (Lines 17 to 19) that was introduced in Definition 10 to optimise the respective final speed for entering the two edges: the function uses an acc-dec-motion explained in Appendix B.2 from the parent state of S to satisfy the restrictions. In the example, backtracking yields the two successors ID6 and ID7 that replace the successors ID4 and ID5, respectively. Note that the speed limit may not be reachable in all cases due to other constraints on the previous edges.

On the other hand, the final speed of a regular successor may be too high to decelerate sufficiently for the speed restrictions on upcoming edges. We use the anticipation scheme introduced in Definition 10 to check for such cases and to compute a motion from state S directly to the start node of that future edge. In the example, this is illustrated with edge (5,7): the length of edge (3,5) does

Algorithm 4: SIMP function `getSuccessors(S)`.

```
   Input: state S
1: successors ← ∅
   ## special case:  wait-activity, switching to next activity
2: if S.activity = "wait-activity" then
3:     S' ← getSuccessorWaitActivity(S)
4:     return S'
   ## get motion restrictions for next edges
5: motionBounds ← getMotionBounds(S)                          # based on v-restrictions
6: if not motionBounds then
7:     return ∅                                               # no way forward
   ## calc motions for next edges.  Start with quickest possible motion
8: motionBounds.sorted()                                      # sort for quickest motion
9: motionPrev ← ∅
10: foreach motionBound in motionBounds do
11:     if motionPrev slower than motionBound allows then
12:         continue                  # previous motion slower than this bound, skip recalculation
13:     motion ← calcMotion(S, motionBound)                   # calculate motion based on bounds
14:     foreach edgeID in nextEdgeIDs do
15:         S' ← createSuccessor(motion)                      # create successor based on motion
16:         successors ← successors ∪ {S'}
        ## use backtracking iff vFinal < vLimit
17:     if vFinal < vLimit then
18:         S'' ← createSuccsBacktrack()
19:         successors ← replace S' with S''
        ## anticipate on upcoming motion restrictions
20:     S' ← createSuccsAnticip()
21:     successors ← successors ∪ {S'}
22:     motionPrev ← motion
23: return successors
   Output: successor-states S' of state S
```

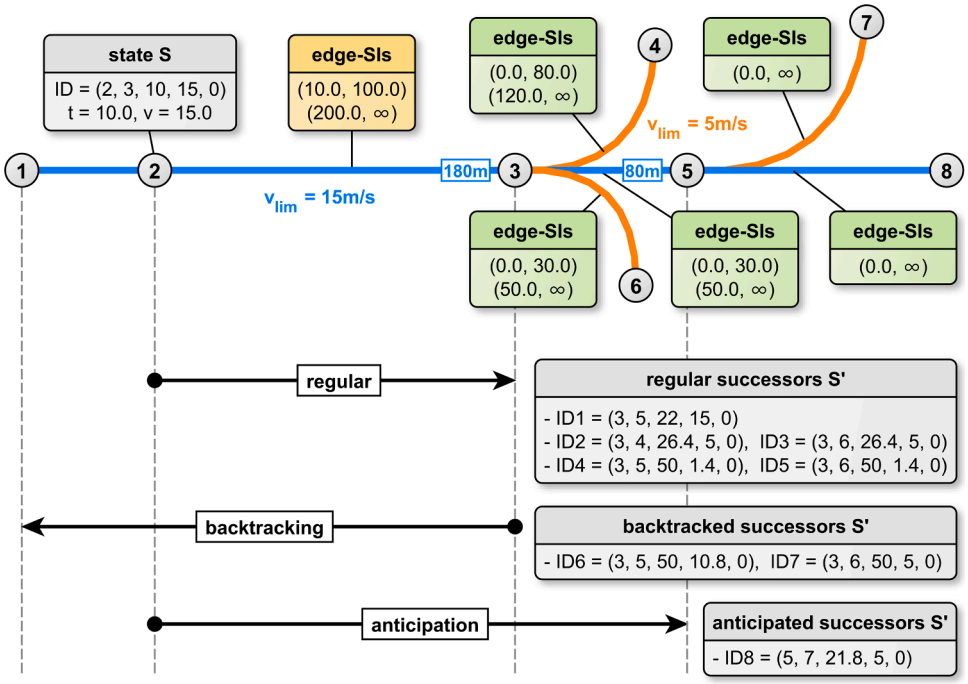

Fig. B.4. Exemplary successor generation. From state S on edge (2,3), different successors are created on edges (3,4), (3,5), (3,6), and (5,7) due to speed limits at straight segments (blue lines, $v_{\text{lim}} = v_{\text{max}}$), curved segments (orange lines, $v_{\text{lim}} = v_{\text{turn}}$), as well as safe intervals (SIs) on edge (2,3) (shaded in yellow) and for entering the upcoming edges (shaded in green). All resulting successors S' are displayed.

not suffice for an agent to decelerate from 15 m/s to the turn speed of 5 m/s. Therefore, the function implementing the anticipation scheme (Lines 20 to 21) creates a successor of state S (ID8) directly for entering edge (5,7).

Motion bounds do not have to be calculated when the agent has to wait at the current node due to a `Wait` activity: the successor can directly be created based on the waiting duration (Lines 1 to 4). If multiple SIs exist in which the agent can enter a subsequent edge, additional successors for holding until each start of these SIs are created as well. Note that holding is only possible until the end of the safe interval at the waiting-node.

B.9. Replanning

When the AS-MAMP algorithm is invoked by the Routing Agent, some agents may already be taxiing. Thus, these agents are likely in motion and in arbitrary places in the taxiway network at the planning time t_p. Since states are always defined on nodes, the Guidance Agents predict when and with which speed these agents will arrive at the next node on their respective paths (see their **P3**). The (node, time, speed)-information is then used as initial conditions when updating their plans in SIMP. To set the reservations correctly, the graph reservations explained in Appendix B.4 include the motion and edges along the already executed part of the route. However, reservations that already ended before t_p are dropped, since these lie in the past and agents cannot infringe them any more. Thus, only those previous reservations that are still relevant during re-planning are accounted for.

Appendix C. AS-MAMP on real-world airport layout

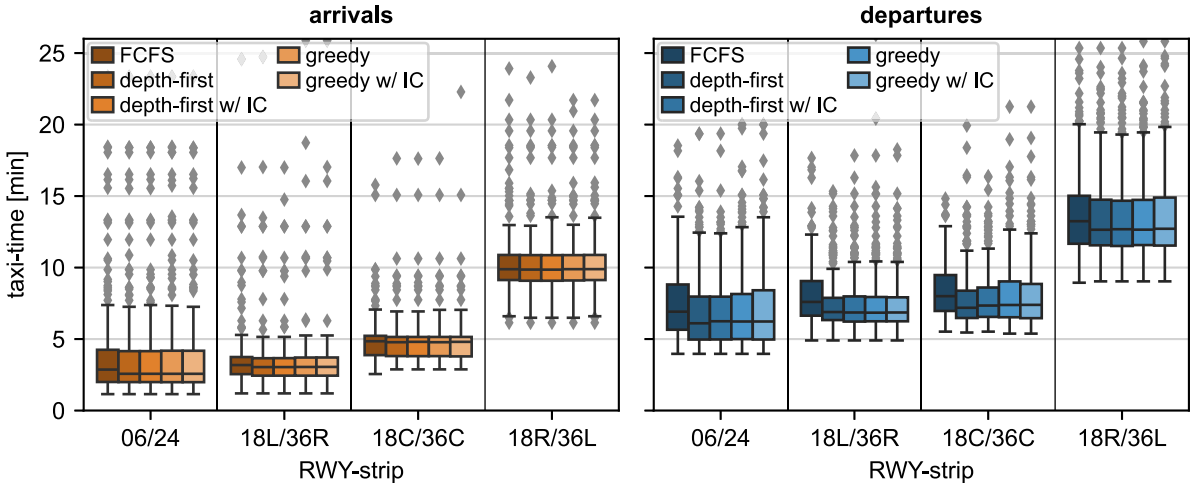

Fig. C.1. Box-and-whisker plots per runway for both inbound and outbound flights, showing the taxi time distributions of simulations of multi-engine taxiing (MET) operations using either FCFS or PBS variants as high-level in AS-MAMP.

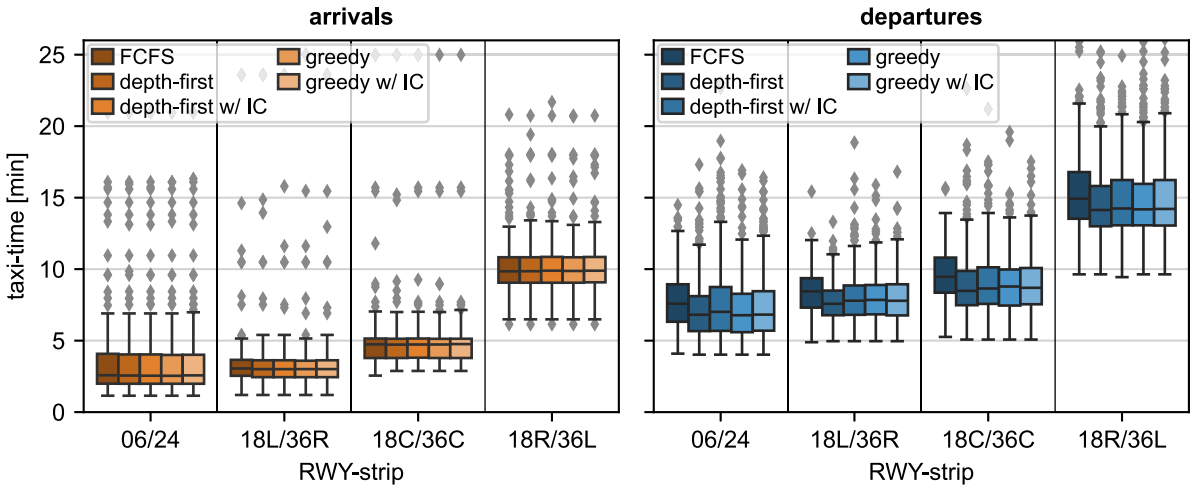

Fig. C.2. Box-and-whisker plots per runway for both inbound and outbound flights, showing the taxi time distributions of simulations of tug-enabled taxiing (TET) operations using either FCFS or PBS variants as high-level in AS-MAMP.

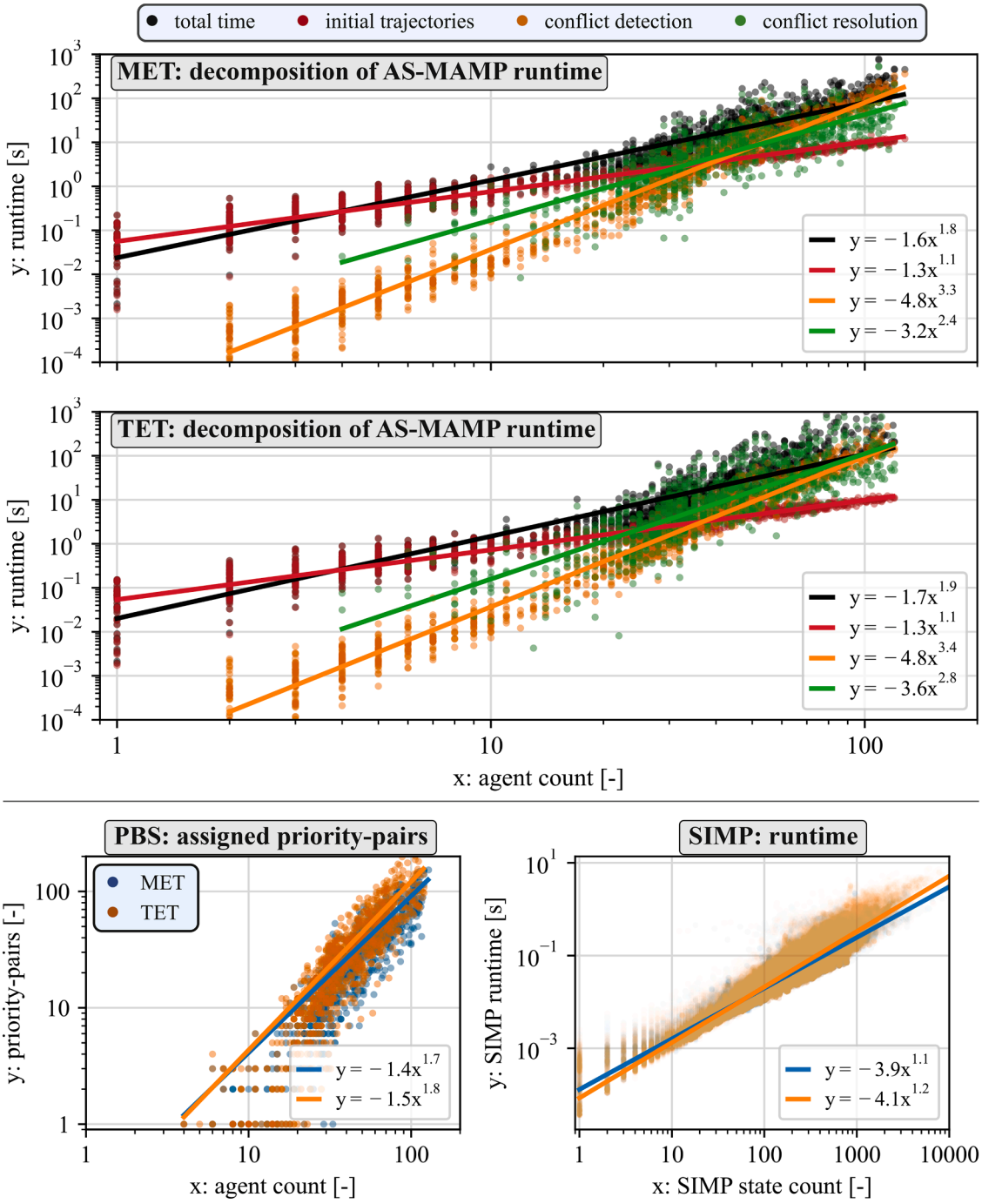

Fig. C.3. AS-MAMP runtime based on 20 simulations (ω_p = 10 to 60 min, 2565 planning rounds, approx. 420 k SIMP-calls) from previous work (von der Burg and Sharpanskykh, 2023, 2024) using historical flight schedules of two of the busiest days at Amsterdam Airport Schiphol to date; MET: multi-engine taxiing, TET: tug-enabled taxiing.

References

ACAAF-TF, 2025. Concept of operations of battery and hydrogen-powered aircraft at aerodromes. *Airport Compatibility of Alternative Aviation Fuels Task Force (ACAAF-TF)*. https://www.iata.org/globalassets/iata/publications/sustainability/concept-of-operations-of-battery-and-hydrogen-powered-aircraft-at-aerodromes.pdf.

AEON, 2023. AEON - Advanced Engine Off Navigation, Retrieved September 25, 2023. https://www.aeon-project.eu/.

Ali, H., Pham, D.-T., Alam, S., Schultz, M., 2022. A deep reinforcement learning approach for airport departure metering under spatial-temporal airside interactions. IEEE Transact. Intel. Transp. Sys. 23 (12), 23933–23950. https://doi.org/10.1109/TITS.2022.3209397

Ali, H., Pham, D.-T., Alam, S., Schultz, M., Li, M.Z., Wang, Y., Itoh, E., Duong, V.N., 2025. Human-AI hybrids in safety-critical systems: concept, definition and perspectives from air traffic management. Adv. Eng. Inform. 65, 103256. https://doi.org/10.1016/j.aei.2025.103256
Ali, Z.A., Yakovlev, K., 2023. Safe interval path planning with kinodynamic constraints. Proceedings of AAAI-23 37 (10), 12330–12337. https://doi.org/10.1609/aaai.v37i10.26453
Alkazzi, J.-M., Okumura, K., 2024. A comprehensive review on leveraging machine learning for multi-agent path finding. IEEE Access 12, 57390–57409. https://doi.org/10.1109/ACCESS.2024.3392305
Andreychuk, A., Yakovlev, K., Atzmon, D., Stern, R., 2019. Multi-agent pathfinding (MAPF) with continuous time. In: Proceedings of IJCAI-19, pp. 39–45.
Andreychuk, A., Yakovlev, K., Boyarski, E., Stern, R., 2021. Improving continuous-time conflict based search. Proceedings of AAAI-2021 35 (13), 11220–11227. https://doi.org/10.1609/aaai.v35i13.17338
Ashok, A., Balakrishnan, H., Barrett, S. R.H., 2017. Reducing the air quality and CO2 climate impacts of taxi and takeoff operations at airports. Transp. Res. Part D: Transp. Environ. 54, 287–303. https://doi.org/10.1016/j.trd.2017.05.013
ASTAIR, 2025. ASTAIR - Auto Steer Taxi at AIRport. https://research.dblue.it/astair/.
Atkin, J. A.D., Burke, E.K., Ravizza, S., et al., 2010. The airport ground movement problem: past and current research and future directions. In: 4th International Conference on Research in Air Transportation (ICRAT), pp. 131–138.
Bakowski, D.L., Hooey, B.L., Foyle, D.C., 2017. Flight deck surface trajectory-based operations (STBO): a four-dimensional trajectory (4DT) simulation. In: 2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC), pp. 1–10. https://doi.org/10.1109/DASC.2017.8102008
Bakowski, D.L., Hooey, B.L., Foyle, D.C., Wolter, C.A., 2015. NextGen surface trajectory-based operations (STBO): evaluating conformance to a four-dimensional trajectory (4DT). Procedia Manuf. 3, 2458–2465. https://doi.org/10.1016/j.promfg.2015.07.506
Balakrishnan, H., Jung, Y., 2007. A framework for coordinated surface operations planning at Dallas-Fort worth international airport. In: AIAA Guidance, Navigation and Control Conference and Exhibit. American Institute of Aeronautics and Astronautics, p. 6553. https://doi.org/10.2514/6.2007-6553
Barer, M., Sharon, G., Stern, R., Felner, A., et al., 2014. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In: Proceedings of SoCS-14, pp. 19–27.
Bennell, J.A., Mesgarpour, M., Potts, C.N., 2011. Airport runway scheduling. 4OR 9 (2), 115–138. https://doi.org/10.1007/s10288-011-0172-x
Bernatzky, T., 2019. Automation concept for cockpit crew integration into trajectory-based dispatch towing. https://doi.org/10.26083/tuprints-00008751.
Brownlee, A. E.I., Weiszer, M., Chen, J., Ravizza, S., Woodward, J.R., Burke, E.K., 2018. A fuzzy approach to addressing uncertainty in airport ground movement optimisation. Transp. Res. Part C: Emerg. Technolog. 92, 150–175. https://doi.org/10.1016/j.trc.2018.04.020
Chan, S.-H., Stern, R., Felner, A., Koenig, S., 2023. Greedy priority-based search for suboptimal multi-agent path finding. Proceedings of SoCS-23 16 (1), 11–19. https://doi.org/10.1609/socs.v16i1.27278
Chua, Z., Cousy, M., Causse, M., Lancelot, F., et al., 2017. Initial assessment of the impact of modern taxiing techniques on airport ground control. In: Proceedings of HCI-Aero 2016. ACM, pp. 1–8. https://doi.org/10.1145/2950112.2964589
Cohen, L., Uras, T., Kumar , T. K., Koenig, S., 2019. Optimal and bounded-suboptimal multi-agent motion planning. In: Proceedings of SoCS-19, pp. 44–51.
Cohen, L., Uras, T., Kumar , T. K., Xu, H., Ayanian, N., Koenig, S., 2016. Improved solvers for bounded-suboptimal multi-agent path finding. In: Proceedings of IJCAI-16, pp. 3067–3074.
Câmara, Álvaro., Silva, D.C., Oliveira, E., Abreu, P.H., 2014. Comparing centralized and decentralized multi-agent approaches to air traffic control. In: Proceedings of the 28th European Simulation and Modelling Conference (ESM-2014), pp. 189–193.
Dhief, I., Wang, Z., Liang, M., Alam, S., Schultz, M., Delahaye, D., 2020. Predicting aircraft landing time in extended-TMA using machine learning methods. In: Proceedings of ICRAT-20, pp. hal-02907597.
Dorri, A., Kanhere, S.S., Jurdak, R., 2018. Multi-agent systems: a survey. IEEE Access 6, 28573–28593. https://doi.org/10.1109/ACCESS.2018.2831228
EUROCONTROL, 2016. A-CDM impact assessment: Final report. https://www.eurocontrol.int/sites/default/files/2019-04/a-cdm-impact-assessment-2016.pdf.
EUROCONTROL, 2021. Data snapshot #22 on lower summer taxi-out times. *Retrieved November 30, 2021*. https://www.eurocontrol.int/sites/default/files/2021-11/eurocontrol-data-snapshot-22-taxi-time-affect-ops.pdf.
EUROCONTROL, 2018. European aviation in 2040 - challenges of growth. *StatisticsForecast Service. Retrieved December 14, 2021*. https://www.eurocontrol.int/sites/default/files/2019-07/challenges-of-growth-2018-annex1_0.pdf.
EUROCONTROL, 2020. Data snapshot #40 on taxi times. *Performance Review Commission. Retrieved May 19, 2023*. https://www.eurocontrol.int/sites/default/files/2023-03/eurocontrol-data-snapshot-40.pdf.
EUROCONTROL, 2023. Airport capacity imbalance. *Retrieved April 4, 2023*. https://www.eurocontrol.int/sites/default/files/2020-12/eurocontrol-prc-technical-note-airport-capacity-imbalance-11122020.pdf.
Felner, A., Stern, R., Shimony, S.E., Boyarski, E., Goldenberg, M., Sharon, G., Sturtevant, N.R., Wagner, G., Surynek, P., 2017. Search-based optimal solvers for the multi-agent pathfinding problem: summary and challenges. In: Proceedings of SoCS-17, pp. 29–37.
Gao, J., Li, Y., Li, X., Yan, K., Lin, K., Wu, X., et al., 2024. A review of graph-based multi-agent pathfinding solvers: from classical to beyond classical. Knowl. Base. Syst. 283, 111121. https://doi.org/10.1016/j.knosys.2023.111121
Gerdes, I., Schaper, M., 2015. Management of time based taxi trajectories coupling departure and surface management systems. In: Proceedings of ATM2015.
González, D., Pérez, J., Milanés, V., Nashashibi, F., 2016. A review of motion planning techniques for automated vehicles. IEEE Transact. Intel. Transp. Sys. 17 (4), 1135–1145. https://doi.org/10.1109/TITS.2015.2498841
Guépet, J., Briant, O., Gayon, J.P., Acuna-Agost, R., 2016. The aircraft ground routing problem: analysis of industry punctuality indicators in a sustainable perspective. Eur. J. Oper. Res. 248 (3), 827–839. https://doi.org/10.1016/j.ejor.2015.08.041
Guépet, J., Briant, O., Gayon, J.-P., Acuna-Agost, R., 2017. Integration of aircraft ground movements and runway operations. Transp. Res. Part E: Logist. Transport. Rev. 104, 131–149. https://doi.org/10.1016/j.tre.2017.05.002
Hayashi, M., Hoang, T., Jung, Y.C., Malik, W., Lee, H., Dulchinos, V.L., 2024. Evaluation of pushback decision-support tool concept for Charlotte Douglas international airport ramp operations. In: Proceedings of ATM2015. https://ntrs.nasa.gov/citations/20160008912.
Helbing, D., 2012. Agent-based modeling. In: Helbing, D. (Ed.), Social Self-Organization: Agent-Based Simulations and Experiments to Study Emergent Social Behavior. Springer, pp. 25–70. https://doi.org/10.1007/978-3-642-24004-1_2
Hoenig, W., Kumar , T. K., Cohen, L., Ma, H., Xu, H., Ayanian, N., Koenig, S., 2016. Multi-agent path finding with kinematic constraints. In: Proceedings of ICAPS-16, pp. 477–485.
Hooey, B.L., Cheng, V. H.L., Foyle, D.C., 2024. A concept of operations for far-term surface trajectory-based operations (STBO). https://ntrs.nasa.gov/citations/20190002687.
IATA, 2021. Net-zero carbon emissions by 2050. *Press Release No. 66. Retrieved April 4, 2023*. https://www.iata.org/en/pressroom/pressroom-archive/2021-releases/2021-10-04-03/.
ICAO, 2004. Advanced Surface Guidance and Control Systems (A-SMGCS) manual.
ICAO, 2016. Aerodrome design and operations. In *Annex 14 to the Convention on International Civil Aviation*. Vol. I of 7 edition.
Jiang, Y., Liu, Z., Hu, Z., Zhang, H., 2021. A priority-based conflict resolution strategy for airport surface traffic considering suboptimal alternative paths. IEEE Access 9, 606–617. https://doi.org/10.1109/ACCESS.2020.3045586
Jung, Y., Hoang, T., Montoya, J., Gupta, G., Malik, W.A., Tobias, L., Wang, H., 2011. Performance evaluation of a surface traffic management tool for Dallas/Fort worth international airport. In: Proceedings of ATM2011.
Lee, H., 2024. Airport surface traffic optimization and simulation in the presence of uncertainties. [Doctoral dissertation, Massachusetts Institute of Technology]. Retrieved May 6, 2024, from http://hdl.handle.net/1721.1/87480.
Li, J., Chen, Z., Harabor, D., Stuckey, P.J., Koenig, S., 2022. MAPF-LNS2: Fast repairing for multi-agent path finding via large neighborhood search. Proceedings of AAAI-2022 36 (9), 10256–10265. https://doi.org/10.1609/aaai.v36i9.21266

Li, J., Gong, M., Liang, Z., Liu, W., Tong, Z., Yi, L., Morris, R., Pasearanu, C., Koenig, S., 2019. Scheduling and airport taxiway path planning under uncertainty. In: AIAA Aviation 2019 Forum. AIAA AVIATION Forum, pp. 1–8. https://doi.org/10.2514/6.2019-2930
Li, J., Hoang, T.A., Lin, E., Vu, H.L., Koenig, S., 2023. Intersection coordination with priority-based search for autonomous vehicles. In: Proceedings of AAAI-23. Vol. 37, pp. 11578–11585. https://doi.org/10.1609/aaai.v37i10.26368
Li, J., Ruml, W., Koenig, S., 2021. EECBS: A bounded-suboptimal search for multi-agent path finding. Proceedings of AAAI-2021 35 (14), 12353–12362. https://doi.org/10.1609/aaai.v35i14.17466
Li, J., Surynek, P., Felner, A., Ma, H., Kumar , T.K., Koenig, S., 2019. Multi-agent path finding for large agents. In: Proceedings of AAAI-19. Vol. 33, pp. 7627–7634. https://doi.org/10.1609/aaai.v33i01.33017627
Li, J., Tinka, A., Kiesel, S., Durham, J.W., Kumar , T. K., Koenig, S., 2021. Lifelong multi-agent path finding in large-scale warehouses. In: Proceedings of AAAI-21. Vol. 35, pp. 11272–11281. 2005.07371.
Lieder, A., Briskorn, D., Stolletz, R., 2015. A dynamic programming approach for the aircraft landing problem with aircraft classes. Eur. J. Oper. Res. 243 (1), 61–69. https://doi.org/10.1016/j.ejor.2014.11.027
Lukic, M., Giangrande, P., Hebala, A., Nuzzo, S., Galea, M., 2019. Review, challenges, and future developments of electric taxiing systems. IEEE Trans. Transp. Electrif. 5 (4), 1441–1457. https://doi.org/10.1109/TTE.2019.2956862
Ma, H., Harabor, D., Stuckey, P.J., Li, J., Koenig, S., 2019. Searching with consistent prioritization for multi-agent path finding. In: Proceedings of AAAI-19. Vol. 33, pp. 7643–7650. arXiv:1812.06356.
Ma, H., Hönig, W., Kumar , T. K., Ayanian, N., Koenig, S., 2019. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In: Proceedings of AAAI-19. Vol. 33, pp. 7651–7658.
Ma, H., Koenig, S., Ayanian, N., Cohen, L., Honig, W., Kumar , T. K., Uras, T., Xu, H., Tovey, C., Sharon, G., 2016. Overview: generalizations of multi-agent path finding to real-world scenarios. In: IJCAI-16 Workshop on Multi-Agent Path Finding (WoMAPF), pp. 1–5.
Ma, H., Kumar , T. K., Koenig, S., 2017. Multi-agent path finding with delay probabilities. In: Proceedings of AAAI-17. Vol. 31, pp. 3605–3612.
Ma, H., Wagner, G., Felner, A., Li, J., Kumar , T. K., Koenig, S., 2018. Multi-agent path finding with deadlines. In: Proceedings of IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, pp. 417–423. https://doi.org/10.24963/ijcai.2018/58
Morris, R., Pasareanu, C.S., Luckow, K., Malik, W., Ma, H., Kumar , T. K., Koenig, S., 2016. Planning, scheduling and monitoring for airport surface operations. AAAI-16 Workshop on Planning for Hybrid Systems .
Okuniek, N., Gerdes, I., Jakobi, J., Ludwig, T., Hooey, B.L., Foyle, D., Jung, Y.C., Zhu, Z., 2016. A concept of operations for trajectory-based taxi operations. In: 16th AIAA Aviation Technology, Integration, and Operations Conference. American Institute of Aeronautics and Astronautics, pp. 1–18. https://doi.org/10.2514/6.2016-3753
Okuniek, N., Zhu, Z., Jung, Y.C., Gridnev, S., Gerdes, I., Lee, H., 2018. Performance evaluation of conflict-free trajectory taxiing in airport ramp area using fast-time simulations. In: 2018 IEEE/AIAA 37th Digital Avionics Systems Conference (DASC). IEEE, pp. 1–10. https://doi.org/10.1109/DASC.2018.8569228
Phillips, M., Likhachev, M., 2011. SIPP: Safe interval path planning for dynamic environments. In: Proceedings of ICRA-11, pp. 5628–5635. https://doi.org/10.1109/ICRA.2011.5980306
Ravizza, S., Atkin, J. A.D., Burke, E.K., 2014. A more realistic approach for airport ground movement optimisation with stand holding. J. Schedul. 17 (5), 507–520. https://doi.org/10.1007/s10951-013-0323-3
Rieth, M., Hagemann, V., 2022. Automation as an equal team player for humans? A view into the field and implications for research and practice. Appl. Ergon. 98 (4), 103552. https://doi.org/10.1016/j.apergo.2021.103552
Rooseleer, F., Treve, V., 2023. RECAT-EU: European wake turbulence categorisation and separation minima on approach and departure. https://www.eurocontrol.int/sites/default/files/2021-07/recat-eu-released-september-2018.pdf.
Rowland, B., 2023. Which part of a flight uses the most fuel? https://www.oag.com/blog/which-part-flight-uses-most-fuel.
Sharon, G., Stern, R., Felner, A., Sturtevant, N.R., et al., 2012. Conflict-based search for optimal multi-agent path finding. In: Proceedings of AAAI-12. Vol. 26, pp. 563–569.
Sharon, G., Stern, R., Goldenberg, M., Felner, A., 2013. The increasing cost tree search for optimal multi-agent pathfinding. Artif. Intell. 195, 470–495. https://doi.org/10.1016/j.artint.2012.11.006
Silver, D., 2005. Cooperative pathfinding. In: Proceedings of the Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE-2005), pp. 117–122.
SKYbrary, 2024a. Accident and serious incident reports: Runway incursion. https://skybrary.aero/articles/accident-and-serious-incident-reports-ri.
SKYbrary, 2024b. Ground operations: Ground collision. https://skybrary.aero/articles/ground-collision.
Stern, R., 2019. Multi-agent path finding — an overview. In: Osipov, G., Panov, A., Yakovlev, K. (Eds.), Artificial Intelligence: 5th RAAI Summer School. Lecture Notes in Computer Science, pp. 96–115. https://doi.org/10.1007/978-3-030-33274-7_6
Stern, R., Sturtevant, N.R., 2019. Multi-agent pathfinding: definitions, variants, and benchmarks. In: Proceedings of SoCS-19, pp. 151–158.
Švancara, J., Vlk, M., Stern, R., Atzmon, D., Barták, R., 2019. Online multi-agent pathfinding. In: Proceedings of AAAI-19. Vol. 33, pp. 7732–7739. https://doi.org/10.1609/aaai.v33i01.33017732
TaxiBot, 2023. TaxiBot. https://www.taxibot-international.com.
Udluft, H., 2017. Decentralization in air transportation. https://doi.org/10.4233/uuid:2af25aa3-be84-4d37-9332-fe2af319db1e.
Paris Agreement, 2015. Retrieved May 17, 2023. https://unfccc.int/sites/default/files/english_paris_agreement.pdf.
von der Burg, M., Sharpanskykh, A., 2023. Multi-agent planning for autonomous airport surface movement operations. In: SESAR Innovation Days 2023, pp. 1–9. https://doi.org/10.61009/SID.2023.1.27
von der Burg, M., Sharpanskykh, A., 2024. Studying the operational consequences of automated engine-off taxiing using multi-agent planning: explorative case study at Amsterdam airport schiphol. In: Proceedings of ICRAT-24, pp. 1–8.
von der Burg, M., Sharpanskykh, A., 2024. Exploring human-automation interactions in next-generation airport surface movement operations: an agent-based modelling approach. In: SESAR Innovation Days 2024, pp. 1–9. https://doi.org/10.61009/SID.2024.1.11
von der Burg , M., Sharpanskykh, A., 2025. Modelling and analysis of autonomous airport surface movement operations based on multi-agent planning. Eur. J. Transp. Infrastruct. Res. 25 (1), 1–23. https://doi.org/10.59490/ejtir.2025.25.1.7459
Wagner, G., Choset, H., 2011. M*: a complete multirobot path planning algorithm with performance bounds. In: Proceedings of IROS-11, pp. 3260–3267.
Walker, T.T., 2022. Multi-agent pathfinding in mixed discrete-continuous time and space.
Walker, T.T., Sturtevant, N.R., 2019. Collision detection for agents in multi-agent pathfinding. https://doi.org/10.48550/arXiv.1908.09707
Walker, T.T., Sturtevant, N.R., Felner, A., 2018. Extended increasing cost tree search for non-unit cost domains. In: Proceedings of IJCAI-18. International Joint Conferences on Artificial Intelligence Organization, pp. 534–540. https://doi.org/10.24963/ijcai.2018/74
Walker, T.T., Sturtevant, N.R., Felner, A., 2020. Generalized and sub-optimal bipartite constraints for conflict-based search. Proceedings of AAAI-2020 34 (05), 7277–7284. https://doi.org/10.1609/aaai.v34i05.6219
Wang, X., Brownlee, A. E.I., Weiszer, M., Woodward, J.R., Mahfouf, M., Chen, J., 2021. A chance-constrained programming model for airport ground movement optimisation with taxi time uncertainties. Transp. Res. Part C: Emerg. Technolog. 132, 103382. https://doi.org/10.1016/j.trc.2021.103382
Wang, Z., Liang, M., Delahaye, D., 2018. Automated data-driven prediction on aircraft estimated time of arrival. In: SESAR Innovation Days 2018, pp. 1–8. https://doi.org/10.1016/j.jairtraman.2020.101840
Woo, C.J., Goh, S.K., Alam, S., Ferdaus, M., Ellejmi, M., 2022. A runway exit prediction model with visually explainable machine decisions. In: Proceedings of ICRAT-22, pp. 1–9.
Yakovlev, K., Andreychuk, A., 2017. Any-angle pathfinding for multiple agents based on SIPP algorithm. In: Proceedings of ICAPS-17. Vol. 27, pp. 586–594.
Yakovlev, K., Andreychuk, A., 2021. Towards time-optimal any-angle path planning with dynamic obstacles. Proceedings of ICAPS 2021 31, 405–414. https://doi.org/10.1609/icaps.v31i1.15986
Yakovlev, K., Andreychuk, A., Vorobyev, V., 2019. Prioritized multi-agent path finding for differential drive robots. In: 2019 European Conference on Mobile Robots (ECMR). IEEE, pp. 1–6. https://doi.org/10.1109/ECMR.2019.8870957

Yin, J., Zhang, M., Ma, Y., Wu, W., Li, H., Chen, P., 2024. Prediction and analysis of airport surface taxi time: classification, features, and methodology. Appl. Sci. 14 (3), 1306. https://doi.org/10.3390/app14031306

Zaninotto, S., Gauci, J., Zammit, B., 2021. A testbed for performance analysis of algorithms for engineless taxiing with autonomous tow trucks. In: 2021 IEEE/AIAA 40th Digital Avionics Systems Conference (DASC), pp. 1–9. https://doi.org/10.1109/DASC52595.2021.9594411

Zheng, Y., Ma, H., Koenig, S., Kline, E., Kumar , T. K., 2023. Priority-based search for the virtual network embedding problem. In: Proceedings of ICAPS-23. Vol. 33, pp. 472–480. https://doi.org/10.1609/icaps.v33i1.27227

Zou, X., Cheng, P., Liu, W.D., Cheng, N., Zhang, J.P., 2018. A two-stage taxi scheduling strategy at airports with multiple independent runways. Transp. Res. Part C: Emerg. Technolog. 95, 165–184. https://doi.org/10.1016/j.trc.2018.07.005

Zou, Y., Borst, C., 2024. Zeta*-SIPP: improved time-optimal any-angle safe-interval path planning. In: Proceedings of IJCAI-24. Vol. 7, pp. 6823–6830. https://doi.org/10.24963/ijcai.2024/754

Zoutendijk, M., Mitici, M., Hoekstra, J.M., 2023. An investigation of operational management solutions and challenges for electric taxiing of aircraft. Res. Transport. Bus. Manage. 49, 101019. https://doi.org/10.1016/j.rtbm.2023.101019