D2.1 Support algorithms for automated tug assignment and path planning

Deliverable ID:	**D2.1**
Project acronym:	**ASTAIR**
Grant:	**101114684**
Call:	**HORIZON-SESAR-2022-DES-ER-01**
Topic:	**HORIZON-SESAR-2022-DES-ER-01-WA1-1**
Consortium coordinator:	**Ecole Nationale de L'Aviation Civile (ENAC)**
Edition date:	**22 May 2025**
Edition:	**01.01**
Status:	**Official**
Classification:	**PU**

Abstract

This deliverable presents support algorithms for automated tug assignment and multiagent path planning. These algorithms were designed on the basis of the algorithms for tug assignment and path planning developed in SESAR AEON project. The former algorithms were extended to take into account diverse and realistic spatiotemporal constraints related to airport surface movement operations (such as airport's traffic rules) and the ASTAIR concept in particular. The developed algorithms dynamically adapt to changes in the environment (such as changes of runway mode of operation) and new

constraints provided by human operators by recalculating their solutions. To address scalability issues of the previously developed tug assignment algorithms and to make them suitable for real-time use, an efficient, meta-heuristics-based approach was developed using Ant Colony Optimisation. The multiagent path planning algorithm developed in AEON was further improved by taking into account more realistic details (such as shapes and improved kinematic models of the vehicles) and enhancing its computational efficiency. During the workshops and interviews conducted in ASTAIR (WP1), preferred interactions of human operators with automated systems and algorithms were identified and described in deliverable D1.2 in the context of eight use cases. In this deliverable, we describe how some of these interactions were modelled and implemented in the multiagent system based on the developed algorithms. The algorithms were integrated in the ASTAIR's validation platform (D4.1) and will be evaluated during the project's validation phase (WP5).

Authoring & approval

Author(s) of the document

Organisation name	Date
TUD	5/3/2025

Reviewed by

Organisation name	Date
ADP	20/3/2025
ENAC	12/03/2025

Approved for submission to the SESAR 3 JU by[1]

Organisation name	Date
ADP	20/03/2025
ENAC	21/03/2025
TUD	20/03/2025
ECTL	20/03/2025
DBL	20/03/2025

Rejected by[2]

Organisation name	Date

[1] Representatives of all the beneficiaries involved in the project

[2] Representatives of the beneficiaries involved in the project

Document history

Edition	Date	Status	Company Author	Justification
00.01	15/1/2025	Initial version	TUD	The table of content and the overall setup
00.05	1/2/2025	Intermediate version	TUD	The initial description of the algorithms finished
01.00	19/3/2025	Pre-final version	TUD	The final version with all comments of the project participants addressed
01.01	22/5/2025	Final version	TUD	The final version with all comments from SJU addressed

ASTAIR

AUTO-STEER TAXI AT AIRPORT

ASTAIR

This document is part of a project that has received funding from the SESAR 3 Joint Undertaking under grant agreement No 101114684 under European Union's Horizon Europe research and innovation programme.

Table of contents

List of figures

List of tables

Executive summary

The goal of the ASTAIR project is to design a seamless partnership between Human and Artificial Intelligence (AI) to manage and perform engine-off and conventional airport surface movement operations at major European airports. ASTAIR original approach to automation is to consider an integrated airport system instead of many separate sub-systems, analyse the level of autonomy an AI system could take on tasks and to make the automation controllable by humans at different levels.

With the introduction of high-level automation for airport surface movement operations, the role of operators and airport operation procedures will significantly change. The key to optimize the overall performance of the collaboration between humans and AI is to adapt intelligent systems to the operators' modus operandi. This will ensure logical consistency across manual and automated control and reduce the cognitive distance between levels of automation by mapping system functions to goals and mental model of operators. In ASTAIR, we will propose interactive tools and adaptative AI algorithms that take advantage of operators' expertise for controlling and engaging with the automation at diverse levels.

This deliverable presents adaptive support algorithms for automated tug assignment and multiagent path planning. These algorithms were designed based on the algorithms for tug assignment and path planning developed previously in SESAR AEON project. The former algorithms were extended to take into account diverse and realistic spatiotemporal constraints related to airport surface movement operations (such as airport's traffic rules and specific constraints of ground traffic controllers identified during interviews) and to the ASTAIR's concept of engine-off taxiing.

The developed algorithms dynamically adapt to changes in the environment (such as changes of runway mode of operation) and new constraints provided by human operators by recalculating their solutions. To address scalability issues of the previously developed tug assignment algorithms and to make them suitable for real-time use, an efficient, meta-heuristics-based approach was developed using Ant Colony Optimisation.

The multiagent path planning algorithm developed in AEON was further improved by taking into account more realistic details (such as shapes and improved kinematic models of the vehicles) and enhancing its computational efficiency.

During the workshops and interviews conducted in ASTAIR (WP1), preferred interactions of human operators with automated systems and algorithms were identified and described in deliverable D1.2 in the context of eight use cases. In this deliverable, we describe how some of these interactions were modelled and implemented in the multiagent system based on the developed algorithms. The algorithms were integrated in the ASTAIR's validation platform (D4.1) and will be evaluated during the project's validation phase (WP5).

1 Introduction

1.1 Purpose of the document

In this deliverable we describe algorithms for the automated assignment of towing vehicles and multiagent path planning, which form the basis for the ASTAIR concept. The developed algorithms are based on the corresponding algorithms developed in SESAR AEON project, described in the AEON's deliverables D2.1 'Multiagent System for Routing' and D2.2 'Model for optimal allocation of towing vehicles'. In this deliverable we describe the developed extensions, in particular which allow taking into account a range of operational constraints relevant to engine-off taxiing and which enable real-time use of the algorithms and adaptation to changes. Furthermore, we describe how some of the interactions with human operators relevant to the ASTAIR concept could be modelled and implemented algorithmically.

1.2 Intended readership

The intended readership also includes:

- researchers developing computational models for sustainable airport operations;
- the key stakeholders targeted by the ASTAIR solution, in particular ground handlers, airport management, airlines, ATC operators and the industry providing green taxiing solutions;
- the overall aviation community interested in the document, as it will be publicly available.

1.3 Related documents

This deliverable builds upon or relates to the following documents:

- AEON project deliverable D2.1 Multiagent System for Routing, edition 00.00.03, edition date 16/9/2022, outlining the routing algorithm used as the basis for the ASTAIR project.
- AEON project deliverable D2.2 Model for Optimal Allocation of Towing Vehicles, edition 00.00.03, edition date 19/9/2022, outlining the algorithm used for the tug allocation used as the basis for the ASTAIR project.
- D1.1 State-of-the-Art, edition 01.00, edition date 26/2/2024, as basis for the algorithms that are described in this document.
- D1.2 Workshops Report, edition 01.00, edition date 27/6/2024, describing operational working practices, constraints and interactions relevant for the algorithms described in this deliverable.
- D4.1 Description of the validation platform, edition 01.00, edition date 27/1/2025, in which the algorithms presented in this deliverable will be integrated and evaluated as a part of the final validation session.

- D5.1 Exploratory Research Plan, which describes the objectives and criteria for the validation of the ASTAIR concept

1.4 Structure of the document

In Section 2 the algorithms for tug fleet management are described. The algorithms for adaptive path planning for aircraft and tugs are detailed in Section 3. In both Sections 2 and 3 we focus on the extensions of the AEON's base algorithms developed specifically for ASTAIR. In Section 3, a special attention is given to the modelling and algorithmic implementation of interactions with human operators in the context of several use cases considered in the ASTAIR's workshops.

2 Algorithms for tug fleet management

Effective deployment of towing vehicles entails assigning them to carry out tow tasks in order to minimize the total amount of fuel burned by taxiing aircraft and ensure that aircraft operate according to schedules, avoiding delays. Furthermore, more temporal constraints related to the assignment of towing vehicles could be provided by human operators (tug fleet managers). These constraints will be taken into account in the model similarly to how flight schedules are considered. Adding to the complexity is the fact that we assume that towing vehicles are electric (electric towing vehicles (ETVs)), requiring charging times to be integrated into their operational schedules. Together, these aspects define the tug scheduling problem that is considered in ASTAIR.

The approach that was followed in the AEON project to model the ETV assignment problem as a general fleet scheduling problem, extending the classic Vehicle Routing Problem (VRP). Such models effectively incorporate charging requirements and can produce optimal schedules for ETVs. However, their applicability in real-world operations is limited by two key aspects. The first is that these models rely on Mixed Integer Linear Programming (MILP) formulations, which are known to suffer from poor scalability. Especially when considering the dynamic nature of airport operations and frequent changes, which may prompt frequent rescheduling of ETVs, this can be undesirable. The second limiting aspect is that these models often follow a two-staged approach, where conflict-free paths are planned for aircraft first, and ETVs are scheduled after. This approach implicitly assumes that paths for aircraft will not change depending on the ETV schedule. In actual operations, however, the ETV schedule determines which aircraft are towed. Due to the kinematic differences between towed and regular taxiing aircraft, the paths found in the path planning stage of the model may no longer be valid.

An alternative approach followed in ASTAIR that effectively addresses the interdependency between aircraft paths and ETV schedules, while also capable of handling dynamic environments, is modeling the problem as a Multi-Agent Pickup and Delivery (MAPD) problem. In MAPD, a group of agents must tend to an incoming stream of pickup and delivery tasks. Tasks must be efficiently allocated to agents, and agents must carry out these tasks without colliding with one another. Typically, a MAPD solution method consists of a task assignment and a path planning module, which solve the problem either jointly (coupled) or through some iterative process (decoupled).

To address the scalability limitations of the tug scheduling algorithm used in AEON, in ASTAIR we used a bio-inspired method called Ant Colony Optimization (ACO), which has proven to be able to generate high quality solutions for VRP-type problems in a relatively short time. This algorithm is described in this section.

2.1 ETV assignment problem description

We consider the problem of dispatching a limited fleet of ETVs to maximize the fuel savings derived from ETV operations. Since the ETVs are electric, they need to recharge during operations. Consequently, the task assignment method must allocate both towing and charging tasks to ETVs. A towing task corresponds to an ETV towing one aircraft to its designated decoupling location. A charging task corresponds to an ETV to charging its battery at the ETV depot for a specified time interval.

Tasks must be allocated efficiently to account for the dynamic nature of airport operations. Specifically, the task assignment method is required to be fast to allow for rapid rescheduling, with a maximum runtime of one minute assumed to be sufficiently low. Additionally, the method should scale well with increasing problem complexity to accommodate variations in operational conditions.

The execution of a tow task is defined as follows. At the start of a tow task, an ETV couples to the aircraft at the stand. It then tows the aircraft along the airport's taxiways to the predefined decoupling point. After decoupling, the aircraft taxis to the runway using its engines. Meanwhile, the ETV enters the service road network via a service road entry point, either to pick up its next assigned aircraft or to return to the ETV depot for charging.

Tow tasks are executed by towing aircraft over the layout of an airport, which is represented as a graph *G = (N, E).* The edges in *E* denote the taxiways and service roads. The nodes in *N* represent stands, decoupling points, service road entry points, the ETV depot, runway entries, and intersections between taxiways and service roads.

Aircraft that are available are prescribed by the flight schedule *F*. Each aircraft can either be towed to multiple decoupling points. The end-point of the tow task determines a number of factors, including the achieved fuel savings, the discharge of the ETV, and which next tow tasks can be reached on time. Therefore, the task assignment method must select which towing end-point will be used. To represent this choice of where to tow the aircraft, we consider multiple potential tow tasks associated with flight f ∈ F, one for each possible towing end-point. All tasks that tow flight f are grouped in A_f^{alt} and the total set of tow tasks is defined as $A = \bigcup_{f \in F} A_f^{\text{alt}}$. All aircraft in F are categorized into regional, narrow-body, and wide-body. The category mass used for the battery modelling of ETVs.

Each tow task a ∈ A has a pickup node $n_a^p \in N$ and drop-off node $n_a^d \in N$. The pickup node is the aircraft stand from the flight schedule and the drop-off node is a decoupling point near its scheduled runway entry or on any of the TRPs. The pickup time of task a is denoted t_a^p as and is equal to the Actual Off-Block Time (AOBT) in the flight schedule. To optimize the assignment of tow tasks to ETVs, it is important to consider three aspects. First, it is necessary to compute paths over *G* to find the duration of tow tasks and the time it takes ETVs to transition between tasks. Second, a method to evaluate the fuel savings that follow from executing a tow task must be defined. Last, the properties of ETVs should be considered, including the battery discharge process that determines how many tow tasks an ETV can execute before requiring charging. These three aspects were considered in D2.2 'Model for optimal allocation of towing vehicles'.

In the following section we present our model and the related algorithms for ETV assignment based on Ant Colony Optimization.

2.2 Ant Colony Approach for the ETV assignment problem

Ant Colony Optimization (ACO) was originally introduced by Dorigo and Gambardella [1] to solve the Travelling Salesman Problem (TSP). ACO is a probabilistic optimization technique inspired by the foraging behaviour of ants. It is a population-based meta-heuristic that simulates the ants' pheromone-laying and following behaviour to explore and exploit a solution space. By iteratively updating pheromone trails based on the quality of discovered solutions, ACO guides the search toward promising regions of the solution space.

ACO is well-suited for combinatorial optimization problems, including the TSP-related Vehicle Routing Problem (VRP), general Multi Robot Task Allocation (MRTA) problems, and airport gate assignment. The ETV assignment problem is essentially about finding an optimal chain of tow tasks for each ETV. This is analogous to the TSP or VRP, where the goal is to determine an optimal sequence of cities or delivery points. Because of these similarities, and the proven effectiveness of ACO solution methods, we developed an ACO-based approach for task assignment of ETVs.

In the ACO method, ants traverse a solution space composed of task nodes to construct a task assignment. The design of this solution space is detailed in Section 2.2.1, while the solution construction process is explained in Section 2.2.2. While traversing the solution space, ants must repeatedly choose which next task node to visit. This task node selection process is detailed in Section 2.2.3.

2.2.1 Solution space

Ants must be able to traverse a solution space in such a way that an optimal task chain can be found. A temporal network is used to construct a solution space that facilitates the exploration of different task chains while adhering to the temporal constraints of the problem.

A schematic overview of the ACO solution space is shown in Figure 1. This example shows a temporal network with five towing tasks, with the start and end node of task *i* denoted by S_i and F_i respectively. The start nodes are positioned along the time axis in correspondence with the task start time and the end node position follows from the task duration. The positioning of nodes along the vertical axis has no meaning in this figure. In addition to the tow task nodes, three charge nodes (C_1, C_2, C_3) are shown. Charge nodes are placed at fixed intervals along the time axis and their associated charge task has a fixed duration. Compared to the mathematical formulation, this approach poses additional constraints as charging ETV must now adhere to these predefined intervals. However, during testing and development, the effect of these predefined intervals on the solution quality was found to be negligible.

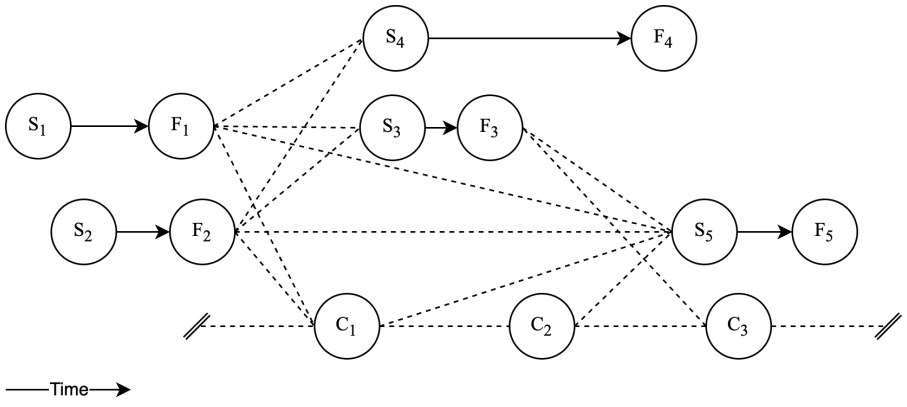


Figure 1: Schematic of the ACO solution space with five tow tasks and three charge tasks.

When exploring the solution space, ants can only travel in the positive time direction, i.e. from left to right. Dashed lines are the edges between nodes and indicate feasible transitions between tasks. For example, from node F_3, node S_5 and C_3 are feasible next tasks. From each node, only the first available charging node is considered as a feasible transition. Only feasible task transitions are included in the

solution space, ensuring that each ant always yields a feasible task assignment. The next section details the process of deriving a task assignment from the explored solution space.

2.2.2 Constructing a Solution

The ACO algorithm constructs task assignments iteratively, with ants exploring the temporal network by stochastically selecting task transitions based on heuristics and pheromones. High-quality solutions are reinforced through pheromone updates, guiding convergence to an optimized assignment. This section outlines the overall procedure for generating a task assignment with ACO, while Section 2.2.3 details the stochastic selection of task transitions and the heuristics and pheromones that influence this process.

The high-level loop for the ACO algorithm is shown in Algorithm 1. This is the original ACO algorithm [1], which remains unchanged in our application. Lines 1-3 initialize the pheromones, heuristics, the best solution x_{best}, and the best objective f_{best}. Then, the algorithm performs a number of N^{it} iterations. In each iteration, a total of N^{ant} ant solutions are constructed (line 6). The objective function value f_k is evaluated for each ant solution x_k. The value of the objective function is the total fuel savings resulting from executing the tow tasks in the solution. If this value is higher than the current best, the best solution and objective value are updated (lines 9 and 10). At the end of each iteration, the pheromones are updated (line 13) according to the process described in Section 4.3. After the maximum number of iterations has been reached, x_{best} and f_{best} are returned (line 16).

Algorithm 1 High-level ACO

```
1:  Initialize pheromones, heuristics
2:  Initialize best solution x_best ← None
3:  Initialize best objective value f_best ← 0
4:  for i = 1 to N^it do
5:      for k = 1 to N^ants do
6:          Construct single ant solution x_k (see Algorithm 2)
7:          Compute objective value f_k for solution x_k
8:          if f_k > f_best then
9:              x_best ← x_k
10:             f_best ← f_k
11:         end if
12:     end for
13:     Update pheromones (see Section 4.3)
14: end for
15:
16: return x_best, f_best
```

To tailor the original ACO algorithm to our specific problem and solution space, we modify the process of constructing a single ant solution. The pseudo-code for this process is provided in Algorithm 2. First, a set with available tow tasks A, a set with charge tasks C, and an empty solution x_k are initialized (lines 1-3). Then, each single ant constructs a task chain for the available ETVs. For each ETV, a route r_j is initialized and the initial conditions for the ETV battery b_j and previous task node n_{prev} are set (lines 5 - 7). When the ETV starts its operations, b_j^0 is set to the maximum battery capacity and the ETV starts from the depot. In the case of replanning or a rolling horizon approach, the initial conditions b_j^0 and n_j^0 are set accordingly.

By using the edges originating from the previous task node, the set with reachable task nodes A^{reach} is constructed (line 9). From those reachable nodes, only the task nodes that are still available in set A are retained (line 10). By taking the intersection with the union between set A and C, it is ensured that the charge nodes are not lost. When battery b_j is at maximum capacity, the set A^{reach} does not contain any charge nodes. If no available tasks remain, the loop terminates (line 12), and the route for ETV j is appended to the ant solution (line 24). The ant then resets to the start of the solution space to construct a task chain for the next ETV.

From the available tasks, only those for which the ETV has sufficient battery are considered feasible. This requires its current state of charge b_j to cover the sum of reaching the task from the previous node $q^S(n_{prev}, a)$, executing the task $q^X(a)$, and returning to the depot $q_l^S(a)$ (line 14). Note that for charge tasks, $q^X(a)$ is negative in this formulation. From the set of feasible tasks, the next task node n_{next} is selected based on a probabilistic selection process (line 15) as described in Section 2.2.3.

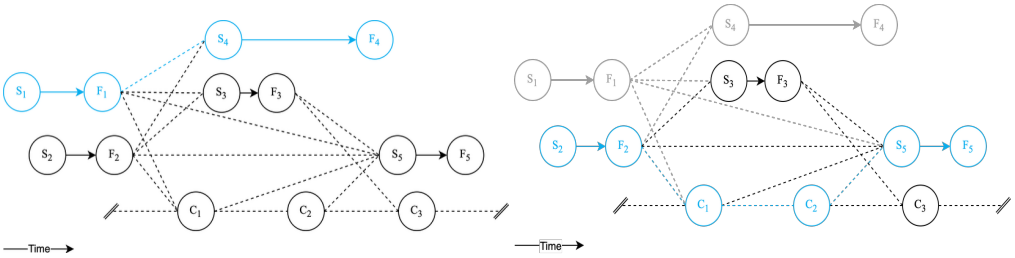
If the selected task node is a tow task, all the alternative tow tasks A_f^{alt} that tow the same flight f are removed from the available set A (lines 16-19). This ensures that each aircraft is assigned to at most one ETV. Charge nodes, if selected, are not removed as multiple ETVs can utilize these concurrently. Lastly, the ETV battery is updated, the selected task is appended to the ETV route r_j, and the previous task node is updated (lines 20 - 22). When the ant has constructed a task chain for all ETVs, the single ant solution x_k is returned (line 27).

Algorithm 2 Construct single ant solution $\mathbf{x}_k$

1: Initialize the set of tow task nodes $A \leftarrow \{\text{all tow tasks}\}$
2: Initialize the set of charge task nodes $C \leftarrow \{\text{all charge tasks}\}$
3: Initialize empty solution $\mathbf{x}_k \leftarrow []$
4: **for** $j = 1$ to N^{ETV} **do**
5: Initialize ETV route $r_j \leftarrow [n_j^0]$
6: Initialize ETV battery $b_j \leftarrow b_j^0$
7: Initialize previous task node $n_{\text{prev}} \leftarrow n_j^0$
8: **while** True **do**
9: Get the set of reachable nodes A^{reach} from previous task node n_{prev}
10: Get the set of task nodes that are still available $A^{\text{avail}} \leftarrow A^{\text{reach}} \cap (A \cup C)$
11: **if** $A^{\text{avail}} = \emptyset$ **then**
12: **break**
13: **end if**
14: Get task nodes for which b_j is sufficient $A^{\text{feas}} \leftarrow \{a \in A^{\text{avail}} \mid b_j > q^S(n_{\text{prev}}, a) + q^X(a) + q_l^S(a)\}$
15: Select transition to next task node n_{next} from A^{feas} (see Section 4.3)
16: **if** $n_{\text{next}} \in A$ **then**
17: Retrieve flight f associated with n_{next}
18: Update $A \leftarrow A \setminus A_f^{\text{alt}}$
19: **end if**
20: Update ETV battery $b_j \leftarrow b_j - \left(q^S(n_{\text{prev}}, n_{\text{next}}) + q^X(n_{\text{next}})\right)$
21: Append n_{next} to r_j
22: Update $n_{\text{prev}} \leftarrow n_{\text{next}}$
23: **end while**
24: Append r_j to $\mathbf{x}_k$
25: **end for**
26:
27: **return** $\mathbf{x}_k$

Figure 2 visualizes the construction of a single ant solution for two ETVs in the example solution space from Figure 3. First, the ant constructs a task chain for ETV 1 in Figure 2(a), starting with task node S_1. From F_1, task node S_4 is selected as the next task. As no task nodes are available from F_4, the task chain for ETV 1 is complete. The ant then resets to the start of the solution space to construct the task chain for ETV 2 in Figure 2(b). As indicated by the grayed-out nodes and edges, the tow nodes already visited by the ant are no longer available. For the second task chain, S_2 is selected as the first task node. The ant then visits two charge nodes and concludes by executing tow task 5.

When constructing the task chain for ETV 2 in Figure 2(a), selecting node S_3 (and subsequently S_5) from F_2 yields a higher objective value than the current solution. This is because executing more tow tasks increases the total fuel savings. However, since task node selection follows a stochastic process, this outcome represents one possible solution generated by an individual ant. Additionally, if the initial battery condition of ETV 2 (i.e., b_2^0) was already low, task node S_3 could be omitted from A^{feas} (line 14 in Algorithm 2), forcing the ant to travel over the charge nodes. The next section further details task node selection, including the pheromone levels and heuristic values that guide this process.



(a) Task chain for ETV 1 (b) Task chain for ETV 2

Figure 2: Single ant solution construction process for two ETVs.

2.2.3 Task Node Selection

As indicated by line 15 in Algorithm 2, an ant selects its next task node from the set A^{feas}. This happens according to the typical node selection process in ACO, where the probability for an ant to move from node *i* to node *j* is defined as p_{ij} in the Equation below:

$$p_{ij} = \frac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{k \in A^{feas}} (\tau_{ik})^{\alpha}(\eta_{ik})^{\beta}}$$

Here, τ_{ij} is the pheromone level and η_{ij} is a heuristic value for edge *(i, j)*. The set A^{feas} contains all nodes an ant can travel to from node *i*. Lastly, α and β are weighing parameters, determining the importance of the pheromones and heuristics respectively. Defining appropriate values for τ_{ij} and η_{ij} is crucial to the ACO solution quality as they largely determine how the algorithm searches through the solution space.

Heuristics

The heuristics are defined by

$$\eta_{ij} = \begin{cases} \dfrac{FS_j}{t_j^d - t_i^d} & \text{if } j \in A \\ 1 - \dfrac{q_i}{Q} & \text{if } j \in C \end{cases}$$

where η_{ij} is the heuristic value for moving from node *i* to node *j*. If *j* is a tow task node, the heuristic value is computed as the immediate reward of executing task *j*, expressed by the associated fuel savings FS_j, divided by the time until the ETV becomes available for its next task. This division promotes tightly planned schedules with less idle time for ETVs. If node *j* is a charging task node, the heuristic value is determined by the ETV's current state of charge q_i relative to its maximum battery capacity Q. As a result, the ant is less likely to select a charging node when the ETV's battery level is high.

From these definitions, it is clear that the heuristic values may vary in scale, which can negatively impact the node selection process. For this reason, when constructing the heuristics matrix, we perform a row-wise scaling for all task nodes to ensure that $\eta_{ij} \in [0, 1]$ for all $j \in AUC$. Performing these scaling operations allows for a fair comparison between heuristic values of different type nodes.

Pheromones

Next, the pheromone update process is described. At the start of the algorithm (line 1 in Algorithm 1), the pheromone matrix is initialized by setting all entries to a small constant value $\tau_0 = 0.1$. Then, during the update pheromone step (line 13) the pheromones are updated using the formula:

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{k=1}^{N^{\text{ants}}} \Delta\tau_{ij}^k$$

Here, ρ is the evaporation rate, and $\Delta\tau^k_{ij}$ is the pheromone deposit of ant k on edge *(i, j)*. The pheromone deposit on edge *(i, j)* is equal to the individual pheromone deposit of ant k, i.e. $\Delta\tau^k$, if edge (i, j) is present in ant solution x_k.

$$\Delta\tau_{ij}^k = \begin{cases} \Delta\tau^k & \text{if } (i,j) \in \mathbf{x}_k \\ 0 & \text{else} \end{cases}$$

This means the ant only deposits its pheromone on the edges it has visited when constructing its solution.

During testing, the ranked-ant pheromone update scheme was found to produce more stable results. The reason for this is the relatively small difference in objective values between high- and low-quality solutions. When pheromone deposits from all ants were considered equally, the pheromone matrix quickly became saturated with deposits from suboptimal solutions, resulting in slow convergence. The ranked-ant scheme mitigates this issue by weighing the pheromone deposits with the ant's relative solution quality, and only considering the solutions of the top N^{rank} ants. In the ranked-ant scheme, the individual deposit by ant k is determined using the equation:

$$\Delta\tau^k = max\left(1 + N^{\text{rank}} - \text{rank}_k, 0\right) c_{\Delta\tau} f_k^{\text{norm}}$$

Co-funded by the European Union

Here, $rank_k$ refers to the ranking of ant k's solution when all solutions are sorted by objective value in descending order. Additionally, f_k^{norm} refers to the objective value of the solution of ant k, normalized between 0 and 1. Normalizing the objective values produces more predictable pheromone deposits making the algorithm more consistent across different problem instances. Finally, $c\Delta\tau$ is a constant, determined through the parameter tuning process.

Charge Node Insertion

To minimize unnecessary idle time and ensure sufficient battery levels for completing future tasks, charge nodes are inserted into the ant's path whenever possible. After the ant selects a tow task node, the algorithm checks for a valid path through charge nodes to reach the selected node. If such a path exists, the charge nodes are added to the ant's path. In the example of Figure 4(b), selecting node S_5 from F_2 causes nodes C_1 and C_2 to be inserted into the ant's solution.

Co-funded by the European Union

3 Adaptive path planning algorithms for aircraft and tugs

The path planning model developed in ASTAIR is an extension of the model presented in deliverable D2.1 'Multiagent System for Routing' of SESAR AEON project. In this section we describe the developed extensions. In particular, in section 3.1 we present an overview of the multiagent system model for path planning used in ASTAIR. During the workshops and interviews conducted in ASTAIR (WP1), preferred interactions of human operators with automated systems and algorithms were identified and described in deliverable D1.2 in the context of eight use cases. In Section 3.2 we describe how some of these interactions were modelled and implemented in the multiagent system based on the developed algorithms. In particular, we describe how the level of conformance to the standard taxiway directions could be adjusted in the developed algorithms.

3.1 Multiagent system model for path planning

The developed multi-agent system (MAS) model for autonomous aircraft taxiing operations has a distributed-hierarchical structure of both centralized and distributed agents, which is illustrated in Figure 3. The centralized Airport Operations Agent defines and updates the flight schedule and runway configuration, the centralized Routing Agent plans conflict-free trajectories for all Aircraft Agents and Towing vehicle Agents which are instructed and monitored by distributed Guidance Agents while executing their planned routes.

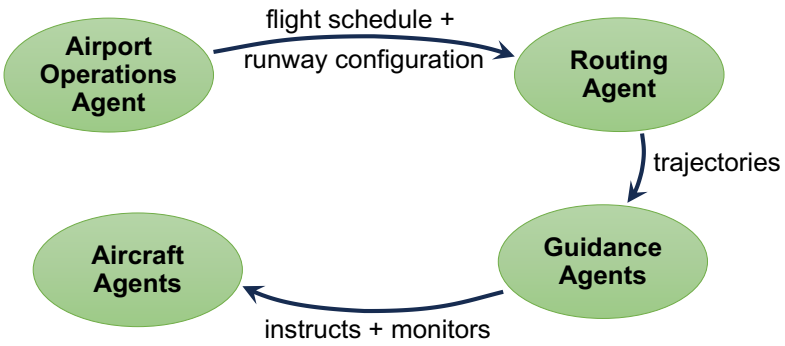


Figure 3: Overview of multi-agent system for autonomous airport surface movement operations

3.1.1 Model specification

The airport taxiing infrastructure is represented by a graph G = (V, E) comprising vertices V and directional edges E. Each bidirectional taxiway segment between two vertices is constructed from two unidirectional edges that connect the vertices. Taxiway edges are constructed using Bezier-curves that closely match the taxiway centrelines from a satellite image of an airport.

The Airport Operations Agent schedules all flights, and updates them whenever new predictions of the underlying A-CDM milestones are available. When the allocated stand of an arriving flight is still

occupied by a departing aircraft, or Eurocontrol issued a Calculated Take-Off Time (CTOT) for a departing aircraft, the agent marks the corresponding flight. Such flights are subject to special routes assigned by the Routing Agent to account for the necessary holding, detour, or prioritization during taxiing. Furthermore, the Airport Operations Agent defines the runways in use, i.e. the runway mode of operation (RMO). Active runways and the resulting flight path of arriving or departing flights must not be crossed. Thus, the Airport Operations Agent blocks such taxiway segments by setting layout constraints on them. This mechanism is also applicable for taxiway segments that are temporarily unavailable.

Both the flight schedule and constraints are shared with the Routing Agent that computes conflict-free routes for all taxiing aircraft within the upcoming planning window w_{plng}. It re-computes the routing plans when it receives updates from the Airport Operations Agent, or latest after the replanning period h_{plng} has passed. We use motion planning to account for vehicle kinematics and shapes in planning. To ensure conflict-free paths, we deploy a two-level search based on Priority-Based Search (PBS) [2] with an augmented version of the Safe Interval Path Planning (SIPP) algorithm [3].

The resulting trajectories are sent to the Guidance Agents which are positioned at every intersection in the taxiway system. Each Guidance Agent controls those Aircraft Agents that are moving towards its location. It instructs them to execute the next part of the planned trajectories, and monitors that the instructions are carried out accordingly. To do so, the Guidance Agents use the airport radar, which reports the position, speed, and heading of all Aircraft Agents while they move over the airport surface. In case the executed movements deviate from the planned routes, the Guidance Agents locally adjust the trajectories to minimize these deviations. However, when the impact becomes too extensive, they request central replanning from the Routing Agent. Once one of the Aircraft Agents has passed the location of a Guidance Agent, it passes the guidance responsibility for that aircraft to the next Guidance Agent along the aircraft's route.

Aircraft Agents represent the aircraft (auto-)pilots and are modelled to be fully cooperative: they thus carry out the instructions as accurately as possible. To account for the different sizes of aircraft, all flights are categorized as one of the 6 aircraft types from the ICAO aerodrome reference codes. They are assumed to have a circular shape with a pre-defined radius according to the type.

When planning the trajectories, a safety zone is added around all agents. To this end, we define a general safety distance, as well as a safety distance that an agent has to keep when it is trailing another aircraft. Both safety measures are defined in relation to the shape radii of the corresponding pair of agents. Moreover, two aircraft that consecutively take off from the same runway must have a minimal separation to mitigate the wake turbulence of the preceding aircraft. We use the time-based separation minima from RECAT-EU for that [4].

The model currently assumes that the A-CDM milestones in the flight schedule as well as the runway exit are predicted with high accuracy. In reality, arriving aircraft may vacate the runway at varying times and exits due to operational factors such as pilot behaviour and weather conditions. The model already includes a positioning tolerance to handle potential noise in position measurements. A similar mechanism could be introduced to accommodate uncertainties in the time of vacating the runway. Once the aircraft has entered the taxiway system, the Guidance Agents can adjust the speed profile to minimize any incurred deviations. As last resort and to cope with an aircraft using an unanticipated runway exit, centralized (partial) replanning could be triggered to adapt the taxi trajectory.

Departing aircraft may incur pushback delays, and the duration of their engine-start may be imprecise. Minor delays could be captured through buffer times added to the offblock-time as well as the engine-start duration. Larger delays could again be counteracted by locally adjusting the trajectory or partial replanning if needed.

3.1.2 Activity sequence of Aircraft Agents

To take the various surface movement operations into account during path planning, the route of an Aircraft Agent is expressed as a combination of the following three activities:

- Go-to activities have one start vertex and a set of goal vertices. Thus, the routing algorithm gets two degrees of freedom: the path between the vertices, and the time to traverse this path. The regular taxiing between one point to another point at the airport is an exemplary go-to activity.
- Follow activities comprise a predefined ordered list of edges that must be part of the route. Therefore, during routing, time is the only remaining variable as the path cannot be changed. Pushback and push-pull manoeuvres of departing aircraft are examples of such.
- Wait activities define a vertex at which an agent has to wait for a fixed duration. For instance, a wait activity is used to specify the place at which the pushback-truck is decoupled from the aircraft, or the necessary direction-switch of the push-pull manoeuvre within the pushback operations occurs.

Using a combination of these activities, the Routing Agent defines an activity sequence for both departing and arriving aircraft, as depicted in Figure 4.

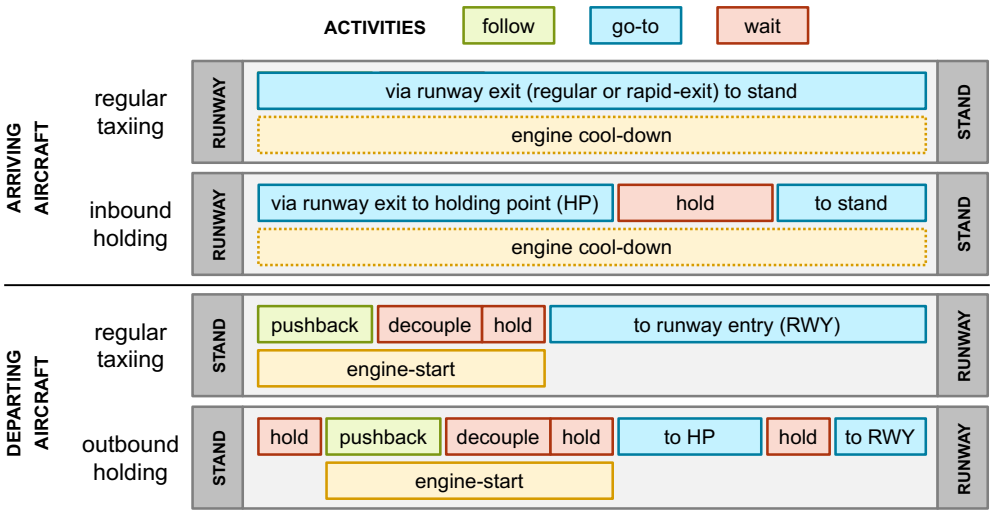


Figure 4. Activity sequence for regular taxiing of arriving and departing aircraft as well as inbound and outbound holding.

1. Engine warmup and cooldown: In the sequence, the warmup and cooldown of the engines represent special cases. The routing algorithm takes the warmup-phase as part of the engine-start manoeuvre and on basis of the aircraft-specific engine-start duration as input value into

account. Therefore, if this duration exceeds the time needed till decoupling from the pushback-truck, additional waiting in form of holding is added to the route. We do not model engine cooldown, as it does not have an influence on the routing regarding the kinematics, since the engines are switched off after standstill at the gate.

2. Inbound holding: When an aircraft arrives at the airport, but its stand is still occupied by a departing flight, the Routing Agent has three options to resolve the anticipated stand-conflict: for long conflict durations (option 1), it sends the arriving flight to the remote holding platform. Otherwise, it defines a detour along the taxiways (option 2), or reduces the agent's taxi speed for short conflicts (option 3). To this end, the Routing Agent first calculates the single-agent route directly to the stand, i.e. the trajectory without accounting for other aircraft agents, to estimate the severity of the stand-conflict. Then, it computes a single-agent trajectory via the remote holding points. When this detour is insufficient to resolve the stand-conflict, the Routing Agent assigns the remaining time as remote holding duration (option 1), and updates the agent's activity sequence accordingly. In contrast, when the taxi duration now exceeds the time at which the departing aircraft has cleared the stand (option 3), it keeps the original activity sequence of the agent.
3. Outbound holding to comply with CTOT-slots: Similar to inbound holding, the Routing Agent deals with departing flights for which Eurocontrol issued Computed Take-Off Times (CTOT-slots). However, as long as no arriving flight requires the stand, it assigns a holding duration at the agent's stand so that the agent arrives at the runway at the beginning of the CTOT-slot. In case an arriving flight is scheduled for the stand, the Routing Agent sends the departing flight to a remote holding location close to the scheduled runway. It updates the activity sequence of the departing flight accordingly.

3.1.3 Routing algorithm

The Routing Agent carries out multi-agent motion planning for all Aircraft Agents that taxi within the planning window. This two-level routing algorithm uses a low-level search to calculate individual trajectories per aircraft, and coordinates all agents in its high-level search to yield conflict-free trajectories. For the low-level, we extended the Safe Interval Path Planning (SIPP) algorithm [3], and adapted the Priority-Based Search (PBS) algorithm [2] to serve as high-level solver.

PBS constructs a priority order between agents to deconflict their space-time trajectories. In its priority tree, each parent-node has up to two child-nodes. Thus, a priority-relation between a conflicting pair of agents is established. In each child-node, one additional priority-pair is added with which one of the two agents that were previously in conflict must give way to the other agent along its entire route. Then, PBS checks the child node that has the lowest sum-of-cost of all agent trajectories for conflicts between those agents that do not yet form a priority-relation with each other. We define the cost of a trajectory as sum of the taxiing duration and travelled distance. Once a child-node is expanded without any collisions, PBS returns the resulting conflict-free trajectories.

In the low-level search, the route of a deprioritized agent has to be adapted, either by changing its path or altering the speed profile along the path. To this end, we translate all paths into a set of graph reservations: an aircraft temporarily blocks a set of edges during each movement between one vertex and another. The blockage times and set of blocked edges are dependent on the agent's shape, velocity profile, the shapes of other agents, and the safety zone between the shapes.

The SIPP algorithm represents moving obstacles as collision intervals and subsequently defines a set of Safe Intervals (SIs) per graph location, representing time intervals during which an agent can occupy that location. Furthermore, states are defined on vertices and motion profiles with piecewise constant acceleration map the trajectory between states. We augmented SIPP to facilitate the activity sequence of an aircraft as defined by the Routing Agent, and to take the travelling direction as well as the kinematic agent properties into account. Additionally, we use SIs also on edges to deal with the reservations of agents higher in priority. In the motion generation, we are bound to the agent's kinematic properties for the current activity and the velocity in the current state. A motion that is part of the follow-activity for pushback is for example constrained by a lower maximum speed than regular taxiing in a go-to activity. In addition, vehicles that have maximum velocity in the current state, might not be able to decelerate enough to satisfy a reservation on the next edge or vertex. In this case, it might be required to start decelerating on the edge before the current state. To efficiently account for this, we anticipate based on the agent's current velocity, braking distance, and reservations or velocity restrictions within the braking distance.

3.1.4 Model calibration

In the MAS model, the agents' motions during route planning are modelled based on constant longitudinal acceleration/deceleration and do not account for slip, i.e. are steady-state motions. Nonetheless, to the best of our knowledge, this is the first study to use such detailed kinematics to compute trajectories of taxiing aircraft. In the following, we thus include an overview of related values found in the literature.

We define a general speed limit of 15 m/s in line with the design taxi speed given in the A-SMGCS manual from ICAO [5]. Except for the dedicated wait-locations, agents must taxi at least with the minimal velocity of 1.5 m/s to avoid stop-and-go during taxiing. For curved taxiway segments, the ICAO manual mentions that speeds up to 10 m/s may occur. Most previous studies on airport surface movement operations define curved segments as turns with a maximal velocity of 5 m/s. Since we model taxiway curves explicitly through Bezier-curves (see Section 2.1), we define a speed limit v_{curve} per edge by using

$$v_{curve} = \sqrt{a_{lat} * r_{curve}}$$

with the lateral acceleration a_{lat} and the radius of curvature r_{curve} of the respective edge. To obtain r_curve, we use the median value of all curvatures per 1 m-segment of the underlying Bezier-curve. For passenger comfort in public transport, [6] provides a range for both longitudinal and lateral accelerations of ±0.9 m/s². Furthermore, they claim that a car driver with a normal driving style experiences a lateral acceleration of up to ±4 m/s² and a longitudinal acceleration of -2 m/s² to 1.47 m/s². In contrast, [7] found in empirical studies that the acceptable limits for passenger comfort are 1.23 m/s2 for longitudinal and 0.98 m/s² for lateral acceleration. As noted above, previous studies did not consider lateral accelerations to define turn speeds. For longitudinal acceleration/deceleration, different values are reported: for example, ±0.98 m/s², or 0.26 m/s² as acceleration and -0.8 m/s² as deceleration.

To find realistic values for the longitudinal and lateral accelerations of taxiing aircraft, we use historic track data from Schiphol captured by ADS-B receivers that record the aircraft positions during taxiing with a rate of 1 Hz. To this end, we map the positions onto the graph representing the taxiway centrelines, and smooth the resulting trajectories with a Savitzky-Golay filter (window length of 11 s, linear polynomial). This yields the travelled taxi distance along the graph edges as well as the speed and acceleration at each time point of the trajectory. However, we only use the data on the edges that correspond to the main taxiways: while the tracks become too noisy in the bay areas and at aircraft stands, the accelerations on runways for takeoff and landing are not representative of those experienced during taxiing.

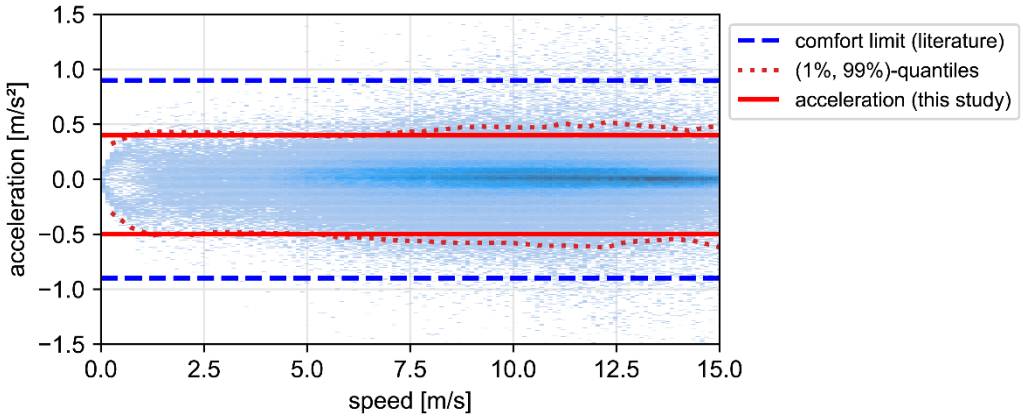


Figure 5. Calibration of longitudinal acceleration and deceleration values with historic track data.

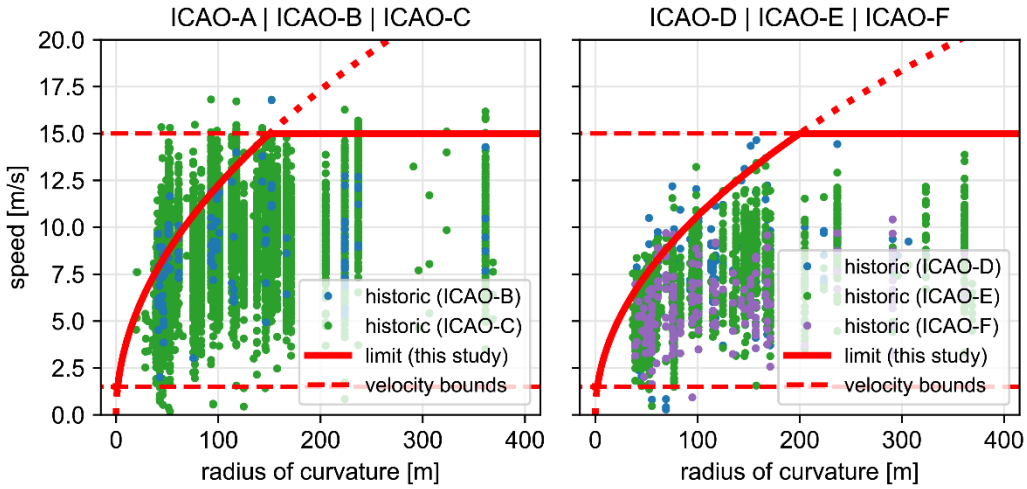


Figure 6. Calibration of curve speed with historic track data

In Figure 5, the acceleration over velocity of each data point is visualized as 2d-histogram. The 1% and 99% percentile lines of the acceleration values per 0.5 m/s step show that the longitudinal acceleration/deceleration remain similar across different taxi speeds. Therefore, we set the acceleration to 0.4 m/s² and deceleration to -0.5 m/s² independent of an agent's speed. While these values seem low compared to those mentioned in the literature, we argue that using these in planning increases the flexibility during execution: the Guidance Agents have more options to locally adjust the trajectories if necessary.

Figure 6 visualizes the historic curve speeds of different aircraft types as average speed along a curved edge with radius r. The average speed is calculated as $\bar{v} = \Delta d / \Delta t$ with the time difference Δt and travelled distance Δd of the data points per edge along each trajectory. Although higher curve speeds exist, we define the speed limit in curves based on a lateral acceleration of 1.5 m/s² for small aircraft (left plot) and 1.125 m/s² for large aircraft (right plot).

parameter	value	unit
maximal speed v_{max}	15	m/s
minimal speed v_{min}	1.5	m/s
curve speed v_{curve}	per edge	m/s
acceleration acc	0.4	m/s²
deceleration dec	-0.5	m/s²
safety distance in general	1	averaged shape radius
safety distance trailing	3	shape radii of preceding aircraft
planning window w_{plng}	15 to 60	min
replanning period h_{plng}	3 to 48	min (= 20 % to 80 % of w_{plng})

Table 1: Kinematic and algorithm parameters that are used in the routing algorithm

Table 1 summarizes the kinematic values and lists the main algorithmic parameters used by the routing algorithm. In general, two aircraft agents have to keep a minimal safety distance between them equal to the average of their shape radii. However, when an aircraft is trailing another agent, it has to keep a safety distance of at least 3-times the shape radius of the preceding aircraft, which is in accordance with experts. The planning window w_{plng} and replanning period h_{plng} are provided as ranges with the requirement that $h_{plng} < w_{plng}$.

3.2 Modelling human-automation interactions using the developed multiagent system model for path planning

In the ASTAIR project, workshops and interviews with human operators were carried out to determine requirements and desirable interactions between humans and automated systems. From these, regular occurring and characteristic situations were synthesized into eight use cases described in deliverable D1.2. In this section, we explore how the findings from these stakeholder workshops and interviews can be embedded into the automated multi-agent system and its path planning algorithms, with a focus on its technical implementation.

The developed multi-agent system provides the following tools to support the decision making of human operators concerning path planning:

1) *Updating Flight Schedule*: The start and goal locations as well as the start times of aircraft are obtained from the flight schedule (FS) provided through the A-CDM milestones and inputs from human operators. When updated schedule information is available, the ATCOs can update the FS entries accordingly.

2) *Adjusting Activity Sequence*: Per aircraft, the MAS creates an activity sequence to account for the different operations such as following a specific path for e.g. pushback, travelling to and holding at a holding point, etc. The ATCOs can adjust this sequence and its elements: they can for instance change the pushback path that must be followed, or the required holding duration. Likewise, data such as the estimated engine start-up duration is stored within the activity sequence and may be altered by human operators.

3) *Setting High-Level Parameters and Constraints*: The ATCOs can set and adjust various high-level parameters. These affect the path planning of all, a group of, or certain aircraft, and are valid for an extended period of time. Examples are:

   - adjusting the conformance level to the standard taxiway directions
   - blocking certain taxiway segments for maintenance work
   - adjusting the speed limits in general or in certain areas such as bay areas e.g. due to adverse weather conditions
   - setting general priority levels between aircraft groups e.g. arriving and departing aircraft

   Moreover, ATCOs can set constraints for specific aircraft that have a direct impact on their route. While some are valid for the entire taxiing, others are timed, i.e. issued for a specific duration. Examples are:

   - constraining the start or goal location in case an aircraft shall leave before or arrive after a certain time point
   - selecting a certain location to be passed during taxiing, affecting the aircraft's activity sequence
   - assigning a specific priority relation between two aircraft, i.e. one has right of way over the other

4) *Fast-forward simulation with/without Change-overlay*: The MAS plans the 4D-trajectories ahead of time, and deconflicts all routes within a pre-defined planning window. Thus, the ATCOs can inspect how the traffic will evolve in the upcoming period through a fast-forward simulation. As visualized in Figure 7 a), a colour scheme is used to indicate the predicted engine status, with the aircraft colour being green when engines are switched off, orange during engine-start, and red when they are running. The aircraft are categorized into one of six ICAO-types with the red circles indicating the associated type sizes. A cyan-coloured tug symbol indicates that a pushback-truck is coupled to the aircraft. As additional functionalities, the differences between two routing plans can

be visualized with a change-overlay as shown in Figure 7 b): one or multiple previous/inferior route alternatives are depicted by shadow-aircraft that are connected with a grey line to the new/superior solution. Furthermore, any aircraft path can be plotted into the visualization with the executed part in darker colour and the remaining path in lighter colour, as shown by the blue and orange trajectories in Figure 7.

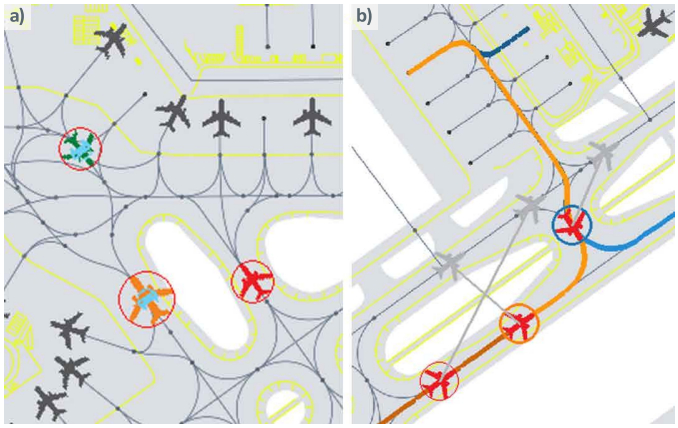


Figure 7. Examples of the fast-forward simulation with aircraft sizes.

5) *Calculation of SAMP Routes*: The MAS can calculate single-agent motion planning (SAMP) routes to quickly assess different routing options. These routes show the fastest possible paths that account for all relevant constraints, but are not yet coordinated with other traffic.

6) *Calculation of MAMP Routes*: Like the regular path planning, the MAS can also calculate alternative multi-agent motion planning (MAMP) routes based on altered input data, updated activity sequences, or changed high-level parameters or constraints like those mentioned in Section II-C3. However, to generate conflict-free trajectories for the selected set of vehicles, more computational time is needed than for calculating alternative SAMP routes.

In the following, we describe several use cases to demonstrate the abilities of the MAS to engage in interactions with human operators using its interaction tools. These use cases are inspired by the ASTAIR project use cases UC1-UC8 described in the deliverable D1.2 “Workshops report”, but adapted here to better illustrate how the algorithms presented in this deliverable work. In the following case descriptions we will refer to relevant UC1-UC8 to make this link more explicit.

3.2.1 Case ”Conformance to Taxiway Procedures of ATC” (related to UC4 “High level taxi strategy tuning” from D1.2)

In today’s operations, ATCOs use procedures and rules to create aircraft flows through the taxiway system. At Amsterdam Airport Schiphol, the main taxiways TWY-A, TWY-B, TWY-C, and TWY-D are used primarily in one direction as visualized in Figure 8. However, ATCOs may deviate from these taxiway procedures on their own discretion. In the historic data, most aircraft indeed follow the standard direction, with a few exceptions to e.g. go around an aircraft that is holding on a taxiway to start its

Co-funded by the European Union

engines. For fully automated operations, adhering to such procedures is not necessary, and paths could be optimized without such constraints, increasing the operational efficiency. However, such free-flow routes may not be comprehensible for human operators, especially when the traffic is dense. Furthermore, should the automation fail, the human operators may not be able to resolve and continue the operations. Therefore, for human-automation teaming, the ATCOs must be able to adjust the conformance level to taxiway procedures of the path planning, which we explore in the following.

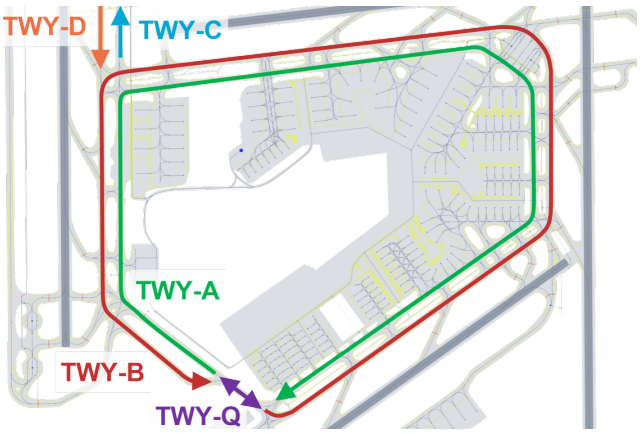


Figure 8. Standard taxiway directions as part of ATC procedures at Amsterdam Airport Schiphol

The standard taxiway directions visualized in Fig. 3 must be known to the MAS. The conformance to these taxiway procedures of ATC is then adjusted by changing the associated cost of traversing a taxiway segment c_{seg} during path planning by multiplying it with a cost factor c_{TWY} :

$$c_{seg} = (t_{taxi} + c_d * d_{taxi}) * c_{TWY}$$

with the taxi time t_{taxi} and taxi distance d_{taxi} along that segment. c_d = 0.1 s/m to convert the distance to unit time. For example, when c_{TWY} = 5, it is five-times more expensive to traverse that taxiway segment in comparison to one with c_{TWY} = 1 for identical taxi time and distance. Since the sum-of-cost is minimized during path planning, non-standard taxiway directions that are assigned higher values for c_{TWY} are less likely chosen.

In the future, instead of letting the ATCOs set the conformance level manually, they could opt for letting the MAS do so dynamically e.g. based on the number of flights to be routed in each planning round and the learned preferences of ATCOs. Moreover, other high-level routing parameters could be adjusted as well, for example:

- Setting default priorities e.g. between arrivals and departures or between aircraft and ground vehicles: giving higher priorities to a certain group will tend to decrease their taxi times. Nonetheless, the ATCOs can deviate from these default priorities by assigning a specific priority-value to any vehicle.

- Setting the general speed limit for the entire taxiway system or certain areas like the bay areas, foremost dependent on the weather situation. Potentially, a data-driven AI subsystem could pose recommendations based on historic weather patterns.

- Setting the minimal or maximal duration that an aircraft holds at a remote holding location. This will impact the decision-support provided by the MAS for e.g. arriving aircraft whose stand is still unavailable for a certain duration.

3.2.2 Case " Departing Aircraft is Delayed" (related to UC2 "Normal operations with re-scheduling" from D1.2)

In real-world airport operations, delays frequently arise out of various reasons, and have to be dealt with. When planning the taxi routes of aircraft, delays occurring prior to the predicted start of the route can be counteracted by updating the prediction and replanning the route accordingly. Many delays may remain unknown to automated systems, and updating the predictions may require coordination among human operators as well as their expertise and problem-solving skills. Therefore, to achieve effective human-automation teaming, such changes and prediction updates must be steadily supplied to the MAS. In the following, we explore one such example.

All routes of aircraft are based on the flight schedule: per flight, the respective start point and time as well as the goal location are extracted from it when forming its activity sequence. The corresponding activities are updated when flight schedule entries change. The updates are then accounted for when the MAS replans the routes of all flights that are or will be taxiing within the planning window. If necessary, the replanning can also be triggered rule-based or manually. In the example, the target off-block time as start time of the route is updated.

In a similar way, any adaptations to the flight schedule are handled, e.g.:

- Assigning a new stand to an arriving aircraft: the goal location in the activity sequence is adapted.
- Allocating deicing to a departing flight in winter conditions: intermediate activities are inserted into the sequence that demand the aircraft to taxi to one of the deicing locations at which it has to hold for a specific time to receive the deicing.
- Changes to the takeoff slot assigned by Eurocontrol (i.e. CTOT-slot): the constraining time to be at the runway is adjusted, potentially also affecting the holding time at the stand and/or remote holding location.

In general, any flight schedule change may lead to knock- on effects, e.g. trigger a stand conflict (i.e. the delayed departing flight is blocking the stand that an arriving aircraft is assigned to) that have to be resolved as well. This may entail further interactions between the MAS and the human operators. However, as such interdependent effects occur mostly after the original cause, there is likely more time for the human operators to request and act upon recommendations from the MAS or make informed decisions to resolve them.

3.2.3 Case "Non-standard Pushback Path" (related to UC4 "High level taxi strategy tuning" from D1.2)

At Schiphol, the standard pushback and push-pull paths dependent on the aircraft types are defined in the airport manuals for all stands. We integrated these paths in the airport layout and use them as part of the activity sequence for outbound flights. However, based on the historic data as well as on

interviews with operational experts, it can be concluded that the ATCOs deviate from these standard pushback procedures in around 20 % to 30 % of times to further optimize the flows and taxi times of the involved aircraft. Thus, the MAS must accommodate to receive and process such informed changes based on the experience of the ATCOs.

By specifying a non-standard pushback path, the activity sequence is automatically updated: during the pushback-activity, the aircraft must follow the new path. While the changes are still pending the acceptance by the ATCO, the MAS keeps both the original as well as the new activity sequences in cache. Likewise, it keeps a copy of the original MAS solution. When queried by the ATCO, it replans the potential routes of all agents using the new activity sequence of the departing aircraft. Once the new solution is accepted by the ATCO, the MAS automatically sends the new routes to the affected aircraft.

3.2.4 Case ” Unavailability of Stand” (related to UC3 “Arriving traffic without parking” from D1.2)

The stands form one of the bottlenecks in the capacity of airports, and the aircraft stand allocation is a manifold problem of its own. Due to delays, arriving ahead of time, or allocation constraints among others, a stand may still be occupied by another aircraft at the time that an arriving aircraft could enter it. The route of the arriving aircraft must thus be adapted. Such cases are visible in the historic data, but were also raised in interviews with experts as both a common operational challenge and interesting use case for airside automation. In the following, we provide an interactive example how the unavailability of a stand can be resolved using the agent-based framework of human-automation teaming.

Consider the following scenario: Shortly before the flight AC- 7 is landing at Schiphol’s runway 06, the ATCO is informed by the ground handler that the chosen stand D04 is blocked / unavailable for another 20 min. Since the aircraft is landing soon, the MAS has already planned a conflict-free route. The ATCO displays the planned route (path displayed in Figure 9a), and notices that the aircraft would arrive approximately 15 min too early at the stand. First, ATCO places a pending goal-constraint at the stand D04 for the blocked time period and requests the MAS to compute initial alternatives without accounting for other traffic, i.e. SAMP routes. The MAS displays three initial solution strategies: (1) assign an alternative stand to the aircraft (path b), (2) let it hold at one of the remote holding points (path c), and (3) let the aircraft take a detour along TWY-B and TWY-A with minimal taxi speed (path d). With these options in mind, ATCO uses the fast-forward simulation to get an impression of the traffic situation in the upcoming 30 min. In turn, she disregards all three options: the alternative stand (1) is in another bay area and will likely not be acceptable for the airline, all remote holding points (2) are already occupied and would require extensive changes to the routes of other aircraft, and she deems (3) to be impractical given the amount of traffic in the upcoming period. From her experience, ATCO knows that the aircraft could alternatively hold in bay area C/D. She selects the chosen point as non-standard holding location at which AC-7 shall hold for 15 min, and lets the MAS compute the SAMP route (path e). After checking her resolution option using again the fast-forward simulation, she accepts the changes to the activity sequence of aircraft AC-7. The MAS carries out partial replanning of the MAMP routes and notifies Anne that the changes have taken effect.

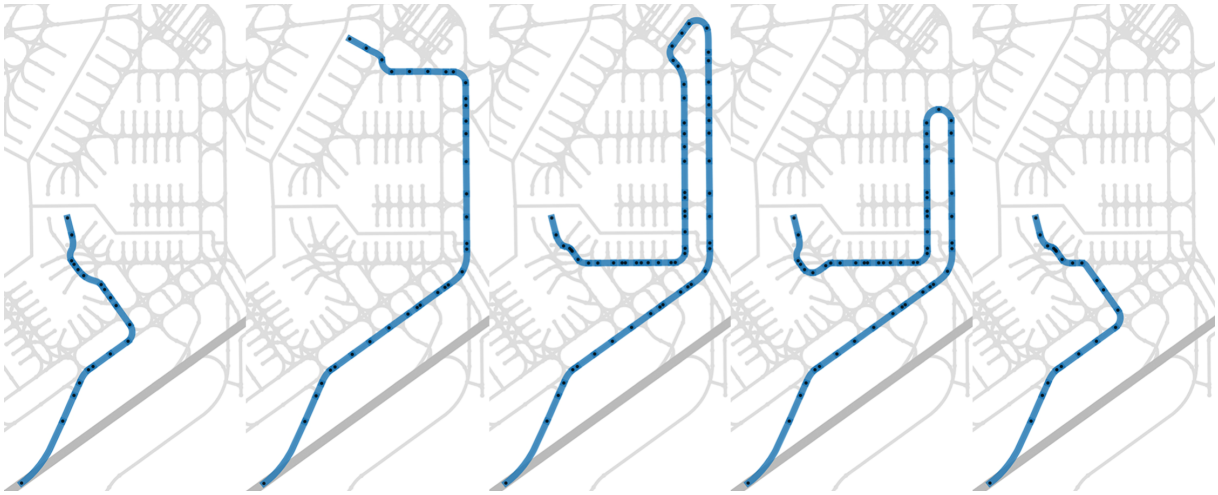


Figure 9. The stand of an arriving aircraft is unavailable: SAMP paths of a) original route, b) rerouting to a different stand, c) holding at a remote holding point, d) detour with minimal speed, and e) holding at a non-standard holding point

In this scenario, through the goal-constraint, AC-7 is not allowed to arrive at the stand prior to the end time, which is taken into account as strict requirement during path planning by the MAS. The MAS uses a set of options such as rerouting to another stand, or holding at a remote holding location among others to create appropriate activity sequences and let the MAS determine the corresponding SAMP-paths. As none of these route alternatives appear suitable to the ATCO, another activity sequence is created from the inputs provided by the ATCO: the chosen holding location is added as intermediate goal at which the aircraft must wait for the selected duration. The MAS carries out a partial replanning by deconflicting the routes of all aircraft that are affected by the new route of AC-7.

3.2.5 Case ”Emergency demands Aircraft to Return to Stand” (related to UC8 “Arriving flight with technical issue” from D1.2)

Non-nominal situations may occur infrequently, but often require non-standard resolution strategies. Especially in such situations, the automated side of the human-automation teaming must provide a flexible interface for effective decision-support. In the following, we consider a scenario in which a departing aircraft must return to its stand due to an emergency.

In the MAS, the activity sequence must be redefined so that the aircraft is not routed further to the runway, but instead back to the original or alternative stand. Dependent on the emergency and general traffic situation, the ATCO can adjust the priority of the aircraft in comparison to other traffic: a high priority yields a fast return route while potentially more traffic is affected, whereas a low priority potentially lengthens its taxi time but creates minimal nuisance to the routes of other aircraft. For various exemplary emergency situations, changing the priority of the emergency-declaring aircraft did not have a significant impact on its own as well as the trajectories of affected aircraft. This suggests that the Routing Algorithm is able to recover to the desired level from such situations. For the default priority, Fig. 9 visualizes the original route of AC-8 (a) as well as its route back to the stand. The effects on other traffic with respect to the original planning are visualized by shadow-aircraft.

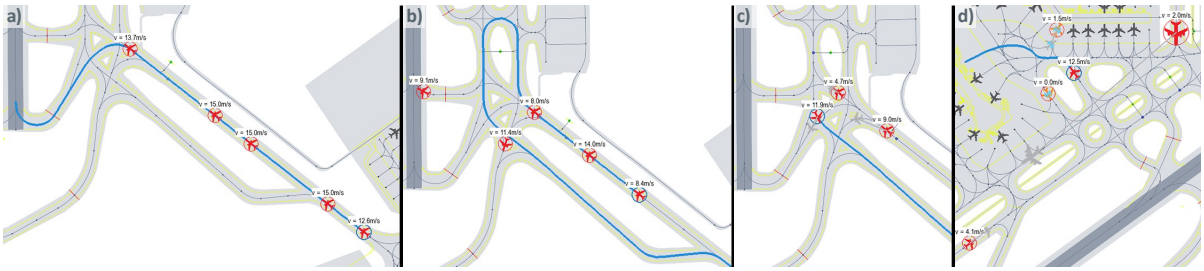


Figure 10. The blue-circled aircraft declares an emergency: a) shows the original route to the runway, b)-d) its route back to the stand with the effects on other traffic shown by shadow-aircraft

References

[1] M. Dorigo and L. M. Gambardella, “Ant colonies for the travelling salesman problem,” *Biosystems*, vol. 43, no. 2, pp. 73–81, Jul. 1997, doi: 10.1016/S0303-2647(97)01708-5.

[2] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, “Searching with consistent prioritization for multi-agent path finding,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, pp. 7643–7650.

[3] M. Phillips and M. Likhachev, “SIPP: Safe interval path planning for dynamic environments,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 5628–5635. doi: 10.1109/ICRA.2011.5980306.

[4] V. Treve and F. Rooseleer, “‘RECAT-EU’ European Wake Turbulence Categorisation and Separation Minima on Approach and Departure.” EUROCONTROL, Aug. 11, 2024.

[5] I. C. A. Organization (ICAO), “Advanced Surface Movement Guidance and Control Systems (A-SMGCS) Manual,” 2004.

[6] I. Bae *et al.*, “Self-Driving like a Human driver instead of a Robocar: Personalized comfortable driving experience for autonomous vehicles”.

[7] K. N. de Winkel, T. Irmak, R. Happee, and B. Shyrokau, “Standards for passenger comfort in automated vehicles: Acceleration and jerk,” *Appl. Ergon.*, vol. 106, p. 103881, Jan. 2023, doi: 10.1016/j.apergo.2022.103881.

List of acronyms

Acronym	Description
ACO	Ant Colony Optimisation
ETV	Electric Towing Vehicle
FS	Flight Schedule
MAMP	Multi-Agent Motion Planning
MAPD	Multi-Agent Pickup and Delivery
MAS	Multi-Agent System
PBS	Priority-Based Search
SAMP	Single Agent Motion Planning
SIPP	Safe Interval Path Planning
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem

Table 2: List of acronyms