Efficient Airport Taxiing Operations: Conflict-Based Multi-Agent Path Finding with Speed Deviations

Axel Ehrnrooth

Vrije Universiteit, Amsterdam, NL

Abstract. This research examines the performance of Conflict-Based Search (CBS) for Multi-Agent Path Finding (MAPF) in airport taxiing operations while accounting for speed deviations. By simulating various traffic density conditions, the study evaluates CBS's computational complexity and path optimality. The results demonstrate that CBS, despite being computationally intensive, effectively handles speed deviations in real-time. Deviations lead to more frequent re-planning, yet re-planning due to deviations requires less CPU time on average than initial planning. Because of this, CBS maintains its performance when accounting for speed deviations, suggesting robustness in adapting to aircraft behavioural changes. While the impact on path optimality is generally insignificant, it becomes notable in high-density traffic scenarios. These findings support CBS as a robust method for real-time automated taxi routing, highlighting its potential to enhance operational and environmental efficiency in complex and dynamic airport environments.

Keywords: Multi-Agent Path Finding · Engine-Off Taxiing · Speed Deviations.

1 Introduction

Conventional airport taxiing operations significantly contribute to airport-related pollution [4], [11]. Initiatives aimed at enhancing the operational and environmental efficiency of airport operations propose methods to optimize taxiing paths through automation [6], [14]. However, a problem arises when aircraft do not strictly follow their planned paths. In an ideal scenario, aircraft adhere to their plans, but in reality, deviations can occur, potentially invalidating the overall plan. This study aims to highlight the effectiveness of a Multi-Agent System for optimizing taxiing operations while accounting for speed deviations.

A Multi-Agent System (MAS) is a system in which multiple autonomous artificial entities, known as agents, collaborate to solve complex problems. The motivation for using multi-agent systems is to tackle otherwise complex problems by dividing them into smaller tasks that agents solve individually, providing coordinated solutions to form complete solutions. MAS has proven especially useful for handling tasks that require intricate planning, coordination, and robustness [24],

[5]. The versatility of MAS stems from the agents' abilities to act autonomously while communicating with each other to optimise their actions. A popular application of MAS is agent-based modelling to simulate complex environments [5]. For example, agent-based simulations can replicate airport operations to study the behaviour of aircraft and ground vehicles. These simulations are valuable for analysing existing systems and evaluating potential MAS solutions to current problems [5]. This research project aims to model the taxiing environment of an airport under various traffic density conditions, with the objective of evaluating the performance of a multi-agent system in coordinating taxiing operations.

A sub-genre of MAS problems is the concept of Multi-Agent Path Finding (MAPF). MAPF presents a system involving multiple agents, each with individual start and goal locations, cooperating to find a collision-free set of paths from their respective starting locations to their destinations [20]. MAPF is suitable for solving the airport taxiing problem as it manages the movement of numerous aircraft within a constrained environment, ensuring that all agents reach their destinations efficiently and without collisions. The MAPF solution is a plan comprising a set of actions assigned to each agent that, when executed, results in each agent reaching its goal position. As the number of agents increases, the likelihood of the optimal path for one agent colliding with that of another agent also increases, leading to conflicts. A MAPF solution is valid only if the set of all paths is free of conflicts [20].

There are numerous ways to handle conflicts in MAPF scenarios, each with varying levels of complexity. The fundamental concept involves imposing constraints on agents to prevent collisions. Two prominent MAPF algorithms are Priority-Based Search (PBS) and Collision-Based Search (CBS). PBS orders agents by priority and in case of conflict, assigns a constraint to the agent with the lower priority, resulting in a depth-first search, finding solutions quickly but not always optimally. The effectiveness of PBS depends heavily on priority assignment, especially as the number of agents increases [9]. Conversely, CBS uses a best-first search approach, assigning constraints to both agents in a conflict and finding a solution with the lowest total cost (e.g. path length). The method always yields an optimal solution but becomes increasingly computationally intensive as the number of agents increases [18]. This research uses CBS to find optimal paths for taxiing aircraft between the runways and gates, prioritising efficiency in taxiing operations. The reasoning behind the decision to use CBS over PBS is based on the inherent weakness of PBS in terms of priority assignment. Misassignment of priorities and the consequent potential for sub-optimal solutions can cause greater inefficiency in the final solution. Avoiding such inefficiencies is crucial for achieving optimal operational standards for airport taxiing.

This research addresses agent deviations as differences in planned and true taxiing speeds [14]. Since MAPF paths are planned in both space and time, speed deviations lead to discrepancies in the planned and actual paths, invalidating the original solution and undermining the efficacy of automated path planning. Given the issues caused by speed deviations, it is essential to investigate their impact on path planning. Current research focuses on automatic generation of

optimal taxiing paths while pilots remain in control of manoeuvring the aircraft. Deviations are at the discretion of pilots and air traffic control operators [6]. When a deviation occurs, the original solution no longer holds, necessitating a re-planning of the system with new start times and locations for all agents to ensure the global solution remains conflict-free.

This project studies the effect of speed deviations on MAPF within semi-automated taxiing operations through agent-based modelling. The analysis focuses on path optimality, computational time, and re-planning frequency, highlighting features most affected by deviations. Additionally, the study examines the impact of deviations on overall system efficiency and explores potential for more robust planning algorithms.

2 Problem Statement and Project Motivation

2.1 Environmental Impact

The global demand for air-travel has grown consistently since the steep decline during the COVID-19 pandemic [10]. As the airline industry expands, so does the associated pollution. To address such topics, various initiatives exist. One such initiative is the Single European Sky ATM Research (SESAR) project, which has recognised the need for digitalisation and automation in modern air travel, with a development time frame established for 2035. The SESAR Concept of Operations 2019 report proposes future surface movement operations aimed at making air traffic more efficient both environmentally and operationally [6].

While engine efficiency is heavily optimised for cruise conditions, conventional jet engines are extremely inefficient for ground propulsion [4]. A significant portion of a flight's emissions is attributed to the time spent on the ground. The most significant determining factor of fuel consumption during taxi operations is taxi time. At large airports designed to accommodate hundreds of flights per day, taxi times can reach up to 37 minutes on average [4], [11].

Initiatives to reduce ground movement emissions include single engine off taxiing and more novel engine-off solutions such as on-board systems and external tug vehicles [8], [21]. The most prominent technology of this category is TaxiBot, a specialised tug vehicle for taxiing operations [8], [16]. When attached to the aircraft, the pilot steers the TaxiBot using the aircraft's conventional controls without requiring modifications to the aircraft. TaxiBot has been tested at major airports, and due to the promising nature of the TaxiBot and the SESAR initiative's intent on preserving pilot control, this research will assume the retention of pilot control during taxiing [6], [17].

Though the TaxiBot concept offers a promising future in terms of emissions reduction, a scheduling and coordination challenge remains. Amsterdam Schiphol Airport often handles up to 100 in- and outbound aircraft per hour during peak hours while slots can remain unused during off-peak hours [13]. Implementing engine-off taxiing during peak hours requires multiple TaxiBots, potentially causing bottlenecks on maintenance roads due to the increased density of active ground vehicles [23].

To effectively address the operational and environmental challenges discussed in this section, it is crucial to implement and automate these methods with a robust planning and coordination solution. Multi-Agent Path Finding (MAPF) offers promising approaches to achieve this. By optimizing the movement of aircraft and autonomous vehicles on the ground, MAPF can significantly reduce taxi times, lower fuel consumption, and minimize emissions. This not only enhances operational efficiency but also supports sustainability goals by decreasing the environmental footprint of airport operations. The adaptability and efficiency of MAPF algorithms make them well-suited to handle the complexities and dynamic nature of airport environments. Therefore, integrating MAPF into airport ground movement systems presents a compelling opportunity to meet both logistical and environmental objectives effectively.

However, since SESAR and similar initiatives are not intending on fully automating taxiing operations but plan to keep pilots in control [6], [14], [8], deviations from planned paths are inevitable as they inherently involve human decision-making. This prioritization of human decisions introduces variability in aircraft movements, which must be accounted for in automated planning systems.

3 Research Questions and Objectives

This research addresses uncertainty in combining automation with human decision-making, focusing on agent deviations. CBS systematically explores the solution space to find the lowest-cost solution, making it optimal and complete. However, it is not explicitly designed for frequent deviations. Unexpected changes in agents' movements can cause new conflicts, invalidating CBS's solution. While CBS is resilient to occasional changes [7], frequent deviations can worsen the overall solution, potentially increasing taxiing time and congestion. Knowing that the involvement of more agents leads to longer processing times, the expectation that deviations affect congestion suggests that they can also lead to a further negative effect on the processing time of CBS.

3.1 Research Questions

To evaluate the impact of speed deviations in automated path planning for taxiing operations, the following research question arises: **How is the overall performance of CBS for automated path planning in airport taxiing operations affected when accounting for speed deviations under various traffic density conditions?**

To evaluate performance under the new conditions, the research question can be split into two sub-questions, each tackling a different aspect of the algorithm's performance. The first sub-question leads as follows: **How does path planning, when accounting for speed deviations, impact the computational complexity of CBS as the number of agents increases?** This question specifically targets the computational performance of the best-first search algorithm,

which suffers as the number of agents increases. Anticipating that deviations will disrupt the initial plans, combined with the lack of anticipatory measures to handle deviations, a vicious cycle of re-planning is expected. The presence of multiple agents deviating at unexpected times leads to constant re-planning, potentially bringing more complex problems and increased computational intensity.

The second sub-question that emerges is: **To what extent is the solution optimality of CBS achieved when accounting for speed deviations?** This question examines the optimality feature of CBS under uncertain conditions. If agents' optimal paths collide, at least one agent may have to take a sub-optimal path. When an agent is forced to take a sub-optimal path and then deviates from it, a new sub-optimal continuation of the path may be planned. The accumulation of sub-optimal paths can lead to consistently worsening solutions, especially as re-planning frequency increases.

3.2 Research Objective

This research addresses computational and operational challenges from speed deviations in automated taxiing. It evaluates the impact of deviations on CBS's computational complexity and path optimality. By simulating different traffic intensities, it assesses conditions that compromise system efficiency. The findings aim to provide insight into the feasibility of a real-time MAPF solution, supporting SESAR's automation initiatives.

4 Related Work

4.1 Multi-Agent Path Finding

"The Multi-Agent Path Finding (MAPF) problem is the fundamental problem of planning paths for multiple agents, where the key constraint is that the agents will be able to follow these paths concurrently without colliding with each other" [20]. The foundation of a MAPF problem is the set of starts and goals of the agents involved, and the formulation of a set of tasks for each agent to reach its goal free of conflicts with other agents and static obstacles. This report adopts the convention of the MAPF problem formulated by Stern et al.

The multi-agent environment is represented as a graph $G = (V, E)$ with k agents. Each agent is mapped to a source vertex in the set $s : [1, \ldots, k] \rightarrow V$ and to a target vertex in the set $t : [1, \ldots, k] \rightarrow V$. At each time step, an agent performs an action $a : V \rightarrow V$ where $a(v) = v'$ and v' is the vertex location after the action. The two basic actions that an agent can perform are to wait and to move. The wait action implies that $v = v'$, while the move action implies that $v \neq v'$ and v' is a vertex adjacent to v. The sequence of tasks performed by an agent i is denoted as $\pi_i = (a_1, \ldots, a_n)$.

To reach a vertex at time x, an agent i will perform a sequence of actions from its source $s(i)$ denoted as $\pi_i[x] = a_x(a_{x-1}(\ldots a_1(s(i))))$ where $\pi_i[x]$ equates

to the target position $t(i)$ if and only if the execution of the set of tasks $\pi_i[x]$ leads to the target. The problem is solved when each agent has a set of actions that brings it to its target [20]. For the purposes of this project and for clarity, the solution for an agent will also be referred to as a path.

Another essential component of any MAPF problem is the concept of conflicts. A MAPF solution is only valid so long as it is free of conflicts. Stern et al. identify five essential categories of conflicts: vertex, edge, swapping, following, and cycle conflicts [20]. Due to the nature of this project, only the first three are relevant, along with several new conflicts that emerge due to the varying speeds of the agents, which will be discussed in further detail in section 5.1.

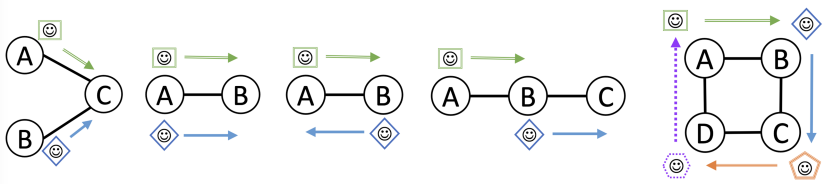

Fig. 1. Conflict types [20]

A vertex conflict occurs when two agents i and j occupy the same node at the same time, formally expressed as a conflict between π_i and π_j where at time x, $\pi_i[x] = \pi_j[x]$. An edge conflict occurs when two agents move on the same edge at the same time, formally expressed as a conflict between π_i and π_j where at time x, $\pi_i[x] = \pi_j[x]$ and $\pi_i[x+1] = \pi_j[x+1]$. It is important to note that disallowing vertex conflicts automatically forbids edge conflicts, as both conditions of an edge conflict are themselves vertex conflicts. Finally, a swapping conflict is similar to an edge conflict in that both agents traverse the same edge at the same time, but their start and end positions are swapped. A swapping conflict is formally expressed as a conflict between π_i and π_j where at time x, $\pi_i[x+1] = \pi_j[x]$ and $\pi_i[x] = \pi_j[x+1]$ [20].

4.2 Conflict-Based Search

Finding a valid solution to a MAPF problem involves resolving all conflicts between agents. The cornerstone of MAPF conflict resolution is the imposition of constraints on the conflicting agents to avoid collisions. For example, if two agents are planned to occupy the same position at the same time, a conflict occurs and must be resolved. Adjusting the plan to ensure that at least one of the conflicting agents is not at the node in question removes the conflict and constitutes a resolution. This is accomplished using constraints that forbid agents from being at the collision location at the time of conflict [9]. Conflict-Based Search (CBS), introduced by Sharon et al., is one of many methods for detecting collisions, applying constraints, and resolving conflicts in MAPF problems [18].

The greatest benefit of CBS is that it is both optimal and complete, meaning that it always finds an optimal solution if a solution exists. CBS achieves optimality by minimising the cost when applying constraints.

Formal Definition of CBS The following description of CBS is based on the formulation of the algorithm in [9], [18]. CBS consists of a high-level and a low-level search. The high-level search is responsible for exploring the solution space of the algorithm, which is created by imposing constraints. The solution space is a search tree structure known as a constraint tree (shown in figure 2), containing nodes with a list of imposed constraints, a set of paths associated with each agent, and a cost determined by the sum of the path lengths.

The root node R initialises the search by finding initial paths for all agents, which can be done optimally using an individual A^* search of the environment. If this set of paths is collision-free, it returns the paths as a solution to the problem. However, if it detects a conflict between agent a and agent b at time t, R is expanded, generating two child nodes Q_1 and Q_2, where Q_1 constrains agent a and Q_2 constrains agent b. Each child node will generate its own set of paths, adhering to the newly imposed constraints. This part of the search is known as the low-level search and works similarly to the initial search, the difference being that it takes constraints into account. The low-level search provides both Q_1 and Q_2 with new costs based on their respective paths.

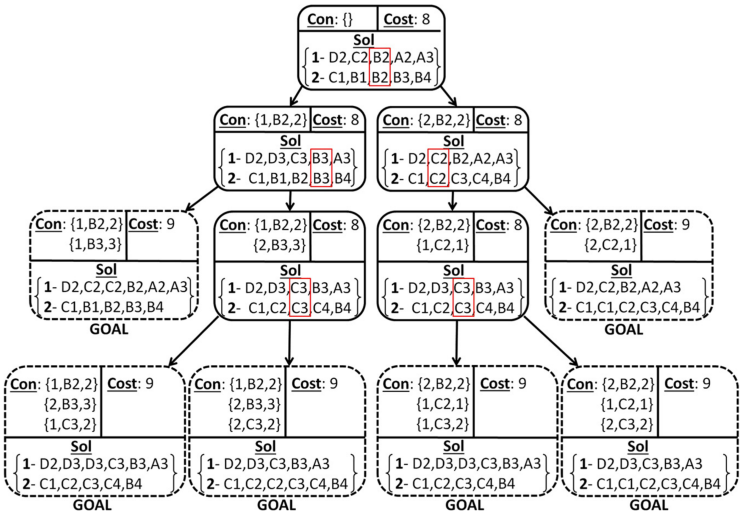

Fig. 2. CBS Constraint Tree [18]

If the paths in a child node are collision free, a solution has been found, and the node becomes a goal node (though a goal node is not necessarily the optimal

solution). The high-level search continues by comparing the costs associated with the child nodes, and if the lowest-cost node is a goal node, its paths are considered an optimal solution. However, if Q_1 is a goal node and Q_2 is not but has a lower cost, CBS will expand Q_2, generating two new child nodes Q_3 and Q_4, repeating the process with the additional nodes. Q_3 and Q_4 have two constraints each: one being the original constraint imposed on Q_2, and the other a unique constraint based on the collision in the paths of Q_2. When the lowest-cost node is a goal node the optimal solution is found, and CBS returns the set of paths in that node to the agents [9], [18].

Testing CBS in benchmark MAPF environments revealed that navigation in open spaces tends to result in more conflicts, making CBS less suitable for such areas. Conversely, CBS exhibited exceptional effectiveness in environments with narrow paths, outperforming other approaches [18]. The algorithm's effectiveness in confined spaces is a promising feature for its application in path planning for taxiing procedures. Given that taxiways are designed as narrow paths with strict operational rules, the entire ground operations environment is comparable to a map consisting of narrow corridors rather than open spaces.

While CBS finds optimal solutions in terms of cost, the optimal solutions assume that all agents will follow the planned paths precisely. Moreover, path optimality does not imply robustness against uncertainty. To explore the use of CBS as a method for taxi operations path planning, it is imperative to understand the impact of uncertainty and unexpected behaviour on the algorithm's effectiveness. This research will utilise agent-based modelling to provide a comprehensive analysis of the effects of agent deviation on CBS.

In addition, Boyarski et al. have expanded Sharon et al.'s work, enhancing the performance of CBS by presenting the *Improved Conflict-Based Search* (ICBS). This new approach aims to optimise the performance of CBS by modifying various essential components of the classical CBS. When all component optimisations are applied, ICBS significantly improves computational time [2]. This project will consider these improvements to further discuss the potential of CBS and its variants for optimising taxiing operations.

4.3 MAPF Applications and Automation at Airports

"Follow the Greens" is an airport taxiing specific concept introduced by the SESAR initiative [14]. Developed to reduce ground controller workload while optimising taxiing procedures and minimising errors, this concept utilises ground lighting aids on taxiways to guide aircraft across the airport. The system illuminates the taxiway centre-line (in green) that an aircraft should follow, adjusting the paths according to the aircraft's requirements. Path planning is determined by a priority-based model and is displayed in real time to the ground controller, who can alter the planned path at their own discretion to better align with their overall strategy. The concept has been tested at multiple airports and has shown success in practical implementations [14].

The success of Follow the Greens demonstrates the potential of automation in taxiing operations. Implementing complete MAPF routing solutions in con-

junction with the Follow the Greens concept could further enhance efficiency and coordination. However, to effectively reduce workload while optimising planning and coordination, it is crucial to fully understand the resilience of MAPF algorithms to sudden, constant changes in behaviour.

MAPF has demonstrated tremendous success in industries where complex coordination and planning are essential. A prominent example of MAPF applications is in large-scale warehouses such as Amazon's [24]. A study conducted by Li et al. showcases the effectiveness of MAPF solutions though a Rolling-Horizon Collision Resolution (RHCR) framework, which enables continuous operation of autonomous warehouse agents. The framework effectively coordinates up to 1000 individual agents, occupying more than a third of the limited space available in the warehouse environment [12].

MAPF implementations have also been heavily motivated and widely proposed for solving complex coordination problems in the aviation industry, relevant to both airside and ground operations. One such project explores resilience of airport surface movements coordinated by MAPF solutions [7]. The study examines the concept of runway reconfigurations, a common environmental change at airports, and their impact on taxiing performance. The study compares the real-world performance of re-coordination after runway reconfigurations to the performance of CBS in the same scenario. Three CBS algorithms are compared, each analysing their environments uniquely. The results demonstrate that CBS can effectively plan according to environmental changes, proving its resilience in this aspect [7]. However, while resilience is proven for environmental changes, agents' speed deviations are not explored. Exploring how CBS handles constant speed deviations provides an additional layer of understanding in terms of its resilience to change.

Chen proposes a multi-agent system as a route planning method for taxiing operations, which uses the agents' flight schedules to assign priorities. Unlike conventional PBS, this method considers factors such as fuel consumption and taxi time, rather than simply assigning priorities. In this proposed solution, agents dynamically adjust their paths based on guidance from an overseeing route management agent that provides time windows for agents to pass through nodes in the airport environment. Allowing dynamic path adjustments drastically reduces taxi time compared to the fixed path method to which it is compared, showing improvements by up to 19.6% [3]. The approach is innovative compared to other MAPF proposals, and the results are impressive, effectively modelling and demonstrating the power of an adaptive environment.

However, Chen's method fails to address the concept of deviations. While the agents' paths are flexible within the time windows, they still adhere to the plan assigned by the MAS. In the real-world applications discussed in this report, human decision-making is prioritized [6], [14], significantly influencing a system's true performance. This This research aims to address the impact if agent deviations on the performance of MAPF systems, providing insights into how such deviations affect overall efficiency.

A study conducted by Siebers addresses planning under uncertainty, aiming to create a robust scheduling system for airport operations, including taxiing [19]. It identifies logistical bottlenecks and weather conditions as sources of uncertainty and proposes an agent-based model to predict arrival and departure times, accounting for these uncertainties. While the model shows robustness in routing and timing, it struggles to detect and resolve agent conflicts [19]. This is where MAPF and CBS excel. Unlike Siebers' research, this study does not account for airside operations and long-term scheduling but optimises taxiing procedures and handles conflicts arising from speed deviations. Siebers emphasises the importance of studying multi-agent planning methods to address the specific problem of conflict management under conditions created by uncertainty [19].

Siebers' suggestions for improvements highlight the existing research gap effectively. Although there are numerous proposals and applications of MAPF for airport taxiing operations and similar domains, the concept of deviations has not been sufficiently addressed. While some research employs predictive methods to anticipate delays or mange uncertainties, others provide theoretically effective and efficient solutions, suggesting a promising future for full automation in airport coordination activities. However, currently any automation must be resilient to uncertainties and unexpected outcomes [6]. This research aims to demonstrate that CBS can serve as a viable MAPF solution for coordinating airport taxiing operations, provided that the planned solutions are adhered to.

5 Research Method

To ensure robustness and applicability, the structure of the environment, visualisations, and the agents involved in the experimentation for this project are built upon a modified version of the existing solution from the graduation work of K. Fines and T. Noortman on airport taxi operations [7], [15]. The CBS algorithm is implemented according to the structure described by Sharon et al. [18]. The environment is modelled as a weighted and undirected graph, representing an abstract taxiway layout (figure 4). The speed of the agents is determined by a speed profile associated with each agent. An agent's speed profile also dictates if a speed deviation will occur and provides reasoning behind the agents' decisions to deviate, replacing random deviations. It is assumed that once an agent has deviated, it will maintain its new speed until it reaches its goal.

5.1 Research Approach and Experimental Setup

The airport taxiing operations coordination problem is presented in this research as a classical MAPF problem and is studied through an agent-based modelling approach. The agents represent aircraft that either arrive or depart at specific times during the simulation and move between runways and the terminal. A unique property of an agent in this specific environment is its speed, which is determined by a speed profile within the agent's internal state. The speed profile

also dictates whether the agent will change speed during taxiing, causing it to deviate from its originally planned space-time trajectory.

The fact that agents move at different speeds means that the usual conflict types explained in [20] no longer sufficiently classify all types of collisions. Three new collision variants are added besides the swapping, edge, and vertex collisions. The root of all new collisions lies in the different time step resolutions for agents travelling at different speeds. Consider agents a and b, where a is faster than b, and a plan for an agent as a list of locations and times: $[(l_1, t_1), (l_2, t_2), \ldots, (l_n, t_n)]$. If agent b has the plan $[(A, 1), (B, 2), (C, 3)]$ and agent a has the plan $[(C, 2), (B, 2.5), (A, 3)]$, the agents are travelling the same paths but in opposite directions. Both agents are planned to travel on the same edge (B to C and C to B), thus swapping nodes and starting at opposite ends but reaching the other side at different times. Recalling the conditions for a swapping collision in [20] in section 4.1, and applying it to the relevant agents, the condition $\pi_a[x+1] = \pi_b[x]$ and $\pi_a[x] = \pi_b[x+1]$ constitutes a swapping collision. When the agents travel at different speeds $\pi_a[x+1] = \pi_b[x]$ might hold, but $\pi_a[x] \neq \pi_b[x+1]$ even if they traverse the same edge. This new condition is also true when the conflict is reversed. If a is faster than b, then if $\pi_a[x] = \pi_b[x+1]$, then $\pi_a[x+1] \neq \pi_b[x]$. This conflict can be seen in figure 3(a).

The second conflict is a vertex conflict where the slower agent b is traveling to a node and agent a catches up and collides with agent b from the rear. Consider the paths $[(A, 2), (B, 2.5), (C, 3)]$ and $[(A, 1), (B, 2), (C, 3)]$ for agent a and b respectively. Even though both agents have the same paths in terms of the nodes they visit, the higher speed of agent a means that $\pi_a[x] \neq \pi_b[x]$ but $\pi_a[x+1] = \pi_b[x+1]$. Since the condition for the collision is solely defined by $\pi_a[x+1] = \pi_b[x+1]$, it becomes a vertex conflict by definition. This conflict is illustrated in figure 3(b).

The final additional conflict type is another vertex conflict that occurs as a consequence of how the previously discussed conflict is traditionally avoided. Because the system tries to find a solution where neither agent is at the same node simultaneously, the cheapest solution becomes the paths $[(A, 2), (B, 2.5), (C, 3)]$ and $[(A, 1), (B, 2), (B, 3), (C, 4)]$. Clearly both agents occupy node B at time 2.5; however, since agent b's path does not explicitly state that the agent is at node B at time 2.5, a conflict is not detected. This new conflict type can be identified and resolved by checking if a location appears consecutively in agent b's plan and constricting agent a's access to the conflicting location during the time that agent b is waiting. This conflict is illustrated in figure 3(c) and can be described using the Stern et al. notation as $\pi_b[x] = \pi_a[y] = \pi_b[x+1]$ and $x < y \leq x+1$, where x and y are time values.

The MAS utilises CBS to plan collision-free optimal paths for each agent. Due to the difficulty of accurately predicting agent arrival times, as discussed by Siebers [19], an arrival/departure schedule is not provided to the planner and a new search is executed when an agent is added to the environment. Agent deviations are modelled as either positive or negative changes in speed. The MAS detects deviations by identifying discrepancies between the planned paths and

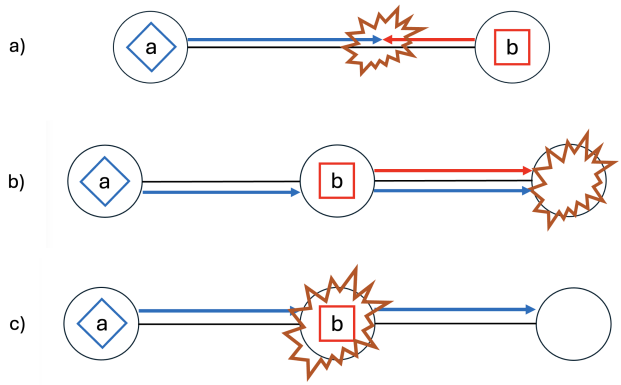

Fig. 3. Additional Conflict Types

the true positions of the agents. If an agent reaches a node earlier than stated by its plan, the system recognises that the agent in question has deviated by increasing its speed. Conversely, if an agent has not yet reached the next node in its planned path, and the current time matches the time step it was meant to arrive according to the plan, then the system recognises that the agent in question has deviated by decreasing its speed.

When an agent deviates, the original solution is no longer valid, necessitating replanning. Since agents move at different speeds, some may still be travelling between nodes when the replanning occurs. Conventional CBS requires agents' source locations to be nodes to generate plans, necessitating a small modification to the CBS algorithm. If an agent is not at a node during replanning, the algorithm calculates the next node and time based on the agent's speed. The predicted locations are then used as source locations and times for the CBS algorithm.

Experimental Setup The airport environment (shown in figure 4) includes five gates and two runways, each with two entry/exit points. Departing aircraft start at a gate and move to the departure runway on the opposite side of the airport. Arriving aircraft start at the arrival runway closer to the gates and move to one of the five gates. The arrival/departure runway configuration is inspired by real-world procedures, where certain runways are designated for specific purposes depending on traffic and weather conditions [1].

The taxiways consist of nodes and weighted, undirected edges. Nodes can be at intersections, runways, midway points, or gates while edges represent the taxiways. During each simulation, agents spawn randomly during the first 20 seconds. This duration was based on a slow agent taking 11 seconds to travel the longest optimal path without deviating. Consequently, even after some early agents have completed their journeys, new ones will spawn, creating a more realistic and continuous segment of an arrival/departure sequence.

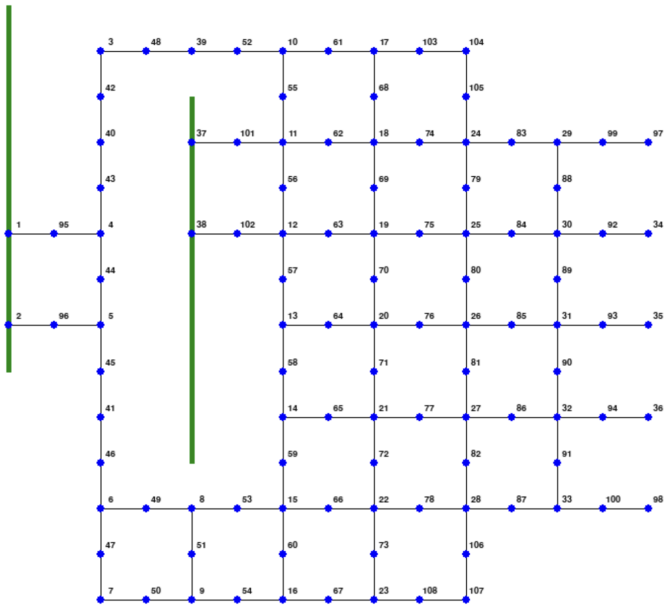

Fig. 4. Airport Environment

To test various traffic scenarios, the number of aircraft in a simulation is increased from 6 to 20. Conflicts began occurring consistently when 6 or more agents were involved, establishing the lower end of the range. The upper end was determined based on the number of agents, allowing aircraft to spawn every second or in multiple large waves, each presenting unique conflict scenarios. The experiments were divided into two categories: half included deviations and the other half did not, serving as a benchmark to assess the effects of deviations on the overall system performance. For each aircraft amount, 15 simulations were conducted, thus resulting in a total of 240 simulations with varying levels of congestion.

Hypothesis: It is posited that accounting for deviations will significantly impact the overall performance of CBS. Specifically, it is expected that deviations will lead to higher re-planning costs and increased processing time due to the need for constant re-planning and resolving of additional conflicts.

5.2 Data Collection and Analysis Methods

Performance metrics include CPU time and path optimality. CPU time was recorded during each planning event (agent spawn or deviation), and indicates the computing time needed to perform a search. Path optimality was assessed by comparing the initial planned path to the actual traveled path, measured in

steps. These metrics address the research questions by evaluating the impact of speed deviations on computational intensity and path optimality, shedding light on their effect on the CBS algorithm's performance.

CPU time was measured using Python's time module and their means and variances were analysed. Optimal path length was determined at agent spawn time through an unconstrained A^* search, while the true path length was determined by the final path length at the end of an agent's taxiing journey. The effects of deviations were evaluated using a Mann-Whitney U-test [22] with the SciPy library. Data was processed using Pandas and NumPy, and graphs were generated using Matplotlib's PyPlot module.

6 Results

The results of the experiments are presented in two separate sections. Section 6.1 presents the recordings of CPU time, aimed at answering the first research sub-question. Section 6.2 presents the results for path optimality, answering the second sub-question. A sample distribution is included alongside the results, providing an overview of the data distribution and assisting with the statistical testing in Section 6.3, conducted to validate the findings.

6.1 CPU Time

CPU time is a crucial metric in this experiment due to the nature of CBS. As noted by Sharon et al. [18], CBS is complete and optimal. While this is beneficial for tasks where optimal solutions are important or necessary, the search space increases drastically with the complexity of the problem. Recording CPU time reveals the effect that the increasing problem size has on computational resources.

CPU time is measured in seconds, with most plannings times being on the order of 10^{-3} seconds. Across 240 simulations, the CBS algorithm can execute up to 1560 times when deviations are not involved, and up to 3120 times in simulations with deviations, given the configurations for this experiment. Since most searches are completed in a fraction of a second, the average CPU time per simulation is recorded. Recording the average allows for an overall assessment of performance, mitigating the influence of a dominant number of fast searches while giving the few particularly long searches the opportunity to increase the average without disproportionately affecting the overall range.

The mean CPU time is 7.75×10^{-3} seconds with deviations and 11.21×10^{-3} seconds without deviations. The variance with and without deviations is 5.08×10^{-5} and 14.91×10^{-5} respectively, and the standard deviation is 7.13×10^{-3} and 12.21×10^{-3} respectively. The distribution of CPU time recordings for all simulations is illustrated in figure 5. Simulations featuring deviations are shown in red, while the benchmark is shown in grey. This colour scheme will remain consistent throughout the rest of the report. The vertical dashed lines represent mean values for both simulations.

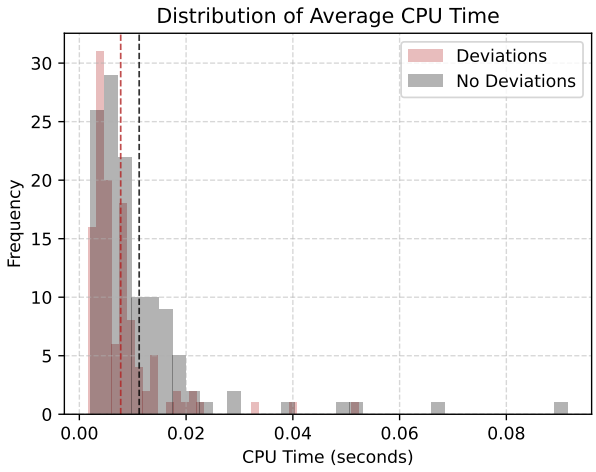

Fig. 5. CPU Time Distribution

The distribution is left-skewed, with most recordings appearing at the lower end of the spectrum. This is logical, as most searches are relatively simple and quick to solve. The left-skew will be discussed further in section 6.3 on statistical testing.

The CPU time recordings are depicted in the line graph in figure 6. It is immediately apparent that there is a consistent gap between the CPU times for the simulation configurations. When accounting for deviations, less CPU time is required for the average search compared to the benchmark. Both configurations exhibit the same general trend of exponential growth in average CPU time relative to the number of agents involved.

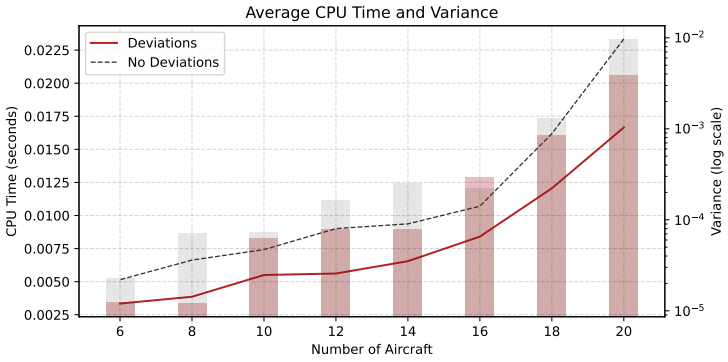

Fig. 6. CPU Time and Variance

In addition to the average CPU times, the variance is shown in the bar plots with an axis on the right-hand side of the graph. It is important to note that the variance displayed here, like the averages, differs from the variance in the sample distribution shown earlier. This is the variance in CPU time averages for each agent count, displayed on a logarithmic scale for visualisation purposes, as it increases so rapidly on a linear scale that the data for the smaller simulations is overshadowed by the larger ones. It is evident that the benchmark generally has a higher variance than the simulations with deviations, except for one simulation configuration.

6.2 Path Optimality

To measure the true cost of increased traffic, path optimality needs to be evaluated. While CBS always finds an optimal solution to a MAPF problem, the true optimal solution for each individual agent can differ significantly from the global optimal solution for the entire multi-agent system. An agent may have to wait or take detours due to other agents occupying space on the map, especially as the number of agents increases. The cost of re-planning is defined as the difference between the true individual, unconstrained optimum and the actual path that the agent takes to reach its goal. While an agent may occasionally follow its optimal path, the difference can be substantial if it does not.

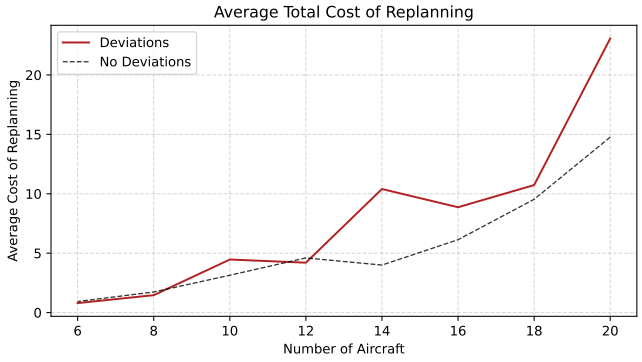

Fig. 7. Cost of Replanning

In a single simulation, the optimal paths of the first and last agents to spawn have a much lower chance of conflicting compared to those that spawn in the middle of the simulation. This results in a majority of zero re-planning costs, particularly when there are fewer agents overall. During simulations featuring deviations, the mean cost of re-planning is 8.00 with a variance of 82.10 and a standard deviation of 9.06. The benchmark simulations had a lower mean of 5.78 with a variance of 47.11 and a standard deviation of 6.86. Initial inspection of

the data indicates that the cost of re-planning is higher on average when agents deviate compared to when they do not. However, this difference is less apparent when the number of agents is smaller.

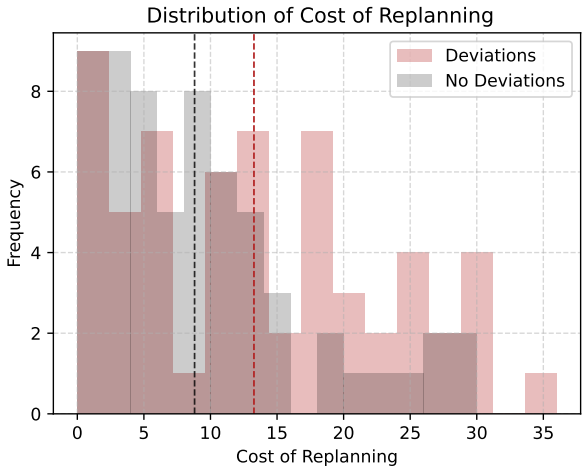

Fig. 8. Distribution of Cost of Replanning

As seen in figure 7, the cost difference only becomes visually apparent when there are more than 12 agents involved. This observation is logical since fewer agents result in less conflict, regardless of the presence or absence of deviations. Consequently, two separate comparisons on the cost of re-planning will be conducted: one for the entire dataset and one for the data with more than 12 agents. The means, variances, and standard deviations mentioned earlier apply to the whole data set, while for the partitioned data, they are 13.27, 92.40, and 9.61, respectively, with deviations, and 8.81, 59.34, and 7.70, respectively, without deviations. The distribution of the partitioned data is illustrated in figure 8.

6.3 Statistical Analysis

For both the CPU time and the cost of replanning, the data follows a left-skewed distribution. This is due to the nature of the measurements, which naturally result in a higher frequency of low values. Because neither sample follows a normal distribution, a typical t-test comparing the means could not be used to validate the results. Instead, a non-parametric test, the Mann-Whitney U-Test, was used to validate the significance of the results [22]. The U-Test compares each data point in both groups and assesses the difference in rank-sum between the groups, rather than the difference in means as a t-test would.

Conducting this test with a significance level of 5%, the null hypothesis is rejected for the CPU times, indicating that the average CPU times are signif-

icantly lower when agents deviate compared to when they do not. The cost of re-planning is less straight-forward because of how it is evaluated in these experiments. As previously mentioned, the cost of re-planning is compared using the entire dataset and with the upper half of the data. When considering the entire dataset, the difference in cost is not significant enough to reject the null hypothesis. However, taking only the data into account where the difference is consistently and visibly present in the graph, the null hypothesis is rejected.

6.4 Summary of Key Findings

Overall Performance Both the benchmark and deviation simulations exhibit an exponentially rising trend with the number of agents. CPU times for simulations with deviations are on average 3.46×10^{-3} seconds faster than the benchmark. The cost of re-planning is not significantly higher when deviations are present, except in scenarios with more than 12 agents.

CPU Time (6.1) and Sub-question 1 CPU time reveals that CBS remains efficient despite increasing complexity. Simulations with deviations show lower average CPU times, attributed to shorter paths during re-planning. However, the frequency of re-planning increases, balancing the overall computational load.

Path Optimality (6.2) and Sub-question 2 Path optimality is not compromised when taking deviations into account but worsens as the number of agents increases. The average re-planning cost is higher for deviating agents, indicating a consistent deterioration in path quality with frequent re-planning. This effect is more pronounced in high-density traffic scenarios.

7 Discussion

7.1 Interpretation of Results

As expected, the performance of CBS decreases with the increasing number of agents involved. This trend is evident from the uptrend in every recorded performance metric. Given that airport environments are limited in capacity, the number of agents that are taxiing simultaneously will not be significantly higher than the number involved in these experiments. This suggests that it is feasible to use CBS as a conflict resolution system in real-world applications.

Speed deviations had the most noticeable effect on CPU time, but the direction of this effect was opposite to the expected outcome. The expectation was that CPU time would be higher than the benchmark when agents deviate from the plans. However, the average CPU time was found to be lower than the benchmark. This can be explained by the fact that when an agent deviates, the path it subsequently takes to reach its goal is shorter than the initially planned path, leading to fewer potential conflicts during the new planning phase.

While it can be stated that CPU time is significantly lower when agents deviate compared to when they do not, it does not represent the complete picture. When agents deviate, the number of times CBS must run increases significantly as illustrated by figure 9.

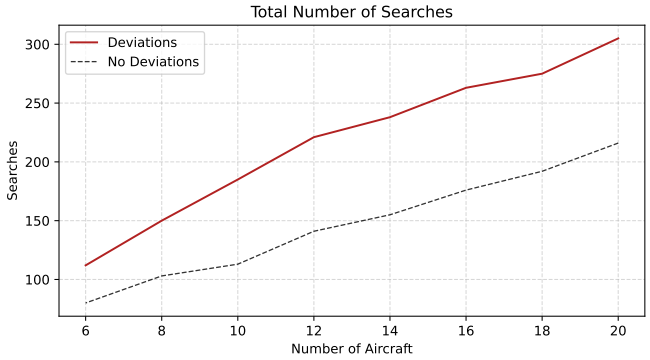

Fig. 9. Total Number of Searches

Combining the number of searches with the CPU time provides a more comprehensive overview of the true impact of deviations on CPU time. Figure 10 shows the total time spent on path planning across 15 simulations for each traffic scenario. It is evident that the faster CPU time is offset by the increased number of searches. The results indicate that CBS spends as much time planning when deviations are involved as it does when they are not, suggesting that speed deviations do not negatively affect CBS performance in terms of CPU time as was expected.

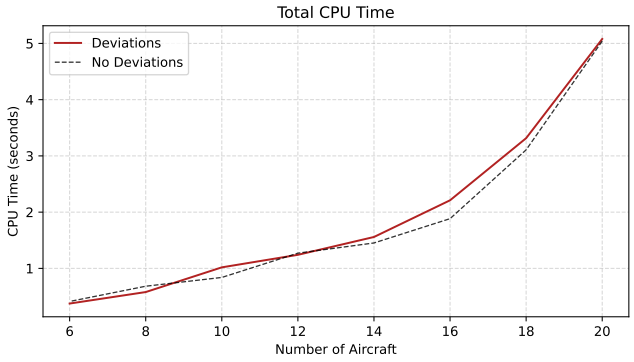

Fig. 10. Total CPU Time

When analysing the cost of deviations in terms of path length, a similar ultimate outcome is observed as with CPU times. For both the benchmark and the deviations, the cost of planning increases exponentially with the number of agents involved. The difference between the deviations and the benchmark is not significant in the entire experiment. However, with more agents involved, the difference in cost becomes higher for the deviating agents. This may be attributed to the previously mentioned concept where agents deviate from their optimal paths, leading to the creation of more expensive paths despite the shorter time taken searching for the paths. Overall, the difference is not statistically significant enough to reject the null hypothesis, indicating that cost performance is also unaffected by CBS.

7.2 Linking to Existing Literature

Existing research has successfully implemented methods that perform flawlessly when faced with large and complex coordination problems. However, only a few have addressed the concept handling uncertainty in real time. Uncertainty, and resilient methods to handle uncertainty, is an aspect that needs thorough exploring if automation is to be applied alongside human decision-making, as suggested by the SESAR project [6]. The following section explains how the results of this experiment relate to the existing literature.

Recalling the related works section, the use of MAPF solutions for solving large warehouse coordination problems has proven extremely successful [12]. The goal was to develop a system capable of continuous operation while knowing only a specific portion of the future tasks. The developed solution, Rolling-Horizon Collision Resolution (RHCR), was a framework based on conflict resolution for a continuous influx of new information while the model was running [12]. This framework demonstrated the strength of CBS and the usefulness of MAPF in scenarios requiring extensive coordination and planning. However, the environment involved fully autonomous agents and did not explore uncertainty and deviation handling. Comparing the findings of both projects, it becomes evident that CBS is not only powerful for long-term continuous planning but can also adapt quickly to scenarios where new plans are needed in real time, without significant changes to the algorithm.

Analysing both Fines' [7] work on CBS for resilient taxi operations and the research by Boyarski et al. [2] on improvements of CBS highlights the algorithm's robustness when modifications are added or changes environmental perception are made. Boyarski et al. demonstrate enhanced performance without considering uncertainty, while Fines illustrates CBS's adaptability in uncertain airport environments. The consistent performance of CBS despite frequent deviations indicates that the method is resilient and effective on its own. However, as shown by the mentioned research, incorporating further enhancements to the classical path finding mechanism can lead to significantly improved performance.

7.3 Implications, Limitations, and Future Work

This experiment demonstrates the adaptability of both Conflict-Based Search and Multi-Agent Systems. Even if agents deviate from their assigned paths, CBS can quickly re-plan, ensuring an optimal solution. Although CBS requires more time to determine a solution compared to other methods, its optimality enhances its resilience in managing deviations. The total CPU performance suggests that handling deviations rarely requires more than a second, indicating the suitability of CBS for automated airport routing due to its fast operation speed and ability to handle complex re-routing problems in real-time.

Despite CBS's resilience, improved versions like those proposed by Boyarski et al. could further enhance performance, potentially reducing computation times while accounting for additional factors [2]. Further study of CBS and its improved versions could enable equally resilient searches with added conditions. Furthermore, while this study demonstrates the effectiveness of MAPF in dynamic and complex environments, the agent-based simulation model used remains abstract compared to the factors involved in the reality. The simulation does not consider differences in weight, thrust, acceleration, turning speed, and other kinematic factors, which uniquely impact the true cost of taxiing. Moreover, in a system where agents are taxiing with at least one engine running, the rate of fuel burn varies between aircraft [11]. Considering all determining factors of fuel consumption, truly optimal solutions may need to account for physical parameters and prioritise planning of agents for which minimal taxi time is most crucial.

8 Conclusion

This project aims to examine the impact of speed deviations on the performance of Conflict-Based Search (CBS) for Multi-Agent Path Finding (MAPF) in airport taxiing operations. The study is motivated by the need for planning and coordination solutions to enhance the efficiency of airport ground operations and reduce environmental impact. With a focus on addressing the computational and operational challenges posed by speed deviations, the research aims to evaluate CBS's computational intensity and path optimality under varying traffic densities.

Two research sub-questions are answered through the experiments:

Sub-Question 1: How does path planning, when accounting for speed deviations, impact the computational complexity of CBS as the number of agents increases?

Sub-Question 2: To what extent is the solution optimality of CBS achieved when accounting for speed deviations?

Through extensive simulations, this research demonstrates that CBS, while computationally intensive, is capable of finding optimal solutions in uncertain environments. Answering the first question, the results indicate that speed deviations lead to an increase in the number of re-plannings, while the CPU time

required for each planning instance was lower on average compared to non-deviating scenarios. This finding suggests that CBS's optimality contributes to its resilience to speed deviations, enabling it to adapt effectively to changes and maintain high performance levels. Answering the second question, the analysis of path optimality revealed that deviations impact the re-planning cost when the number of agents is high. Although the overall difference in re-planning costs was not statistically significant across all data, it became significant in denser traffic scenarios.

The research supports the notion that CBS is a robust and effective method for MAPF in complex and dynamic environments. The study's implications suggest that CBS, with potential improvements and modifications for more complex real-life models, is a strong candidate for real time automated airport routing services.

References

1. Amsterdam Airport Schiphol: Gebruiksprognose 2024. Tech. rep., Schiphol Group (2023)
2. Boyarski, E., Felner, A., Stern, R., Sharon, G., Betzalel, O., Tolpin, D., Shimony, E.: ICBS: The Improved Conflict-Based Search Algorithm for Multi-Agent Pathfinding. Proceedings of the International Symposium on Combinatorial Search **6**(1), 223–225 (Sep 2021). https://doi.org/10.1609/socs.v6i1.18343, https://ojs.aaai.org/index.php/SOCS/article/view/18343
3. Chen, Fu: Aircraft taxiing route planning based on multi-agent system. In: 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC). pp. 1421–1425. IEEE, Xi'an, China (Oct 2016). https://doi.org/10.1109/IMCEC.2016.7867448, http://ieeexplore.ieee.org/document/7867448/
4. Deonandan, I., Balakrishnan, H.: Evaluation of Strategies for Reducing Taxi-Out Emissions at Airports. In: 9th AIAA Aviation Technology, Integration, and Operations Conference (ATIO). American Institute of Aeronautics and Astronautics, Hilton Head, South Carolina (Sep 2009). https://doi.org/10.2514/6.2009-6926, http://arc.aiaa.org/doi/10.2514/6.2009-6926
5. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-Agent Systems: A Survey. IEEE Access **6**, 28573–28593 (2018). https://doi.org/10.1109/ACCESS.2018.2831228, https://ieeexplore.ieee.org/document/8352646/
6. Ferro, D., Schuller, D., Zaki, G., Kuren, I., Stridsman, L., Huart, O., Ranieri, A., Schnabel, O., Genchev, M., Loscos, e.M., Gringinger, E., Bruni, F., Champagne, C., Sanchez-Palomo Bermudez, L., Nacchia, F.: SESAR Concept of Operations (CONOPS 2019). Tech. Rep. D2.5 (May 2019), edition 01.00.00
7. Fines, K., Sharpanskykh, A., Vert, M.: Agent-Based Distributed Planning and Coordination for Resilient Airport Surface Movement Operations. Aerospace **7**(4), 48 (Apr 2020). https://doi.org/10.3390/aerospace7040048, https://www.mdpi.com/2226-4310/7/4/48
8. Guo, R., Zhang, Y., Wang, Q.: Comparison of emerging ground propulsion systems for electrified aircraft taxi operations. Transportation Research Part C: Emerging Technologies **44**, 98–109 (Jul 2014). https://doi.org/10.1016/j.trc.2014.03.006, https://linkinghub.elsevier.com/retrieve/pii/S0968090X14000722

9. Hönig, W., Li, J., Koenig, S.: Overview of Multi-Agent Path Finding (MAPF). Tech. rep., University of Southern California (2020)
10. International Air Transport Association (IATA): Air Passenger Demand Set to Reach 4 Billion in 2024. Tech. rep., IATA Pressroom (Jan 2024), https://www.iata.org/en/pressroom/2024-releases/2024-01-31-02/
11. Khadilkar, H., Balakrishnan, H.: Estimation of aircraft taxi fuel burn using flight data recorder archives. Transportation Research Part D: Transport and Environment **17**(7), 532–537 (Oct 2012). https://doi.org/10.1016/j.trd.2012.06.005, https://linkinghub.elsevier.com/retrieve/pii/S1361920912000612
12. Li, J., Tinka, A., Kiesel, S., Durham, J.W., Kumar, T.K.S., Koenig, S.: Lifelong Multi-Agent Path Finding in Large-Scale Warehouses (2020). https://doi.org/10.48550/ARXIV.2005.07371, https://arxiv.org/abs/2005.07371, version Number: 2
13. LVNL (Air Traffic Control Netherlands): Runway Use. Tech. rep., LVNL (Air Traffic Control Netherlands), https://en.lvnl.nl/runway-useanchor-14efe196d2b34726906304b2f602dff7
14. Navia, I., HungaroControl: SESAR Solution PJ.02-W2-21.4 Contextual Note V3: Full Guidance Assistance to mobiles using 'Follow the Greens' procedures on Airfield Ground Lighting. Tech. rep., SESAR Joint Undertaking (2019), https://www.sesarju.eu/sesar-solutions/full-guidance-assistance-mobiles-using-follow-greens-procedures-based-airfield, acronym: AART
15. Noortman, T.: Agent-Based Modelling of an Airport's Ground Surface Movement Operation: Understanding the principles and mechanisms of decentralised control. Master's thesis, Delft University of Technology (May 2018), http://resolver.tudelft.nl/uuid:96c4fc2f-8ed1-4be6-a5e7-9b19ba71954b
16. Royal Schiphol Group: Factsheet Taxibot. Tech. rep., Royal Schiphol Group (2020), https://www.schiphol.nl/en/download/b2b/1594319068/40LYaERw5x8BuRuXpVQkiK.pdf
17. Schiphol Group: Sustainability at the Airport. Tech. rep., Schiphol Group, https://www.schiphol.nl/en/schiphol-group/page/sustainability-at-the-airport/
18. Sharon, G., Stern, R., Felner, A., Sturtevant, N.R.: Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence **219**, 40–66 (Feb 2015). https://doi.org/10.1016/j.artint.2014.11.006, https://linkinghub.elsevier.com/retrieve/pii/S0004370214001386
19. Siebers, J.: Airport Surface Planning under Uncertainty. Master's thesis, Delft University of Technology (Jan 2017), http://resolver.tudelft.nl/uuid:dd051acf-7adf-46a2-8223-b0bb16824518
20. Stern, R., Sturtevant, N., Felner, A., Koenig, S., Ma, H., Walker, T., Li, J., Atzmon, D., Cohen, L., Kumar, T.K.S., Boyarski, E., Bartak, R.: Multi-Agent Pathfinding: Definitions, Variants, and Benchmarks (2019). https://doi.org/10.48550/ARXIV.1906.08291, https://arxiv.org/abs/1906.08291, version Number: 1
21. Tao, J., Guo, J., Liu, C.: A review of powered wheel for aircraft. In: 2016 IEEE International Conference on Aircraft Utility Systems (AUS). pp. 378–383. IEEE, Beijing, China (Oct 2016). https://doi.org/10.1109/AUS.2016.7748078, http://ieeexplore.ieee.org/document/7748078/
22. Triola, M.F.: Elementary statistics. Pearson Education UK, Harlow, twelfth edition edn. (2014), oCLC: 1015886727
23. Van Oosterom, S., Mitici, M., Hoekstra, J.: Dispatching a fleet of electric towing vehicles for aircraft taxiing with conflict avoidance and efficient battery charging. Transportation Research Part C: Emerging Tech-

nologies **147**, 103995 (Feb 2023). https://doi.org/10.1016/j.trc.2022.103995, https://linkinghub.elsevier.com/retrieve/pii/S0968090X22004089
24. Wurman, P., D'Andrea, R., Mountz, N.: Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses. AI Magazine **29**(1), 9–19 (Mar 2008)