Multi-Agent Task Allocation and Path Planning for Autonomous Ground Support Equipment

Master of Science Thesis

M. van der Zwan

Technische Universiteit Delft

TU Delft

Multi-Agent Task Allocation and Path Planning for Autonomous Ground Support Equipment

Master of Science Thesis

by

M. van der Zwan

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Tuesday December 19th 2023.

Student number:	4654196	
Project duration:	September 2022 – December 2023	
Thesis committee:	Ir. M.J. Schuurman	Chair - Aerospace Structures and Materials
	Dr. O.A. Sharpanskykh	Supervisor - Air Transport and Operations
	Dr. G. Ermiş	Additional - Air Transport and Operations
	Dr. M. Lourenço Baptista	Examiner - Air Transport and Operations

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

TU Delft

Acknowledgements

As I bring to a close this significant chapter of my academic journey with the submission of my thesis, I reflect on a rewarding six-year journey at TU Delft's Faculty of Aerospace Engineering. I want to thank everyone with whom I have had the privilege of meeting and collaborating over the last six years. The experiences and opportunities offered to me during this time have been invaluable in shaping both my academic pursuits and my personal growth.

My Master of Science is concluded with this thesis called "Multi-Agent Task Allocation and Path Planning for Autonomous Ground Support Equipment". I want to express my gratitude to my supervisors, Alexei and Gulcin. Your guidance and expertise have been essential to my research and my professional development. I also would like to express my appreciation to Jeroen, whose operational expertise has been really valuable to my research. His contributions have enriched my understanding of airside operations and added depth to my work.

A special thanks goes out to my friends who have been working on their theses in NB2.56. Our friendship, discussions, and shared experiences have been a source of motivation and have greatly enriched the past year. To my family - my parents, my sister Terri, and Gaby - your unconditional support and encouragement have been the foundation of my achievements. Finally, I want to thank Jilles, for your understanding, patience, and endless support. I could not have done all this without you and I am very thankful for that.

As I conclude, I am filled with a bittersweet feeling. Although I am sad that this chapter of my life is ending, I am excited for what lies ahead. The knowledge, experiences and memories I have gained here will be a guiding light in my future endeavors. Thank you, one and all, for being a part of my journey at TU Delft.

Manouk van der Zwan
Delft, December 2023

Contents

List of Figures

List of Tables

List of Abbreviations

A-CDM	Airport Collaborative Decision Making
AAS	Amsterdam Airport Schiphol
ABM	Agent Based Model
ACO	Ant Colony Optimization
APU	Auxiliary Power Unit
ASP	Answer Set Programming
ASU	Air Start Unit
CA*	Cooperative A*
CAT	Conflict Avoidance Table
CBM	Conflict-Based Min-Cost-Flow
CBS	Conflict Based Search
CBS-TA	Conflict-Based Search Task Allocation
CBSwP	Conflict Based Search with Priorities
CD	Complex Dependencies
CNP	Contract Net Protocol
CO-WHCA*	Conflict Oriented WHCA*
CO-WHCA*P	CO-WHCA* Prioritization
CP	Constraint Programming
CT	Constraint Tree
DARP	Dial-A-Ride Problem
DP	Dynamic Programming
EAPA*	Enhanced Partial Expansion A*
ECBS	Enhanced Conflict Based Search
EPA	Equipment Parking Area
ERA	Equipment Restraint Area
ESA	Equipment Staging Area
FIFO	First In First Out
GCBS	Greedy Conflict Based Search
GPU	Ground Power Unit
GSE	Ground Support Equipment
HCA*	Hierarchical Cooperative A*
HWY	Highways
IA	Instantaneous Assignment
ICBS	Improved Conflict Based Search
ICT	Increasing Cost Tree
ICTS	Increasing Cost Tree Search
ID	In-schedule Dependencies
ILP	Integer Linear Programming
LKH-3	Lin-Kernighan-Helsgaun 3 solver
LNS	Large Neighborhood Search
MA-CBS	Meta-agent Conflict Based Search
MAPF	Multi Agent Path Finding
MAPP	Multi-agent Path Planning
MDD	Multi-valued Decision Diagram
MILP	Mixed Integer Linear Programming
MR	Multi Robot task
MRTA	Multi-Robot Task Allocation
MT	Multi Task robot
ND	No Dependencies
NP	Non-deterministic Polynomial-time
NPA	No Parking Area
OSF	Operator Selection Function
PBB	People Boarding Bridge
PBS	Priority-Based Search
PCA	Pre-Conditioned Air
PDP	Pickup and Delivery Problem
PT	Priority Tree
RMTA	Regret-based Marginal-cost Task Assignment
SAT	Boolean Satisfiability Problem
SCIPP	Soft Conflict Interval Path Planning
SIPP	Safe Interval Path Planning

SIPPwRT	SIPP with Reservation Table
SPDP	Single vehicle Pickup and Delivery Problem
SR	Single Robot task
SSC	Sequential Single Cluster
SSI	Sequential Single-Item
ST	Single Task robot
STN	Simple Temporal Network
TA	Task Allocation
TAT	TurnAround Time
TEA	Time Extended Assignment
TePSSI	Sequential Single-Item with Precedence and Temporal constraints
TeSSI	Sequential Single-Item with Temporal constraints
TPTS	Token Passing with Task Swaps
TS	Tabu Search
TSP	Traveling Salesman Problem
VRP	Vehicle Routing Problem
VRPPD	Vehicle Routing Problem with Pickups and Deliveries
VRPTW	Vehicle Routing Problem Time Windows
VRSP	Vehicle Routing and Scheduling Problem
WHCA*	Windowed Hierarchical Cooperative A*
XD	Cross-schedule Dependencies

I

Scientific Paper

Multi-Agent Task Allocation and Path Planning for Autonomous Ground Support Equipment

M. van der Zwan*

Delft University of Technology, Delft, The Netherlands

Abstract

Many large airports aim to have complete autonomous airside operations in the future. Amsterdam Airport Schiphol (AAS) for example, launched the Autonomous Airside Operations program to achieve this goal. Our main contribution is to present a Multi-agent Pickup-and-Delivery (MAPD) model that uses a centralized task allocation mechanism to improve the performance of integrated task allocation and path planning for autonomous ground handling operations compared to previous research. This study models a global multi-vehicle Pickup and Delivery Problem with Time Windows (PDPTW) for the scheduling of autonomous ground handling tasks. A warm start multi-objective mixed integer linear programming model is proposed to solve the scheduling problem where the initial feasible solution is obtained by an insertion heuristic. This multi-agent task allocation model, when combined with multi-agent path planning, forms a MAPD model for modeling autonomous ground handling operations. Multi-agent path planning is solved using prioritized Safe Interval Path Planning (SIPP). A replanning model is developed to assess the resilience of our model to disruptions of operations. Also, a mixed integer nonlinear programming model, which includes an additional non-linear objective, is proposed to generate more realistic task assignments by minimizing the waiting time of vehicles on the aircraft stands. In this study, a four-hour planning window with three aircraft stands at AAS is used for the experiments. The results show that the proposed approach improves the computational time of the task allocation model with 48% for the normal traffic scenario, compared to the previously published results. The conflict-free routes of all ground support equipment (GSE) vehicles are all successful and close to the shortest path results, with an average increase of 0.04% and 10% for the path length and the duration of the path, respectively. Our model is therefore able to generate complete, high quality solutions in less than three minutes.

1 Introduction

Fully autonomous operations is a goal that is aspired by many airports around the world. At Amsterdam Airport Schiphol (AAS) for example, Royal Schiphol Group presented the Vision 2050 initiative, which aims to transform AAS into the most sustainable and high-quality airport in the world [1]. To achieve this goal, Royal Schiphol Group launched the Autonomous Airside Operations program in 2020 with the goal of replacing all airside vehicles with interconnected, autonomous and emission-free vehicles [2]. For airside operations, specifically ground handling operations, there is a limited number of vehicles shared between different aircraft stands to complete all scheduled ground handling tasks. Ground handling operations refer to the various services and activities carried out on the ground to ensure safe, efficient and timely arrival and departure of the aircraft. Ground handling vehicles must travel without conflicts between the aircraft stands and the terminal. Creating such an interconnected autonomous fleet requires a framework that allocates tasks to vehicles and a traffic management system that enables the execution of tasks. Fortunately, most procedures within aircraft ground operations are standardized and aircraft servicing locations have similar placements on each aircraft, making it easier to implement automation in the short term [3]. Trials with an autonomous baggage tractor are already being carried out at the Amsterdam Airport Schiphol, which loads and unloads baggage at the terminal and the aircraft stands [4].

Existing research on task allocation and path planning of autonomous vehicles is mainly focused on warehouse environments [5–8]. In the literature, this problem is known as the Multi-Agent Pickup-and-Delivery (MAPD) problem, where agents (e.g., autonomous vehicles) collaborate with each other to execute tasks in a known environment. In the context of aircraft ground handling, the formulation of MAPD problems remains relatively new. Chen et al. [9] (hereafter CES) introduce a multi-agent model that combines an auction-based task allocation mechanism with prioritized multi-agent path planning. The MAPD problem captures tasks performed by

*Msc Student, Air Transport and Operations, Faculty of Aerospace Engineering, Delft University of Technology

autonomous ground support equipment (GSE) vehicles during the turnaround time. CES used a single vehicle pickup and delivery model to generate a bid in each round of the auction. This model re-optimizes the existing route after including the auctioned task, which increased the computational time. Furthermore, the decentralized approach coupled with time window constraints, limited resources, and a time-limited, optimization-based bid generation, led to some tasks remaining unallocated, resulting in incomplete solutions. In this work, we overcome this drawback by developing a centralized optimization approach for allocating and scheduling tasks across multiple vehicles.

We propose a global multi-vehicle optimization technique to address the issues of the auction-based task allocation mechanism. Global multi-vehicle optimization shows promising results for the allocation and scheduling of ground handling tasks, as it allows the consideration of relationships and restrictions among all ground handling tasks and generates complete solutions [10]. Combining this approach with prioritized route planning results in a novel method of scheduling and conflict-free routing to coordinate autonomous ground handling vehicles. We compare the performance of our proposed method to the MAPD framework of CES due to its close resemblance to our research and show that our method performs better in terms of allocation rate, computational time, and total completion time.

For this research, Amsterdam Airport Schiphol is used to evaluate the performance of the model. Flight schedules, GSE vehicles, and tasks are generated based on the infrastructure and available resources at AAS. Then a global task allocation model, a Pickup and Delivery Problem with Time Windows (PDPTW) [11], is activated that assigns all tasks to GSE vehicles while minimizing the total completion time of all vehicles. In this model, we first use an insertion heuristic to find an initial feasible solution [12]. We use this solution as a warm start solution of a Mixed Integer Linear Programming (MILP) model. The schedule forms the basis for the path planning model, which generates conflict-free routes for all GSE vehicles between the terminal and the aircraft stands. Path planning is solved similar to CES using a combination of prioritized planning and safe interval path planning (SIPP). The output of the complete model is a solution that contains both a schedule and an itinerary for each GSE vehicle.

MAPD algorithms can be classified into online and offline models. Online algorithms are able to take into account tasks that are revealed at certain points in time, while offline algorithms are aware of all tasks to be scheduled beforehand. As flight schedules and ground handling tasks are known in advance, this research falls into the latter category. To assess the feasibility of real-time implementation, we simulate disruptions that would initiate a replanning process. Tasks with significant delays are replanned using an adapted version of the insertion heuristic, and then new paths are generated using the path planning model. Each part of the model is verified using various verification techniques, such as open source data sets, animations, and unit tests. The performance of the model is assessed using key performance indicators such as the average completion time, allocation rate, and computational time in different traffic scenarios. We show that the proposed global optimization approach improves the computational time with 48% on average for the normal traffic scenario and the task allocation rate is 100% for all instances. In addition, we perform a sensitivity analysis on the most influential variable, the number of GSE vehicles.

The paper is structured as follows. Section 2 gives a definition of the problem followed by design considerations and assumptions made to model ground handling operations at AAS. The complete multi-agent model is introduced in Section 3, including a detailed description of each of the components of the model. Subsequently, the verification and validation procedures of the model are discussed in Section 4. The experimental setup and results of the experiments are discussed in Section 5, and the sensitivity analysis is discussed in Section 6. Finally, Section 7 presents a discussion of the findings, and Section 8 concludes this research.

2 Case Description

Airside operations and their environment are complex and dynamic, which makes them difficult to model for automated ground handling. This section provides a detailed description of the problem and design considerations for modeling autonomous ground handling at AAS. In this study, for simplicity, we leave the movement of aircraft and related operations, such as taxiing and towing, out of scope. The environment of this model includes the aircraft stands and the perimeter road that connects them. The aircraft is assumed to be stationary on the stand, ignoring its motions during its arrival (or departure) before (or after) a turnaround process starts (or ends).

2.1 Problem Definition

Aircraft ground handling is performed between two flights to prepare the aircraft for the next flight. We consider flights within a four-hour planning window. The flight schedules are generated using a uniform distribution. Between flights, a 10-minute break is included to accommodate pushback and docking of the aircraft. There exists a set of GSE vehicles to handle ground handling tasks on a pier in an airport. These vehicles are used to perform various tasks in and around the aircraft. In this research, five types of GSE vehicles are considered, a fuel filling vehicle (RE), a catering truck (CA), a baggage tractor (BA), a portable water tanker (WA), and a lavatory truck (WC) that perform refueling, catering, baggage handling, water service and lavatory service tasks. Each of these tasks has to be serviced at a specific location on the aircraft stand and within specified time windows.

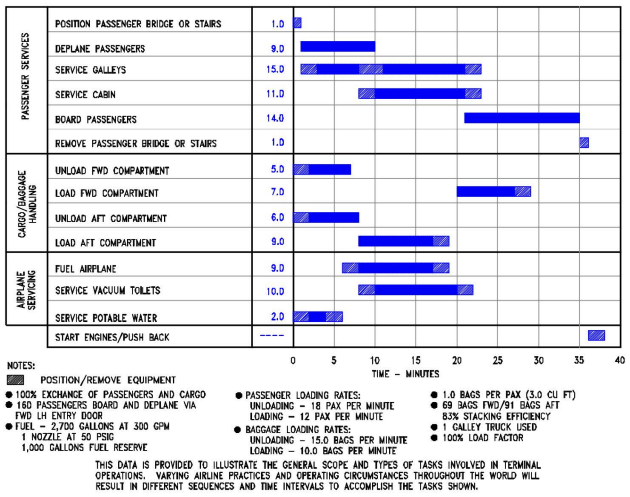
The time intervals between flights define the time window constraints to complete ground handling tasks. Let F_1, F_2, F_3, F_4 be the time intervals during which an aircraft is standing on an aircraft stand within the planning window. Ground handling tasks must be completed for a single aircraft per time interval on a stand with the available GSE vehicles. The number of GSE vehicle is limited, therefore, vehicles are shared and reused throughout the airport. There can be several vehicles for each vehicle type, and each vehicle can only be used to perform a specific type of task within the vehicle capacity limits. To achieve efficient operations, the allocation of tasks should be optimized. Furthermore, vehicle movement on the pier must be coordinated to ensure safe and timely operations. The combination of task allocation and coordination of movements can be modeled using a Multi-Agent Pickup-and-Delivery (MAPD) problem approach.

The multi-agent task allocation and path planning model aims to generate a schedule and conflict-free paths for autonomous GSE vehicles while adhering to constraints and restrictions. GSE vehicles have capacity and precedence constraints and tasks must be completed within specific time windows. We model the task allocation problem as a multi-vehicle Pickup and Delivery Problem with Time Windows (PDPTW) because of it's ability to incorporate all the aforementioned constraints. A PDPTW model assigns a set of tasks that need to be picked up from designated locations and delivered to the corresponding destinations within task-specific time windows. Therefore, each task is divided into a paired pickup and delivery task to solve the problem using a PDPTW model. There is a separate PDPTW model for each type of GSE vehicle. Within a PDPTW model, vehicles start at a virtual start node located at the depot, move around to serve tasks, and end their route back at the depot in a virtual end node. Each model aims to minimize the total completion time of all vehicles of the same type and minimize the start time of each task. Vehicle paths are planned to coordinate vehicle movements around the airport. Paths are determined based on the results of the task allocation model. Mobility restrictions for GSE vehicles on a pier are taken into account when planning routes. There are directional restrictions on the service road and parts of the stand that are occupied by static obstacles, such as the aircraft or passenger boarding bridges.

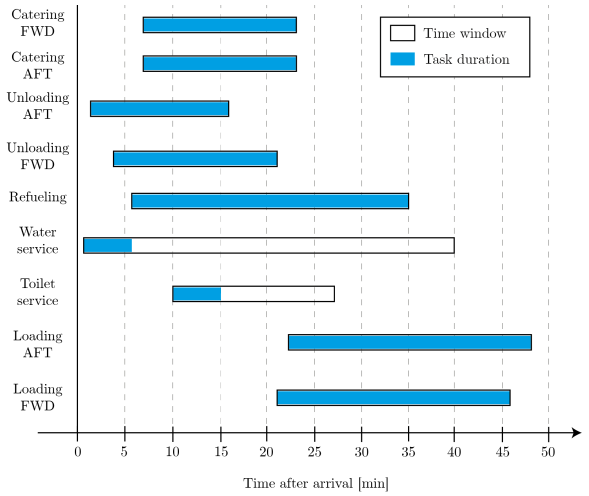
2.2 Ground Handling Activities

Ground handling activities are tasks that must be completed on the apron, within a turnaround process between the arrival and departure of an aircraft. A distinction can be made between short- and long-haul flights in terms of ground handling activities. Typically, short-haul flights have a much tighter schedule due to the profitability of maximizing aircraft utilization. Tight schedules require more attention to planning and scheduling to prevent delays and ensure the most effective utilization of resources. Therefore, in this research, the short-haul flights for the Boeing 737-800 are considered. The ground handling activities for an aircraft can be grouped into two categories; over the wing and under the wing. This research focuses on the latter, which includes refueling of the aircraft, filling the water tank, emptying the waste tank, handling of baggage, and loading of catering galleys. Seasonal activities such as de-icing are not considered in this study. In addition, it is assumed that short-haul flights are not used for freight transportation. Thus, in baggage handling, only passenger luggage requires loading and unloading, and only a baggage tractor with carts is necessary.

Turnaround time (TAT) is the time required to complete the scheduled ground handling activities between two legs of the flight. The schedule of tasks in the TAT of Boeing 737-800 is shown in Figure 1(a). Based on the schedule shown in Figure 1(a) and discussions with experts in the field, we use the turnaround time schedule shown in Figure 1(b) in this research.



(a) Turnaround time schedule Boeing 737-800 [13].



(b) Adapted turnaround time schedule for this research based on internal documents.

Figure 1: Turnaround time schedules containing the task duration of each task and time windows each task needs to be executed in.

In Figures 1(a) and 1(b), the duration of the tasks and the time windows in which these tasks have to be performed are given. Unlike the other vehicles, baggage handling and catering vehicles have two service locations, in front and aft of the aircraft, each of which must be visited to complete the services, and these service times are included in turnaround time.

2.3 Infrastructure

In this research, the north side of pier B in AAS is used as case study. The infrastructure of this pier is shown in Figure 2. Pier B is a linear pier with six aircraft stands on each side. Each stand on the north side has a Passenger Boarding Bridge (PBB) through which all passengers board the aircraft. Furthermore, each of these stands is equipped with an underground fueling system. Therefore, only hydrant vehicles are necessary to refuel an aircraft. This pier is made up of stands that are connected to a single perimeter road. GSE vehicles can use this road to move between stands and to/from the terminal. Each stand has a designated entrance and exit point to the perimeter road.

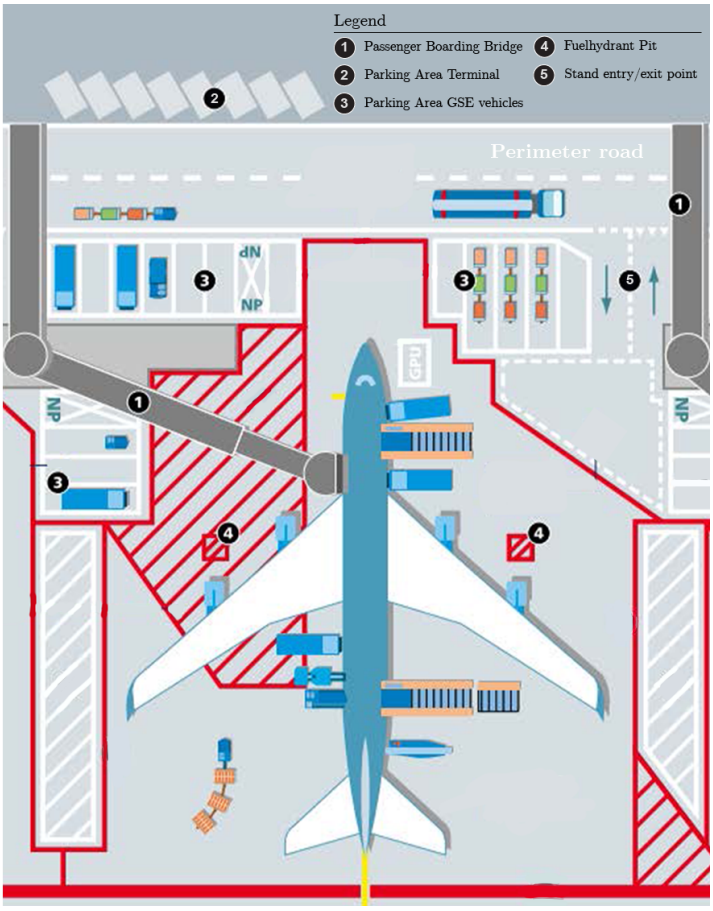


Figure 2: Aircraft stand infrastructure pier B north side at AAS. Adapted from [14].

2.4 Ground Handling Equipment

Each of the aforementioned activities requires a specific ground handling vehicle. This means that five unique GSE vehicles are required to perform all ground handling tasks. Table 1 shows the characteristics of the required GSE vehicle types; the number of vehicles needed to carry out all ground handling activities for a single aircraft, their priorities, and the capacity of each vehicle. The priority is used to resolve possible conflicts during movement, and the capacity is expressed in the number of tasks it can perform with full load. The capacities of the GSE vehicles are based on discussions with experts and the domain knowledge shared in the work by Tabares et al. [3]. For baggage handling, two tractors are required per aircraft, one for loading and one for unloading the baggage. Each tractor can link up to five carts full of baggage, which is sufficient to carry all passengers' baggage from or to an aircraft. Two catering trucks can serve three aircrafts; therefore, the capacity is defined as 1.5 per vehicle. Furthermore, it is assumed that all GSE vehicles are shared between aircraft stands on the same pier and all GSE vehicles serving the same task type are homogeneous.

Table 1: Required resources to perform under the wing ground handling activities on a short-haul aircraft. Based on [3] and discussions with experts.

GSE vehicle	Amount	Priority	Capacity
Fuel filling vehicle (RE)	1	1	∞
Catering truck (CA)	2	2	1.5
Baggage tractor (BA)	2	3	1
Portable water truck (WA)	1	4	20
Vacuum lavatory truck (WC)	1	5	25

2.5 Simulation Environment

The simulation environment of this research is a simplified grid-based version of the aforementioned airport infrastructure in Section 2.3. The environment includes three aircraft stands, a perimeter road, and a terminal parking area. Using both the B737-800 documentation[13] and Figure 2 allowed creating a grid-based model

of an aircraft stand. In the 16x16 grid-map of a single aircraft stand, each cell of the grid-map has the size of 4 m x 4 m and static obstacles, such as the PBB and the parked aircraft, which cover around 15-20% of the environment are included. The service locations occupy single cells and their locations within the aircraft stand are based on B737-800 documentation. The complete airside environment including the terminal, the service road, and several aircraft stands on a pier is shown in Figure 3.

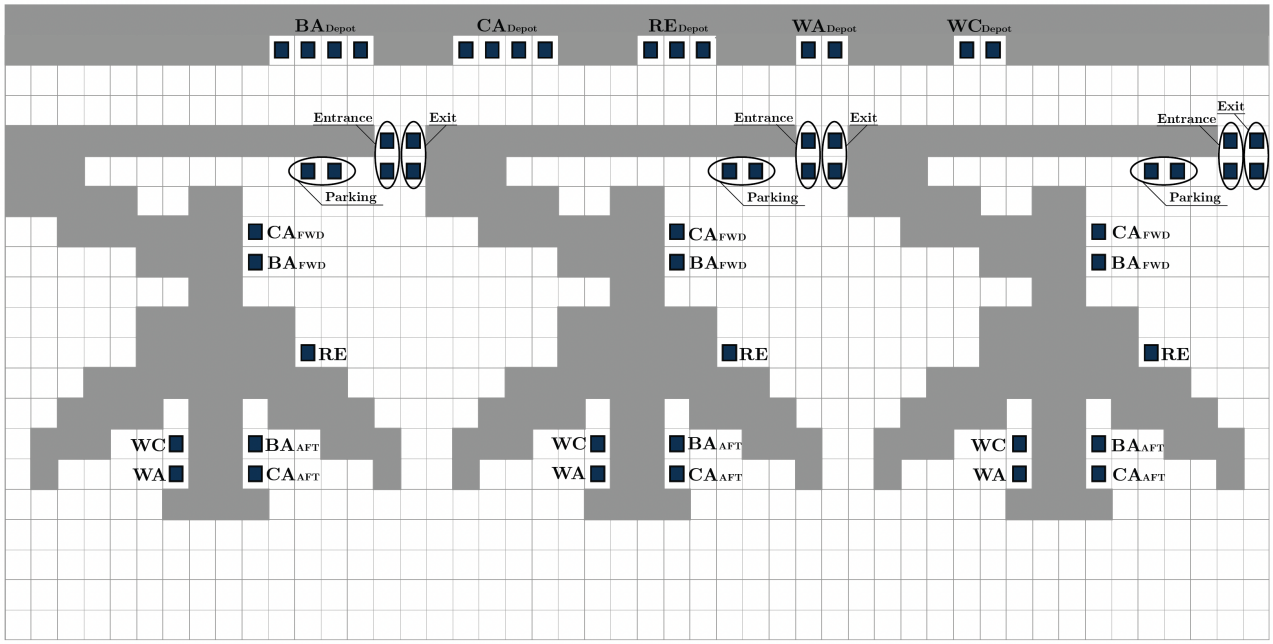


Figure 3: Three aircraft stands modeled in a grid-based environment with cells of 4 m x 4 m, including service and depot locations.

The perimeter road and terminal parking area are modeled with the same cell sizes as the corresponding aircraft stands. Directional constraints are implemented in the grid-map of the service road to ensure that GSE vehicles are driving on the right side of the road. GSE vehicles are shown as rectangular in shape and occupy a single cell. The model adopts a simplified movement scheme for the vehicles, where they navigate exclusively in four cardinal directions: north, south, east, and west. In this approach, rotational movements, acceleration, and deceleration of vehicles are not considered. The speed limit of vehicles on an aircraft stand is 15 km/h [14], therefore the speed of the vehicles is assumed to be one cell per second which is equivalent to 14 km/h.

3 Model Description

This section provides a detailed description of the algorithms proposed to schedule and coordinate autonomous ground handling operations. Autonomous ground handling can be expressed as an MAPD (multi-agent pick-up and delivery) problem. A set of agents $A = \{a_1, a_2, ..., a_n\}$ are expected to perform a set of tasks $T = \{t_1, t_2, ..., t_m\}$ for which the delivery and pickup locations are represented by the nodes of a network. This network is denoted as a complete graph $G = (V, A)$, where the vertices V represent the node locations and the arcs A represent the links between the nodes. The set of arcs A include bidirectional links. Each task $t \in T$ is characterized by a pickup vertex $i \in V$ and a delivery vertex $j \in V$. Generally, the number of tasks is greater than the number of agents, which means that agents must complete a set of tasks while avoiding collisions with other agents. This work models the MAPD problem in an offline setting, which means that all the characteristics of the tasks are known before planning. A solution to the MAPD problem involves assigning agents to tasks and creating a conflict-free routing plan for all agents in the model.

This research introduces a novel algorithmic approach, as depicted in Figure 4. Inputs for the models, such as flight schedules and environment specifics, are presented in Section 2.1. The inputs are used to assign agents to tasks of the same type using a multi-vehicle Pickup and Delivery Problem with Time Windows. An insertion heuristic is used to generate an initial feasible solution to accelerate the PDPTW optimization. This warm start PDPTW model creates the schedules for each GSE vehicle and is explained in detail in Section 3.2. The conflict-free paths for these schedules are obtained using a multi-agent path planning algorithm, which combines prioritized planning with safe interval path planning. This approach is explained in Section 3.3. The replanning

model, which is designed to evaluate the real-time performance of the model, is described in Section 3.4. This model is activated during operations if a significant effect is measured due to a delayed task. For the replanning of tasks, an adapted version of the insertion heuristic is used. The updated schedules are then inserted into the multi-agent path planning algorithm to obtain updated conflict-free paths.

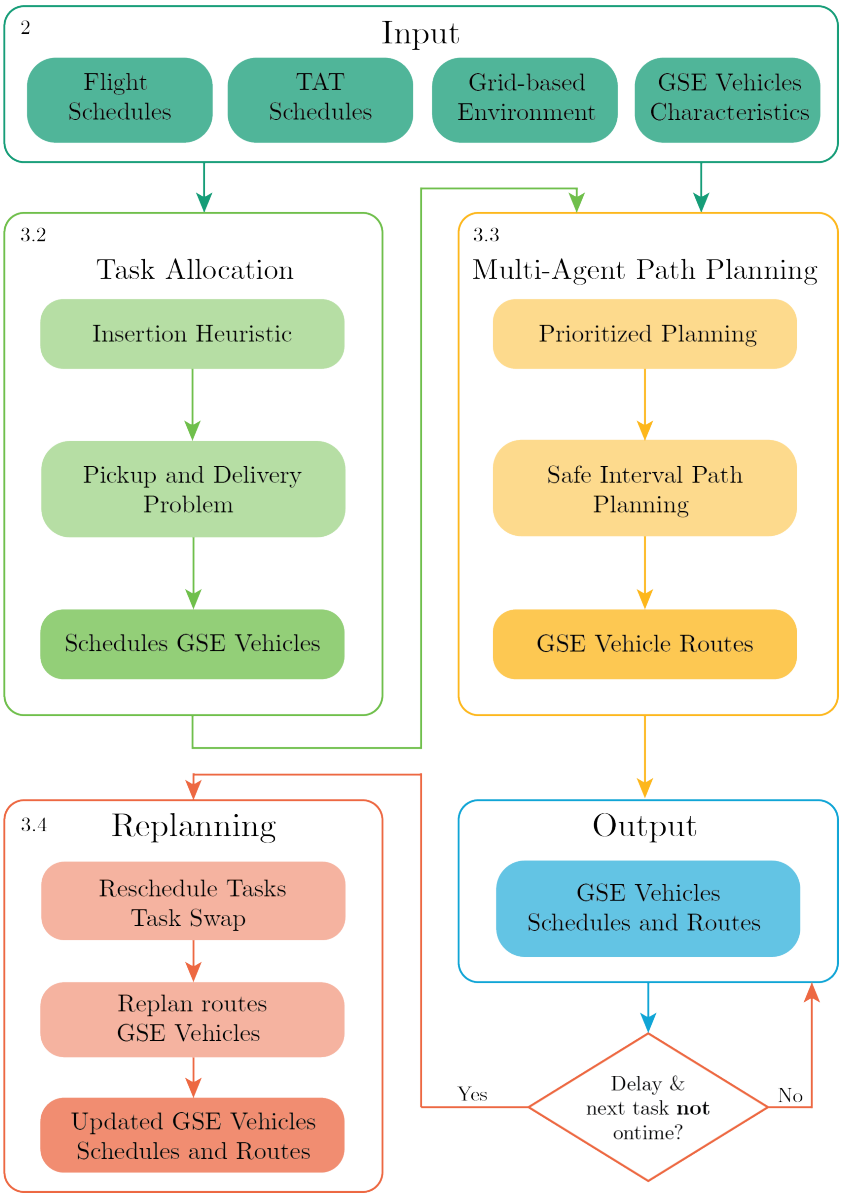


Figure 4: Overview of the task allocation and path planning method. Numbers in the figure refer to the sections they are discussed in.

3.1 Multi-Agent System Specification

An MAPD model can be broken down into four distinct parts: the environment, the agents, the features of the agents, and the interaction between the environment and the agents. The MAPD model for autonomous ground handling operations is visualized in Figure 5. This model has three types of agents that interact with each other and their environment. The task allocation agent is responsible for the task allocation and scheduling of all tasks. This schedule is then communicated to the GSE vehicle agents, who each plan their paths and incorporate them into their itinerary. When the itinerary is executed, the vehicles will begin to move within the environment. The path planning coordinator agent is then also activated and ensures safe movements by resolving conflicts.

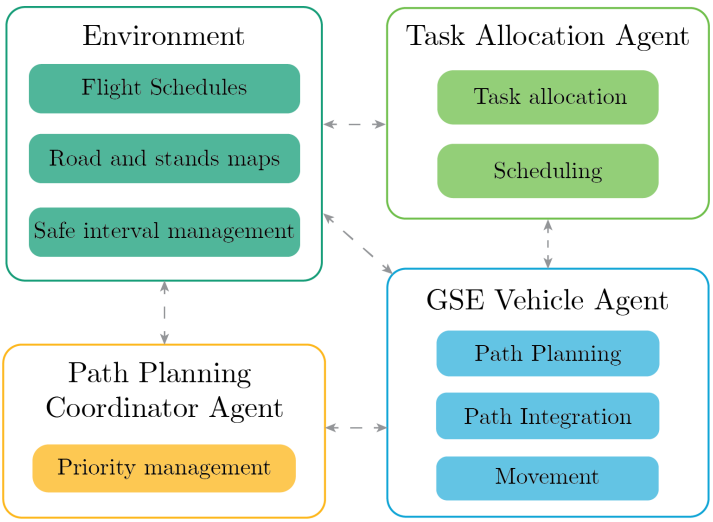


Figure 5: Overview of the Multi-Agent Model and its properties.

3.2 Task Allocation

Turnaround tasks, such as catering, consist of multiple subtasks. Catering galleys need to be loaded onto the catering truck, and this truck delivers the galleys to the corresponding aircraft stand to be loaded into the aircraft. Thus, ground handling tasks can be divided into pick-up and delivery tasks, with each having a distinct service location. An overview of the pickup and delivery subtasks for different ground handling tasks is presented in Table 2. In this table, the baggage handling is divided into a loading and unloading task. Both unloading and baggage loading must be performed during turnaround. Therefore, four separate tasks must be created; two pickup and two delivery tasks. Additionally, each baggage tractor needs to deliver carts to both the front and aft service location. Figure 6 demonstrates the baggage tractor moving between service locations during loading and unloading baggage.

Table 2: Description of pickup and delivery tasks for different ground handling tasks, including capacity of corresponding GSE vehicles.

Task Type	Pickup Task	Delivery Task	Capacity
BA load	Load baggage from terminal	Load baggage into aircraft	1
BA unload	Unload baggage from aircraft	Unload baggage into terminal	1
CA	Load catering galleys at terminal	Swap catering galleys in aircraft	1.5
WA	Fill up water tank of truck at terminal	Fill up water tank of the aircraft	20
WC	Empty waste tank of aircraft	Empty tank of vacuum lavatory truck tank	25
RE	N.A.	Fuel aircraft using an underground fueling system	∞

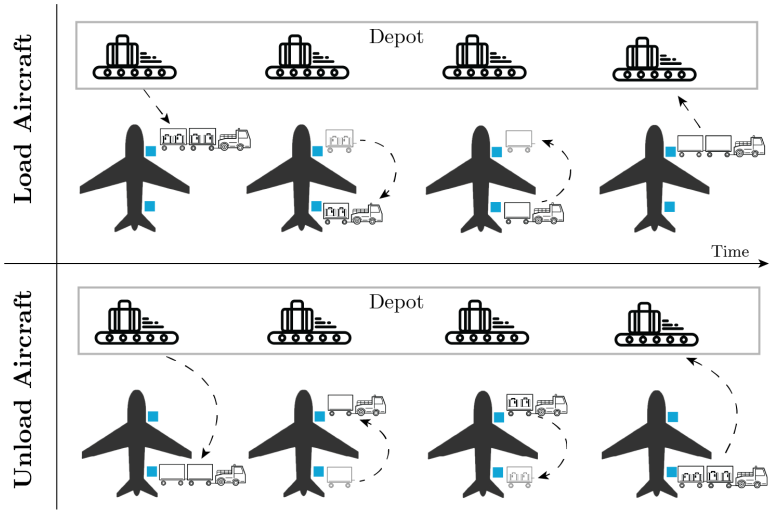


Figure 6: Schematic representation of baggage handling loading and unloading tasks.

Creating separate pick-up and delivery tasks enables us to model the scheduling problem as a pick-up and delivery problem. In this study, each GSE vehicle is restricted by the amount of tasks it can perform with a full load, the time windows linked to each task, and the precedence constraints between the pickup and delivery tasks. The remainder of this section describes the model developed to solve the task allocation problem for autonomous ground service equipment.

3.2.1 Mathematical Model

The model is based on the multi-vehicle paired pickup and delivery problem with time windows presented in a survey of pickup and delivery problems by Parragh et al. [11]. A paired problem implies that the number of pickup tasks is equal to the number of delivery tasks. The PDPTW is modeled on a complete graph $G = (V, A)$ where V is the set of all vertices and A the set of all arcs. A virtual start node 0 and a virtual end node m are added to the graph.

Parameters
The parameters of the model are presented in Table 3. As discussed in Section 2.3, each vehicle must enter and exit an aircraft stand using the designated entry and exit point. The travel distance between two vertices is computed using the Manhattan distance, while taking into account the entry/exit locations it must visit within the environment. Additionally, the vehicle speed is assumed to be uniform, resulting in the travel time being equivalent to the travel distance. As each vehicle of the same type is homogeneous, the travel times are the same for each vehicle and therefore t_{ij}^k equals t_{ij} $\forall$ $k \in K$. Travel times in the real world for GSE vehicles can vary due to congestion on the perimeter road or other unexpected delays of GSE vehicles. To take into account uncertainties, a distance buffer coefficient is utilized. CES proposes a randomly generated variable following a normal distribution with a mean of 1.4 and a standard deviation of 0.2; $Y \sim N(1.4, 0.2^2)$.

Table 3: Notation for sets and parameters of the pickup and delivery problem with time windows. Adapted from [11].

Sets	**Description**
V	set of all vertices $\{0, m\} \cup P \cup D$
A	set of all arcs $\{(i,j) : i,j \in V, i \neq m, j \neq 0, i \neq j\}$
P	set of pickup tasks $\{1, ..., n\}$
D	set of delivery tasks $\{n+1, ..., n+n\}$
V_{depot}	set of all depot vertices
V_{stand}	set of all aircraft stand vertices
K	set of vehicles
Parameters	**Description**
n	number of pickup/delivery tasks; total number of tasks $= 2 \cdot n$
q_i	demand at vertex i; pickup vertices are associated with a positive value and the delivery vertices with a negative value; the demand at the start depot 0 and the end depot m is equal to 0
e_i	earliest time to begin service at vertex i
l_i	latest time to begin service at vertex i
d_i	service duration at vertex i
t_{ij}	travel time from vertex i to vertex j
C^k	capacity for vehicle k

Decision Variables
The decision variables of the model are presented in Equation 1.

$$x_{ij}^k = \begin{cases} 1, & \text{if arc } (i,j) \text{ is traversed by vehicle } k \\ 0, & \text{else} \end{cases} \tag{1a}$$

$$Q_i^k = \text{load of vehicle } k \text{ after leaving vertex } i \tag{1b}$$

$$B_i^k = \text{starting time of service of vehicle } k \text{ at vertex } i \tag{1c}$$

Objective function
The classic goal in vehicle routing problems is to minimize the total distance or time taken to complete all tours. A PDPTW is bound by time windows; therefore, the time to service all tasks, known as the makespan, is used

as objective. The makespan also serves as the primary objective of our model. We define the makespan for each vehicle as the difference between the start times of the virtual start and end node visited by the vehicle, with the aim of minimizing the total makespan of all vehicles. Equation 2 shows the mathematical formulation of the objective function.

$$z_1 = \sum_{k \in K} B_m^k - B_0^k. \tag{2}$$

We use a second objective function to minimize the start times of all tasks in the stands. The mathematical formulation is shown in Equation 3. This objective function enforces that all tasks must be performed as early as possible so that service locations on the stands are not occupied for too long and the turnaround time of each aircraft is minimized.

$$z_2 = \sum_{k \in K} \sum_{i \in V_{stand}} B_i^k. \tag{3}$$

A third objective function is defined to create a solution that improves compliance with the regulations on ground handling in real life. Regulations state that vehicles should minimize their time on an aircraft stand. Vehicles are therefore rewarded for waiting at a parking location instead of the service location on a stand. This is achieved by subtracting the waiting time at a parking location from the minimization objective. The waiting time is calculated using the mathematical formulation shown in Equation 4. The third objective function is non-linear, changing the existing MILP model into a Mixed-Integer Non-Linear Programming (MINLP) model. In our case, the linearization of the non-linear objective function is automatically handled by the Gurobi solver. Although Gurobi is not a non-linear programming solver, it can deal with simple non-linear constraints or objectives using its features for quadratic programming.

$$z_3 = \sum_{k \in K} \sum_{i \in V_{depot}} \left(\sum_{j \in V_{stand}} (B_j^k - t_{ij} - d_i) \cdot x_{ij}^k - \sum_{j \in V} B_i^k \cdot x_{ji}^k \right). \tag{4}$$

In experiments, we use a combination of z_1 and z_2 as a baseline and only include the third objective z_3 in a single experiment to measure improvement in compliance with ground handling regulations. Therefore, the mathematical formulations of the two versions of the objective function are $z = z_1 + z_2$ and $z = z_1 + z_2 - z_3$, respectively.

Pickup and Delivery Problem with Time Windows
The complete PDPTW model is given in Equations 5 to 15.

$$\min \quad z \tag{5}$$

subject to:

$$\sum_{k \in K} \sum_{j:(i,j) \in A} x_{ij}^k = 1, \qquad \forall i \in P \cup D \tag{6}$$

$$\sum_{j:(0,j) \in A, j \neq m} x_{0j}^k = 1, \qquad \forall k \in K \tag{7}$$

$$\sum_{i:(i,m) \in A, i \neq 0} x_{im}^k = 1, \qquad \forall k \in K \tag{8}$$

$$\sum_{i:(i,j) \in A} x_{ij}^k - \sum_{i:(j,i) \in A} x_{ji}^k = 0, \qquad \forall j \in P \cup D, k \in K \tag{9}$$

$$x_{ij}^k = 1 \rightarrow B_j^k \geq B_i^k + d_i + t_{ij}, \qquad \forall (i,j) \in A, k \in K \tag{10}$$

$$x_{ij}^k = 1 \rightarrow Q_j^k = Q_i^k + q_i, \qquad \forall (i,j) \in A, k \in K \tag{11}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\{C^k, C^k + q_i\}, \qquad \forall i \in V, k \in K \tag{12}$$

$$\sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(n+i,j) \in A} x_{n+i,j}^k = 0, \qquad \forall i \in P, k \in K \tag{13}$$

$$B_i^k \leq B_{n+i}^k, \qquad \forall i \in P, k \in K \tag{14}$$

$$e_i \leq B_i^k \leq l_i, \qquad \forall i \in V, k \in K \tag{15}$$

(16)

Constraint 6 states that every vertex must be served exactly once. Constraints 7 and 8 ensure that each vehicle starts at the depot and ends at the depot. To ensure that each vehicle is used, the route from depot 0 to depot m is forbidden. Constraint 9 ensures the continuity of the flow between vertices, while 10 utilizes time variables to eliminate any sub-tours, provided that $t_{ij} + d_i > 0$ for all $(i, j) \in A$. Constraint 10 is linearized using the big M method; $(1 - x_{ij}^k) \cdot M + B_j^k \leq B_i^k + d_i + t_{ij}^k$. Both 11 and 12 are required to guarantee that the capacity of each vehicle is not exceeded. In paired pickup and delivery, both the pickup and delivery tasks must be served by the same vehicle, and a delivery task can only start after the pickup task is finished. The constraints 13 and 14 are related to these rules. Finally, constraint 15 guarantees that each task must be served within its time window.

In a PDPTW model, the vehicles are homogeneous. In our problem, we have a heterogeneous fleet of GSE vehicles. For each type of GSE vehicle, there is a homogeneous set of GSE vehicles. Therefore, we solve a separate PDPTW model for a homogeneous GSE fleet for each type of GSE vehicle. This allows us to adapt the PDPTW model to be adapted for each type of vehicle to better fit the specifications of the related type. Table 4 shows an overview of the adaptations per type of GSE vehicle. Catering galleys and baggage of passengers must be delivered to the correct aircraft and, therefore, the pickup and delivery tasks are paired. Trucks used for watering and toilet service have a large tank that is used to fill and empty aircraft tanks, respectively. Here, only capacity limits the ability to serve a task on an aircraft, so tasks are unpaired. This enables us to simplify the watering and toilet servicing model by removing constraints 13 and 14. Finally, as discussed in Section 2.3, AAS features an underground fueling system, and therefore fuel hydrant vehicles without capacity constraints are used for fueling tasks. This allows us to remove capacity constraints 11 through 14 and the load decision variable Q_i^k. Figure 7 shows a visual representation of a possible partial solution for the task allocation model.

Table 4: Adaptations to PDPTW model for each type of GSE vehicle.

GSE vehicle	Time Windows	Capacitated	Paired
Baggage tractor (BA)	✓	✓	✓
Catering truck (CA)	✓	✓	✓
Portable water truck (WA)	✓	✓	
Vacuum lavatory truck (WC)	✓	✓	
Hydrant vehicle (RE)	✓		

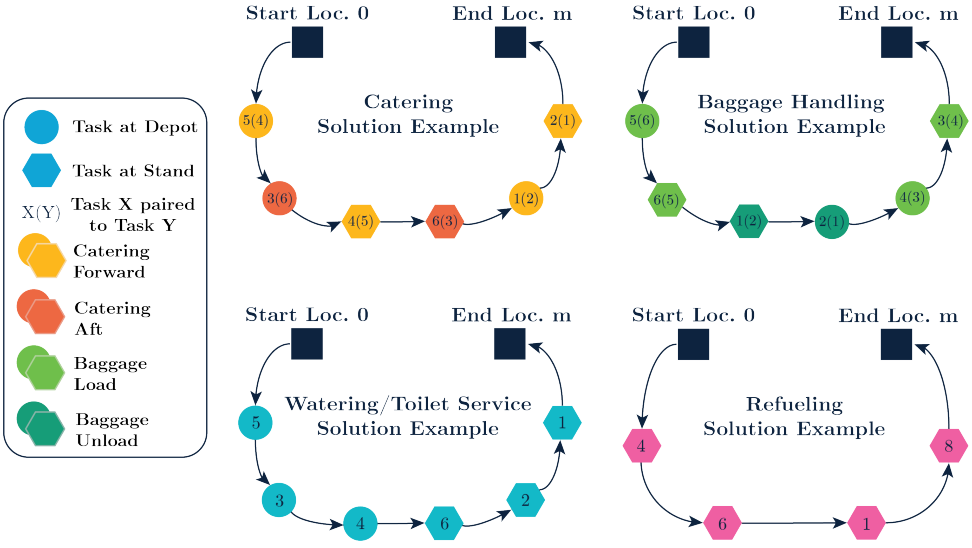


Figure 7: Graph-based representation of task allocation solutions for each type of GSE vehicle.

3.2.2 Insertion Heuristic

Vehicle routing problems are known to be NP-hard, PDPTW is also known to be NP-hard. We use an insertion heuristic to generate the initial feasible solution for the solver that we use to solve the mixed-integer programming model of the PDPTW. In this way, we eliminate the time the solver spends while searching an initial feasible solution, which can be costly for problem instances which have specific features or considerably larger sizes, and the mixed integer programming solver initiates a search starting from the warm start solution we provide.

For vehicle routing problems with time windows, Solomon [12] concluded that an insertion-type heuristic is the most suitable. Three versions of an insertion heuristic were studied. The first version maximizes the benefit of adding a set of tasks to an existing route in comparison to a separate route. The second version focuses on minimizing the route distance, and the third version includes urgency to complete a task. The primary focus of this research is to minimize the makespan and completion times of tasks, so the third version is chosen. The insertion cost of any task u that is to be placed between task i and j is obtained by Equation 17 where $c_1(i,u,j) = d_{iu} + d_{uj} - d_{ij}$, $c_2(i,u,j) = b_{j_u} - b_j$, and $c_3(i,u,j) = l_u - b_u$. $\alpha_1 + \alpha_2 + \alpha_3 = 1$, $\alpha_1 \geq 0$, $\alpha_2 \geq 0$, and $\alpha_3 \geq 0$.

$$c(i,u,j) = \alpha_1 \cdot c_1(i,u,j) + \alpha_2 \cdot c_2(i,u,j) + \alpha_3 \cdot c_3(i,u,j). \tag{17}$$

The urgency of the tasks, the distance and the service time of the subsequent tasks are considered in c_1, c_2, and c_3, respectively. The three constants α_1, α_2, and α_3 are used to weigh the importance of each sub-equation. For autonomous ground service vehicles, the goal is to minimize time, rather than distance. Therefore, α_1 is set to 0 and α_2 and α_3 are set to 0.5 each. To solve the autonomous ground handling task allocation problem using Solomon's insertion heuristic, we made some adaptations. In the insertion heuristic of Solomon, a single task is evaluated per iteration and there is an unlimited number of homogeneous vehicles available. In our work, we evaluate the pick-up and the corresponding delivery tasks together to insert them into a route per iteration. We run the insertion heuristic separately for each type of GSE vehicle, where there is a finite set of vehicles for each type. A description of the adapted insertion heuristic can be found in Algorithm 1.

Algorithm 1: Adapted Insertion Heuristic

Input : GSE vehicle data, Flight Schedule, TAT schedule
Output: Schedule for GSE vehicles

```
Solution = ∅
Insertion_cost = ∞
Best insertion_cost = ∞
while Unassigned tasks do
    P, D ← Pickup and delivery task with earliest deadline
    for k ∈ K do
        foreach pos_p in route vehicle k do
            P.Insertion_cost(pos_p) = c(i, P, j)
            foreach pos_d following pos_p in route vehicle k do
                D.Insertion_cost(pos_d) = c(i, D, j)
            PD.insertion_cost = P.Insertion_cost + D.Insertion_cost
            if PD insertion_cost < Insertion_cost then
                Insertion_cost = PD Insertion_cost
        if Insertion_cost < Best insertion_cost then
            Best insertion_cost = Total insertion_cost
            Best vehicle = k
    if Best insertion_cost ≠ ∞ then
        Unassigned tasks.remove(P, D)
        Solution[best vehicle].insert(P, D)
        Insertion_cost = ∞
        Best insertion_cost = ∞
```

3.3 Integrated Path Planning

Following the task allocation model, a path planning algorithm is implemented based on the work of CES. This algorithm, called Prioritized SIPP, combines prioritized planning as the high-level solver with Safe Interval Path Planning as the low-level solver, resulting in conflict-free individual vehicle paths. Safe Interval Path Planning is a single agent path finding algorithm first introduced by Phillips and Likhachev [15] and considers other agents within the environment as dynamic obstacles. Dynamic obstacles in the environment change their position from one time step to the next, requiring the creation of a large number of constraints based on discrete time steps. To simplify this process, rather than discrete timesteps, safe time intervals are used. A safe interval is a continuous period of time during which it is safe for an agent to occupy the grid. This reduces the number of constraints due to the usage of time intervals rather than time steps. Combining SIPP and prioritized planning requires storing routes of higher priority vehicles and using them to modify safe intervals for agents with

lower priority. Ma et al. [16] proposed a reservation table that enables the integration of prioritized planning and SIPP. This allows vehicles to move between their starting and end points without the occurrence of collisions.

The schedules generated in the task allocation model contain a list of tasks to perform, the time the task should be performed, and the load per GSE vehicle. To move from one task location to another, it is necessary to visit intermediate stops such as the entrance/exit of an aircraft stand. The route can be broken down into two segments: the service road environment and the aircraft stand environment. This allows separate path finding, as discussed in Section 2.5. The working structure of Prioritized SIPP is demonstrated in Algorithm 2. The model uses the set of agents $A = a_1, a_2, ..., a_n$, the schedules generated in the task allocation model, the maps and safe intervals of the service road, and the aircraft stands, as input. The order in which the paths of the agents are planned is based on the priorities of the GSE vehicles, as listed in Table 1. Once the paths are planned, safe intervals are updated to ensure conflict-free paths. The output is the set of paths for each agent.

Algorithm 2: Prioritized Safe Interval Path Planning

Input : Set of agents A, Schedule of GSE vehicles, Map of service road, Map of aircraft stand, Safe intervals of road, Safe intervals of stands

Output: Set of paths P_a for agent a, $a \in A$

```
Safe_Interval_road = (0, ∞)
Safe_Interval_stand = [(0, ∞), (0, ∞), (0, ∞)]
foreach a ∈ A do
    foreach segment ∈ schedule of agent a do
        if segment ∈ Map_road then
            path = SIPP(segment_start, segment_goal, Map_road, Safe_Interval_road)
            P_a.append(path)
            Safe_Interval_road = Update_Safe_Interval(path, Map_road, Safe_Interval_road)
        else if segment ∈ Map_stand then
            idx = number of aircraft stand
            path = SIPP(segment_start, segment_goal, Map_stand, Safe_Interval_stand[idx])
            P_a.append(path)
            Safe_Interval_stand[idx] = Update_Safe_Interval(path, Map_stand, Safe_Interval_stand[idx])
```

This algorithm differs from the SIPP algorithm of Phillips and Likhachev [15] to better fit the path planning of GSE vehicles. The determination of successors with the function *getSuccessor* and possible moves are also restricted due to directional constraints as discussed in Section 2.5. For a detailed description of the differences, the reader is referred to the research by CES [9].

3.4 Replanning

Ground handling is susceptible to delays due to a variety of shareholders, including passengers, airlines, and GSE vehicle providers. The turnaround schedule is tightly planned, so if one task is delayed, it could cause a domino effect and delay other tasks on the GSE vehicle schedule. Consequently, the ability to reassign tasks to different vehicles in order to reduce total delays is critical. We evaluate the feasibility of real-time implementation of our model by simulating disruptions that trigger a replanning process. This process is activated when two conditions occur: (1) several tasks are delayed due to the disruption, (2) the total deviation is substantial. An overview of the process is visualized in Figure 8.

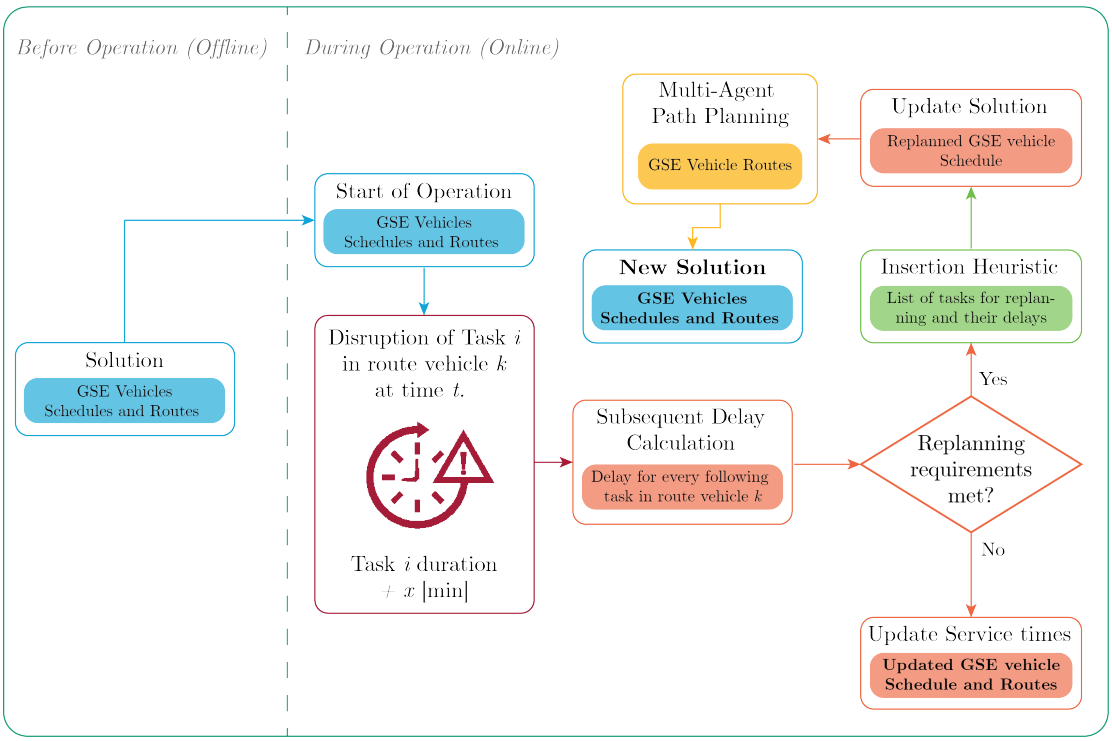


Figure 8: Flowchart of replanning process.

When a disruption occurs, all subsequent tasks must be re-evaluated for replanning. However, the pickup subtasks that have already been completed prior to disruption will remain as originally planned. Significantly delayed tasks caused by the disruption are re-planned using an adapted version of the insertion heuristic presented in Section 3.2.2. The inner workings of the heuristic remain the same, only the insertion cost equation is adapted. The time windows in the heuristic are widened, so that it can try to find a better solution, but it can still assign the task to the original vehicle, ensuring a complete solution. The assignment of a delayed task adds a penalty to the insertion cost. The adapted insertion cost equation is shown below.

$$c(i,u,j) = \quad \alpha_1 \cdot c_1(i,u,j) + \alpha_2 \cdot c_2(i,u,j) + \alpha_3 \cdot c_3(i,u,j) + t_{delay} * \rho. \tag{18}$$

The newly found solution is then inserted into the path planning algorithm to obtain updated paths, and operation can continue using these updated paths. The performance of the model is measured in Section 5.3.

4 Verification and Validation

In our research, we want to make sure that the multi-agent task allocation and path planning model works properly and the results are realistic. Autonomous ground handling is not yet performed in real life, and there are no other open-source models for multi-agent task allocation and path planning, therefore, only validation by industry experts can be performed. For this model, all input parameters are determined together with an operational expert. In addition, modeling approaches are evaluated to ensure close resemblance with actual ground handling operations. In the remainder of the section, we focus on the verification process explaining the verification steps for each part of the model. First, we discuss the verification steps and results for the task allocation model. Following this are the integrated path planning and replanning verification steps discussed.

The pickup and delivery problem with time windows (PDPTW) is a well-known subclass of the vehicle routing problem with pickups and deliveries. Li and Lim [17] created a metaheuristic to solve PDPTW problems. In addition, they created data sets to test the performance of the metaheuristic for solving large multi-vehicle PDPTW problem instances. These data sets are currently used as a benchmark in the literature. These data sets are used as a verification method for the PDPTW problem developed for this research. Our model assigns 48 tasks at most for a single GSE type with a four-hour planning window. Therefore, instances with 100 tasks are chosen for verification due to their size similarity. The objective of this benchmark consists of two goals: (1) minimizing the number of vehicles, (2) minimizing the total distance. The best-known solutions are

published online[1]. Adapting our model to match the objective used in the benchmark instances is achieved by allowing the usage of arc $(0, m)$ in constraints 7 and 8. The new constraints become $\sum_{j:(0,j)\in A} x_{0j}^k = 1$ and $\sum_{i:(i,m)\in A} x_{im}^k = 1$ for k in K. If a vehicle uses the $(0, m)$ arc, this implies that the vehicle does not leave the depot and is therefore not used. Using 100 randomly chosen task benchmark instances as input for our task allocation model resulted in an identical number of vehicles and negligible ($< 0.1\%$) differences in distance compared to the best known solutions. Although our PDPTW model has a limited solution time, it can still find solutions similar to the state-of-the-art metaheuristics. From these results, we conclude that the developed PDPTW model is verified. The objectives used for ground handling operations, as defined in Section 3.2.1, are verified using a small input data set for which the solution is known. The insertion heuristic is verified using a combination of unit and integration tests using the same small input data set.

The small input data set was also used to confirm that the path planning model works as intended. First, we conducted unit tests, which helped us examine each part of the model in isolation to ensure that they functioned correctly. Then, we performed integration tests to assess the performance of all the model's components together. Furthermore, we use dynamic visualization to make the verification process more informative. This involved creating animations that showed the behavior of the model over time. These animations not only confirmed that the model was working correctly, but also allowed us to observe the inner workings in the simulation environment. Finally, the replanning model reuses verified algorithms. Therefore, only integration tests using a small input data set were necessary.

5 Results

To quantify the performance of the multi-agent model, several experiments are executed. Each experiment is performed using the infrastructure, the TAT schedule, and vehicle characteristics presented in Section 2. Aircraft flight schedules are generated in a planning window of four hours. The turnaround times of the aircrafts are determined using a uniform distribution, while the duration and time windows of the tasks are identical for all flights (see Figure 1(b)). Table 6 shows the time windows for each task with respect to the arrival of the aircraft at the stand. The total number of GSE vehicles for each type is presented in Table 5. Gurobi 10.0.1 is used for task allocation, all experiments are carried out on an 8-core Apple M1 Pro chip with 16GB RAM.

Table 5: Resources available for three aircraft stands in a four-hour planning window based on discussions with experts. Resources are presented in order of priority.

GSE	Total	Capacity
RE	3	∞
CA	4	1.5
BA	4	1
WA	2	20
WC	2	25

Table 6: Time windows of start time for each task in seconds after the arrival of an aircraft.

Task	Time Window [s] (fwd, aft)
RE	[300, 420]
CA	[360, 480], [360, 480]
BA load	[1260, 1380], [1260, 1380]
BA unload	[240, 260], [60, 180]
WA	[60, 2100]
WC	[600, 1320]

5.1 Task Allocation Experiments

Evaluating the task allocation mechanism is done with two experiments. The first experiment focuses on different traffic demands. Amsterdam Airport Schiphol is a hub-and-spoke airport, which means that there are peak periods where passengers transfer from long-haul to short-haul aircraft and vice versa. During these periods, there is a higher traffic demand. To test whether the model can handle both peak and normal traffic demand, two scenarios are created. To simulate a peak period, the turnaround time of an aircraft is shortened while keeping the planning window constant, resulting in more flights to be serviced. The range of the turnaround time in the normal scenario is between 50 and 60 minutes; $U \sim (50, 60)$. In the peak scenario, the turnaround time is between 45 and 55 minutes; $U \sim (45, 55)$. This results in 12 flights in the normal scenario and 14 flights in the peak scenario.

The Key Performance Indicators (KPIs) that are used to quantify the performance of the turnaround time model are those shown in Table 7. The first four metrics measure the makespan and the average start time of tasks in a stand that are minimized in the baseline objective; Equations 2 and 3, respectively. The statistical significance of the increase in flights is tested using the non-parametric Wicoxon signed-rank test. This statistical test works well with non-normally distributed, paired data. Results are expressed as p-values and the statistical

[1] https://www.sintef.no/projectweb/top/pdptw/100-customers/

significance is concluded for $p < 0.05$. In addition, the Vargha-Delaney A-values are computed. This test measures the magnitude of the statistical difference between scenarios. There is a large effect size for $A > 0.71$ or $A < 0.29$.

Table 7: Key Performance Indicators for Task Allocation Experiments.

KPI	Description
$\bar{M}_{tot}$	Average makespan for all GSE vehicles [min]
$\bar{M}_{GSE}$	Average makespan for a single type of GSE vehicles [min]
$\bar{s}_{tot}$	Average start time of a task on a stand for all GSE vehicles [min]
$\bar{s}_{GSE}$	Average start times of a task on a stand for a single type of GSE vehicles [min]
$\bar{t}_{tot}$	Average computational time of all GSE vehicles [s]
t_{GSE}	Computational time of a single type of GSE vehicles [s]
W_{tot}	Wilcoxon signed rank test p-value for all GSE vehicles [-]
W_{GSE}	Wilcoxon signed rank test p-value of a single type of GSE vehicles [-]
A_{tot}	Vargha-Delaney A-value for all GSE vehicles [-]
A_{GSE}	Vargha-Delaney A-value of a single type of GSE vehicles [-]

The number of flights handled within a four-hour window is the primary factor that affects KPIs. We hypothesize that the more flights that need to be handled, the higher our KPIs will be. Our hypotheses are as follows:

$\mathbf{H}_{2.1}$: *The average makespan ($\bar{M}_{tot}$) under the peak traffic demand is higher than under the normal traffic demand.*
$\mathbf{H}_{2.2}$: *The total computational time (t_{tot}) under peak traffic demand is higher than under the normal traffic demand.*

Before starting the optimization model, an initial solution is generated using the insertion heuristic. Inserting an initial solution into the optimization model improves computational time, increases efficiency, and, therefore, improves the quality of the solution. The average computational time of the heuristic to generate a solution for all GSE vehicles is 0.08 and 0.09 seconds for the normal and peak scenario, respectively. The runtime of the optimization model is limited to 30 seconds per GSE type. Analysis of the improvement of the objective function over time shows that the specified runtime is sufficient to obtain high-quality results.

Results on the makespan, the computational time and the average start time of the two scenarios are shown in Table 8. The average makespan of all GSE vehicles decreases with 5% in the peak traffic scenario compared to the normal traffic scenario. Looking at the makespan for each type of GSE vehicle separately shows a slight increase in makespan for all types except for the GSE vehicles used for toilet service. Here, we measure a decrease of 32% in makespan. The resulting GSE vehicle routes show that in the peak scenario, the second toilet service vehicle is used only for a single task, significantly decreasing the makespan of this vehicle while keeping the makespan of the second vehicle constant. Therefore, our first hypothesis, $\mathbf{H}_{2.1}$, is rejected. Due to the limitation of the runtime to 30 seconds, the difference in runtime is minimal. The computational time of the peak traffic scenario is higher, so the second hypothesis $\mathbf{H}_{2.2}$ is supported. The main difference is that for the catering tasks, an optimal solution is found for the normal scenario within the time limit.

Table 8: Results for task allocation experiments.

KPI	Normal	Peak	KPI	Normal	Peak	KPI	Normal	Peak
$\bar{M}_{RE}$	235.15	215.61	t_{RE}	0.038	0.041	$\bar{s}_{RE}$	129.89	133.00
$\bar{M}_{CA}$	246.03	253.03	t_{CA}	10.723	30.370	$\bar{s}_{CA}$	130.83	134.10
$\bar{M}_{BA}$	251.16	270.30	t_{BA}	30.282	30.381	$\bar{s}_{BA}$	135.89	139.08
$\bar{M}_{WA}$	134.20	136.28	t_{WA}	30.041	30.071	$\bar{s}_{WA}$	126.63	129.09
$\bar{M}_{WC}$	201.62	136.03	t_{WC}	30.051	30.057	$\bar{s}_{WC}$	125.83	129.00
$\bar{M}_{tot}$	213.62	202.65	$\bar{t}_{tot}$	20.225	24.185	$\bar{s}_{tot}$	129.82	132.85

To assess whether increasing the number of flights causes a significant difference in performance, statistical tests are performed. For each type of GSE vehicle, the p-value and the A-value are calculated for the makespan, as well as the overall average. The computational times for each set of vehicles of the same type are compared for both traffic scenarios and result in a single p-value and A-value. These results are summarized in Table 9. Statistically significant results are highlighted in bold. No statistical difference is measured for the makespan, the p-value of $\bar{M}_{tot}$ is higher than 0.05 and the A-value is lower than 0.71, and therefore confirms that there is no significant performance difference in terms of the makespan between the two traffic scenarios. Although there

are no statistically significant differences in computational times, the magnitude of the difference is significant. From this we can deduce that the difference in computational time for increasing the amount of flights is relevant for real-world applications.

Table 9: Statistical results for task allocation experiments. N = normal traffic scenario, P = peak traffic scenario.

KPI	N vs. P	KPI	N vs. P
$W_{\bar{M}_{RE}}$	0.75	$A_{\bar{M}_{RE}}$	0.56
$W_{\bar{M}_{CA}}$	0.63	$A_{\bar{M}_{CA}}$	**0.75**
$W_{\bar{M}_{BA}}$	0.38	$A_{\bar{M}_{BA}}$	0.69
$W_{\bar{M}_{WA}}$	1.00	$A_{\bar{M}_{WA}}$	0.50
$W_{\bar{M}_{WC}}$	1.00	$A_{\bar{M}_{WC}}$	0.50
$W_{\bar{M}_{tot}}$	0.74	$A_{\bar{M}_{tot}}$	0.60
$W_{t_{tot}}$	0.13	$A_{t_{tot}}$	**0.72**

5.1.1 Non-linear objective function

The inclusion of the third objective, Equation 4, in the task allocation model is also tested. The included objective minimizes waiting times in the stands. The normal scenario is used for this experiment and metrics are compared between the baseline ($z = z_1 + z_2$) and the extended objective ($z = z_1 + z_2 - z_3$). The actual waiting times at the stand resulting from the task allocation model are calculated separately for each type of vehicle to detect improvements. In addition, other metrics are used, such as the makespan and computational time. The optimization time is limited to 60 seconds for the non-linear objective function, due to the increase in complexity of the model.

Table 10: Key Performance Indicators for Task Allocation Change Objective Experiments.

KPI	Description
$\bar{M}_{tot}$	Average makespan for all GSE vehicles [min]
$\bar{M}_{GSE}$	Average makespan for a single type of GSE vehicles [min]
$\bar{D}_{tot}$	Average waiting time at the depot for all GSE vehicles [min]
$\bar{D}_{GSE}$	Average waiting time at the depot for a single type of GSE vehicles [min]
$\bar{S}_{tot}$	Average waiting time at a stand for all GSE vehicles [min]
$\bar{S}_{GSE}$	Average waiting time at a stand for a single type of GSE vehicles [min]

Making this added objective work, requires locations where vehicles can stay longer periods. The depot locations of each vehicle are used as a waiting location. All vehicles must visit these locations to load or unload, except the uncapacitated refueling vehicles. Therefore, additional tasks are created at the depot for the refueling vehicles. The following hypotheses can be formulated.

$\mathbf{H}_{2.3}$: *The average makespan ($\bar{M}_{tot}$) of the model with the non-linear objective function is higher than that of the linear objective function.*
$\mathbf{H}_{2.4}$: *The average waiting time ($\bar{S}_{tot}$) at the stand of the model with the non-linear objective function is lower than that of the linear objective function.*

The KPIs for both objectives are shown in Table 11. The difference in makespan is only 0.11% and therefore negligible. Therefore, hypothesis $\mathbf{H}_{2.3}$ is rejected. Significant results are found for the average waiting times, with an increase in waiting times of 65% at the depot and a decrease waiting times of 98% at the stand for the non-linear objective function. Thus, hypothesis $\mathbf{H}_{2.4}$ is accepted. The average computational time is 20.23 and 60.16 s for the linear and nonlinear model, respectively.

Table 11: Results for Change Objective Experiments.

KPI	MILP	MINLP	KPI	MILP	MINLP	KPI	MILP	MINLP
$\bar{M}_{RE}$	235.15	235.24	$\bar{D}_{RE}$	31.43	114.92	$\bar{S}_{RE}$	85.62	0.00
$\bar{M}_{CA}$	246.03	246.04	$\bar{D}_{CA}$	105.45	130.09	$\bar{S}_{CA}$	24.17	0.00
$\bar{M}_{BA}$	251.16	252.12	$\bar{D}_{BA}$	52.68	90.575	$\bar{S}_{BA}$	43.03	4.93
$\bar{M}_{WA}$	134.20	134.22	$\bar{D}_{WA}$	55.32	83.78	$\bar{S}_{WA}$	27.59	0.00
$\bar{M}_{WC}$	201.62	201.65	$\bar{D}_{WC}$	98.97	151.98	$\bar{S}_{WC}$	51.52	0.00
M_{tot}	213.62	213.85	D_{tot}	68.77	114.27	S_{tot}	46.39	0.99

5.2 Integrated Path Planning Experiments

The performance of the path planning model is assessed by computing the KPIs defined in Table 12. These KPIs will be calculated for the regular traffic situation and compared with the solution with the shortest path. The shortest path solution calculates the paths of all agents separately without considering conflicts, which allows us to measure the decrease in optimality of the scenario solution. The difference between the two solutions indicates whether there are many conflicts during operation and how these conflicts are resolved. The CES path planning results showed that the increase in path length for a traffic scenario with 16 flights was 0.12%, which is negligible; however, the duration increased with 28%. We expect to see the same behavior here and therefore the following hypotheses are defined.

Table 12: Key Performance Indicators for Integrated Path Planning Experiments.

KPI	Description
$\bar{l}_{path}$	Average route length for all GSE vehicles [m]
$\bar{d}_{path}$	Average route duration for all GSE vehicles [s]
$\bar{a}_{delay}$	Average delay per agent [s]
$\bar{T}_{delay}$	Average delay per task [s]
T_{ontime}	Task ontime rate [-]
a_{succes}	Agent success rate [-]
t_{tot}	Total computational time of all GSE vehicles [s]

$\mathbf{H}_{3.1}$: *The increase in the average route length $\bar{l}_{path}$ for Prioritized SIPP is negligible (<1%) compared to the shortest path length.*
$\mathbf{H}_{3.2}$: *The increase in the average route duration $\bar{d}_{path}$ for Prioritized SIPP is significant (≥10%) compared to the duration of the shortest path.*

Table 13: Results for Path Planning Experiments. S.P. = Shortest Path, P.SIPP = Prioritized Safe Interval Path Planning.

	Normal Scenario		**Peak Scenario**		**Difference**	
KPI	*S.P.*	*P. SIPP*	*S.P.*	*P. SIPP*	*Normal*	*Peak*
$\bar{l}_{path}$	1794.93	1794.93	2230.67	2232.26	+0.00%	+0.07%
$\bar{d}_{path}$	423.8	423.93	527.07	634.20	+0.03%	+20.3%
$\bar{a}_{delay}$	0.00	0.00	0.00	4.53	–	–
$\bar{T}_{delay}$	0.00	0.00	0.00	0.37	–	–
T_{ontime}	1.00	1.00	1.00	0.98	0.00%	−2.00%
a_{succes}	1.00	1.00	1.00	1.00	–	–
t_{tot}	3.113	3.120	4.018	4.023	+0.21%	+0.12%

From the result in Table 12, it can be concluded that the paths formed using prioritized SIPP are identical to the shortest path for the normal traffic scenario. No conflicts occur during the planning window, forcing vehicles to wait or move around, and all tasks are served on time. Therefore, hypothesis $\mathbf{H}_{3.1}$ is accepted and $\mathbf{H}_{3.2}$ is rejected for the normal traffic scenario. However, we see different behavior for the peak traffic scenario. There is a significant increase in the duration of the route and a negligible increase in the length of the route. Thus, both hypotheses are accepted for the peak traffic scenario.

In order to identify any bottlenecks and analyze the routes of all vehicles, a heat map of the path planning model of both traffic scenarios is created. Since the heat maps of both scenarios are similar, only the heat map of the normal traffic scenario is displayed in Figure 9. In this heat map, all paths are plotted for 100 simulations to improve visualization. The main result of the heat maps is the high occupancy of parking locations. These parking locations are located at the top of the aircraft stand, where GSE vehicles are parked at airports. To avoid conflicts, the refueling vehicles and baggage handling vehicles move to a parking location after finishing their task. Other types of tasks can wait at the depot for a longer period of time to avoid this waiting time at the aircraft stand. This behavior differs from the path planning results of CES. There, vehicles waited next to their service location instead of waiting at the depot. Furthermore, similar to the results in CES, vehicles prefer to wait until a conflict is resolved rather than take a detour, as only the shortest paths are found on the heat map.

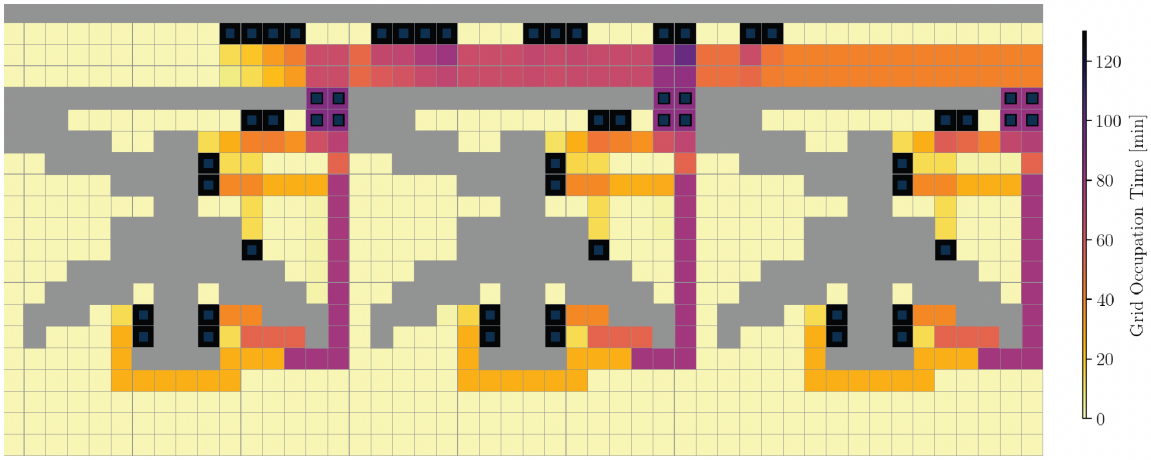


Figure 9: Heatmap of the planned paths for the normal traffic scenario. The squares represent the depots, service locations, and entrances and exits of bays.

5.3 Replanning Experiments

With replanning experiments, we measure the robustness of the model. Disruptions in the form of delays are modeled for ground handling tasks. Baggage handling tasks are chosen for this experiment due to the large number of external factors. Passengers could be a no-show at the gate, a transfer flight could be delayed, or luggage loading takes longer, each resulting in delays. A single disturbance is modeled during a loading task. The delay is defined as a uniform distribution, with a lower limit of 30% and an upper limit of 50% of task duration; $U \sim (0.3 \cdot d_i, 0.5 \cdot d_i)$. The increase in task duration is updated in the route, and checks are performed to assess whether subsequent tasks can still be executed on time. If multiple subsequent tasks are delayed and the delay is more than five minutes, replanning is activated. All tasks scheduled after the disruption are replanned using the adapted insertion heuristic described in Section 3.4. The KPIs used for these experiments are the single-type GSE vehicle metrics used in the task allocation experiments, see Table 7. The heuristic finds feasible solutions, but this solution is not optimized for the makespan. Thus, it is expected that the average makespan of the replanned schedule is higher than that of the original solution. The main criterion for the insertion heuristic is the service time of tasks, and which is why we expect the average start time of tasks to be lower than the original solution. The following two hypotheses are defined on the basis of this reasoning.

$\mathbf{H}_{4.1}$: *The average makespan $\bar{M}_{BA}$ of the replanned schedule is higher than under the disrupted schedule.*
$\mathbf{H}_{4.2}$: *The average task start time $\bar{s}_{BA}$ of the replanned tasks is lower than under the original schedule.*

The results are presented in Table 14. In this experiment, a disruption causes a delay of six minutes for a single task. Due to this delay, two tasks are delayed with a delay of 317 seconds total. Since baggage handling vehicles have waiting times at parking locations, these times are used to compensate for the delay. Therefore, only two tasks are delayed. This is also the reason for the makespan of the disrupted routes to be equal to the original solution. The makespan of the replanned routes is 4.38% higher than the original and disrupted routes, supporting hypothesis $\mathbf{H}_{4.1}$. The average start time for replanned tasks is increased by 0.05% compared to the original solution and rejects hypothesis $\mathbf{H}_{4.2}$.

Table 14: Results for replanning experiments.

KPI	Replanned	Disrupted	Original
$\bar{M}_{BA}$	262.17	251.16	251.16
$\bar{s}_{BA}$	815.72	820.60	815.32

6 Sensitivity Analysis

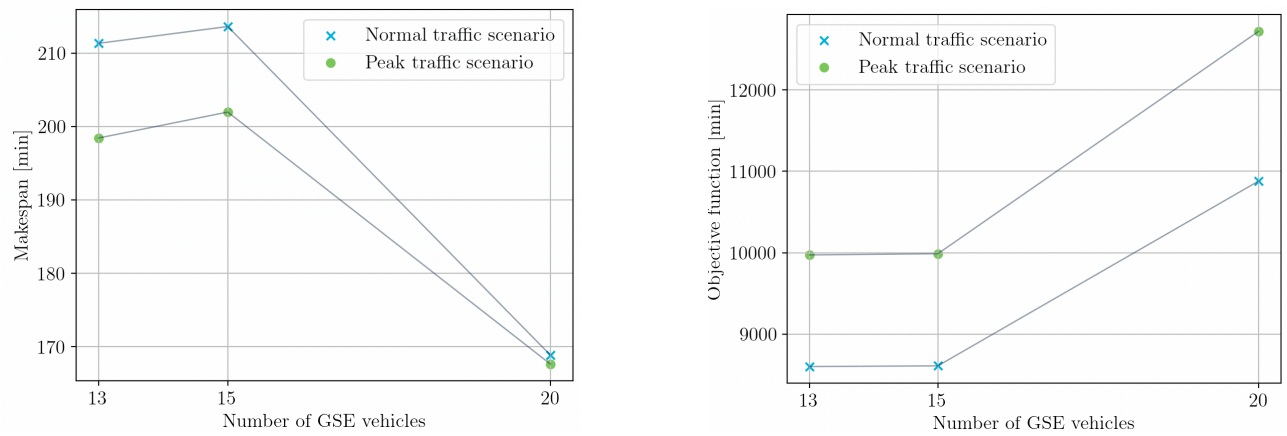
To assess the significance of the input parameters, a sensitivity analysis is performed. This allows us to evaluate how changes in input parameters or underlying assumptions influence the model's performance, providing insights into its robustness and reliability. The parameters that influence task assignment the most are the number of available GSE vehicles. Currently, the number of GSE vehicles is based on discussions with experts

and the study by Tabares et al. [3]. However, understanding the performance of the model with different numbers of GSE vehicles is valuable for the implementation of autonomous ground handling vehicles in the real world. Both traffic scenarios from the task allocation experiment are used to assess the performance of the model with different numbers of vehicles.

To find the minimum number of GSE vehicles, Constraints 7 and 8 of the PDPTW are adapted to allow the vehicles to remain at the depot and not be used. Not using a vehicle results in a lower overall makespan, and therefore a minimization of the number of vehicles is initiated. The number of vehicles is increased by one for each type of GSE vehicle in the second experiment. An overview of the number of vehicles is presented in Table 15. Minimizing the number of GSE vehicles resulted in a total of 13 GSE vehicles for all vehicle types. The tasks of filling the water tank and emptying the waste tank can be performed with a single vehicle for both scenarios. The results of each experiment for both scenarios in terms of the makespan and the baseline objective function ($z = z_1 + z_2$) are presented below.

Table 15: Number of vehicles available for three aircraft stands in a four-hour planning window.

GSE	Minimum	Baseline	Maximum
RE	3	3	4
CA	4	4	5
BA	4	4	5
WA	1	2	3
WC	1	2	3
Total	13	15	20



(a) Makespan for different number of vehicles.

(b) Objective function for different number of vehicles.

Figure 10: Sensitivity Analysis of number of GSE vehicles for the normal and peak traffic scenario.

Figures 10(a) and 10(b) show similar trends with an increasing number of GSE vehicles for both traffic scenarios. The makespan decreases slightly with the minimal number of GSE vehicles by assigning tasks more efficiently to the watering and toilet service vehicles. The decrease in the objective function with minimal number of GSE vehicles is negligible due to the increase in the start times at which tasks are served. The makespan then decreases significantly for 20 GSE vehicles with a decrease of 27% and 20% for the normal and peak traffic scenario, respectively. Therefore, the increase in the number of GSE vehicles shows that shorter and more efficient trips are planned. Interestingly, this results in higher start times for tasks, therefore increasing the value of the objective function.

7 Discussion

We can conclude from this study that the use of a global optimization method for task allocation of ground handling tasks improves the performance of the multi-agent model. The performance of the model is measured using several experiments that focus on the scalability, robustness, and quality of the model. The first experiment compared the makespan and computational time of two traffic scenarios, from which it could be concluded that the increase in the number of tasks does not lead to an increase in makespan, therefore rejecting

Hypothesis 2.1. The additional tasks present in the peak traffic scenario can be easily inserted into vehicle routes, thus limiting the increase in makespan. This statement is supported by the decrease in average total waiting times of 25% for the peak traffic scenario. The average makespan of each scenario is 213.62 and 202.65 minutes, respectively. Comparing these values with CES shows an increase of 39% and a decrease of 5% in the makespan for each traffic scenario. The increase in makespan can be explained by the differences between our traffic scenarios. This study includes additional turnaround tasks compared to CES. Baggage handling is divided into forward and aft baggage handling, and catering tasks are performed on the front and aft of the aircraft, doubling the number of tasks. Furthermore, the duration of tasks is significantly longer in this study, with an average increase of 40%. Finally, this study uses a total of 15 vehicles compared to the 12 vehicles used in CES. These factors could therefore explain the increase in makespan in the normal traffic scenario, however this effect is not apparent for the peak traffic scenario. This decrease is caused by the use of global optimization compared to a distributed auction method. In our experiment, we decrease the makespan by adding more tasks, thus enhancing the efficiency and utilization of the vehicles.

The Wilcoxon signed-rank test resulted in a non-significant p-value, suggesting that there is no statistically significant difference in the makespan between the two scenarios. However, the A-value from the Vargha-Delaney test, with a value of 0.60, reveals a moderate effect size. It implies that while the number of vehicles remains constant, the changes in task complexity, represented by the additional flights, do impact the overall performance of our task allocation model. Looking at the specific A values of the GSE types, Table 9, it can be seen that this effect size is only present for catering and baggage handling. This indicates that these tasks are the most tightly planned and have little flexibility with an increasing number of tasks.

In addition to comparing the makespan, we performed a similar analysis of the computational time required for both scenarios. The computational time increases 20% with the peak traffic scenario, and therefore we accept Hypothesis 2.2. Comparison of computational times with CES shows a decrease of 48% and 71% for the normal and peak traffic scenarios, respectively. The statistical tests yielded a non-significant p-value, which, similar to the makespan, suggests that there is no statistically significant difference in computational time between the two scenarios. The A value of the Vargha-Delaney test showed a significant value of 0.72, indicating a large effect size. Although the p-value did not identify a significant difference, the effect size suggests that there could be a substantial and meaningful difference in the computational time required for the two scenarios. It is therefore important to further investigate the implications of the large statistical significance of increasing the number of flights. Consequently, more research is needed to evaluate the scalability of the model to determine the practical use of the task assignment model.

An additional objective function is introduced to improve compliance with airport regulations for ground handling vehicles. The average makespan and average waiting time are compared for both objectives. The increase in average makespan for the additional objective is negligible, thus rejecting Hypothesis 2.3. Including the minimization of waiting times at the aircraft stand does not worsen the minimization of the makespan, it changes the original solution by shifting waiting times to the depot tasks. This can be concluded from the 98% decrease in waiting time at the aircraft stands compared to the solution with the original objective. This additional objective shows promising results for generating more realistic task assignments that comply with airport regulations. Gurobi is able to handle this non-linear objective, but further research into possible linearization of this objective should be performed to decrease the complexity of the model. In addition, depot tasks are used as parking locations to limit adaptations between the two models. However, creating parking locations at the aircraft stand where vehicles can wait, would improve the accuracy of the model representation. These parking locations should be accessible to all vehicles, and the duration for the vehicles to stay should be unlimited. Implementing these locations causes the optimization model to no longer fit the category of pickup and delivery problems. Further research on optimization problems including parking locations can assess the possibility of implementing parking locations in the current model.

Path planning experiments are also performed to assess the ability to properly execute the created schedules of vehicles. In this research, the same model for path planning as CES is used, and only parking locations are added for baggage tractors and fuel hydrant trucks. Therefore, the hypotheses were formed under the assumption that the outcome of our experiments would be similar. The results showed that the solution for prioritized SIPP is identical to the shortest path solution for the normal traffic scenario. Thus, Hypothesis 3.1 is accepted and Hypothesis 3.2 is rejected. The heat map, Figure 9, shows that vehicles only use the shortest paths and do not use detours, which is also observed by CES. However, CES found an increase in path duration of 28% due to the resolution of conflicts by waiting. The absence of an increase in the duration of the route in our model indicates that there are no conflicts during the planning period. The results for the peak traffic scenario indicate that the behavior of the model changes with increasing number of tasks. The duration of the route increases significantly with 20.3%, while the length of the route increases only with 0.03%. This increase

indicates that vehicles still prefer to wait to resolve conflicts. In addition, baggage handling tasks seem to arrive early and therefore wait 1 cell next to the service location until the earliest starting time is reached. However, this is observed to a lesser extent due to the introduction of parking cells. All tasks are performed in time with a success rate of 100%, and the routes are created in less than 5 seconds. The peak traffic scenario experiences average delays of 4.53 and 0.37 seconds per agent and task, respectively. These numbers are within acceptable ranges for real-world cases. Therefore, it can be concluded that prioritized planning in combination with SIPP is a suitable approach for path planning of ground handling vehicles. Improvements to the modeling environment should be made in further research to create a more realistic model of the airside of AAS. Further research into a more detailed environment could focus on accurate vehicle sizes and kinametics of vehicles, such as rotational motions, acceleration, and deceleration.

The robustness of the model is measured with replanning experiments. The quality of response to disturbances is measured for baggage handling tasks. The average vehicle makespan and the average start time of tasks are calculated for three scenarios. The three scenarios are routes without disruption, delayed routes without replanning, and replanned routes, respectively. The replanning model behaves as expected, with a negligible increase of 0.05% in task start times and an increase of 4% in makespan compared to the original model, so Hypotheses 4.1 is supported. However, Hypothesis 4.2 is rejected due to the small increase in the sum of start times. The increase in start times is minimized by the insertion strategy, but could not be avoided. Therefore, the model still behaves as expected, but the hypothesis is rejected. With a run time of 8.9E-4 s, replanning can be performed during real-time operations without further delay in operations. For this experiment, a single loading task was selected to simulate a disruption. Choosing the task is initially done randomly, however, many disrupted tasks do not cause any subsequent delays, therefore no replanning is activated. Therefore, the baggage vehicle with the lowest parking duration is chosen. Within the route of this vehicle a task is selected halfway through the planning window. The delay is modeled using a stochastic modeling approach because of the unavailability of real-time data. Therefore, multiple recommendations can be made to improve applicability in the real world. First, accurate delay probabilities for baggage handling tasks would improve the delay estimation of the disrupted task. Furthermore, the relationship between delayed baggage handling tasks and other types of tasks would allow modeling the response to delays for the entire model.

Varying the total number of GSE vehicles demonstrated the sensitivity of our model to the number of GSE vehicles. Minimization of GSE vehicles showed that two types of tasks could be performed with a single vehicle; watering and toilet service tasks. This can be explained by the wide time windows and relatively short duration of these tasks compared to the other tasks, see Figure 1(b). Using only 13 vehicles instead of 15 results in a decrease in makespan and a slight increase in the start times of the tasks, resulting in an objective value remaining constant. Increasing the total number of vehicles to 20 results in an increase in the objective value. This is counter intuitive as the makespan decreases. However, this increase in task start times is probably due to a limitation of computational time. Increasing the number of vehicles significantly increases the solution space, therefore creating a less bounded model. This causes the model to require longer computational times to obtain a high-quality solution. As autonomous vehicles have not yet been introduced into ground handling operations, this sensitivity analysis can help the decision-making process of choosing the total number of GSE vehicles.

8 Conclusion

This research has shown that the utilization of a global optimization technique for the assignment of ground handling tasks improves the computational time of the multi-agent task allocation model with 48% for normal traffic and has an allocation rate of 100% for all instances compared with CES. Near-optimal routes are generated using the schedule produced by the task allocation model. The non-linear objective function decreased waiting times on the aircraft stands with 98%. Experiments show that there are large waiting times between tasks, the model can therefore quickly reassign tasks if necessary without sacrificing the quality of the solution, making it a robust model. Finally, the total number of GSE vehicles is an important decision to make for airports, airlines, and other stakeholders, as it significantly influences the quality of the solution. In conclusion, our analysis has highlighted the promising potential of the proposed approach. However, more in-depth research is essential to fully assess its practical applicability in real-world scenarios. The optimization model could be adapted to comply with operational regulations. Research on the implementation of more accurate, data-driven delay and task duration estimation could further improve practical applicability. Additionally, a more detailed environment must be created to assess autonomous movement in real-time on the airside of an airport. Furthermore, the kinematics of autonomous vehicles could be implemented to more accurately represent the movement of vehicles. While this multi-agent model provides a solid foundation for autonomous ground handling, it also marks the beginning of a promising journey towards its full realization.

References

[1] Royal Schiphol Group, *Vision 2050; storyline*, Amsterdam Airport Schiphol, Schiphol, The Netherlands, 2020.

[2] Schiphol Innovation, *An autonomous airport in 2050*, 2021. [Online]. Available: https://www.schiphol.nl/en/innovation/blog/an-autonomous-airport-in-2050/ (visited on 11/24/2022).

[3] D. A. Tabares and F. Mora-Camino, "Aircraft ground handling: Analysis for automation," in *17th AIAA Aviation Technology, Integration, and Operations Conference, 2017*, 2017. DOI: 10.2514/6.2017-3425.

[4] Schiphol News Room, *Schiphol tests self-driving baggage tractor*, 2021. [Online]. Available: https://news.schiphol.com/schiphol-tests-self-driving-baggage-tractor/? (visited on 10/09/2023).

[5] J. Li, A. Tinka, S. Kiesel, J. W. Durham, T. K. Satish Kumar, and S. Koenig, "Lifelong multi-agent path finding in large-scale warehouses," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, vol. 2020-May, 2020.

[6] Z. Chen, J. Alonso-Mora, X. Bai, D. D. Harabor, and P. J. Stuckey, "Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, 2021, ISSN: 23773766. DOI: 10.1109/LRA.2021.3074883.

[7] H. Ma and S. Koenig, "Optimal target assignment and path finding for teams of agents," in *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS*, 2016.

[8] M. Liu, H. Ma, J. Li, and S. Koenig, "Task and Path Planning for Multi-Agent Pickup and Delivery," in *18th International Conference on Autonomous Agents and Multiagent Systems*, 2019. [Online]. Available: www.ifaamas.org.

[9] S. T. Chen, G. Ermis, and A. Sharpanskykh, "Multi-agent planning and coordination for automated aircraft ground handling," *Robotics and Autonomous Systems*, vol. 167, Sep. 2023, ISSN: 09218890. DOI: 10.1016/j.robot.2023.104480.

[10] S. Padrón, D. Guimarans, J. J. Ramos, and S. Fitouri-Trabelsi, "A bi-objective approach for scheduling ground-handling vehicles in airports," *Computers and Operations Research*, vol. 71, 2016, ISSN: 03050548. DOI: 10.1016/j.cor.2015.12.010.

[11] S. N. Parragh, K. F. Doerner, and R. F. Hartl, "A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations," *Journal fur Betriebswirtschaft*, vol. 58, no. 2, 2008, ISSN: 03449327. DOI: 10.1007/s11301-008-0036-4.

[12] M. M. Solomon, "Algorithms for the vehicle routing and scheudling problems with time windows.," *Operations Research*, vol. 35, no. 2, pp. 254–265, 1987, ISSN: 0030364X. DOI: 10.1287/opre.35.2.254.

[13] Boeing Commercial Airplanes, "737 max airplane characteristics for airport planning," Boeing, Tech. Rep., 2022, Document number: D638A004, pp. 100–115.

[14] HSE Office, *Safety and security pocketguide*, Amsterdam Airport Schiphol, Schiphol, The Netherlands, 2022.

[15] M. Phillips and M. Likhachev, "SIPP: Safe interval path planning for dynamic environments," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2011. DOI: 10.1109/ICRA.2011.5980306.

[16] H. Ma, W. Hönig, T. K. Satish Kumar, N. Ayanian, and S. Koenig, "Lifelong path planning with kinematic constraints for multi-agent pickup and delivery," in *33rd AAAI Conference on Artificial Intelligence, AAAI 2019*, 2019.

[17] H. Li and A. Lim, "A Metaheuristic for the Pickup and Delivery Problem with Time Windows," in *Proceedings 13th IEEE International Conference on Tools with Artificial Intelligence.*, 2001, pp. 160–167.

II

Literature Study
previously graded under AE4020

1 Introduction

Airports in Europe were confronted with the immense shortage of ground services staff spring 2022, resulting in long security lines, delayed flights and lost luggage. This problem occurred due to layoffs at airports during the COVID-19 pandemic. However, this shortage was already predicted by unions due to the relatively low salary and heavy work conditions for ground personnel[70]. Amsterdam Airport Schiphol(AAS) is one of these airports that had struggles with passenger amounts returning to pre-COVID-19 numbers, resulting in the cancellation of a large amount of flights. Research has shown that passenger amounts will return to pre-COVID-19 in 2024, so resolving these problems is currently top priority for AAS[33]. Airports such as AAS are now fast tracking innovations to make ground handling autonomous, requiring less personnel.

Additionally, airports around the world need to decrease their emissions drastically in the coming years in order to achieve sustainable aviation. Amsterdam Airport Schiphol, which is part of Royal Schiphol Group has announced that they strive to become the most sustainable airport of the world by 2050[65]. The first step announced by Schiphol Group is that by 2030 all the vehicles on airside should be running on electricity or hydrogen; emission free. Most of these ground support equipment(GSE) vehicles currently run on diesel and have not changed in design since the introduction of the vehicles in the 60s and 70s[26]. A large amount of GSE vehicles are now also available in electric variants but radical innovation is needed in order to achieve all the proposed goals. Becoming autonomous, improving safety and efficiency is also part of the plans for 2050 presented by Royal Schiphol Group. To achieve this, utilizing self-propelled, emission-free autonomous vehicles on the airside is required.

This study focuses on the airside operations that are related to ground handling of aircraft between flights. Currently, personnel are scheduled to ground handling tasks and they need to find a suitable GSE vehicle to execute the task, however this needs to change when ground handling becomes autonomous. In order to create a complete autonomous system, it is needed to design a framework which creates schedules for each GSE vehicle and safe, collision-free paths to and from each task. The aim of this literature study is to create a research proposal for designing a safe and efficient autonomous ground handling at Amsterdam Airport Schiphol that includes task allocation and path planning. Previous work by Chen et al.[13] and Padron et al. [56, 57] showed that it is possible to develop a framework for autonomous aircraft ground handling. This research will focus on extending the work of Chen et al. by coupling task allocation and path planning.

To be able to write a research proposal, it is necessary to start with researching the current operations at AAS in detail. Chapter 2 describes the conventional ground handling operations at an aircraft ramp as well as the current infrastructure and ground handling approach of Amsterdam Airport Schiphol(AAS). Furthermore, the research regarding automated ground handling operations will be discussed. From this chapter are requirements deducted which need to be satisfied by the task allocation and path planning approaches, which analyzed in chapter 3 and chapter 4 respectively. There are a lot of state-of-the-art task allocation and path planning algorithms and to determine which approaches fit the problem best, chapter 3 first starts with finding the suitable type of task allocation algorithm. These algorithms are categorized in centralized and decentralized approaches. From each approach is one variation further analyzed and the chapter is concluded with a decision for a task allocation approach. With the task

allocation approach known, the next step is to determine the path finding algorithm. Chapter 4 starts with an extensive definition of multi-agent path finding followed by a overview of all the types of path finding algorithms. Each type of algorithm is discussed in detail and the chapter is concluded with a trade-off of suitable path finding algorithms. In order to increase the efficiency and optimality of existing task allocation and path finding combinations, research proposed to couple the two algorithms. The current frameworks incorporating this strategy are discussed in chapter 5. A research proposal for the remainder of this thesis is written with the gathered knowledge of the previous chapters and this proposal together with the methodology and research planning are presented in chapter 6.

2

Aircraft Ground Handling

Aircraft ground handling entails a wide range of services that are provided to an aircraft when parked at an aircraft stand between their arrival and next departure. An aircraft stand is a parking space in which passengers can deplane and board, and ground services are executed. The goal of these services is to prepare the aircraft for the next flight leg within the allocated time. In order to simplify and speed up this process are Ground Support Equipment(GSE) vehicles developed. The development of GSE vehicles dates back to the beginning of the 20^{th} century when the Duat Tow Tractor was build which pulled freight, industrial materials and later on also baggage at airports. The development of specified GSE vehicles significantly increased during the World War II as flying became more common. Around the 1960s, GSE vehicles were adapted from military to daily use and were widely used in commercial aviation[26]. Since then, GSE vehicles have not changed apart from the transition of fossil fuels to electric vehicles.
This study considers solely ground handling services that are performed at a parked aircraft in a stand at Amsterdam Airport Schiphol(AAS). In section 2.1 are the different ground services and the required equipment introduced and analyzed. Section 2.2 discusses the infrastructure of Amsterdam Airport Schiphol. Current innovations that are implemented at AAS are mentioned in section 2.3 and research that focuses on further innovations and the transition towards autonomous GSE vehicles can be found in section 2.4.

2.1. Overview of aircraft ground handling operations

Ground handling services can be categorized into two categories, namely the cabin and ramp activities. Cabin activities refer to the services that are performed inside of the aircraft, such as the boarding and deplaning of passengers and catering galleys. The services that are performed on the outside of the aircraft such as refueling and cargo handling are part of the ramp activities. A complete overview of the performed activities can be seen in Figure 2.2. This figure not only shows the services performed, but also the order in which they are performed.

Figure 2.1: Aircraft stand Schiphol Airport. Retrieved from [35].

Ground handling procedures starts when the aircraft approaches the aircraft stand [4]. From there, the aircraft is marshalled into the docking location by airport personnel or using a digital docking system to ensure that the aircraft is positioned along the indicated markings that correspond with the type of aircraft. These markings ensure that the aircraft have sufficient separation between aircraft stands. An example of these markings can be found in Figure 2.1. When the aircraft is parked in the proper position, engines are shut off, beacon lights are turned off and the parking brakes(chocks) are installed. From this point onward, the ramp and cabin handling from Figure 2.2 can be initiated. This starts with connecting the Ground Power Unit(GPU), the Pre-Conditioned Air(PCA) unit and the aircraft access such as a Passenger Boarding Bridge(PBB) or mobile stairs. Once connected, passenger deplaning and cargo unloading is executed. Following this are the various cabin and ramp activities executed such as cleaning and refueling which prepares the aircraft to depart again. The next step is to load cargo into the compartments and let passengers board the aircraft. When all the previously mentioned tasks are completed, it is time to decouple all the ground support equipment. Finally, the aircraft will start one engine using an Air Start Unit(ASU) and the pilot will give permission to disconnect the GPU, PCA and parking brake which makes it possible to initiate pushback.

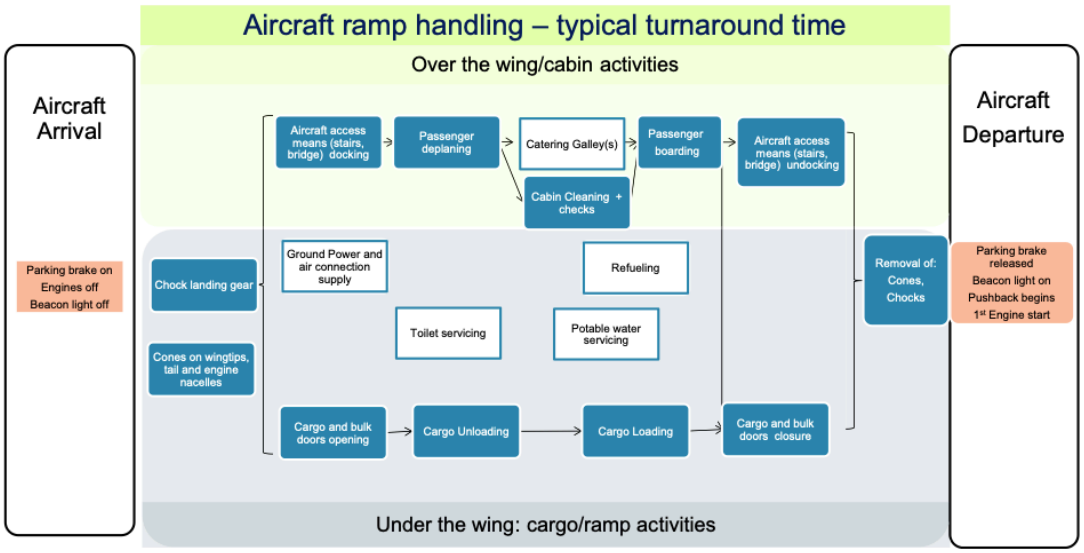


Figure 2.2: Task sequence of a typical turnaround time. Retrieved from [84].

In some cases, not all aforementioned tasks need to be performed. Short haul flights do not always burn up all their fuel which gives airlines an choice where and when to refuel. This is also the case for other tasks such as emptying the waste tank. The choice is mainly dependent on the available infrastructure, the flight schedule and the status of the tanks within the aircraft. All optional tasks are colored in white in Figure 2.2.

For each of the tasks that need to be performed when servicing an aircraft is special equipment developed. All of these equipment together is called Ground Support Equipment(GSE). In Figure 2.3 is the servicing arrangement of a Boeing 737-800 visualized. There are in total nine different GSE vehicles illustrated in the figure, however only six GSE vehicles will be discussed in more detail. The GPU, PCA and ASU are not always required due to the fact that the Auxiliary Power Unit(APU) inside of the aircraft is often used instead.

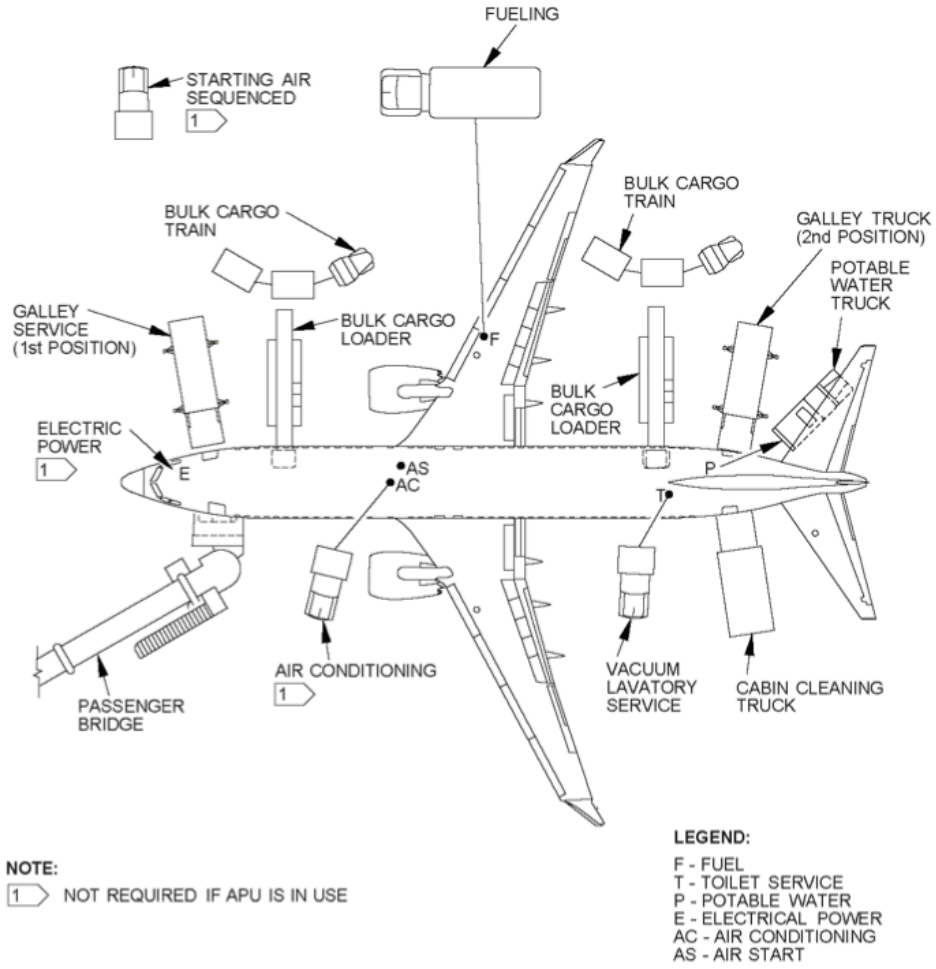


Figure 2.3: Ground Support Equipment Servicing Locations for a Boeing 737-800. Retrieved from [10].

- **Galley service:** this vehicle collects meal trolleys from the previous flight and replaces them with new meal trolleys for the departing flight.
- **Bulk cargo loader and train:** this systems exists of three separate parts. First there is the train, which consists of a tow tractor and baggage carts. Once the train has arrived at the ramp, the loader is utilized. The loader is a mobile conveyor belt which is used to move cargo from the train to the cargo hold of the aircraft. In the cargo hold is the baggage carefully placed and secured by personnel. In long range aircraft are Unit Load Devices(ULD) used. The ULD is already filled with cargo beforehand and is then driven to the aircraft and placed in the cargo hold, making it much more time and space efficient.
- **Fueling truck:** there are two different methods of fueling an aircraft and it depend on the infrastructure of the airport which of them is used. If an airport has an underground network of fuel lines, the fuel hydrant dispenser is used. This is a small truck connecting the fuel line of the airport with the fuel tanks of the aircraft. When this network is not present, a refueling truck is used.
- **Portable water truck:** a truck containing a tank filled with water and connects to a valve on the outside of the aircraft in order to fill the water tank within the aircraft.
- **Cabin cleaning truck:** this truck is very similar to the galley service truck, only they serve a different purpose. This truck assists the cleaning crew by delivering cleaning supplies into the cabin.
- **Vacuum lavatory service:** the toilet service is performed with a truck carrying an empty tank to which a hose is connected. The other end of the hose is connected to the aircraft to empty the waste tank.

During operational hours, multiple aircraft stands are in need of GSE vehicles, this is why it is desired to assess how many GSE vehicles of each type are needed at one stand to have smooth ramp operations with minimal delays. With the data provided in [83] and Figure 2.3, the amount of GSE vehicles is determined together with the information of how they are shared within an airport in Table 2.1. In addition to the amount and sharing strategy, is also priority of each GSE vehicle determined. This priority is utilized to resolve a occurrence of a conflict when moving between aircraft stands.

Table 2.1: GSE vehicle resources for a short range aircraft ramp activities and GSE vehicle sharing strategy including priority. Adapted from [83] and internal documents.

Ground Support Equipment(GSE)	Amount	Shared Resources Strategy	Priority
Passenger Boarding Bridge(PBB)	1	Specific to aircraft stand	2
Belt loader	2	Specific to aircraft stand	5
Catering truck	1 - 2	Shared with the whole airport	3
Cleaning truck	0 - 1	Shared with the whole airport	4
Fueling vehicle	1	Shared with the whole airport	1
Portable water truck	1	Shared with the whole airport	6
Toilet servicing truck	1	Shared with the whole airport	7
Baggage tractor	2	Shared with the whole airport	8
Baggage cart	6 - 8	Shared with the whole airport	9

Since fueling is not only essential but can also be dangerous, it has the highest priority. Safety regulations state that there should always be a refuel equipment hazard zone starting from the fuel tank valve until the fuel access point. In this zone are no vehicle other allowed. Also a free escape corridor has to be present to make sure the vehicle can move out of the way in case of an emergency. The determination of priority does not only depend on safety but also on time management as airlines desire to minimize the time on the ground to maximize profit.

2.1.1. Turnaround Time

In order to achieve minimal ground time, it is needed to plan all the necessary tasks in a manner which is fast, but also in line with the existing regulations. The time span in which the aircraft is on-chocks is called the TurnAround Time(TAT). The time spent on each task and the TAT is estimated by the aircraft manufacturer, in this case Boeing. This estimation of the TAT is published as a part of the 'Airplane Characteristics for Airport Planning' document[10]. From this document is the estimated TAT of a Boeing 737-800 retrieved and presented in Figure 2.4.

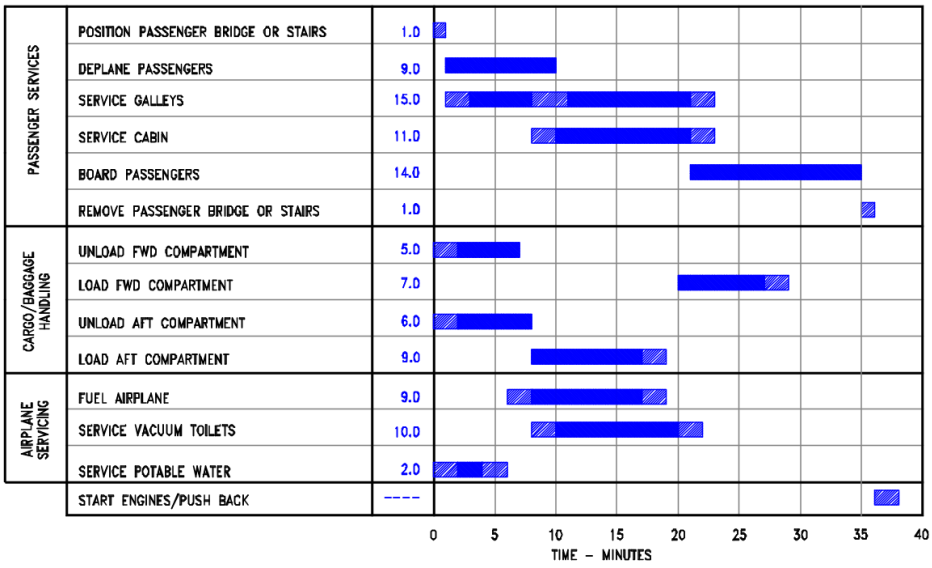


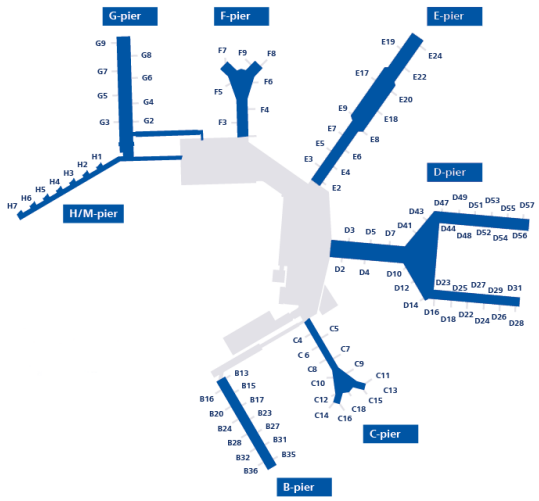
Figure 2.4: Time Schedule of the TAT of a Boeing 737-800. Retrieved from [10].

As can be seen from Figure 2.4, the passenger services determine the TAT and are called the critical path. Minimizing this critical path has a high priority, which is why the catering and cleaning truck have priority over the cargo handling and servicing vehicles(see Table 2.1). Another interesting thing to mention is that in Figure 2.3 it can be seen that the toilet service connection and the water connection are relatively close to each other. This results in hygiene regulations which prohibit servicing the toilets simultaneously with the portable water. Furthermore, the aft compartment is fully loaded before the forward compartment is loaded. To ensure that the center of gravity remains within acceptable range, it is required to load the aircraft in this order.

However, even the exact same aircraft type can vary significantly in turnaround time. The TAT is dependent on different factors such as the layout of the cabin, the resources available at the airport and airline operations. Not only the duration of a task but also the schedule within the TAT is not constant and each airline has their own strategy. In addition to this are airlines also forced to contract different providers of ground handling tasks for different type of tasks at each airport in their network. This is due to the fact that most ground handling providers provide a single task in which they are specialized.

2.2. Infrastructure Amsterdam Airport Schiphol

As previously mentioned, Amsterdam Airport Schiphol(AAS) is the airport that will be used as the environment for this study. In order to correctly model this environment, it is required to describe the current infrastructure of AAS. Schiphol has seven piers with 91 (semi-)connected gates as seen in Figure 2.5a. This means that passengers can board the aircraft using a PBB or walk to the mobile stairs[1]. There are also parking spaces for aircraft to reside to when the aircraft stand is needed for another aircraft.



(a) Pier layout with gate numbers. Retrieved from [67].

(b) Aerial view of pier B. Retrieved from [15]

Figure 2.5: Infrastructure of Amsterdam Airport Schiphol

For the remaining part of this study is only pier B considered, which is a linear pier with 13 short haul aircraft platforms, see Figure 2.5b. Each of the aircraft platforms that correspond to the gate can be accessed by GSE vehicles using the one perimeter road which runs around the pier. A clear layout of the aircraft stand at AAS is visualized in Figure 2.6. This figure shows the area in which GSE vehicles can enter and exit the platform(8), as well as the region in which they are allowed to wait, to park and, to maneuver called the Equipment Staging Area(ESA), Equipment Parking Area(EPA) and, Equipment Restraint Area(ERA) respectively. When GSE vehicles are not in use at the platform or when the task has been completed, the GSE vehicles can be parked in the designated EPA(3,7,9). Within this

environment are speed limits enforced to ensure that operations can be performed safely. The speed limit on the perimeter road is 30 km/h and for the platform and baggage area the limit is 15 km/h[30].

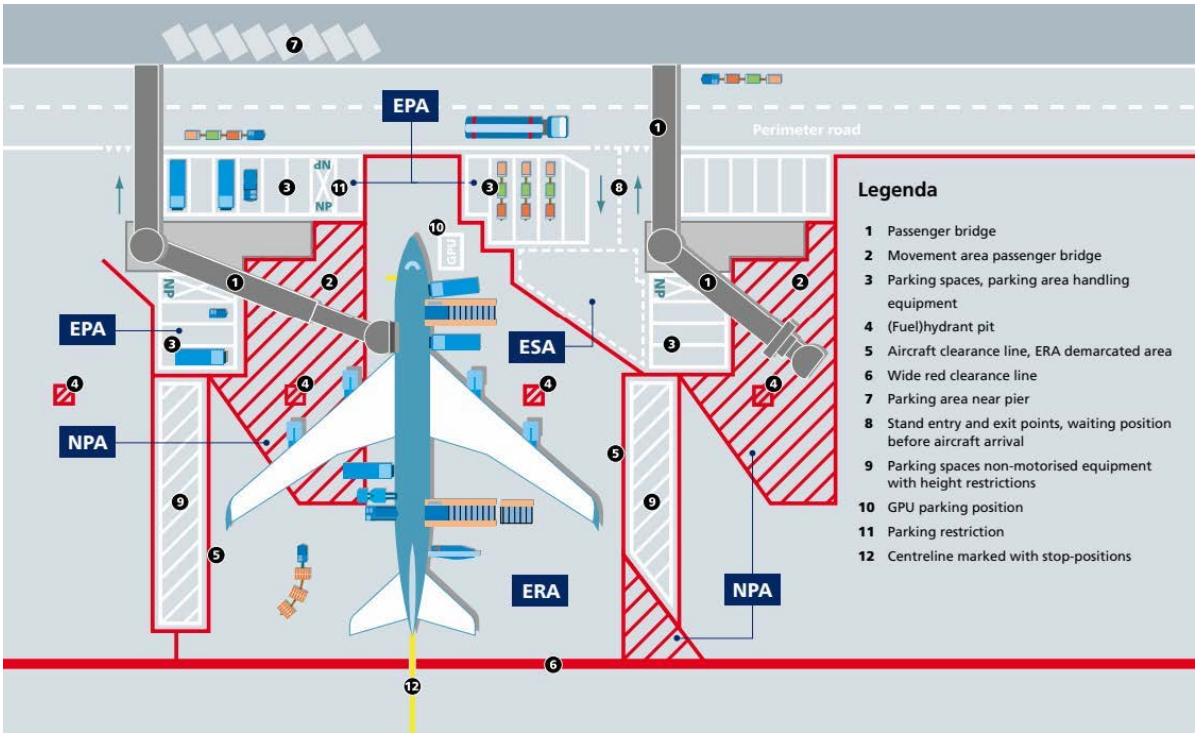


Figure 2.6: Layout of an aircraft stand at AAS with markings. Retrieved from [31].

2.3. Current approaches on aircraft ground handling

At Schiphol there are many providers that perform ground handling tasks. Some providers are able to perform all necessary activities, but most specialize in a selection of the tasks due to the different type of GSE vehicles needed for each task. Each airline is responsible for contracting their own service provider at an airport which can lead to a having a lot of providers present at an airport such as AAS. For example, KLM has their own ground services department at Schiphol, however, they do have to contract other providers at their destination airports.

The scheduling of the usage of GSE vehicles around an airport is done by scheduling personnel that drive the GSE vehicles. The personnel is assigned to perform a task on an aircraft at a specific time and location. This makes the personnel responsible for finding the correct GSE vehicle and traveling to the platform while still arriving on time. If a GSE vehicle is not available due to for example delays of other aircraft, the task cannot be performed on time delaying all following aircraft in the schedule as well. This makes the efficiency of an airline very dependent on the availability of GSE vehicles and the punctuality of the personnel.

Since 2018, AAS has implemented the concept Airport Collaborative Decision Making(A-CDM)[20]. This concept was originally developed by Eurocontrol with the goal to improve efficiency and resilience of airport operations. This is achieved by more transparent communication between all stakeholders such as airlines and airport operators. By knowing more accurately when each GSE vehicle is required to be at the specific platform makes it possible to schedule the GSE vehicles more efficient and resulting in less delays due to delayed or unavailable GSE vehicles. In the years that this concept has been carried out, accurate data on the turnaround time has been gathered which can be used as a benchmark to evaluate the performance of the model for this study.

2.4. Research on aircraft ground handling

Optimizing aircraft ground handling is something that airlines have been working on for years to enable growth of the amount of flight executed per day. Global air traffic has increased on average over 7% in the period of 2014-2019[32] and this growth is expected to continue in 2024 when the traffic levels have returned to the pre-COVID-19 levels[33]. This makes the bottlenecks which were present in 2019 and the research regarding this still relevant. Research mainly focuses on utilizing current infrastructure more efficient, such as taxiways, runways and ramps. As mentioned before, a delay in one of the tasks that are part of the TAT, delays the complete aircraft and also the other aircraft within the schedule. This is why most of the research focuses on increasing accuracy of the estimation of the TAT[73] as well as estimating the amount of each type of GSE vehicle needed in different scenarios of traffic[68]. By improving these estimations, GSE vehicle tasks can be planned with higher accuracy making it possible to minimize delays of aircraft.

Other solutions regarding the turnaround time have been studied such as utilizing a booking system in which airlines can book GSE vehicles[62] upfront and creating mathematical optimization models with the goal of optimizing GSE scheduling[57][27]. Also, new concepts of GSE vehicles are researched which would enable different type of tasks to be performed by one GSE vehicle. This would decrease the amount of traffic around an aircraft platform which then decreases the probability of delays caused by unavailability of GSE vehicles[69]. Finally, research has been focusing recently on the transitioning from personnel handled to autonomous ground support equipment [52]. Due to airports having difficulties with sustaining growth and with hiring and reducing workload of personnel, autonomous vehicles have gained interest with all stakeholders. To determine if switching to autonomous vehicles is a good investment, the opportunities and challenges will be discussed.

2.4.1. Opportunities for automated aircraft ground handling

The rise of autonomous vehicles has been present for the last couple of years in order improve efficiency, sustain growth and reducing human workload within operations of different fields. Technology needed for autonomous vehicles is widely available, affordable and properly tested for environments similar to an aircraft platform[83]. Airlines suffer more and more from a shortage in personnel, stricter rules on heath and safety for personnel and difficulties with accommodating the growth in flights. In addition to this, the way aircraft handling is performed as described in section 2.1 has not significantly changed since it was introduced over 50 years ago but costs of autonomous versus manned vehicles has.

Furthermore, aircraft have changed in the past five decades and became more and more similar which is favorable for automation. First of all, aircraft doors and couplings for GSE vehicles have been standardized, making it possible to perform the same tasks on different aircraft type with the same GSE vehicle. Commercial aircraft also have very similar servicing point locations as the shape of aircraft do not vary significantly. In Figure 2.7 are the locations of portable water servicing and passenger doors plotted on relative halfspan and length of an aircraft. From this figure can be concluded that over the years the servicing locations converged making it possible to determine an accurate servicing area for each turnaround task.

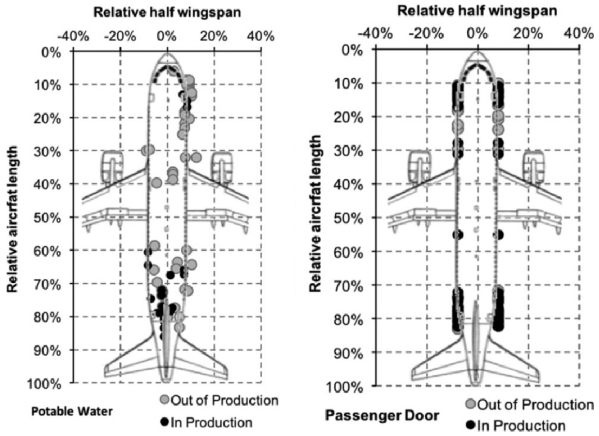


Figure 2.7: Servicing locations for short-to-medium haul aircraft. Retrieved from [68].

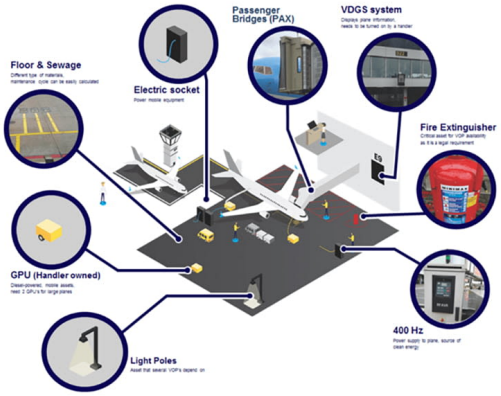


Figure 2.8: The aspects of an aircraft stand that are monitored by Smart VOP 1.0. Retrieved from [64].

Finally, within the airport a transition to automation has already been initiated like the self service check-in and baggage drop off inside the terminal. Next to this, AAS has also invested in state-of-the-art CT scanners used for security which run on wind energy[66], developed a tool which predicts the waiting times at security and advises travelers when to go through security[63] and implemented a data-driven monitoring system to measure the performance of an aircraft stand called Smart VOP 1.0, see Figure 2.8[64].

Future outlook

The opportunities show that automation within aircraft ground handling is the next step. Amsterdam Airport Schiphol has the ambitious goal to have fully automated ground handling in 2050 which are interconnected, self-propelled and emission free.[65]. In order to achieve this, it is needed to map out the developments that need to take place between now and 2050. Tabares et al. propose short, mid and long term changes which enable an autonomous fleet[83].

On short term, targets can be designed which would guide autonomous vehicles in order to find the correct location for GSE vehicle docking. The next step is to robotize aircraft systems such as opening of doors and panels, automatic connection systems and automated cargo loading systems in aircraft which makes it possible to perform turnaround tasks without human interaction. Finally, it is needed to change processes within the aircraft in order to ease automation. An example is the catering service, which is now performed by swapping individual trolleys and could be simplified by swapping the complete galley. Next to this, the traffic rules for the infrastructure at AAS needs to be adapted to autonomous vehicles. Specific predetermined routes could be designed in order to simplify the path planning, as well as have uniform rules on speed and right of way.

2.4.2. Challenges for automated aircraft ground handling

Transitioning to autonomous vehicles for ground operations has become an interesting investment time and money wise for airlines and airports. However there are challenges in fully automating the aircraft ground operations. According to Tabares et al.[83] can the challenges be divided into three categories; aircraft related, operational and scientific.

Aircraft related challenges

A difficulty that arises within aviation is the duration and strict rules of aircraft certification. Servicing doors that can be controlled remotely is technically feasible but has not been certified yet, making it impossible to execute TAT tasks without human interaction. Fortunately, for most tasks within the TAT are the servicing doors the only aspect that require human interaction with the exception of baggage handling in aircraft that cannot fit ULDs in their cargo hold such as the B737.

Operational challenges

Airport operations has many stakeholders with different interests and priorities and is also a very conservative industry. This makes it difficult to convince each stakeholder to invest in the transition to autonomous vehicles as it will not align with the priorities of all stakeholders. Furthermore, there is not a proper insight into the actual costs of this transition, making it difficult for for example GSE vehicle manufacturers to determine if they can afford to develop autonomous GSE vehicles. Next to this is it impossible to switch from manual to fully autonomous without a transition period. In this transition period, a hybrid between the two scenarios is used and for this hybrid period are no concrete plans present yet. Also government rules regarding autonomous vehicles at airports are not developed yet but could possibly cause difficulties.

Scientific challenges

Instead of scheduling GSE vehicle personnel, it is required to assign tasks to the different types of GSE vehicles and the GSE vehicles should be able to drive between aircraft stands conflict-free. Development of an algorithm which tackles these two requirements is required in order to ensure efficiency and safety at an airport. Autonomous vehicles moving around need to be able to plan trajectories and solve any conflicts that might arise within their path. Next to this should it be possible to adapt to any real time changes which can for example be caused by delayed aircraft or delayed passengers. So the airport should be modeled as a hybrid dynamic system with real time control of a heterogeneous fleet of GSE vehicles. Another challenge is defining how the vehicles should response to incidents and disruptions and how they should communicate as well as defining how this automated system should be monitored.

2.5. Final remarks on aircraft ground handling

This study focuses on developing a solution for the previously mentioned scientific challenges. An algorithm that satisfies all the requirements mentioned in this chapter has yet to be developed, however research regarding automating airport ground services exists. One interesting study by W.H. Ip et al.[34] suggest building an Agent Based Model(ABM) to model the planning of aircraft ground services. Agent based modeling is a preferable modeling technique for modeling traffic because of the following reasons; ABM is able to capture emergent phenomena, it is able to describe a system in a natural manner and ABM is seen as a flexible, low cost and time efficient technique[7]. As both of these papers state that in situations very closely related to the environment of this study that ABM is suitable, it is decided that from this point onward an ABM will be developed. This model has to tackle two separate aspects; task assignment and path planning. In the following chapters are state-of-the-art task allocation and path planning algorithms analyzed in order to decide which algorithms are suitable for the remainder of this study. When researching algorithms, the following requirements derived from this chapter should be taken into account:

- Heterogeneous fleet
- Heterogeneous capacities
- More tasks than agents
- Priority for tasks/agents
- Release time of task
- Task execution time window
- Handle possible delays
- Handle changes in schedules
- Different environments; road and ramp
- In line with regulations

3

Task Allocation

From the analysis of aircraft ground handling in chapter 2 was concluded that there are many challenges that need to be tackled in order to successfully run autonomous ground operations at AAS. One of the scientific challenges is to change the method of planning ground handling tasks. Efficiently assigning tasks to agents is essential for optimizing resource utilization, reducing operational costs, improving system performance, and achieving overall objectives. Therefore this problem arises in diverse fields as well, including robotics, distributed computing, transportation logistics, and workforce management[18, 25].

For this research, the task allocation method has to assign each task to a GSE vehicle of the correct type and the vehicle should be able to execute the tasks in the corresponding time window in order to prevent delays. The first step to finding a suitable method is to identify the type of task allocation problem of this research. An overview of the existing types is presented in section 3.1. Over the years, the study of multi-agent task allocation has evolved significantly, driven by advances in computing power, communication technologies, and the growing need for intelligent and collaborative systems. Initially, task allocation problems were primarily addressed in a centralized manner, but recently research has shifted to decentralized approached. All the solution techniques that are suitable for the type of problem of this research are discussed in section 3.2.

One type of task allocation technique that seems to fit this problem well is the Vehicle Routing Problem (VRP). A VRP focuses on allocating tasks to a fleet of vehicles with limited capacities, aiming to minimize travel distances or time while adhering to various constraints. Therefore detailed research in section 3.3 presents a suitable version of a VRP called a Pick-up and Delivery Problem. However, previous research focussing on airport operations mainly use auction based techniques, such as Single Sequential Item auctions. Therefore this technique is discussed in more detail in section 3.4. Finally, in section 3.5 are both previously mentioned methods compared and a decision is made on which method will be used in the remainder of this research.

3.1. Taxonomy

Task allocation in a multi-agent system generates a task division for all agents to collaboratively achieve a predetermined goal. In this research, this is done by assigning tasks to agents making it an explicit cooperation problem also known as a multi-robot task allocation problem(MRTA). In the work of Gerkey and Matarić [23] MRTA problems are categorized based on three domains; robot type, task type and allocation type.

Single task robot (ST)	vs.	Multi-task robot (MT)
Single robot task (SR)	vs.	Multi-robot task (MR)
Instantaneous assignment (IA)	vs.	Time extended assignment (TA)

In the multi-task domain, the agents are capable of performing tasks simultaneously while only one task can be performed by a single task agent. As the name suggests, in the single agent task domain is a task completed by a single agents and multiple agents are required in the multi-agent task domain. The final division is based on the available information. In an instantaneous assignment domain is the information on tasks, agents and the environment not continuous making it impossible to include future

allocations. On the other hand exists the time extended assignment in which all information is readily available about current and future tasks. The allocation of GSE vehicles for aircraft ground handling is according to this taxonomy and the information found in chapter 2 a single-task, single robot, time extended assignment problem [ST-SR-TA].

However, Korsah, Stentz et al. states that the three domains are limited in scope. Newer task allocation problems that experience interrelations in utilities and constraints require a more detailed taxonomy. They propose a taxonomy called iTax which includes these interrelations and makes it possible to categorize the problem in more detail[38]. iTax is a two level taxonomy and the first level is newly developed and expresses the degree of interdependence and the four classes can be found in Figure 3.1. The second level is identical to the taxonomy of Gerkey and Matarić. Including the first level of iTax is relevant for ground handling activities as there are multiple different type of dependencies present within the tasks that need to be performed by GSE vehicles. An example is the baggage handling operations for which unloading the baggage of the arriving flight is required to be executed before loading baggage of a departing flight into the aircraft.

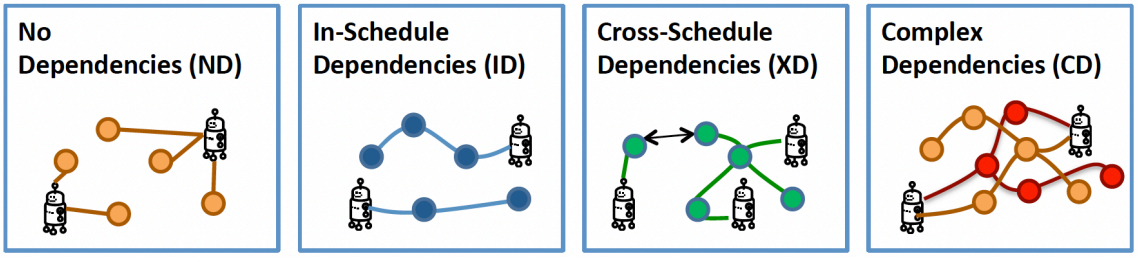


Figure 3.1: Examples illustrating the degree of interdependence of a task allocation problem. Circles represent tasks and solid lines are agent routes. Arrows between two tasks indicates a constraint. The routes in the rightmost square illustrate possible task decompositions. Retrieved from [38].

The task allocation problem of this study can be placed in the XD [ST-SR-TA] class. This class is characterized by agents not only being dependent on their own schedule but also on the schedules of other agents. One of the common cases of this class given by Korsah, Stentz et al. states that single-agent tasks have inter-task constraints like precedence and simultaneity. In this study, different type of tasks need to be allocated to one of the the same type of GSE vehicle. To make sure the allocation is possible, it is needed to check if the previous task is completed before starting a new task, that only one agent is performing the task and that the unloading task is performed before the loading task. So it can be concluded that the task allocation of GSE fits the class XD [ST-SR-TA]. Therefore, the remainder of the chapter will only focus on solution techniques for this class of task allocation problems.

3.2. Solution Techniques

In literature are an overwhelming amount of task allocation algorithms found. In order to give an overview of the different algorithms a categorization based on *control* is performed[36]. The term *control* refers to the degree in which the decision making is performed by one single entity or locally, centralized and decentralized respectively. In a centralized approach, a central entity such as a coordinator or a planner is responsible for allocating tasks to the agents in the system while considering factors such as the agents' capacity and requirements that are linked to the tasks. A decentralized MRTA approach is characterized by each agent being responsible for making its own allocation decisions based on local information, without the need for a central entity. Each agent communicates with its neighbors to exchange information about the tasks and agents and collaboratively decide on task allocations.

A fairly new category in multi-agent task allocation is the usage of machine learning which can be both centralized and decentralized. Reinforcement learning is the kind of machine learning mostly used in literature[53][61]. Within this learning process, agents learn to communicate with neighboring agents in order to create appropriate policies for allocating the tasks. However this research is mostly limited to homogeneous agents which makes it not yet interesting for TA of GSE vehicles.

The centralized approach can be useful in situations where there are a large number of agents and tasks to be allocated, or when there are complex dependencies between the tasks. However, this approach also has limitations, such as the possibility of bottlenecks and the potential for the central entity to become a single point of failure in the system. On the other hand, a decentralized approach can be more flexible and adaptable, and can operate even when communication links are unreliable or unavailable. But, a decentralized approach may suffer from the lack of global information and coordination, which can result in suboptimal task allocations. In order to decide which approach suits the case of ground operations best, both approaches will be further researched. This chapter will continue with elaborating on examples of centralized approaches followed by decentralized approaches in subsection 3.2.1 and subsection 3.2.2.

3.2.1. Centralized Approaches

There are many centralized methods developed in the last decades with each a different application or different environment. Most centralized approaches used for situations similar to task allocation of GSE vehicles are optimization based. Optimization based approaches aim to find the optimal solution for a given problem. The optimal solution is found for the predetermined objective function and within the restrictions set by constraints[5]. A well known optimization based algorithm is the Hungarian Method, first introduced by Kuhn[39] which uses linear-programming based matching. This method has been extensively researched and improved upon but is difficult to utilize in lifelong problems as well as difficult to generalize.

Another well known generalized optimization based approaches in literature is the vehicle routing problem(VRP). A vehicle routing problem is a combinatorial optimization problem that finds the most efficient way to route a fleet of vehicles to a set of destinations. The goal is to minimize the total distance traveled, time taken, or other relevant cost metric, while satisfying various constraints such as vehicle capacity, time windows, and customer demand. The problem is often encountered in transportation and logistics, where it is necessary to plan the optimal delivery routes for goods or services. VRP can take on many different forms, such as the capacitated VRP, the VRP with time windows, and the VRP with multiple depots. Due to the ability of including various constraints and other characteristics in a VRP, it is an interesting approach to consider for this research. Solving a VRP is however computationally challenging due to it being a NP-hard problem. Fortunately, a wide range of algorithms have been developed to find high quality solutions, including exact algorithms, heuristics, and metaheuristics. Exact solutions include branch-and-bound, Integer Linear Programming (ILP), Dynamic Programming(DP) and Constraint Programming(CP)[60]. These solutions guarantee optimality but remain computationally expensive and lack robustness which makes applying the model in dynamic environments difficult.

Heuristics and Metaheuritics

Heuristics can provide a fast and efficient way to approximate an optimal solution, especially for large or complex optimization problems. Heuristics are in VRPs usually used to estimate the travel distance for each agent to all tasks[77]. Examples are savings, nearest-neighbor, insertion and sweep heuristics. Metaheuristics are typically used when the problem of task allocation is complex and involves a large number of interdependent agents with conflicting objectives. Examples of metaheuristics commonly used in multi-agent task allocation include genetic algorithms, simulated annealing, and particle swarm optimization[51][89]. The combination of a (meta)heuristic in combination with a VRP shows to be a promising approach for the task allocation of autonomous GSE vehicles[2]. In section 3.3 are various VRP types presented as well as interesting heuristics to use.

3.2.2. Decentralized Approaches

The main advantage of decentralized approaches is the redundancy; if one robot unexpectedly fails, all other robots are still up and running. As a solution, robots can be easily added and removed from the set which makes this type of approach also scalable and flexible. There are multiple types of decentralized approaches for multi-robot task allocation found in literature, however the most suitable approaches all fall under market-based approaches. The advantages and disadvantages of a market-based approach are listed in Table 3.1.

Table 3.1: Advantages and disadvantages of a market-based approach. Based on information from [36].

Advantage	**Explanation**
Efficiency	Due to being partially centralized but mostly decentralized, it has a faster runtime than centralized approaches.
Robustness	Decentralized approaches do not have a single point of failure, as other robots still function when one fails.
Scalability	New robots can be easily added or removed.
Online use	Tasks that are generated during the simulation are easily added to the task allocation.
Uncertainty	Market-based approaches are able to work with dynamic and unknown environments.
Disadvantage	**Explanation**
Suboptimality	Optimality cannot be guaranteed, and highly-suboptimal solutions are often the result.
Communication	Market-based approaches might require a lot of communication, demanding high computational power.
Difficult to design	The design requirements are not easily captures in cost and revenue functions.

Market-Based Approaches

Market-based approaches are inspired by the principles of economics where the assignment of a task is determined by the willingness of a robot to execute it. The most commonly used market-based approach within multi-agent task allocation is auctions. Within an auction are a set of tasks or goods assigned to bidders based on the received bids and other auction rules. The first simulated auction by Smith dates back to 1980 which uses contracts to assign tasks to agents and is still the foundation of most auction frameworks[75]. This framework is called the Contract Net Protocol(CNP) and it works as follows. The auction exists of four phases; announcement, submission, contract and selection respectively. A schematic of the CNP can be found in Figure 3.2 below. In the first phase an auctioneer announces the task(s) that have become available for bidding. Next, each agent determines their bid based on their bidding strategy and communicate this bid to the auctioneer. When the auctioneer has received all bids, it will determine the winner based on a predetermined strategy. The winning agent is assigned to the task with a contract in the final phase. Afterwards, this process can be repeated for the remaining tasks.

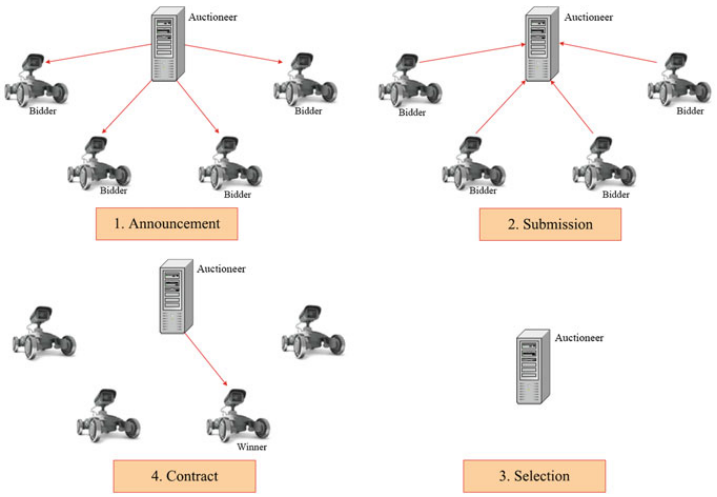


Figure 3.2: Steps of Contract Net Protocol algorithm. Retrieved from [36].

Task allocation of autonomous GSE should occur in a cooperative setting, meaning that only cooperative market-based approaches are considered. Agents share all their information with the auctioneer in order to achieve the best possible solution. In addition to this, a bid strategy for agents can be developed which helps with generating a fair bid. Each auction algorithm has varying bid and auctioneer strategy depending on the situation it is used in which makes it easy to convert existing frameworks to the autonomous GSE vehicle task allocation. Examples of possible auction mechanisms for this study are presented in section 3.4.

When there is no central auctioneer, each agent will assign tasks based on available information. When conflicts between the assignment of tasks occurs, a consensus is made using an predefined algorithm. This type of market-based approach is called the consensus-based auctions. Due to having no central auctioneer, not all agents have the same available information leading to solutions far from optimal. Situational awareness is in the situation of autonomous GSE vehicles paramount and therefore consensus-based auctions will not be considered in this research.

3.3. Vehicle Routing Problem

There are many form of vehicle routing problems, but for this research the VRP with Pickups and Deliveries (VRPPD) are most interesting to look into as these model goods being transported, as the name suggest, between pick-up and delivery locations. Different versions of this problem are elaborate upon in the second part of the survey of Parragh et al.[58] which are sub-categorized in two classes; unpaired and paired. As this research focuses on heterogeneous goods, paired pick-up and delivery locations are considered. Within this subclass are the classical Pickup and Delivery Problem (PDP) and the Dial-A-Ride Problem (DARP) and these problems focus on goods and people respectively. From this survey can be concluded that this research fits best with the PDP category.

Below, the notation and mathematical model are presented for a pickup and delivery problem with time windows. It can be assumed that the edges are symmetric; $t_{ij}^k = t_{ji}^k$ and $c_{ij}^k = c_{ji}^k$ which reduces the amount of variables required significantly. The model exists of a graph $G = (V, A)$ where V is the set of all vertices $V = \{0, n+\tilde{n}+1\} \cup P \cup D$, and A the set of all arcs; $A = \{(i,j) : i,j \in V, i \neq n+\tilde{n}+1, j \neq 0, i \neq j\}$.

n	number of pickup vertices
$\tilde{n}$	number of delivery vertices
K	set of vehicles
q_i	demand/supply at vertex i; pickup vertices are associated with a; positive value, delivery vertices with; a negative value; at the start depot 0; and the end depot $n+\tilde{n}+1$; the demand/supply is zero, $q_0 = q_{n+\tilde{n}+1} = 0$
P	set of pickup vertices, $P = 1, \ldots, n$
D	set of delivery vertices, $D = n+1, \ldots, n+\tilde{n}$
e_i	earliest time to begin service at vertex i
l_i	latest time to begin service at vertex i
d_i	service duration at vertex i
L_i	maximum ride time of user i
c_{ij}^k	cost to transverse edge (i,j) with vehicle k
t_{ij}^k	travel time from vertex $i \rightarrow j$ with vehicle k
C^k	capacity of vehicle k
T^k	maximum route duration of vehicle/route k

Within the mathematical model are three decision variables that are varied in order to find the optimal solution. The first decision variable saves all arcs taken by each vehicles, the second variable determines the load of the vehicle at each vertex. Finally, the third variable saves the time that each task is executed.

$$x_{ij}^k = \begin{cases} 1, & \text{if } \mathrm{arc}(i,j) \text{ is traversed by vehicle } k \\ 0, & \text{else} \end{cases}$$

Q_i^k	load of vehicle k when leaving vertex i
B_i^k	beginning of service of vehicle k vertex i

The mathematical model is initialized with the objective function (1) which minimizes the total routing cost. The remaining formulations are constraints set to the model. The first constraint (2) guarantees that each vertex is serviced only once and constraints (3) and (4) ensure that every vehicle starts at the depot and ends at the depot. To ensure the flow, constraint (5) is formulated. Constraint (6) is formu-

lated to ensure that time is properly taken into account. Constraint (7) and (8) defines the capacity limit for vehicles and ensures that at least the required load for the next vertex is loaded. Constraint (9) ensures that each arc is only visited once or never. Constraint (10) and (11) ensure that the pickup and delivery locations are served by the same vehicle and that delivery occurs after pickup respectively. Finally constraints (12) and (13) include time window and maximum route duration restrictions[58]. Note that constraints (6) and (7) are non-linear but can be linearized using the big M notation.

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij}^k x_{ij}^k \tag{1}$$

subject to:

$$\sum_{k \in K} \sum_{j:(i,j) \in A} x_{ij}^k = 1 \qquad \forall i \in P \cup D, \tag{2}$$

$$\sum_{j:(0,j) \in A} x_{0j}^k = 1 \qquad \forall k \in K, \tag{3}$$

$$\sum_{i:(i,n+\tilde{n}+1) \in A} x_{i,n+\tilde{n}+1}^k = 1 \qquad \forall k \in K, \tag{4}$$

$$\sum_{i:(i,j) \in A} x_{ij}^k - \sum_{i:(j,i) \in A} x_{ji}^k = 0 \qquad \forall j \in P \cup D, k \in K, \tag{5}$$

$$x_{ij}^k = 1 \Rightarrow \quad B_j^k \geq B_i^k + d_i + t_{ij}^k \qquad \forall (i,j) \in A, k \in K, \tag{6}$$

$$x_{ij}^k = 1 \Rightarrow \quad Q_j^k = Q_i^k + q_j \qquad \forall (i,j) \in A, k \in K, \tag{7}$$

$$\max\{0, q_i\} \leq Q_i^k \leq \min\left\{C^k, C^k + q_i\right\} \qquad \forall i \in V, k \in K, \tag{8}$$

$$x_{ij}^k \in \{0,1\} \qquad \forall (i,j) \in A, k \in K, \tag{9}$$

$$\sum_{j:(i,j) \in A} x_{ij}^k - \sum_{j:(n+i,j) \in A} x_{n+i,j}^k = 0 \qquad \forall i \in P, k \in K, \tag{10}$$

$$B_i^k \leq B_{n+i}^k \qquad \forall i \in P, k \in K \tag{11}$$

$$e_i \leq B_i^k \leq l_i \qquad \forall i \in V, k \in K, \tag{12}$$

$$B_{n+\tilde{n}+1}^k - B_0^k \leq T^k \qquad \forall k \in K. \tag{13}$$

Building a PDP for autonomous GSE vehicle task allocation does need additional conditions in order to create a suitable model. The first addition should be time windows, which ensure that the tasks are performed between a predefined earliest and latest starting time[76][19]. Secondly, precedence constraints should be included for some types of tasks[12]. For example, the baggage carts need to be unloaded after pickup from the depot before they can unload an aircraft. Finally, the capacity of each GSE type should be taken into account as these are different for each type[40]. Additionally, a VRP is either static or dynamic. With a static vehicle routing problem are all tasks known before the optimization is initialized while a dynamic VRP has tasks that come in during the day, meaning that it should be able to add new tasks to the predetermined schedules. Aircraft that should be handled are known beforehand meaning that a static VRP is suitable for this research[76].

Another interesting paper defining a multi-objective VRP with simultaneous delivery and pickup and time windows (MO-VRPSDPTW)[85]. Five objectives are taken into account; number of vehicles, total travel distance, makespan, total waiting time and, total delay time. Being able to change the weights of these objective and therefore the importance of the objective can give interesting insights in the sensitivity of the choice of objective on the quality of the solution.

3.3.1. Heuristics

The defined capacitated multi-vehicle pickup and delivery problem with simultaneous deliveries, precedence and time windows is a complex optimization problem which is a NP-hard problem which cannot

be solved in an exact manner. In order to solve this created problem a heuristic or metaheuristic is required. First, a few examples of heuristics is presented followed by examples of metaheuristics.

Solomon [77] designed and analyzed four different heuristics and variants to calculate initial routes for a VRSP with time windows. Due to the time dependence, the routing and scheduling costs now include waiting time costs as well and not many heuristics are suitable for this type of VRP. Within this work is a homogeneous fleet assumed and are free-routing parallel procedures and sequential methods considered for the heuristic. The heuristics are presented in Table 3.2. Each heuristic is then tested on the same problem sets, which exist of random, semi-clustered and clustered generated geographical data using a random uniform distribution. Additionally, is the scheduling horizon varied and are the time windows randomly generated with a uniform distribution and randomly assigned to customers. From these experiments is concluded that the insertion heuristic with criterion 1 is very successful. Time driven insertions proved to work well for heavily time constrained problems, which our research will be. This heuristic is therefore an interesting option for the VRP of GSE vehicles. It however should be researched how this heuristic can work with a heterogeneous fleet.

Table 3.2: Types of heuristics for VRSPTW. Based on information from [77].

Heuristic Type	Definition
Savings *free-routing parallel*	a. Evaluates if adding one customer to the existing route results in savings compared with serving each agent separately. b. Vehicle capacity and time window constraints are taken into account, and waiting times are limited.
Time-Oriented, Nearest-Neighbor *sequential*	a. Finds the nearest unrouted customer to the depot to add to the route. b. When no customer can be added due to capacity and time window constraints, a new route is started. c. Metric used to find closest customer is interchangeable.
Insertion *sequential*	a. Initializes a route and evaluates this route with two criteria to determine if a customer should be added into the route at the evaluated instance. b. The first criteria determines the best insertion point in the route. c. The second criteria determines the best customer to insert.
Insertion criterion 1	Maximize the benefit of servicing a customer on the partial route instead of a direct route.
Insertion criterion 2	Minimize a measure of total route distance and time of adding a customer.
Insertion criterion 3	In addition to criterion 2, the temporal aspect also accounts for urgency of servicing a customer.
Time-Oriented, Sweep	a. Exists of a clustering and scheduling stage. First stage is same as original sweep heuristic. b. Scheduling stage uses insertion criterion 1. c. Process is repeated until all tasks are assigned.

In the paper of Bianchessi and Righini[8] are various heuristics presented, one of which is local search. In local search, a initial solution is generated which then is improved by using a neighborhood structure that defines a set of neighboring solutions to a given solution. Each iteration the solution is replaced by the best neighbor according to the objective function. This continues until a predetermined criterion is met. However, due to it being a greedy algorithm, it might get stuck in a local optimum. Therefore the usage of metaheuristics called tabu search might be interesting to ensure a global optimum is found.

The first metaheuristic discussed is called Tabu Search(TS)[45][8], in which a candidate solution is iteratively improved by applying local search moves that modify the current solution. However, certain moves are prohibited, or "tabu", for a certain number of iterations to prevent the algorithm from getting stuck in a local optimum. TS maintains a tabu list that records the moves that have been recently applied and prohibits them from being applied again for a certain number of iterations. This encourages the algorithm to explore new regions of the search space and avoid revisiting the same solutions. The algorithm also employs an aspiration criterion that allows a tabu move to be applied if it leads to a

new best solution. This allows the algorithm to escape from local optima and continue to explore the search space.

For usage in combination with a VRP, TS can be used to iteratively improve a set of vehicle routes by swapping tasks between routes or inserting tasks into routes, while maintaining the constraints of the problem such as capacity and time windows. The tabu list is used to prohibit moves that would violate these constraints, and the aspiration criterion is used to allow moves that improve the current best solution. TS is a powerful metaheuristic that can efficiently explore a large search space and converge to high-quality solutions for complex combinatorial optimization problems such as VRP. However, the performance of TS can depend on the choice of algorithmic parameters such as the tabu tenure and the neighborhood structure used to generate moves, which must be carefully selected for each problem.

An interesting area of metaheuristics used in VRP in order to reduce computational time are evolutionary and swarm algorithms[51]. The genetic algorithm is a well known evolutionary algorithm, which is inspired by the process of natural selection. It works by creating a population of candidate solutions to a problem, and then repeatedly applying selection, crossover, and mutation operators to the population to generate new candidate solutions. Selection involves choosing the best solutions from the population based on a fitness function, which measures how well each solution solves the problem. Crossover involves combining two solutions from the population to create a new solution that shares some of the characteristics of both parents. Mutation involves randomly changing some of the characteristics of a solution to introduce new variation into the population. After each iteration of selection, crossover, and mutation, the fitness of the new population is evaluated and the process is repeated until a satisfactory solution is found or a termination condition is met.

The idea behind a genetic algorithm is that by applying selection, crossover, and mutation operators to a population of candidate solutions, it is possible to generate new solutions that are better adapted to the problem at hand. In the context of using it as a metaheuristic for a VRP, it works by evolving a population of candidate solutions over multiple generations, with the aim of finding an optimized solution to the VRP. In order for any metaheuristic to work properly, is it very important to choose the characteristics such as population size carefully[45].

Particle swarm algorithm is a stochastic optimization approach and is part of the population based family[51]. This algorithm is based on the navigation mechanism of birds in nature and is inspired on equations that simulate the interaction of individuals in flying swarms. In the context of using it as a metaheuristic for a Vehicle Routing Problem (VRP), a swarm algorithm works by simulating the collective behavior of a swarm of agents, with the aim of finding an optimized solution to the VRP. In a typical swarm algorithm, a population of agents (also known as particles) is initialized with random positions and velocities in the search space, each representing a candidate solution to the VRP. These agents then move through the search space according to certain rules, which are separation, alignment, and cohesion. With separation is force applied to prevent collisions, alignment adjusts flying direction based on neighbors and cohesion maintains a predefined distance to prevent isolation.

The position of each agent is updated at each iteration based on its current position and velocity, as well as the positions of other agents in the swarm. The objective function is used to evaluate the fitness of each agent at each iteration, and the agents adjust their movements in response to this evaluation. The swarm algorithm can use various mechanisms to promote exploration of the search space and exploitation of promising solutions. For example, the algorithm may use a neighborhood topology that restricts the agents' interactions to a subset of other agents, or it may use a global best position to guide the movements of the agents. The swarm algorithm continues to iterate until a stopping criterion is met, such as a maximum number of iterations or a solution that meets a certain threshold of fitness.

Lastly, a variation on the particle swarm algorithm is the Ant Colony Optimisation (ACO) which is insipred on as the name suggest, ants[51]. This algorithm exists of three phases; construction phase, pheromone update phase and optional daemon action phase. A population of virtual ants is initialized, and each ant starts at a random location in the search space. The ants move through the search space according to a probabilistic rule that is guided by pheromone trails left by previous ants. The pheromone

trail strength is proportional to the quality of the solution found by the ant. As the ants move through the search space, they construct candidate solutions by selecting edges to visit based on a combination of pheromone trail strength and a heuristic value that measures the desirability of visiting each edge. After all ants have constructed a solution, the pheromone trails are updated by depositing a certain amount of pheromone on the edges visited by the best ant(s) and evaporating a certain amount of pheromone from all edges. The algorithm iteratively repeats this process until a stopping criterion is met, such as reaching a maximum number of iterations or finding a solution that meets a certain threshold of quality.

Relating this to a vehicle routing problem, the ants represent the GSE vehicles and the edges represent the routes between goal locations. The objective is to find a set of routes that minimize the total distance traveled by the vehicles while servicing all customers. The pheromone trails represent the quality of the routes, and the heuristic value represents the desirability of visiting each goal location.

3.4. Auctions

As mentioned before, it is important to have a central auctioneer which makes decisions based on the global objective. Two of such types of auction-based multi-agent coordination systems are proposed for multi-robot task allocation. There is single-item auctioning where separate auction for each task are held and combinatorial auctioning in which auctions are held with grouped tasks[71]. Within the domain of single-item auctioning exists the parallel and sequential single-item auction. The parallel auction holds multiple auctions at the same time which speeds up the process but make it impossible to incorporate synergies. Without these synergies, the auction cannot ensure high quality solutions so this type of auction is not suitable for this research. Sequential single-item(SSI) auctions fortunately are able to incorporate synergies which ensures that the solution is bounded suboptimal. Combinatorial auctions are able to take synergies into account resulting in the ability of finding an optimal solution. However, the amount of bids generated grows exponentially with the number of tasks which results in high computational costs for larger systems. Therefore, combinatorial auctions are also not a suitable option for this research. A combination of sequential single-item and combinatorial is called the sequential single cluster(SSC). Here the agents only bid on a subset of tasks reducing the computational time and increasing the synergy compared to SSI[37][41]. In the remainder of this section are only SSI and SSC discussed and evaluated.

Table 3.3: Overview of types of auctions.

Type of auction	**Suitable(Y/N)**	**Reason**
Parallel single-item	N	Synergies of tasks are not captures
Sequential single-item	Y	Takes parts of synergies into account
Combinatorial	N	Amount of bids exponential with number of tasks
Sequential single-cluster	Y	Synergy of combinatorial and less bids required

3.4.1. Sequential Single-Item Auction

Every task is initially not assigned to any agent but allocated in a single multi-round auction. Every round all agents need to bid on the task for which they can bid the lowest. The bid is determined by the marginal increase in travel distance if the task would be allocated to that agent. Synergies are taken into account which bounds the solution slightly, the solution is at most twice the minimal travel distance independent of the precision of the distance calculation[37].

Sequential Single-Item auctions can be solved with three team objectives; MINISUM, MINIMAX, MINI-AVE which minimizes the sum of travel costs, minimizes the maximum travel cost of one agent and minimizes the average travel cost of all targets respectively. For each of the team objectives are bidding rules defined as well to reach the shared goal. Within this problem are the following mathematical definitions formed[41].

- A set of agents $A = a_1, a_2, ..., a_n$
- A set of targets $T = t_1, t_2, ..., t_m$
- A non-negative cost function $c(i, j)$, $i, j \in R \cup T$, and $c(i, j) = c(j, i)$

As proven by Lagoudakis et al.[41], SSI is a NP-hard problem which required a heuristic to approximate the optimal paths for all agents. This heuristic can be changed and this will not worsen the suboptimality bounds as long as it returns values of higher quality than the heuristic proposed in this research. In addition to changing the insertion heuristic, are also several improvements developed such as including rollouts, bundle bid or regret based winner determination. An interesting adaptation for the application with autonomous GSE is SSI with temporal constraints included (TeSSI) due to the ability of including time windows for tasks.

TeSSI is developed by Nunes et al.[54] and included time windows in which tasks should be executed. Agents keep track of the time windows in a simple temporal network(STN) and the cost function is based on the makespan in combination with the total distance traveled. Generating the bids is now dependent on the agent's schedule and the time window of the to be allocated task. The bidding is now also restricted by duration and travel time constraints associated with the task. When an agent determines the bid on a task, it first looks at it's current schedule in the STN. An example STN is shown in Figure 3.3 and this figure used to explain the steps taken by an agent. This example contains three tasks, the task for which the bid is generated needs to be fitted in this schedule while complying with the duration and travel time constraints. For the task which complies with the constraints and results in the minimal marginal increase of costs is a bid generated; the corresponding cost.

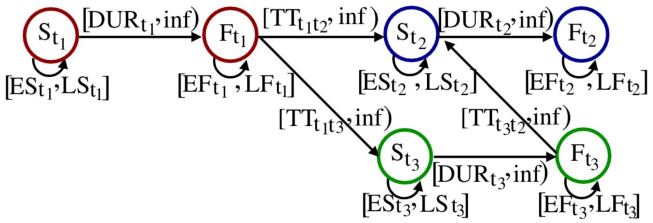


Figure 3.3: A simple temporal network of an agent. Retrieved from [54].

Experiments show that TeSSI works well in situations where the tasks are all known beforehand but also when tasks are introduced over time and it outperforms other algorithms as shown in the paper of Nunes et al. Nunes et al. [55] also created an extension which takes precedence into account in addition to temporal constraints, TePSSI. It is similar to TeSSI with the addition that first the tasks with higher priority are entered into the auction followed by the tasks with lower priority. This approach seems very suitable for autnomous GSE vehicles, however there is aspect which makes it impossible to utilize this approach. Within TeSSI and TePSSI, when no agent can fit a task within their schedule, the task is discarded which is undesirable as it cannot guarantee completeness.

3.4.2. Sequential Single-Cluster Auction

The SSC auction type is said to take synergies better into account compared to SSI and its extensions which should result in lower suboptimal bounded results. Heap and Pagnucco.[28] developed this Sequential Single-Cluster auction and form clusters before the auctions which are fixed. These clusters are based on single locations of the tasks meaning that tasks that have to be performed at the same location are clustered. However clustering for pairs of pick-up and delivery locations is a bit more difficult and uses a two-level clustering approach. First clustering on pick-up locations is performed, follow by clustering on delivery locations within the first clusters while taking into account capacity constraints. In the paper of Heap and Pagnucco this clustering approach works very well as pick-up and delivery locations are scattered in the environment, however at an airport the pick-up locations are centralized while the delivery locations are scattered which possibly worsens the results. Additionally, SSC does not take temporal constraints into account which makes it unsuitable for this research.

3.5. Final remarks on Task Allocation

Previous work on task allocation for aircraft ground handling by Chen et al. [13] proposes a combination of a TeSSI auction and an optimization solver based on a single vehicle pick-up and delivery problem (SPDP). Using the SPDP solver for the generation of bids ensures that each task is assigned to an

agent, making TeSSI suitable for the task allocation of GSE vehicles. In order to speed up the model, a bundling method is used. Bundles are generated by gathering all neighboring tasks of the same type which are evaluated based on time windows, duration of the tasks and travel distance. If the bundle is deemed feasible, a bundle value and the optimal order of tasks is saved. The size of the bundles as well as the total amount of tasks bundled is predetermined in order to generate bounded suboptimal bundles. Experiments show that the bundling method is far from optimal and actually decreases the optimality of the overall solution and the computational time does not decrease due to the bundling, which was not expected. A different bundling heuristic is required in order to use this method proposed by Chen et al. properly due to the extensive computational time that it created in the SPDP model.

Other research focused on scheduling aircraft ground handling operations uses a combination of a VRPTW and a metaheuristic called improved Sequential Iterative Method iSIM. This research is performed by Padron et al.[56, 57] the VRPTW is solved by first generating a quick initial solution using the I3 insertion heuristic, discussed in subsection 3.3.1 Table 3.2. This is then followed by a local search process based on a LNS together with CP to further improve the solution. The VRPTW is not only solved for the minimizing the total completion time but also for minimizing a combination of the waiting time and total reduction of time windows of each task which makes it a bi-objective model. This research shows that the combination of a VRP together with a metaheuristic is a suitable solution and variations of this concept might improve the performance of this research further.

The task allocation proposed for the remainder of this research is using a Pickup and Delivery Problem (PDP) with time windows, capacity and temporal precedence constraints as presented in section 3.3 in combination with a metaheuristic. This decision has been made based on the following reasons; the vehicle routing problem framework fits really well with the autonomous GSE vehicles problems as it is able to include for example time and capacity restrictions. In addition are different and multiple objectives possible which gives the opportunity of an extensive analysis concluding which objective fits best. Furthermore, there is a lot of literature available as well as benchmark models and other open source solvers which could be used as support and comparison in the performance analysis.

Vehicle routing problems are however known to become computationally expensive with increasing complexity as well as increasing vehicles. Fortunately, this can be prevented by using a well formed metaheuristic such as tabu search which provides a high quality initial solution, reducing the runtime of the VRP. If the PDP approach proves to be too computationally expensive, then the approach of Chen et al.[13] will be adapted to this research.

4

Multi-Agent Path Finding

The next chapter for ground support equipment vehicles is to become autonomous. Each vehicle should be able to move around the airport without vehicles moving into any stationary or moving objects. To enable this, it is required to develop an algorithm which makes it possible for each vehicle to plan their own conflict-free path from their parking space towards the aircraft stand where they need to perform turnaround tasks and vice versa. This problem is not unique to autonomous aircraft ground handling, as is also common in warehouses[87], airport surface movements[22], factories and other autonomous vehicles such as UAVs[90]. This complex logistical problem is therefore widely researched and there are various disciplines within path finding.

The discipline that will be focused on for this study is multi-agent path finding and the definitions within this technique will be explained in section 4.1. Within this discipline are many different solving methods found in literature which can be divided within optimal and suboptimal solvers. Sections 4.3 - 4.6 describe the multiple types of multi-agent path finding solvers and finally, all solution techniques are compared against each other in section 4.8 to find the most suitable approach for airport ground handling.

4.1. Definition

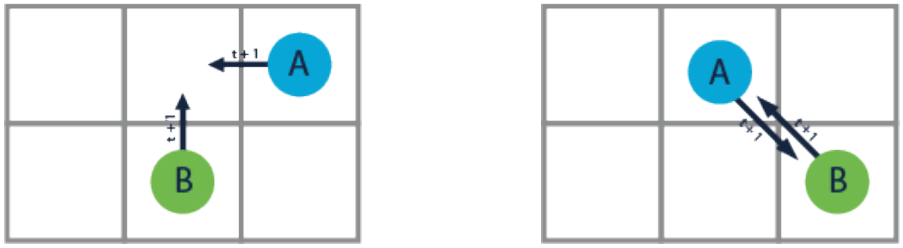
Aircraft ground handling makes use of different types of GSE vehicles, as discussed in section 2.1, which need to travel from their parking location to the aircraft ramp using the existing infrastructure of AAS which has been researched in section 2.2. In order to ensure that autonomous GSE vehicles can move around the airport without the occurrence of conflicts, each vehicle should be able to plan their own collision-free path which can be driven simultaneously in the same environment. From this requirement can be derived that also the time dimension should be considered next to the spacial three dimensions. This type of problem is in literature referred to as a Multi-Agent Path Finding (MAPF) problem.

In all the papers discussing multi-agent path finding, different terms are used to describe the same concept. To simplify, the definitions that are used in the rest of this study are described below and are adapted from Stern et al.[80] and Felner et al.[21]. The environment is defined as an undirected graph $G = (V, E)$ where V refers to vertices and E to edges. In this environment are k agents; $a_1 \ldots a_k$ that move from a start position; $s_i \in V$ to a goal position; $g_i \in V$. Time is assumed to be discreet and each time step all agents perform an action. This action is either remaining at its current location or moving from its current position v to an adjacent vertex v' called $wait$ and $move$ respectively. An vertex is adjacent if and only if there exists an edge between the two vertices; $(v, v') \in E$. A sequence of actions of a single agent is called a $path(\pi)$ and the set of k paths of which each path belongs to a different agent is called a $solution$.

The goal of a MAPF problem is to find a valid solution where there are no collisions between agents. When a conflict does occur, a valid solution can be reached by defining constraints which aims to resolve conflicts between agents. Within a grid based MAPF problem exist two different types of conflicts; $vertex$ and $edge$ conflict. A visualization of the two conflicts can be found in Figure 4.1. A conflict is resolved by creating a constraint for each of the agents involved. The mathematical definitions are the

following:

Vertex conflict: Agents a_i and a_j plan to both occupy the same vertex at the same time, i.e. $a_i(t) = a_j(t)$
Edge conflict: Agents a_i and a_j plan to transverse the same edge at the same time. $\rightarrow a_i(t) = a_j(t+1)$
Constraint: A tuple $\langle a_i, v, t \rangle$ which prohibits agent a_i from occupying vertex v at time step t.



(a) Vertex conflict (b) Edge conflict

Figure 4.1: Different types of conflict in a grid based environment.

In order to find the most desirable valid solution, it is needed to determine an objective function. Within the MAPF problems are two objective functions frequently used which are the makespan and the sum of costs. Makespan is defined as the number of time steps required for all agents to reach their final goal location. The mathematical definition of the makespan is $max_{1 \leq i \leq k} \mid \pi_i \mid$ and is mostly used for compilation-based MAPF algorithms. The second objective function is the sum of costs which sums the time steps required for all agents to reach the final location of their path and is mathematically defined as $\Sigma_{1 \leq i \leq k} \mid \pi_i \mid$. This objective is mostly used for search-based MAPF algorithms, but there is also research which combines the makespan and sum of costs[82] or minimizes the distance traveled by agents or limits the makespan by adding deadlines to the tasks[48].

Planning paths in a real environment is complex and computationally expensive, which is why in MAPF problems various assumptions are made to simplify the problem[3]. Below are the most common assumptions listed and these assumptions are valid for all the solvers discussed in the remainder of this chapter unless specifically stated.

- Time is discrete.
- Agents occupies a single vertex each time step and have an uniform size and shape.
- Uniform transverse edges.
- Kinematics of an agent is neglected.
- Only one action can be performed per time step by one agent.
- Plan execution is perfect.

MAPF solvers can be categorized in multiple settings, but one of the most common is; centralized and decentralized[21]. In the centralized setting are all agents controlled by one decision maker while in the decentralized problem each agent decides on their own which can be cooperative or self-interested. This study exists of agents which strive for a global optimum, making self-interested decentralized MAPF problems out of the scope of this study. A distinction can also be made based on the amount of tasks can be assigned to an agent. If an agent has only one task and stays at the goal location once reached, it is a one-shot problem. Literature speaks of lifelong problems when agents can be assigned to multiple tasks that have to be executed in the same graph. This type of problem is developed for autonomous warehouses[80]. Lifelong planning problems are very suited for the modeling of autonomous GSE vehicles as it is known that there is a surplus of tasks compared to the amount of GSE vehicles so each vehicle needs to assigned to more than one task.

Other important properties are completeness, optimality and soundness. Soundness refers to the validity of the solution which in MAPF is defined as a set of k collision free paths for each agent. An

algorithm is complete when a solution is found if it exists. Finally, optimality is reached when the best possible solution is always returned by an algorithm. Identifying these properties for a solver is an important step in choosing a MAPF solver for autonomous aircraft ground handling. There are three types of solvers that are of interest for this literature study; optimal, suboptimal and bounded suboptimal solvers.

An optimal solver will go through all possible routes and solutions in order to guarantee the optimal solution for a predetermined objective function if it exists. Suboptimal solutions are complete and sound but do not guarantee optimality as the name suggests. The solver returns a solution if it exists but it may not be the solution which fits the objective best. To ensure a certain quality level of a suboptimal solution, it is possible to predetermine bounds in which the solution must fit and this type of solver is called bounded suboptimal.

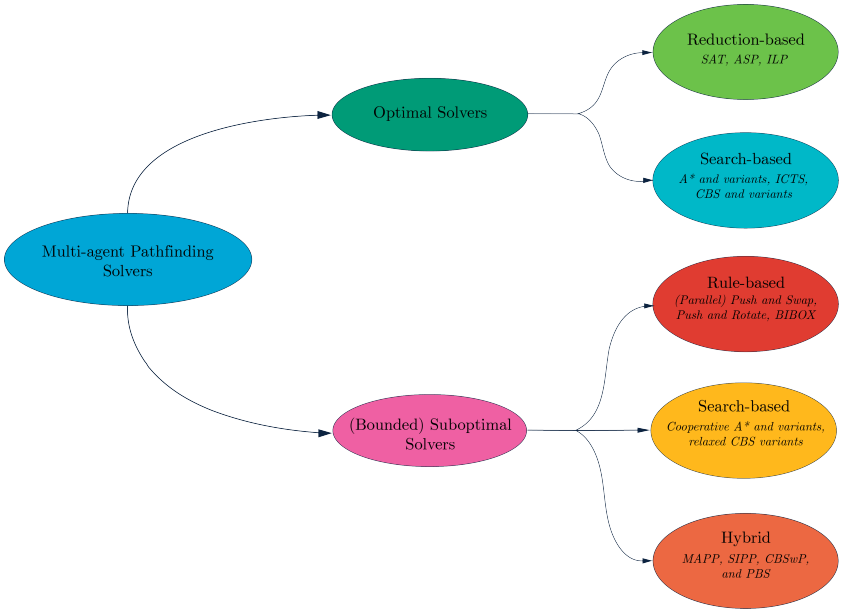


Figure 4.2: Different types of MAPF solvers found in literature

The MAPF problem is also a NP-hard problem [81][88] which makes it very difficult to find an optimal solution within a limited runtime when the amount of agents increases. This is why large MAPF problems are mostly solved using (bounded) suboptimal solvers. However, for problems with a small amount of agents is it a possibility to use optimal solvers. Figure 4.2 shows categories in which MAPF solvers can be placed and each of these categories will be discussed in the following sections.

4.2. Reduction-based Solver

Another type of solver reduces the MAPF to standard, better known problems. This type of solver is therefore called a reduction-based solver. This solver is known to return optimal, high quality solutions and is optimized for makespan. Changing the objective to sum-of-costs is unfortunately not done easily, as the reduction needs to be redone for each objective. There are multiple examples of reduction-based solvers in which Integer Linear Programming, Boolean Satisfiability Problem or Answer Set Programming is used to solve the MAPF problem[21][82][46]. However, due to the inability to switch easily between objectives and the difficulty of reduction of a MAPF problem will this field of solvers not be explored in more detail.

4.3. Optimal Search-based Solvers

Optimal solver guarantee to return the solution which fits the objective function best. Because of this property, a lot of research regarding optimal solvers has been published. There are three types of optimal search based solvers that are commonly used. Subsection 4.3.1 discusses the optimal A*

solver and variants, subsection 4.3.2 discusses the Increasing Cost Tree Search(ICTS) solver and this has been further developed into Conflict Based Search(CBS), which is discussed subsection 4.3.3 along with interesting adaptations.

4.3.1. A* based solver

The most common best-fit search based algorithm is A*[79]. This method makes decisions using a heuristic to estimate the cost of the remaining path in addition to the cost of reaching the current location. The agent will determine the possible moves from the current location and for each of these moves the cost of the remaining path is calculated. A* then executes the move with the lowest cost and the process is repeated from this new location. Due to the low complexity of this A* algorithm it is an very interesting approach to consider however, it becomes inefficient with an increasing amount of agents and large environments due to the size of the search space and branching factor(b_{agent}) growing exponentially with the number of agents; $|V|^k$ and b_{agent}^k respectively.

Decreasing the size of the two bottlenecks is the first step that is required in order to be able to use A* in airport ground operations. There are three improvements developed to successfully decrease the search space and branching factor. The first approach is called Operator Discomposition(OD) developed by Standley[78] which is a framework that can be used as an addition to a path finding algorithm such as A* and aims to decrease the branching factor. OD decouples the problem into k single-agent path finding problems and priority is assigned to each agent. Paths of agents are planned according to this priority, so an agent with lower priority needs to avoid previously planned paths of other agents. Once the paths of all agents are planned, a new state is generated resulting in a branching factor that is independent of the amount of agents.

Enhanced Partial Expansion EPEA* developed by Goldenberg et al.[24] is another approach to decrease the branching factor. In this approach, there are only nodes created when the total cost of all agents' paths of the child node is equal to the total cost of the parent node. The costs of these child nodes are found using an Operator Selection Function (OSF) which uses a priori domain knowledge. This significantly reduces the size of the tree resulting in less memory required. The third approach, also developed by Standley is called Independence Detection(ID) and focuses on decreasing the search space. Agents are grouped together when there is a conflict somewhere along their path. New paths will then be formed optimally using A* for the newly formed group. This process repeats until there exists a valid solution. The size of the problem is then dominated by the largest amount of agents in a group k' reducing the search space to $|V|^{k'}$.

As discussed before, the two biggest downsides of A* solvers is when the amount of agents increases, that both the memory required and the branching factor increase exponentially. Solutions which decrease the effective amount of agents with the use of merging/bundling agents is an interesting area to explore. Independent Detection(ID) for example merges two agents when a conflict is found between the two. The path of these agents will then be planned as one group, reducing the total amount of agents. The paths of agents in this group will then be found using a version of A*. This process of merging agents into groups is then repeated until a valid solution is found. The runtime is dominated by the group containing the most agents[21][72].

4.3.2. The Increasing Cost Tree Search

In order to speed up the path finding process, an algorithm has been developed which uses a two level approach. The first level that the algorithm initiates, called the high-level searches the increasing cost tree (ICT) and every node of this tree consists of a vector $[C_1, C_2, ..., C_k]$ which represents all possible solutions for each agent with their associated cost. In the root node of the tree are the solutions with the lowest cost, the optimal solution, saved for every agent. Children of this root node are generated by creating a node where one agent's cost is increased by one unit of cost. So if there are three agents, then for each node there are three children created[72]. An example of an ICT with three agents is visualized in Figure 4.3. The high level makes use of a breadth-first approach which ensures that the earliest found valid solution is also the optimal one.

The low level of ICTS finds the valid paths for all agents with the costs defined by the high level. Each

possible path of agent a_i with cost C_i is generated and are stored in a Multi-valued Decision Diagram (MDD). Once there exists a MDD for each agent, a search can be performed on the cross product of the MDD which checks if there exists a combination of k paths for which the costs are equal to the determined value in the high level and do not conflict. If a combination of valid paths is found, then the lower level returns true and the search is concluded. Otherwise the search continues with the higher level continuing with the next node of the ICT[21].

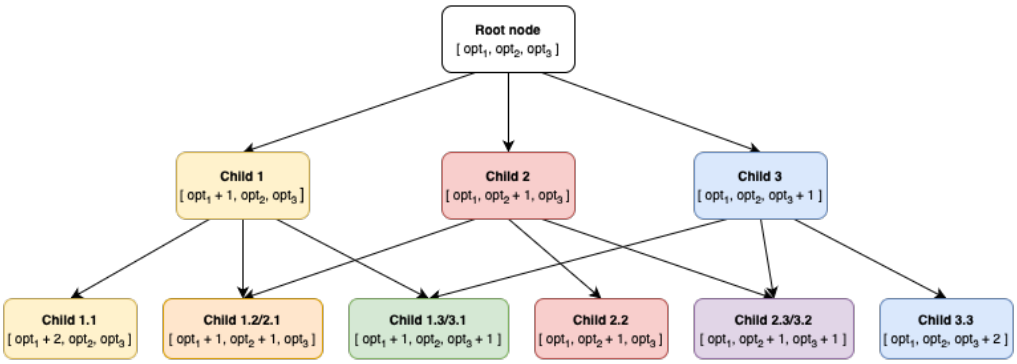


Figure 4.3: Increasing Cost Tree for three agents.

4.3.3. Conflict Based Search

Conflict Based Search is an optimal MAPF algorithm that is first described in literature by Sharon et al.[72]. The algorithm consists of two levels which solves a MAPF problem by converting the problem into constrained single-agent pathfinding problems. CBS is initiated by searching a constraint tree(CT) in the high level. This tree is binary and each node of the tree contains three properties. An example of a high level CT can be found in Figure 4.4 and the three properties are: constraints(Con), a solution(Sol) and total cost(Cost). The constraint property of node N is called $N.constraints$ and contains the constraints that are assigned to agents in order to obtain a conflict-free solution. $N.solution$ is a set of k consistent paths, one path for each agent which adhere to the constraints and $N.cost$ is a summation of all agents paths costs also known as the f-value of node N. A node in the CT is a goal node if the generated solution has no conflict, i.e. is valid. Below, in Algorithm 1 is the pseudo code for CBS presented.

Algorithm 1: High level of CBS

```
Input  : MAPF instance
Output: Collision free paths for all agents
Root.constraints = ∅
Root.solution = find individual paths by the low level()
Root.cost = SIC(Root.solution)
insert Root to OPEN
while OPEN not empty do
    P ← best node from OPEN //lowest solution cost
    Validate the paths in P until a conflict occurs.
    if P has no conflict then
        return P.solution // P is goal
    C ← first conflict (a_i, a_j, v, t) in P
    foreach agent a_i in C do
        A ← new node
        A.constraints ← P.constraints + (a_i, v, t)
        A.solution ← P.solution
        Update A.solution by invoking low level(a_i)
        A.cost = SIC(A.solution)
        if A.cost < ∞ //A solution was found then
            Insert A to OPEN
```

The algorithm start with a single node for which a solution, the cost and conflicts are found and added to the node. To attempt to resolve the earliest conflict, the algorithm will proceed with generating two new nodes which each contain a constraint which resolves the found conflict. An example of the higher level CT is found in Figure 4.4 where the constraints are found in the top left corner, the cost in the top right and below the two is the solution written for the two agents of this example. As can be seen, at time step $t = 3$ are both agents in vertex D. This conflict needs to be resolved by adding the constraint that either agents 1 or 2 cannot be in vertex D at $t = 3$ and this can be found in the two new nodes.

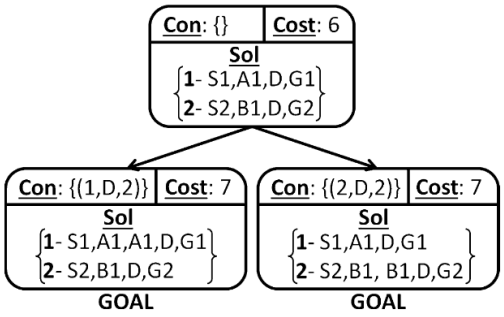


Figure 4.4: An example of a complete Constraint Tree. Retrieved from [72].

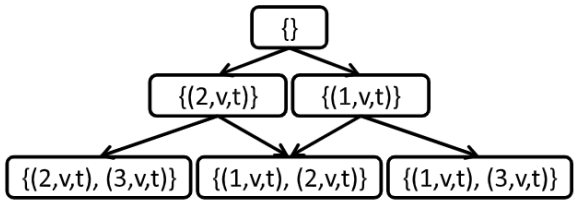


Figure 4.5: A binary Constraint Tree where three agents are involved in a conflict. Retrieved from [72].

When there are more than two agents involved in the conflict, then $k - 1$ constraints are added to the k nodes generated using a binary tree. This is shown in Figure 4.5 where there are three agents which are involved in a conflict at vertex v and time step t and in the three nodes are each two constraints added.

The paths which are part of the solution of a node are found by initiating the lower level of the algorithm. In the lower level, each agent generates their path which satisfies the constraints assigned to it while completely ignoring the other agents. The solution created by the low level will then be validated in the high level. The generation of the paths can be done by any pathfinding algorithm and in this paper A* in combination with a shortest path heuristic is utilized. When the created path does not satisfy the constraints, it will be discarded and otherwise it is added to the node in the CT. If two nodes have the same cost, a tie-breaking strategy is applied which is based on Standley's tie-breaking conflict avoidance table(CAT)[78]. This entails that the node in which the conflict involves the least amount of agents is preferred.

This will results in a set of collision-free paths that satisfy all the generated constraints. Due to splitting the problem, it is possible to lower the complexity of the problem as a whole resulting in a lower computational time compared to A*-based algorithms. Another interesting aspect is that CBS is proven to be an optimal and complete MAPF algorithm and performs well in environments containing narrow corridors and crossings i.e. bottlenecks. This type of environment can be compared to the roads leading from and to the aircraft ramps of AAS. However, there are also situations in which A* outperforms CBS. When an environment contains an open space that agents have to cross, A*-based techniques will find a conflict solution more quickly than CBS as it eliminates conflicting solutions immediately instead of generating new nodes from these solutions.

Meta-agent Conflict Based Search

The aircraft ramp is considered a large open space in which agents can move from and to their goal location from a single point of entry. In this type of environment is CBS not an efficient algorithm as in open spaces there will be a lot of internal conflicts between agents i.e. strongly coupled agents, resulting in generating many new nodes which will increase the computational time. Sharon et al. have developed a variation of CBS which deals with these strongly coupled agents by merging them into one meta-agent called Meta-agent Conflict Based Search(MA-CBS). The strategy of merging agents is defined as follows. The first component of the strategy is the merging policy which describes when a meta-agent will be formed while the second component describes how constraints are made for meta-

agents.

Two agents are merged into one meta-agent when the number of conflicts between the two agents exceed a predetermined parameter B. If this parameter is not exceeded, then the node is branched into two nodes. When two or more agents are merged, the low-level is invoked to find a conflict-free path for the agents within the meta-agent using a MAPF solver. The constraints of the individual agents need to be merged as well to ensure that the generated paths of the meta agent are valid. Internal conflicts i.e. conflicts which involve only the agents within the meta-agent are neglected. External conflicts however need to be taken into account but should only assigned to the agent involved and not the entire meta-agent. The proof optimality and completeness for CBS applies to MA-CBS as well. Experiments comparing CBS(MA-CBS($B = 0$)), EPEA*(MA-CBS($B = \infty$)), ICTS and, MA-CBS with varying values for B in three different environments show that MA-CBS outperforms all other low-level solvers when a value for B is chosen between 0 and ∞. From these experiments can be concluded that MA-CBS tackles the issues that arise with open spaces and still performs well in bottleneck situations.

Improved CBS

A drawback of MA-CBS is that the the computational effort of planning paths for agents prior to the merge is unnecessary and could have been prevented by merging the agents at the root node of the CT. A new, improved version of MA-CBS is proposed by Boyarski et al. called Improved Conflict Based Search (ICBS)[11]. With ICBS is the CT reset after the decision has been made to merge agents. This strategy is called Merge and Restart (MR) and aims to improve the computational time significantly by merging the agents into a meta-agent in the root node of the CT. Another improvement of ICBS compared to MA-CBS is that the splitting of the CT is done more careful by prioritizing and bypassing conflicts. Prioritizing nodes makes the algorithm choose to continue with the node in CT which has a minimal increase in cost by adding the constraint in question. The final improvement proposed is to change a path of an agent slightly in order to bypass a conflict making it unnecessary to split a node resulting in a smaller CT. There are three conditions stated that need to be met for a valid bypass. The new proposed path should exclude the found conflict, the cost needs to be unchanged and the new path is consistent with the previously determined constraints. It is shown that for the same experiments as mentioned in the previous sections that ICBS significantly improves the runtime in the same environment as well as the success rate of the algorithm.

4.4. Rule-based Solver

Rule-based solvers do not execute a search but instead develop movement rules for agents for different environments and scenarios. These type of solvers are known to be computationally efficient, however this solution may be far from optimal and is not always complete. A well known rule-based solver which is complete is an algorithm by Kornhauser et al. but is complex to implement, making it a unattractive option for this research. Other variants include Push-and-Swap, Push-and-Rotate, Parallel Push-and-Swap and BIBOX[21]. All of these variants are proven to be complete only when there are two unoccupied vertices in the graph. However, the quality of the solution cannot be guaranteed and optimizing for makespan or sum of cost is an NP hard problem[79]. For these reasons are rule-based solvers not further taken into account for this research.

4.5. Suboptimal Search-based Solvers

In this study, the amount of autonomous GSE vehicles considered is relatively small compared to the large MAPF problems found in literature such as automated warehouses where they use suboptimal solvers. However, when the scale of the study would be increased to airport wide MAPF, the problem becomes significantly larger, making it undesirable to utilize optimal solvers. Suboptimal search based solvers could be an option to tackle the scalability problem of optimal search based solvers. First, different types of suboptimal A* based solvers are discussed in subsection 4.5.1, followed by bounded suboptimal adaptations of CBS in subsection 4.5.2.

4.5.1. A* based Suboptimal solvers

Each autonomous GSE vehicle has knowledge of the paths of other vehicles as they are of cooperative nature. This makes it possible to split the large MAPF problem into single-agent searches and each

agent can take the paths of other agents into account in their search. Cooperative A*(CA*) combines this feature with the A* search framework and is developed by David Silver [74]. It is required to predetermine priority for each agent, such that each agent knows who to give priority. Each single agent planned route is saved in a reservation table and these entries need to be avoided by agents with lower priority. This reservation table needs to be a three-dimensional grid; two spatial and one time dimension. The heuristic used to determine the order in which agents plan their path is very important as it will determine the quality and completeness of the solution. Silver proposes Hierarchical CA*(HCA*) which uses a heuristic that improves the completeness, called Reverse Resumable A* search. However this improvement still cannot guarantee a complete solution.

Windowed Hierarchical Cooperative A*

In order to improve on optimality and completeness as well as decreasing the required solution space, Silver developed Windowed HCA*(WHCA*). Instead of taking every movement of other agents into account, only a small time window of movements is considered. The path outside of this window is planned without taking other agents paths into account. The window will shift periodically and each time agents will re-plan their paths within this window cooperatively. Priority of agents can be changed in each time window which could improve the qualitiy of the solution significantly. The search space is reduced to a interval w-step time window and the success rate increases compared to HCA*. Completeness can unfortunately still not be guaranteed.

There are multiple adaptations to WHCA* that aim to improve the performance of the solution. One that is worth discussing is Conflict Oriented WHCA*(CO-WHCA*)[9], which only creates a reservation table when a conflict is found. This table has the size of a predetermined interval w with the time of the conflict at the center of the interval. Prioritization(CO-WHCA*P) can be applied to determine which agent needs to replan based on the likelihood of finding a detour. These adaptations improve the solution cost and success rate significantly and prioritization reaching the best solution quality, but at the cost of computational time.

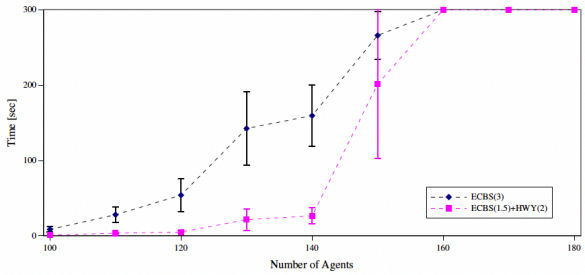
4.5.2. CBS based Suboptimal Solvers

From subsection 4.3.3 can be concluded that there are improvements developed which make CBS faster and more successful, however the scalability of CBS and ICBS could be improved on. Various attempts are proposed in literature such as Greedy CBS, Bounded CBS and Enhanced CBS by Barer et al.[6]. Greedy CBS(GCBS) relaxes the search of both the high and low level in order to speed up the search. Nodes that seem to be closer to the goal node in the high level CT should be prioritized to achieve better computational times. For the high level are five heuristics developed and tested in order to find a solution the fastest. It was found that number of pairs of conflicting agents obtains the best results in terms of simplicity and performance. The low level is relaxed by using a sub-optimal A* based solver which prioritizes on minimum number of conflicts. GCBS maintains completeness and improves runtime but the solution is not bounded, making it impossible to guarantee a solution which is close to the optimal solution.

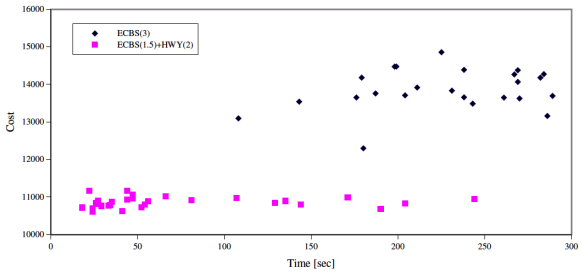
Bounded CBS is an extension of GCBS where both levels are bounded to a suboptimality bound by using focal search. For the high and low level are weighting factors chosen, w_H and w_L respectively. Both of these weights should together be equal to the suboptimality bound w, so $w_H \cdot w_L = w$. Choosing the values of these weights determines the performance of the algorithm, so they are important decision variables of the algorithm. Enhanced CBS(ECBS) is an advanced version of BCBS which uses the same weight for the low-level, but the high-level bound is equal to the sum of lower bound costs of consistent paths of all agents. The high level bound varies throughout the CT ensuring that only solutions within the set bound are returned. Other research, performed by Li et al.[43] mentions a lifelong version of ECBS called Bounded-Horizon ECBS. Instead of finding collisions on the complete path, only the conflicts in the first x timesteps are found and resolved. Due to this slight change of the algorithm is the high-level search tree significantly smaller. This adaptation makes ECBS lifelong in addition to being complete, bounded suboptimal, successful and computationally cheap. Bounded-Horizon ECBS is therefore interesting for modeling of autonomous GSE vehicles.

One other adaptation that should be considered is the usage of highways in the ECBS framework[16].

The usage of highways should reduce the runtime in situations where agents are tightly coupled, as it was found that ECBS performs significantly worse when agents need to move around in narrow corridors. From simulations performed in a Kiva-like domain which resembles a warehouse can be concluded that ECBS+HWY performs better regarding solution cost as well as runtime, as can be seen in Figure 4.6b and Figure 4.6a respectively. ECBS with highways is therefore also an interesting candidate to combine with Bounded-Horizon to even further improve the performance.



(a) Runtimes with varying number of agents. Each data point is the median runtime over 10 trials. A vertical bar around a data point indicates the median absolute deviation.



(b) Performances regarding costs vs. time. Each data point is an instance with 150 agents solved within 5 minutes.

Figure 4.6: Results of ECBS and ECBS+HWY on the Kiva-like domain. Retrieved from [16].

4.6. Hybrid solvers

The final category in Figure 4.2 is the hybrid solvers, which as the name suggests, is a combination of previously discussed solvers. It is common within this category that a search based solver is combined with a rule- or reduction-based solver in order to speed up the algorithm[46]. An interesting domain within hybrid solvers are the priority based solvers. Agents are assigned a priority which determines which agent will obtain the constraint from an occurring conflict. Insubsection 4.5.1 are already priority based A* algorithms discussed, but other existing algorithms such as CBS can be adapted to becoming prioritised planners and these are discussed in subsection 4.6.1. Other examples are combining CBS with reduction based solvers such as SAT or MILP and Multi-agent path planning(MAPP)[86]. It was decided in section 4.2 to not utilize reduction based solvers, so this area of hybrid solvers is also out of scope for this literature study.

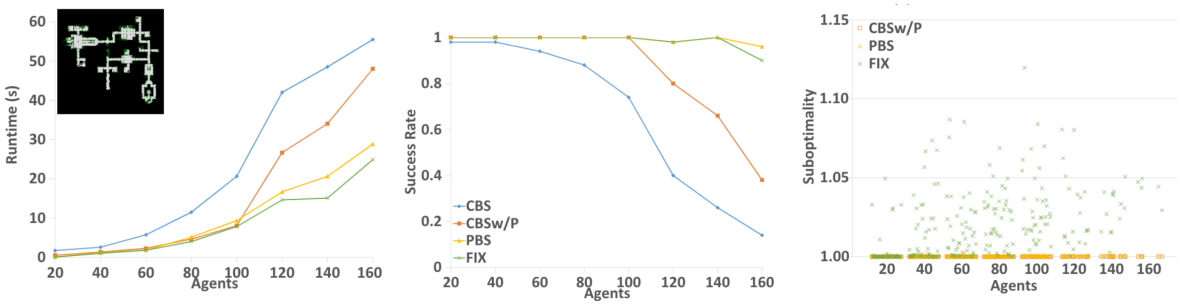
4.6.1. Priority-based CBS solvers

There are two methods proposed by Ma et al.[49] to include prioritization within the existing CBS framework. The first adaptation is proposed is called CBS with Priorities(CBSwP) and only gives an agent priority when paths collide. An ordering of priority is then added to information stored in the node, in addition to constraints, cost and solution. The determination of the priority can be done with any heuristic and only the child node for the agent with the lowest priority is created. This reduces the CT significantly results in improvement in scalability and runtime.

Priority-Based Search(PBS) is the second method discussed in this research and differs from CBSwP on a few aspects. In the high-level is a depth-first search performed in a Priority Tree(PT) rather than a best-first search in a CT. The high-level produces a priority order to find a solution, other aspects remain equal to the original CBS framework. The low-level now functions in the same manner as Co-operative A* but with a predefined priority in each node. PBS works as follows. In the low-level are paths constructed for each agent based on the priority instead of constraints and this set of paths is then validated in the high-level. When a conflict is found, it is clear that at least one of the agents was not included in the priority list of the node. From this node are then two children generated in which each child has a different agent with higher priority. The low-level is then initiated for both children and the cycle continues with the child that has the lowest solution cost.

Within PBS it is possible to have a initial priority order, however it may lead to suboptimal solutions or unfeasible solutions. The ability to add priority order is an interesting aspect for autonomous GSE vehicle modeling as it was found in chapter 2 that some types of GSE vehicles have a higher overall

priority. When all agents are included in the initial priority order, it behaves the same as CA* and this variant is called FIX. An experiment has been executed which compares CBS, CSBSwP, PBS and FIX. The results regarding runtime, succes rate and suboptimality within a map that resembles an airport environment the most can be found in Figure 4.7. It can be seen in Figure 4.7a that both prioritized CBS variant perform better in terms of runtime and that PBS with varying priority performs slightly better than FIX. However a distinction between CBSwP and PBS can be found in Figure 4.7b, as the success rate of PBS stays close to 1 for a much larger amount of agents. Finally, the suboptimality is heavily influenced by the initial prioritizing which is visible in Figure 4.7c where FIX suboptimality increases quickly with increasing amount of agents.



(a) Runtimes for increasing amount of agents.

(b) Success rate for increasing amount of agents.

(c) Suboptimality for increasing amount of agents.

Figure 4.7: Results of experiments in the brc202d map of the game Dragon Age: Origins. The lines correspond to the following solvers; blue = CBS, orange = CBSwP, yellow = PBS and green = FIX. Retrieved from [49].

As discussed in subsection 4.5.2, in order to make a CBS approach lifelong, a bounded horizon is needed to be implemented[43]. Bounded-Horizon PBS would be possible by limiting the timesteps that are taken into account for the high-level and the conflict resolution based on priority within the low-level. Experiments show that the runtime improves as well as the scalability.

4.6.2. Safe Interval Path Planning

Another interesting adaptation of A* is called Safe Interval Path Planning(SIPP) developed by Phillips and Likhachev[59]. Within this single agent path finding algorithm are other agents, with higher priority within the environment treated as dynamic obstacles. The priority is predetermined by an arbitrary prioritization method and this method is interchangeable. The main difference between an priority based A* algorithm such as HCA* and SIPP is that SIPP uses intervals in order to model dynamic obstacles instead of blocking a new location each timestep. This results in a much smaller search space, as SIPP only searches through intervals of a configuration which exists of information regarding the agent such as position, heading while other algorithms search each timestep. An interval is defined as a continuous period for a configuration which is either safe or occupied and these are called safe intervals and collision intervals respectively. An example of such a configuration and the corresponding timeline can be found in Figure 4.8.

The differences between Safe Interval Path Planning and A* are as follows. In this algorithm is $g(s)$ the shortest path from the start state to state s and with a heuristic function $h(s)$ is an estimate of the costs of the remaining path from s to goal state calculated. The cost of transitioning from state s to a new state s' is defined as $(c(s, s'))$. Finally, the costs of the path from the start state to the goal state is $f(s)$. The assumptions made in this model is that an agent is able to wait and acceleration/deceleration is negligible. Algorithm 2 shows the pseudocode for SIPP which is identical to A* except for the method to find successors and updating the time variable of state s; line 6 and line 12 respectively. The function to find successors is shown in Algorithm 3 which returns all possible moves from state s which comply with the safe interval(s) of each corresponding configuration.

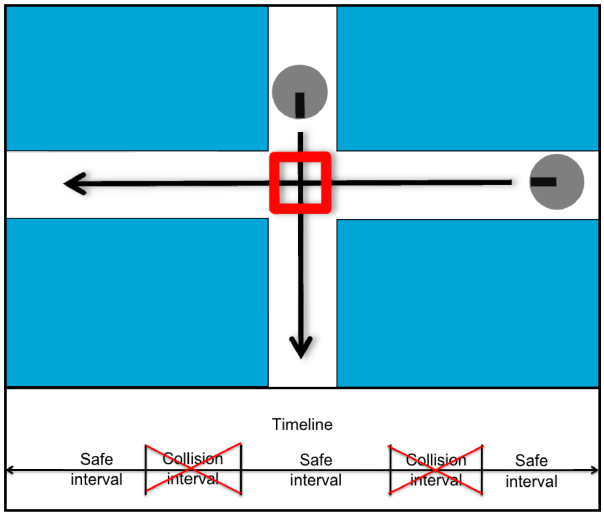


Figure 4.8: An example environment with one configuration highlighted and the corresponding timeline. Retrieved and adapated from [59].

Phillips and Likhachev compared the performance of SIPP with HCA* (subsection 4.5.1 in an indoor, narrow environment as well as an outdoor, open environment. The environment of this research consists of both narrow and open spaces; the perimeter road and bay area and therefore it is important that the model functions well in both tested environments. In both environments SIPP performed significantly better with a 100% completion rate within 5 minutes and an improved average run time with a factor of 30 and 16 respectively for the narrow and open environment which shows that SIPP indeed reduces the amount of expansions necessary.

Algorithm 2: Safe Interval Path Planning

```
Input  : Dynamic obstacles, start location,
         goal location
Output: Collision free path
g(s_start) = 0
OPEN = ∅
insert s_start into OPEN with f(s_start) =
 h(s_start)
while s_goal is not expanded do
    Remove s with the smallest f-value
     from OPEN
    successors = GetSuccessors(s)
    foreach s' in successors do
        if s' was not visited before then
            f(s') = g(s') = ∞
        if g(s') > g(s) + c(s,s') then
            g(s') = g(s) + c(s,s')
            updateTime(s')
            f(s') = g(s') + h(s')
            insert s' into OPEN with f(s')
```

Algorithm 3: Getsuccessors(s)

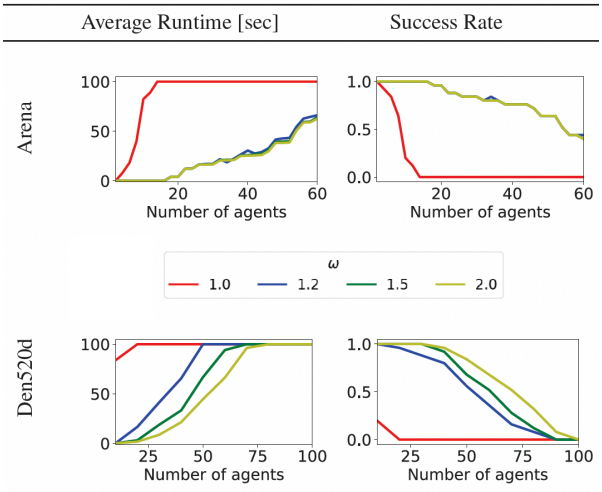
```
Input  : State with smallest f-value from
         OPEN, configurations
Output: List of successors
successors(s) = ∅
foreach m in M(s) do
    cfg = configuration of m applied to s
    t_m = time to execute m
    t_start = t(s) + t_m
    t_end = endTime(interval(s)) + t_m
    foreach safe interval i in cfg do
        if startTime(i) > t_end or endTime(i)
         < t_start then
            Continue
        t = earliest arrival time at cfg
         during interval i with no collisions
        if t does not exist then
            Continue
        s' = state of configuration cfg with
         interval i and time t
        insert s' into successors
```

Seeing that SIPP outperforms HCA* in both narrow and open spaces, SIPP may be a very suitable lower level search method for this research. However, the runtime of SIPP needs to be improved for larger environment. There are many adaptations created in order to apply this technique to different, larger environments. The first variation is called SIPP with Reservation Table (SIPPwRT) which aims

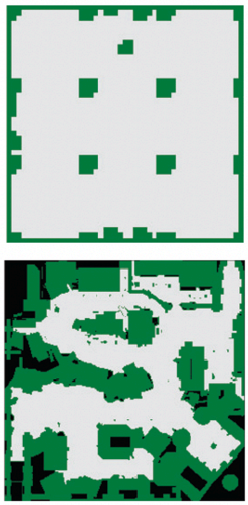
to reduce the run time by saving the previously determined intervals for each configuration.

Ma et al.[50] propose this generalized version of SIPP using a reservation table for a lifelong multi-agent pickup and delivery problem. In the original SIPP model, the path of dynamic obstacles are stored using a chronologically ordered list of the cells it occupies each timestep. These lists need to be iterated over to compute safe intervals, making SIPP computationally expensive in large environment with many agents. A reservation table tackles this problem by storing the previously computed safe intervals ordered by their lower bound for each cell. This table is then updated after each new constructed path and irrelevant intervals can be removed from the table to reduce the size. Adding time offsets to the intervals is also proposed in this research in order to account for different size agents. From an experiment performed by Ma et al. where the environment size, number of agents and tasks velocity were changed was found that the planning time required for 250 agents and 2000 tasks was less than 16 seconds. These low planning times show that the introduction of the reservation table is an useful addition to the SIPP algorithm.

Soft Conflict SIPP(SCIPP) has been developed by Cohen et al.[17] in order to be able to use SIPP as a low-level search for suboptimal CBS solvers such as ECBS. The previously discussed versions of SIPP are unable to deal with soft conflicts defined in the high-level and is therefore not suitable for focal search. As mentioned subsection 4.5.2, defining conflicts as soft creates the opportunity to return bounded suboptimal solutions instead of no feasible solution. Instead of an optimal solution, SCIPP outputs a w-suboptimal solution for a predetermined value of w which does not violate any hard constraints. Experiments using PR2 motion primitives in two different environments (see Figure 4.9b) returned the following results found in Figure 4.9a. From these results can be concluded that using SCIPP significantly improves the success rate as well as the average runtime with increasing number of agents. It can therefore be concluded that SCIPP is a suitable low-level search solver for suboptimal CBS solvers.



(a) Results using four different values for the suboptimality bounds w.



(b) Arena maps with 49x49 cells(top) and Den520d map with 256x256 cells(bottom).

Figure 4.9: Experimental results of average runtime and success rate of ECBS-CT(a) for two different environments(b). Retrieved from [17].

4.7. Real World Application of MAPF

All of the discussed MAPF solvers show that they work well within simulated environments, however they do not represent real world applications due to the assumptions made. Recent research focuses more on making these assumptions, found in section 4.1 more realistic. One area of MAPF solvers now focuses on developing adaptation which allow the solver to be become continuous. An example of this is SIPP which is applied to more and more existing solvers[3]. The second assumption made is that an agent occupies only a single vertex per timestep and each agent has the same size. It is clear that

this assumption is also not realistic. An improvement that has been proposed is adapting the conflict definition to also take into account when agents are close to each other, so not only when they occupy the same vertex. There is also a new adaptation of MAPF which uses large agents, which takes size of an agent into account[42].

The third assumption states that each agent can perform one action each timestep which are five options; moving North, South, East, West, or wait. The length of each edge is also uniform within classical MAPF problems. There is research which focuses on weighing edges and including travel times, however this is not very pressing to be implemented when a model uses a grid-based environment. As the aircraft ramp is best modeled using a grid-based approach, this research will not be elaborated upon. Kinemetics are assumed to be neglected, meaning that agents can accelerate, decellerate and change direction instantly. Many research focuses on including kinemetic constraints in existing MAPF solvers, however this is out of the scope of this research. Finally, classic solvers also do not account for possible delays or other uncertainties. Airport operations are known to be very prone to delays due to delayed flight or many other causes and research shows that delays are mostly taken into account by adding buffers to time constraints or travel times.

4.8. Trade-off

Now that an elaborate overview has been created of different types of MAPF solvers, a comparison can be performed to decide which of these solvers would fit best to the path planning of autonomous GSE vehicles. As explained in previous sections, it is decided that only suboptimal solvers are considered for this application. The solvers considered are CO-WHCA*, CO-WHCA*P, bounded horizon(BH) ECBS, ECBS+HWY, CBSwP, and PBS. The low-level solver considered in this trade-off for the all CBS solvers is SIPP. There are two categories considered in the trade-off; Implementation and performance. Each of these categories has multiple subcategories which are described below.

Implementation

This category evaluates the level of difficulty of the implementation of the algorithm. Three criteria are considered: complexity, availability and feasibility. The complexity of the algorithm relates to the required amount of programming, variables and steps. Complex algorithms requires more complex programming become less understandable and reproducible. Moreover, higher complexity increases the risk of not completing the research within the given nine-month timeframe. Availability refers to both online and offline resources and tools that can be of assistance with the implementation. Online support includes literature, sample codes, and online assistance, while offline support includes the knowledge and assistance available within the research group. Feasibility refers to the applicability of the algorithm to the considered situation, in this case, the AAS layout represented by a grid-based graph that offer limited diversion possibilities. As such, algorithms that are efficient in open spaces or require extensive space for diversion are less suitable for this research.

Performance

Scalability is a measure of the algorithm's ability to handle an increased number of agents with only minor increase in computation time. Ideally, there should be a linear relationship between the number of agents and computation time, whereas an exponential or logarithmic relationship is less desirable. Optimality refers to the algorithm's ability to identify optimal or near-optimal solutions. Success rate denotes the number of agents that can be effectively coordinated without conflicts to reach their goal node within the designated timespan. To enable a proper comparison of success rates across algorithms, a dense environment with numerous corridors is preferred if possible. Lastly, completeness refers to the ability of finding a solution if one exists.

Table 4.1: Trade-off Multi-Agent Path Finding Solvers

	CO-WHCA*	CO-WHCA*P	BH ECBS	ECBS+HWY	CBSwP	PBS
Implementation						
Complexity	4	3	3	4	2	3
Availability	3	3	4	4	3	4
Feasibility	4	4	3	3	4	5
Performance						
Optimality	3	3	4	4	5	4
Scalability	4	3	4	4	4	4
Success rate	4	5	3	4	4	5
Completeness	2	2	5	5	5	5
Total Score	24	23	26	28	27	30

In this research, the environment is divided in two sections; the road between the depot and aircraft ramps and the aircraft ramp. Both sections are grid based but a different path finding solver might be more suitable for each section. The total scores in Table 4.1 show that each of the considered solvers could be implemented in this research as they deviate slightly from each other. However, Priority Based Search is the solver that scores the highest, due to the high performance it shows in literature. ECBS with highways also scores high with regards to performance and is deemed to be less complex than PBS due to the highways reducing the search space. Implementing highways could be feasible for the former section as it consists primarily of a perimeter road which could be modeled as a highway with different exits for every aircraft ramp. Therefore a combination of PBS and highways is an interesting option to explore.

As mentioned in subsection 4.6.1, combining bounded horizon with PBS is also an option to implement in order to improve the scalability and decrease the computational costs. Using bounded-horizon is especially useful with highly coupled paths which is the case for GSE entering and exiting the aircraft ramp. So using bounded horizon PBS in this section might be worth implementing. PBS based solvers require a low level path finding algorithm as well and for this SIPP might be an interesting option or the less complex version WHCA* which uses priority in order to resolve conflicts as well which is realistic for GSE vehicle conflicts. Furthermore, using SIPP instead of a A* solver as the low-level solver will also improve the scalability and computational costs and therefore bounded-horizon may not be required. To conclude, a combination of PBS with SIPP as a low level solver will be implemented with an option to extend the model by creating highways or using bounded horizon.

5 Coupled Task Allocation and Path Finding

Previous chapters have elaborated on the different types of path finding solvers and task allocation techniques which could be adapted for the autonomous aircraft ground handling environment. However, even if both task allocation and path finding algorithms ensure optimality, it may lead to an overall suboptimal solution. For example, if an optimal task allocation is used as input for the path finding algorithm, it could lead to suboptimal paths due to arising conflicts which would result in an overall suboptimal solution. A solution for this suboptimality is coupling the task allocation and path finding algorithms resulting in optimality and increased overall performance.

When task allocation and path finding are coupled, agents can be assigned tasks that are close to their current location, reducing travel time and increasing the speed of task completion. Additionally, agents can be assigned tasks that are compatible with their individual capabilities, which can result in better overall performance and higher task completion rates. Furthermore, by considering the tasks and paths of multiple agents simultaneously, the system can avoid conflicts and collisions, optimize the use of resources, and ensure that tasks are completed within their respective time windows. This is especially important in complex and dynamic environments, making it a desirable approach for many applications, such as robotics, transportation, and logistics.

Various researchers have developed coupled solvers by combining known task allocation and path finding algorithms. The most interesting type of solver within this domain is the combination of a task allocation algorithm with a multi-agent pick-up and delivery problem(MAPD) solver which is a lifelong version of one-shot MAPF problems. In this chapter are multiple coupled solvers presented, starting with the CBS-TA solver in section 5.1, which couples task allocation within the existing CBS framework.

5.1. Conflict Based Search - Task Allocation

In order to create a coupled solver, Hönig et al.[29] included a task allocation mechanism within the existing CBS framework (see subsection 4.3.3). Instead of generating all possible assignments and choosing the best generated path, a new assignment is generated on demand, decreasing the search space significantly and making this approach feasible. Fortunately, only the high level of the CBS solver needs to be adapted to make it a coupled approach. Instead of one search tree, a search forest is created which only creates new nodes on demand. Both CBS and CBS-TA start with a single root node which contains the optimal task assignment, however, when one or multiple conflicts are found, not only the root node is expanded but a second root node is created which contains the second-best task assignment. The difference between the two search methods is visualized in Figure 5.1.

Computing a next best task assignment on demand works as follows. A cost matrix C is computed for the best assignment when the solver is initialized. This cost matrix will be duplicated and the costs for the current solution will be set to infinity, to ensure that a new solution will be found. Then the optimal assignment algorithm will be executed which return a next best solution. In this research, the Hungarian Method is chosen as optimal assignment algorithm. This assignment algorithm can be

instructed to forbid or force assignments in order to speed up the process based on the finding of the previous search tree.

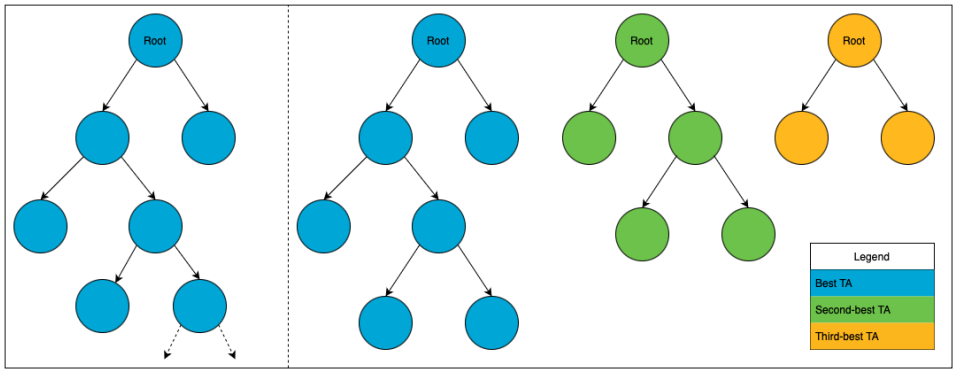


Figure 5.1: A search tree(left) versus a search forest(right) used in CBS and CBS-TA respectively.

A proof of optimality and completeness is provided with the objective function being the sum-of-cost as well as an example of the steps taken by the algorithm. This example is shown in Figure 5.2 and shows a small environment containing two agents which need to move from their start position towards their goal. The model start with generating the best assignment of tasks to agents. This output is then used for the root node of the CBS framework. A conflict is found at $t = 2$ seconds which results in the generation of two next-best assignments (step 3). Here the child with the lowest total cost is chosen as the second root node for the CBS search forest (step 4). This results in three nodes with the same cost and all of them contain a conflict. The tie breaker in this algorithm is FIFO, which results in the decision to expand the second root node in the search fores (step 6). Both children also found conflicts, which results in the decision to generate a new next best assignment. This new assignment is found to have a higher cost than the previously generated child, cost of seven and cost of six respectively. The assignment with a cost of six (from step 3) is now chosen for the third root node. There are again three nodes with equal costs and using the tie breaker strategy, two children are generated of the second root node (step 9). Again, conflicts are found in both nodes and the algorithm moves to the root node of the third search tree (step 10). The paths within this node contain no conflicts and is the final solution of this problem.

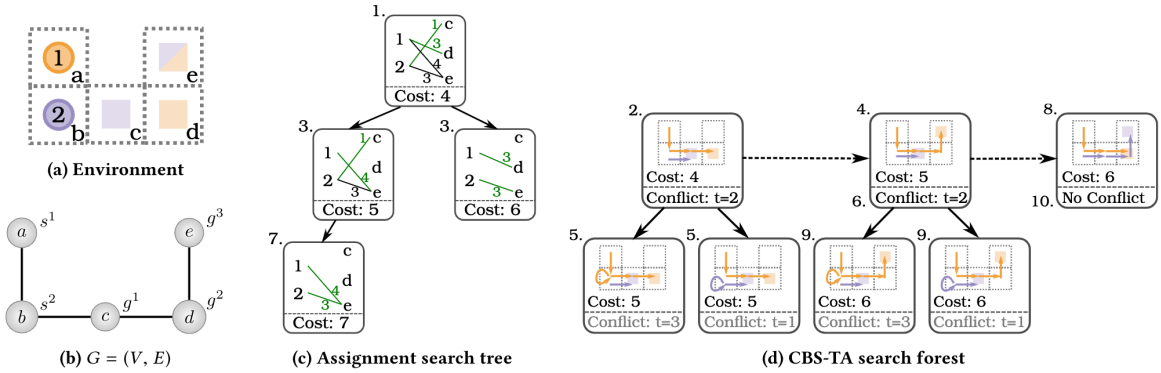


Figure 5.2: Example execution of CBS-TA. The environment (a) can be represented as a graph (b) with some vertices being a start and/or goal vertice. The assignment search tree and path finding search forest are visualized in (c) and (d) respectively. The numbers show in which order steps are taken. Retrieved from [29].

The same paper also presents an extension to their framework which uses ECBS instead of CBS. ECBS uses focal search in both levels resulting in including only solutions which are bounded by a single suboptimality factor w. This decreases the size of the search tree and forest significantly, improving the computational time of the solver. To see if these proposed solvers perform better than the decoupled solvers, benchmark experiments have been conducted. In Figure 5.3 is CBS-TA compared to the

decoupled TA and CBS as well as ILP. From this figure can be seen that CBS-TA has a higher success rate with increasing amount of agents, returns lower costs and has a lower runtime than the decoupled solver.

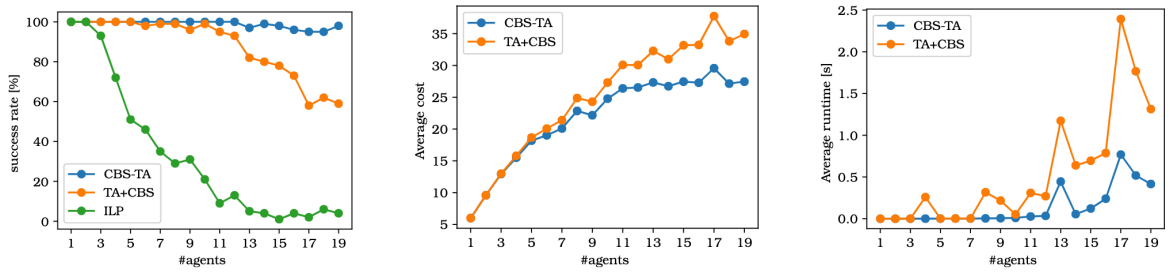


Figure 5.3: Benchmark results comparing CBS-TA with task assignment followed by decoupled TA and CBS and IPL. Each data point summarizes 100 randomly generated 8 X 8 4-connected grids with random start and goal locations.

In larger environment is ECBS-TA a more suitable variant as it runs significantly faster. For a benchmark with a 32x32 4-connected grid environment outperforms ECBS-TA in cost and runtime Conflict-Based Min-Cost-Flow(CBM) with higher amounts of agents. A trade-off needs to be made with respect to the choice of suboptimal bound, the higher the bound the lower the runtime. Due to the performance of ECBS-TA in large environments, it is an interesting approach for the handling of autonomous GSE vehicles.

5.2. Lifelong Relative Regret-Based Marginal-Cost Based Task Assignment

Chen et al.[14] propose a coupled MAPD solver which uses a marginal-cost assignment heuristic and a Large Neighborhood Search(LNS)-based meta-heuristic improvement strategy. A lifelong version is created using a Regret-based Marginal-cost based Task Assignment(RMTA) algorithm. The decision of which agent will take the next task is determined by comparing the marginal cost of adding the task to the best and next best robot. The robot with the lowest marginal cost is then chosen and the task is included in the route of this robot. The steps taken in this solver are visualized in Figure 5.4.

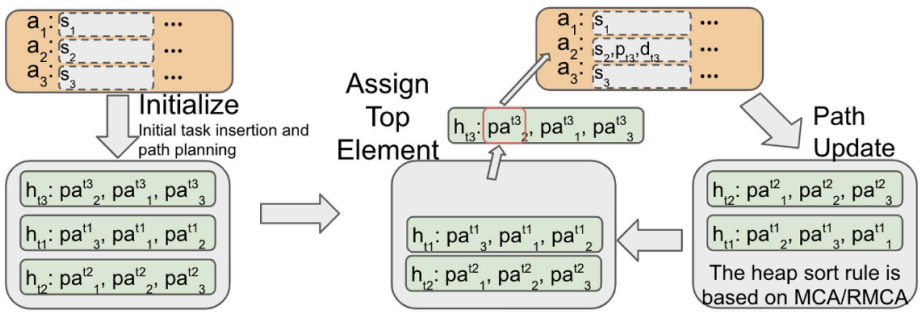


Figure 5.4: The flowchart of MCA/RMCA for assigning three tasks/packages t_1,t_2,t_3 to three robots 1,2,3. The gray box is priority heap $\mathcal{H}$, greenbox is potential assignment heap h, orange box is current assignment set $\mathcal{A}$,dashed border box is ordered action sequence o_i for each robot i,s_i is is initial location, and p_{t3} and d_{t3} are respectively the pick-up and destination locationof task t_3. Retrieved from [14].

The performance of this model is compared to the decoupled algorithm Token Passing with Task Swaps(TPTS) and the coupled algorithm called CENTRAL developed by Ma et al.[47]. The experiments exist of a grid presentation of a warehouse with 500 tasks that are made available at different timesteps and have to be executed by a varying amount of agents. RCMA shows significant improvement compared to CENTRAL and TPTS in runtime, makespan and total travel delay. Due to the complexity and, the

difficulty of implementing requirements and constraints into the tasks, it is decided that this solver will not be suitable for ground handling at airports.

5.3. Task Allocation Prioritized and Hybrid

The third coupled approach that is presented is the solver developed by Lui et al.[44]. Within this paper are two versions presented; TA-Prioritized and TA-Hybrid and the difference is only in the path planning aspect of the solver. Both solvers first compute a set of task sequences, one for each agent using a special TSP. Afterwards, paths are planned according to the task sequence of the agent. The task allocation algorithm minimizes the maximum makespan of an agent as well as the sum of makespan of all agents. This special TSP is solved on a directed weighted graph for the defined objectives using the LKH-3 TSP solver and Hamiltonian cycles. Each task has a release time which can influence the execution time however collisions are not yet taken into account at this stage which results in possible deviations of actual execution time.

TA-Prioritized uses an improved version of prioritized planning where the order of paths planned is based on the execution times. The agent with the highest execution time has the highest priority in order to minimize the constraint for these agents which might result in lower actual execution times. The improvement of this solver is choosing the next agent to be planned only after the path of the agent is planned which results in decisions made based on the actual execution times. For TA-Hybrid is a MAPF-based path planning approach used which divides the agents into two groups; new task agents and free agents. The new task agents have a set goal location and for this group is ICBS used. Once the agent reaches its goal location, it becomes a free agent. A path from their current location to a pickup location needs to be planned but is prone to change as tasks might swap in the meantime. A polynomial min-cost max-flow algorithm plans paths at every time step where the set of free agents has changed. Both solvers use reserving dummy paths as deadlock avoidance and in order to guarantee completeness.

TA-Hybrid shows in benchmark experiments in a large warehouse that it performs better than the CENTRAL algorithm as well as TA-Prioritized. With increasing amount of agents TA-Hybrid shows to return the lowest makespan values with reasonable increase in runtime compared to TA-Prioritized and the makespan values are only slightly higher than the optimal makespan returned by the TSP. TA-Hybrid is an interesting option to explore as it shows the possibility of coupling a large optimization algorithm with a path finding algorithm, has a relatively low runtime and is scalable.

5.4. Final Remarks on coupled Task Allocation and Path Finding

There are four coupled algorithms presented which each uses a different combination of task allocation and path finding algorithms. An overview is presented in Table 5.1 from which can be concluded that many different combinations between task allocation and path finding algorithms is possible. The TA Prioritized and TA-Hybrid use a combination with an optimization based TA algorithm which is most closely related to this research and will therefore be used as starting point for developing the model in the remainder of the thesis.

Table 5.1: Comparison algorithms used in coupled approaches.

Coupled Algorithm	Task Allocation	Path Planning
CBS-TA	Hungarian Method	CBS
RMTA	Regret-based Marginal Cost Assignment	A*
TA-Prioritized	TSP	Prioritized Planning
TA-Hybrid	TSP	ICBS

6

Research Proposal

With the collected information of the previous chapters, a research objective together with more detailed research questions can be constructed. Chapter 2 provides an overview of all the operational tasks that take place on and around an aircraft at the gate. The different type of GSE vehicles and the location where they perform their tasks is also discussed. Together with a detailed description of the current infrastructure of Amsterdam Airport Schiphol an outline of the environment, tasks and types of agents have been determined. Next are multi-agent task allocation techniques introduced and analyzed in chapter 3 to determine which technique fits the problem best. Following this chapter are first in chapter 4 various types of multi-agent pathfinding approaches discussed and from an evaluation is one approach chosen. Finally, another domain which combines task allocation and path finding in one framework is explored in chapter 5.

Before a research objective can be determined, it is needed to establish the research gap this research aims to close. Section section 6.1 describes this gap based on the gathered knowledge in the previous chapters. The next step is to determine the research objective and this objective is presented in section 6.2 together with a main research question and supporting research questions. Finally, the methodology for the research following from this literature study is presented in section 6.3 together with the planning.

6.1. Research Gap

In order to develop a research proposal, it is important to identify the gaps within the available research. The gap within modeling autonomous GSE vehicles can be identified using the information of the previous chapters, chapter 4, 3, and 5 respectively. Within the field of multi-agent path finding are many approaches developed with the goal of creating the fastest, most efficient and highest quality solution for a predefined environment. Some approaches are optimized for warehouse environment and others for large open spaces, but there has not been any research which uses the airside of an airport as their environment. There are a few aviation related applications found in literature but these focus on aircraft taxiing[22] and not on aircraft ground operations. There are currently one study by Chen et al.[13] that developed a decoupled task allocation and path finding framework for autonomous GSE vehicles. However, this study concluded that the model is not performing as expected and that other approaches are advised to be explored.

Within the multi-agent task allocation are many techniques developed which focus mostly on warehouse application in which there exists only one type of agent which can execute each available task. In this research however, there are multiple types of GSE vehicles which need to be assigned to the corresponding type of task. Additionally are there also multiple precedence constraints and time windows that need to be taken into account as the turnaround schedules of aircraft at the ramp are very tight. Developing a task allocation mechanism which fits these requirements and creating the possibility to couple the MAPF model to the task allocation is a gap in research that this thesis will attempt to fill in addition to the application.

6.2. Research Objective and Research Questions

The goal of this literature is to define a method which can be created to fill the aforementioned research gap. This goal is written down in a research objective below:

> *"Design and evaluate a coupled task allocation and path planning algorithm for autonomous GSE vehicles at Amsterdam Airport Schiphol using a multi-agent based modeling approach."*

With the research objective known, it is important to formulate a main research question which further defines the scope of the research. The main research question for the thesis is:

> *"What is the effectiveness of a coupled multi-agent task allocation and path planning approach for autonomous GSE vehicles performing aircraft ground services, in comparison to a decoupled approach, in terms of achieving efficient task allocation and path planning?"*

Supporting research questions are formulated which serve as support for answering the main research question. These questions can be related to the following aspects of the research; development and performance of the (de)coupled models.

Development of models

1. Which heuristic is most suitable for calculating the initial travel times?
2. What objective(s) are most suitable for the task allocation of GSE vehicles?
3. What type of constraints need to be defined for each type of GSE vehicle?
4. How should the task allocation and path planning be coupled?
   (a) Which coupling approach suits the decoupled models best?

Performance of models

1. Which Key Performance Indicators can be defined in order to measure the effectiveness of the coupled method in comparison to the decoupled method?
   (a) What is the effect on the KPIs of coupling task allocation with path planning?
   (b) What is the effect of changing parameters such as the amount of available GSE vehicles on the performance of the models?
2. Which criteria can be defined to evaluate the necessity of activating the coupling mechanism?
   (a) What is the effect of changing the criteria on the KPIs of the models?

6.3. Methodology

This section starts with presenting the scope of the research in subsection 6.3.1 including the environment in which the model will be tested and the assumptions made. The modeling approach is discussed in subsection 6.3.2. Then, subsection 6.3.3 describes the steps that need to be taken for the remainder of the thesis in order to fulfill the research objective as well as a timeline in which this will be achieved.

6.3.1. Research Scope

In order to scale the research gap to a reasonable research scope, it is needed to make assumptions with regards to ground handling activities and the environment AAS. A few assumptions have already been defined in chapter 2 which are restated in this section. Assumptions regarding multi-agent path finding can be found in section 4.1 and for multi-agent task allocation in section 3.3.

Assumptions Ground Handling Activities

The first assumption is that within the model only the eight activities listed below are considered. All other activities are assumed to be executed without any delays or disruptions and seasonal activities such as deicing are neglected. Furthermore, it is also assumed that each GSE vehicle can be shared with all the aircraft ramps and refueling/recharging of GSE vehicles is not taken into account.

- Galley service
- Portable water service
- Cargo/baggage loading and unloading
- Fueling
- Cleaning service
- Lavatory service
- (Dis)Connecting Passenger Boarding Bridge

Additionally, assumptions need to be made with respect to the aircraft stands and service roads of AAS. First, as discussed in chapter 2 only pier B of Amsterdam Airport Schiphol is considered in this simulation. This pier mainly serves short-haul aircraft which have tight schedules and therefore only ground handling operations of short-haul aircraft is considered. The most frequently parked type of aircraft at pier B is a Boeing 737 which is why the model will only include this type of aircraft and its servicing locations. Each aircraft ramp is assumed to have one PBB which passengers use to plane and deplane the aircraft and each ramp has access to the underground fuel system. Aircraft are also assumed to be stationary in the ramp meaning that towing and marshalling is not considered. Finally, due to safety reasons refueling cannot occur when people are onboard of the aircraft and GSE vehicles have designated entry and exit locations at the aircraft ramp.

Assumptions Task Allocation and Path Planning

From previous chapters can be concluded that various types of algorithms are developed for different environments and assumptions. In this research, the following characteristics are to be assumed in the algorithms.

- Heterogeneous fleet
- Heterogeneous capacities
- More tasks than agents
- Priority for tasks/agents
- Release time of task
- Task execution time window

For task allocation, each type task is assumed to have the same duration and these are estimated using data from Boeing, see subsection 2.1.1.

6.3.2. Modeling Approach

In addition to the research scope is it also possible to present the preliminary modeling approach of the remainder of this thesis. An schematic overview of the model can be found in Figure 6.1. Both the coupled and decoupled model start with calculating initial travel distances for each task using a to be determined heuristic. These travel distances together with the other inputs will initiate the Pickup and Delivery Problem. The output of the PDP is a schedule for each GSE vehicle serving all the tasks within the time windows. This schedule then serves as an input for planning collision free paths for all GSE vehicles together with the simulation environment and priority between GSE vehicles.

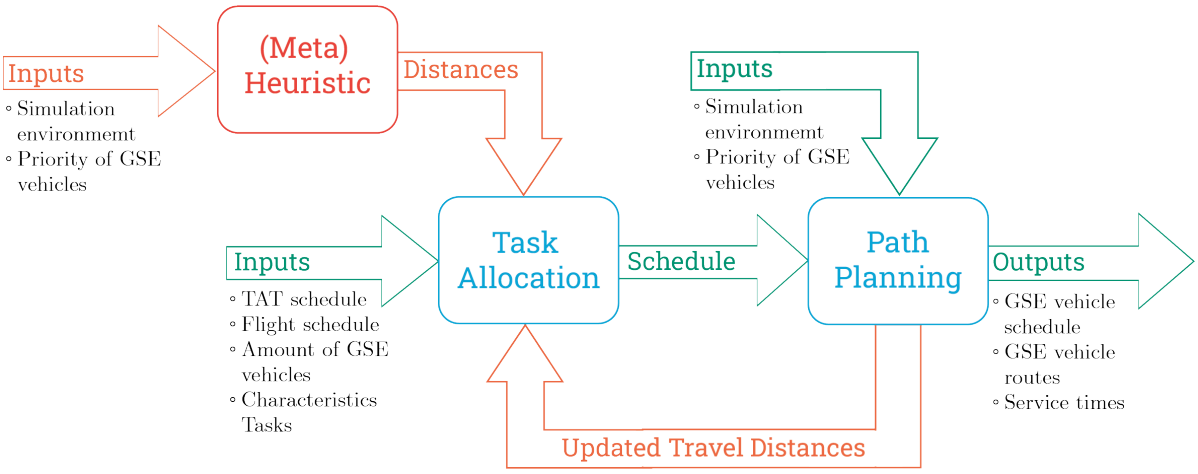


Figure 6.1: Flow chart of the (de)coupled models.

From this point onward will the coupled and decoupled model differ. The decoupled method will return the collision free paths with the corresponding service times, however the coupled method will have

an extra step. If the newly calculated paths will results in significant changes in the service times and therefore in the schedules, a decision can be made to initiate the task allocation mechanism once again to find a better solution with a different schedule. This process will terminate when the quality of the schedule, routes and service times is high enough. The requirements for this need to be determined at a later stage of the thesis.

6.3.3. Research Planning

This section will present a preliminary planning for the remainder of this thesis containing the modules that need to be developed. This thesis will use the simulation environment of Chen et al.[13] as a starting point of the model. This environment is later adapted to fit seamlessly with the newly developed model. The modules that need to be completed in order to develop a coupled task allocation and path planning model for autonomous GSE vehicles are summarized in Table 6.1 including the estimated time needed to complete it. Additionally, for project management purposes a Gantt Chart is developed which includes the inter-dependencies between the modules presented in Table 6.1 and is shown in Figure 6.2.

Table 6.1: Planning Thesis.

Module	Goal of module	Duration (weeks)
1. Orientation and Conceptualization	Familiarizing with the existing model in order to establish integration of to be developed models with simulation environment.	2
2. Development of Task Allocation Model	Develop a PDP algorithm which is in line with all requirements and assumptions.	10
3. Development of Path Planning Model	Develop a path planning algorithm which plans collision free paths for all agents.	7
4. Development of Coupled Model	Incorporate a coupling method for combining (2) and (3) into one model.	5
5. Development of Extensions to the Model	Develop possible extensions such as delays into the existing models and reallocation of tasks.	3
6. Simulation	Run multiple scenarios with different parameters for decoupled and coupled method as well as for a benchmark model.	3
7. Analysis	Analyze the results of all three models and perform a sensitivity analysis.	4
8. Finalizing	Document everything into a paper and prepare defense.	10

MSc. Thesis Planning

Coupled Task Allocation and Path Planning for automated Aircraft Ground Handling at Amsterdam Airport Schiphol

Manouk van der Zwan (STUDENT NAME) | 12-Sep-22 (START DATE) | 27-Nov-23 (END DATE)

Task ID	Task Name	Type	Start Date	End Date	Planned Duration (In Weeks)
1.	**LITERATURE STUDY PART 1**	Task	12-Sep-22	2-Dec-22	12
a	Context, developments and modelling of ground services	Sub-task	12-Sep-22	7-Oct-22	4
c	Research path planning algorithms	Sub-task	26-Sep-22	28-Oct-22	5
c	Research task allocation algortihms	Sub-task	10-Oct-22	11-Nov-22	5
d	Research coupled task allocation and path planning algortihms	Sub-task	24-Oct-22	11-Nov-22	3
b	Define research gap	Milestone	14-Nov-22	14-Nov-22	0
e	Draft research objective and questions	Sub-task	14-Nov-22	25-Nov-22	2
g	In-depth research path planning algorithms	Sub-task	14-Nov-22	2-Dec-22	3
f	Trade-off path planning algorithms	Milestone	28-Nov-22	28-Nov-22	0
i	HOLIDAY	Holiday	5-Dec-22	2-Jan-23	5
2.	**LITERATURE PART 2 + KICK-OFF + RESEARCH METHODOLOGIES**	Task	9-Jan-23	14-Apr-23	14
b.	In-depth research task allocation algortihms	Sub-task	9-Jan-23	27-Jan-23	3
g	Define methodology thesis	Sub-task	6-Feb-23	3-Mar-23	4
a.	HOLIDAY	Holiday	23-Jan-23	27-Jan-23	1
h	Trade-off task allocation algorithms	Milestone	30-Jan-23	30-Jan-23	0
c.	Writing literature study	Sub-task	30-Jan-23	31-Mar-23	9
d.	HOLIDAY	Holiday	4-Mar-23	9-Mar-23	1
e.	Hand-in draft report literature study	Milestone	27-Mar-23	27-Mar-23	0
g.	Submit TEP form	Deliverable	27-Mar-23	27-Mar-23	0
f.	Research methodologies	Sub-task	27-Mar-23	7-Apr-23	2
f.	Prepare Kick-off presentation	Sub-task	27-Mar-23	7-Apr-23	2
h.	Hand-in final assignment research methodologies	Deliverable	14-Apr-23	14-Apr-23	0
g.	Give Kick-off presenation	Milestone	14-Apr-23	14-Apr-23	0
3.	**PART 1 RESEARCH**	Task	17-Apr-23	21-Jul-23	14
b.	Develop task allocation algorithm	Sub-task	17-Apr-23	2-Jun-23	7
e.	Verification of task allocation algorithm	Sub-task	1-May-23	2-Jun-23	5
e.	Task Allocation algorithm done	Milestone	5-Jun-23	5-Jun-23	0
c.	Develop path planning algorithm	Sub-task	5-Jun-23	7-Jul-23	5
e.	Verification of path planning algorithm	Sub-task	19-Jun-23	7-Jul-23	3
e.	Path planning algorithm done	Milestone	10-Jul-23	10-Jul-23	0
d.	Possible delay/extra time to program both algorithms	Sub-task	10-Jul-23	21-Jul-23	2
e.	Prepare midterm presentation	Sub-task	10-Jul-23	21-Jul-23	2
f.	Give midterm presentation	Deliverable	17-Jul-23	17-Jul-23	0
4.	**PART 2 RESEARCH**	Task	21-Jul-23	12-Oct-23	12
a.	Familiarise with Simulation model	Sub-task	24-Jul-23	4-Aug-23	2
a	Coupled task allocation and path planning algorithm	Sub-task	24-Jul-23	25-Aug-23	5
e.	Path planning algorithm done	Milestone	28-Aug-23	28-Aug-23	0
b	Writing thesis paper	Sub-task	7-Aug-23	29-Sep-23	8
a.	HOLIDAY	Holiday	28-Aug-23	1-Sep-23	1
c	Validation/Analysis of model	Sub-task	28-Aug-23	29-Sep-23	5
a	Hand-in draft report thesis	Deliverable	2-Oct-23	2-Oct-23	0
b	Prepare green light presentation	Sub-task	2-Oct-23	13-Oct-23	2
c	Give green light presentation	Milestone	9-Oct-23	9-Oct-23	0
5.	**THESIS DEFENSE**	Task	12-Oct-23	8-Nov-23	4
a.	HOLIDAY	Holiday	12-Oct-23	18-Oct-23	1
b.	Prepare thesis defense	Sub-task	19-Oct-23	8-Nov-23	3
c.	Perform thesis defense	Deliverable	8-Nov-23	8-Nov-23	0

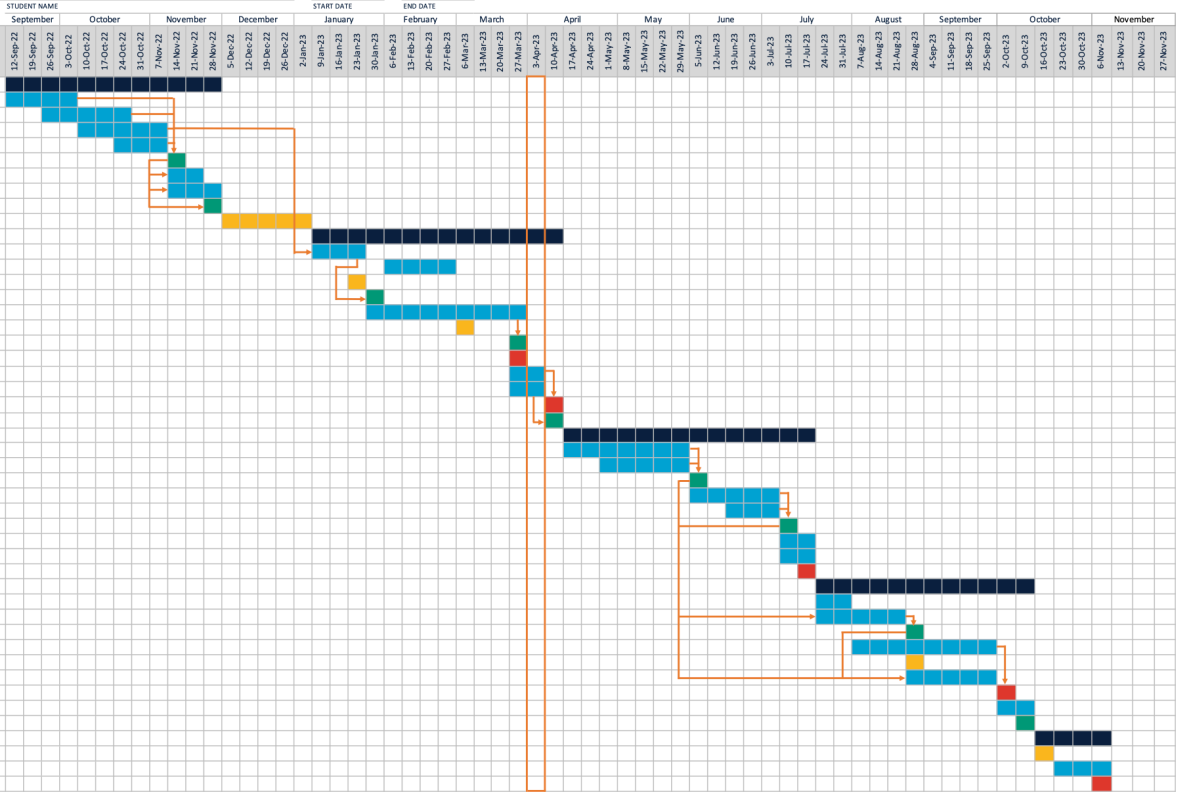


Figure 6.2: Gantt Chart Thesis 'Coupled Task Allocation and Path Planning'.

Bibliography

[1] Gaby Allaart and Joanne Kaijen. Aircraft Stand Table. Amsterdam Airport Schiphol, Schiphol, The Netherlands, 2022.

[2] Giovanni Andreatta, Luigi De Giovanni, and Michele Monaci. A Fast Heuristic for Airport Ground-Service EquipmentandStaff Allocation. Procedia - Social and Behavioral Sciences, 108:26–36, 1 2014. ISSN 18770428. doi: 10.1016/j.sbspro.2013.12.817.

[3] Anton Andreychuk, Konstantin Yakovlev, Dor Atzmon, and Roni Sternr. Multi-agent pathfinding with continuous time. In IJCAI International Joint Conference on Artificial Intelligence, volume 2019-August, 2019. doi: 10.24963/ijcai.2019/6.

[4] Aviation Learnings Team. A guide to airport ramp operations, ground handling & ground support equipment(gse), 2020. https://aviationlearnings.com/ground-handling-ramp-operations-aircraft-ground-support-equipment-gse-\machines-that-supplement-the-airplane/#Common_Terminologies.

[5] Mohamed Badreldin, Ahmed Hussein, and Alaa Khamis. A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation. Advances in Artificial Intelligence, 2013:1–11, 11 2013. ISSN 1687-7470. doi: 10.1155/2013/256524.

[6] Max Barer, Guni Sharon, Roni Stern, and Ariel Felner. Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem. In Proceedings of the 7th Annual Symposium on Combinatorial Search, SoCS 2014, volume 2014-January, 2014. doi: 10.1609/socs.v5i1.18315.

[7] Ali Bazghandi. Techniques, Advantages and Problems of Agent Based Modeling for Traffic Simulation. International Journal of Computer Science Issues, 9(1), 2012.

[8] Nicola Bianchessi and Giovanni Righini. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. Computers and Operations Research, 34(2):578–594, 2 2007. ISSN 03050548. doi: 10.1016/j.cor.2005.03.014.

[9] Zahy Bnaya and Ariel Felner. Conflict-oriented windowed hierarchical cooperative. In Proceedings - IEEE International Conference on Robotics and Automation, 2014. doi: 10.1109/ICRA.2014.6907401.

[10] Boeing Commercial Airplanes. 737 max airplane characteristics for airport planning. Technical report, Boeing, 2022. Document number: D638A004.

[11] Eli Boyarski, Ariel Felner, Roni Stern, Guni Sharon, David Tolpin, Oded Betzalel, and Eyal Shimony. ICBS: Improved conflict-based search algorithm for multi-agent pathfinding. In IJCAI International Joint Conference on Artificial Intelligence, volume 2015-January, 2015.

[12] David Bredström and Mikael Rönnqvist. Combined vehicle routing and scheduling with temporal precedence and synchronization constraints. European Journal of Operational Research, 191(1), 2008. ISSN 03772217. doi: 10.1016/j.ejor.2007.07.033.

[13] Szu-Tung Chen, Alexei Sharpanskykh, and Gülcin Ermis. Multi-agent Planning and Coordination for Automated Aircraft Ground Handling. Master's thesis, Delft University of Technology, 2022.

[14] Zhe Chen, Javier Alonso-Mora, Xiaoshan Bai, Daniel D. Harabor, and Peter J. Stuckey. Integrated Task Assignment and Path Planning for Capacitated Multi-Agent Pickup and Delivery. IEEE Robotics and Automation Letters, 6(3), 2021. ISSN 23773766. doi: 10.1109/LRA.2021.3074883.

[15] Sam Chui. Amsterdam air traffic control visit, 2020. https://samchui.com/2020/01/20/amsterdam-air-traffic-control-visit/#.Y2t9iS8w30p.

[16] Liron Cohen, Tansel Uras, and Sven Koenig. Feasibility Study: Using Highways for Bounded-Suboptimal Multi-Agent Path Finding. In Proceedings of the 8th Annual Symposium on Combinatorial Search, SoCS 2015, volume 2015-January, 2015. doi: 10.1609/socs.v6i1.18363.

[17] Liron Cohen, Tansel Uras, T. K. Satish Kumar, and Sven Koenig. Optimal and bounded-suboptimal multi-agent motion planning. In Proceedings of the 12th International Symposium on Combinatorial Search, SoCS 2019, 2019.

[18] Antonello Contini and Alessandro Farinelli. Coordination approaches for multi-item pickup and delivery in logistic scenarios[Formula presented]. Robotics and Autonomous Systems, 146, 2021. ISSN 09218890. doi: 10.1016/j.robot.2021.103871.

[19] Anders Dohn, Matias Sevel Rasmussen, and Jesper Larsen. The vehicle routing problem with time windows and temporal dependencies. In Networks, volume 58, 2011. doi: 10.1002/net.20472.

[20] EUROCONTROL Airport CDM Team. Airport CDM Implementation. EUROCONTROL, Brussels, Belgium, 5.0 edition, 2017.

[21] Ariel Felner, Roni Stern, Solomon Eyal Shimony, Eli Boyarski, Meir Goldenberg, Guni Sharon, Nathan Sturtevant, Glenn Wagner, and Pavel Surynek. Search-based optimal solvers for the multi-agent pathfinding problem: Summary and challenges. In Proceedings of the 10th Annual Symposium on Combinatorial Search, SoCS 2017, volume 2017-January, 2017.

[22] Konstantine Fines, Alexei Sharpanskykh, and Matthieu Vert. Agent-based distributed planning and coordination for resilient airport surface movement operations. Aerospace, 7(4), 2020. ISSN 22264310. doi: 10.3390/aerospace7040048.

[23] Brian P. Gerkey and Maja J. Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. International Journal of Robotics Research, 23(9), 2004. ISSN 02783649. doi: 10.1177/0278364904045564.

[24] Meir Goldenberg, Ariel Felner, Roni Stern, Guni Sharon, Nathan Sturtevant, Robert C. Holte, and Jonathan Schaeffer. Enhanced partial expansion A*. Journal of Artificial Intelligence Research, 50, 2014. ISSN 10769757. doi: 10.1613/jair.4171.

[25] Matthew C. Gombolay, Ronald J. Wilcox, and Julie A. Shah. Fast Scheduling of Robot Teams Performing Tasks with Temporospatial Constraints. IEEE Transactions on Robotics, 34(1), 2018. ISSN 15523098. doi: 10.1109/TRO.2018.2795034.

[26] GSE Solutions. Aviation ground support equipment: brief history, 2018. https://www.aerotime.aero/articles/21568-aviation-ground-support-equipment-brief-history.

[27] Daniel Guimarans and Silvia Padrón. A stochastic approach for planning airport ground support resources. International Transactions in Operational Research, 29(6), 2022. ISSN 14753995. doi: 10.1111/itor.13104.

[28] Bradford Heap and Maurice Pagnucco. Sequential single-cluster auctions for robot task allocation. In AI 2011: Advances in Artificial Intelligence: 24th Australasian Joint Conference, Perth, Australia, December 5-8, 2011. Proceedings 24, pages 412–421. Springer, 2011.

[29] Wolfgang Honig, Scott Kiesel, Andrew Tinka, Joseph W. Durham, and Nora Ayanian. Conflict-based search with optimal task assignment. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, volume 1, pages 757–765, 2018.

[30] HSE Office. Schipholregels. Amsterdam Airport Schiphol, Schiphol, The Netherlands, 30.2 edition, 2022.

[31] HSE Office. Safety and Security Pocketguide. Amsterdam Airport Schiphol, Schiphol, The Netherlands, 2022.

[32] International Air Transport Association (IATA). Industry statistics fact sheet, 2019. https://www.iata.org/en/iata-repository/publications/economic-reports/fact-sheet-industry-statistics-dec19.pdf.

[33] International Air Transport Association (IATA). Air passenger numbers to recover in 2024, 2022. https://www.iata.org/en/pressroom/2022-releases/2022-03-01-01/.

[34] W. H. Ip, Vincent Cho, Nick Chung, and George Ho. A Multi Agent Based Model for Airport Service Planning. International Journal of Engineering Business Management, 2(2):93–100, 2010.

[35] KCM for Economy Class and Beyond and Kevincm. A brace of klm boeing 737s operating short haul services from amsterdam schiphol, 2018. https://economyclassandbeyond.boardingarea.com/2018/07/14/train-vs-plane-a-bit-of-airplane-photography-at-amsterdam-schiphol\-airport-airplane-art-extra/img_3816-2/.

[36] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. Multi-robot task allocation: A review of the state-of-the-art. Studies in Computational Intelligence, 604, 2015. ISSN 1860949X. doi: 10.1007/978-3-319-18299-5{_}2.

[37] Sven Koenig, Pinar Keskinocak, and Craig Tovey. Progress on Agent Coordination with Cooperative Auctions*. In Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, pages 1713–1717. Association for the Advancement of Artificial Intelligence, 2010. www.aaai.org.

[38] G. Ayorkor Korsah, Anthony Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. International Journal of Robotics Research, 32(12), 2013. ISSN 02783649. doi: 10.1177/0278364913496484.

[39] Harold W. Kuhn. The Hungarian method for the assignment problem. In 50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art. 2010. doi: 10.1007/978-3-540-68279-0{_}2.

[40] Suresh Nanda Kumar and Ramasamy Panneerselvam. A Survey on the Vehicle Routing Problem and Its Variants. Intelligent Information Management, 04(03):66–74, 2012. ISSN 2160-5912. doi: 10.4236/iim.2012.43010.

[41] Michail G. Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton Kleywegt, Sven Koenig, Craig Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In Robotics: Science and Systems, volume 1, 2005. doi: 10.15607/rss.2005.i.045.

[42] Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma, T. K. Satish Kumar, and Sven Koenig. Multi-agent path finding for large agents. In 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, 2019. doi: 10.1609/socs.v10i1.18483.

[43] Jiaoyang Li, Andrew Tinka, Scott Kiesel, Joseph W. Durham, T. K. Satish Kumar, and Sven Koenig. Lifelong multi-agent path finding in large-scale warehouses. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, volume 2020-May, 2020.

[44] Minghua Liu, Hang Ma, Jiaoyang Li, and Sven Koenig. Task and Path Planning for Multi-Agent Pickup and Delivery. In 18th International Conference on Autonomous Agents and Multiagent Systems, 2019. www.ifaamas.org.

[45] Ran Liu, Xiaolan Xie, Vincent Augusto, and Carlos Rodriguez. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. European Journal of Operational Research, 230(3):475–486, 11 2013. ISSN 03772217. doi: 10.1016/j.ejor.2013.04.044.

[46] Hang Ma. Graph-Based Multi-Robot Path Finding and Planning. Current Robotics Reports, 3 (3):77–84, 6 2022. doi: 10.1007/s43154-022-00083-8.

[47] Hang Ma, T. K.Satish Kumar, Jiaoyang Li, and Sven Koenig. Lifelong multi-Agent path finding for misc pickup and delivery tasks. In Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, volume 2, 2017.

[48] Hang Ma, Glenn Wagner, Ariel Felner, Jiaoyang Li, T. K. Satish Kumar, and Sven Koenig. Multi-agent path finding with deadlines. In IJCAI International Joint Conference on Artificial Intelligence, volume 2018-July, 2018. doi: 10.24963/ijcai.2018/58.

[49] Hang Ma, Daniel Harabor, Peter J. Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, 2019.

[50] Hang Ma, Wolfgang Hönig, T. K. Satish Kumar, Nora Ayanian, and Sven Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, 2019.

[51] Seyedali Mirjalili. Evolutionary Algorithms and Neural Networks Theory and Applications, volume 780. Springer, Warsaw, 2019. http://www.springer.com/series/7092.

[52] Robert Morris, Corina S. Psreanu, Kasper Luckow, Waqar Malik, Hang Ma, T. K. Satish Kumar, and Sven Koenig. Planning, scheduling and monitoring for airport surface operations. In AAAI Workshop - Technical Report, volume WS-16-01 - WS-16-15, 2016.

[53] Dhouha Ben Noureddine, Atef Gharbi, and Samir Ben Ahmed. Multi-agent deep reinforcement learning for task allocation in dynamic environment. In ICSOFT 2017 - Proceedings of the 12th International Conference on Software Technologies, pages 17–26. SciTePress, 2017. ISBN 9789897582622. doi: 10.5220/0006393400170026.

[54] Ernesto Nunes and Maria Gini. Multi-Robot Auctions for Allocation of Tasks with Temporal Constraints. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, pages 2110–2116, 2015. www.aaai.org.

[55] Ernesto Nunes, Mitchell McIntire, and Maria Gini. Decentralized multi-robot allocation of tasks with temporal and precedence constraints. Advanced Robotics, 31(22), 2017. ISSN 15685535. doi: 10.1080/01691864.2017.1396922.

[56] Silvia Padrón and Daniel Guimarans. An Improved Method for Scheduling Aircraft Ground Handling Operations from a Global Perspective. Asia-Pacific Journal of Operational Research, 36(4), 2019. ISSN 02175959. doi: 10.1142/S0217595919500209.

[57] Silvia Padrón, Daniel Guimarans, Juan José Ramos, and Salma Fitouri-Trabelsi. A bi-objective approach for scheduling ground-handling vehicles in airports. Computers and Operations Research, 71, 2016. ISSN 03050548. doi: 10.1016/j.cor.2015.12.010.

[58] Sophie N. Parragh, Karl F. Doerner, and Richard F. Hartl. A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. Journal fur Betriebswirtschaft, 58(2), 2008. ISSN 03449327. doi: 10.1007/s11301-008-0036-4.

[59] Mike Phillips and Maxim Likhachev. SIPP: Safe interval path planning for dynamic environments. In Proceedings - IEEE International Conference on Robotics and Automation, 2011. doi: 10.1109/ICRA.2011.5980306.

[60] Sameera S Ponda. Robust Distributed Planning Strategies for Autonomous Multi-Agent Teams. ProQuest Dissertations and Theses, 0828990, 2012.

[61] Wei Qin, Yan-Ning Sun, Zi-Long Zhuang, Zhi-Yao Lu, and Yao-Ming Zhou. Multi-agent reinforcement learning-based dynamic task assignment for vehicles in urban transportation system. International Journal of Production Economics, 240:108251, 2021.

[62] Siddhanta Saggar, Maurizio Tomasella, Giovanni Cattaneo, and Andrea Matta. Enhanced operational management of airport ground support equipment for better aircraft turnaround performance. In Proceedings - Winter Simulation Conference, volume 2021-December. Institute of Electrical and Electronics Engineers Inc., 2021. ISBN 9781665433112. doi: 10.1109/WSC52266.2021.9715320.

[63] Schiphol Innovation. Predicting waiting times at security, 2020. https://www.schiphol.nl/en/innovation/blog/predicting-waiting-times-at-security/.

[64] Schiphol Innovation. Real-time data for smarter working: Smart vop 1.0, 2020. https://www.schiphol.nl/en/innovation/blog/smart-vop/.

[65] Schiphol Innovation. An autonomous airport in 2050, 2021. https://www.schiphol.nl/en/innovation/blog/an-autonomous-airport-in-2050/.

[66] Schiphol Innovation. High-tech travel at schiphol, 2022. https://www.schiphol.nl/en/innovation/blog/high-tech-travel-at-schiphol/.

[67] Schiphol Newsroom. 2 april 2020 - update: Schiphol zet volgende stap naar kern schiphol en sluit pieren, 2020. https://nieuws.schiphol.nl/schiphol-blijft-open-met-fors-kleinere-operatie/?

[68] Michael Schmidt. A review of aircraft turnaround operations and simulations. Progress in Aerospace Sciences, 92, 2017. ISSN 03760421. doi: 10.1016/j.paerosci.2017.05.002.

[69] Michael Schmidt, Philipp Nguyen, and Mirko Hornung. Novel aircraft ground operation concepts based on clustering of interfaces. In SAE Technical Papers, volume 2015-September, 2015. doi: 10.4271/2015-01-2401.

[70] C. Schrijver. Fnv: Weer chaos op schiphol op komst door personeelsgebrek, 2022. https://www.fnv.nl/nieuwsbericht/sectornieuws/vervoer/2022/04/weer-chaos-op-schiphol-op-komst-door-personeelsgeb.

[71] N. Seenu, R. M. Kuppan Chetty, M. M. Ramya, and Mukund Nilakantan Janardhanan. Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems. Industrial Robot, 47(6), 2020. ISSN 0143991X. doi: 10.1108/IR-04-2020-0073.

[72] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R. Sturtevant. Conflict-based search for optimal multi-agent pathfinding. Artificial Intelligence, 219, 2015. ISSN 00043702. doi: 10.1016/j.artint.2014.11.006.

[73] Kaveh Sheibani. Scheduling Aircraft Ground Handling Operations Under Uncertainty Using Critical Path Analysis and Monte Carlo Simulation. International Journal of Business Strategy and Automation, 1(1):37–45, 12 2019. ISSN 2644-2094. doi: 10.4018/ijbsa.2020010103.

[74] David Silver. Cooperative pathfinding. In Proceedings of the 1st Artificial Intelligence and Interactive Digital Entertainment Conference, AIIDE 2005, 2005.

[75] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. IEEE Transactions on Computers, C-29(12):1104–1113, 1980. doi: 10.1109/TC.1980.1675516.

[76] M Sol and M W P Savelsbergh. The General Pickup and Delivery Problem. Transportation Science, 29(1):17–29, 1995.

[77] Marius M. Solomon. ALGORITHMS FOR THE VEHICLE ROUTING AND SCHEDULING PROBLEMS WITH TIME WINDOW CONSTRAINTS. Operations Research, 35(2):254–265, 1987. ISSN 0030364X. doi: 10.1287/opre.35.2.254.

[78] Trevor Standley. Finding Optimal Solutions to Cooperative Pathfinding Problems. In Twenty-Fourth AAAI Conference on Artificial Intelligence, pages 173–178, 2010. www.aaai.org.

[79] Roni Stern. Multi-agent path finding An Overview. In Gennady S. Osipov, Aleksandr I. Panov, and Konstantin S. Yakolev, editors, Artificial Intelligence, volume 11866 LNAI, chapter 6, pages 96–115. Springer, Cham, 7 2019. doi: 10.1007/978-3-030-33274-7{_}6.

[80] Roni Stern, Nathan R. Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T. Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, T. K. Satish Kumar, Eli Boyarski, and Roman Barták. Multi-agent pathfinding: Definitions, variants, and benchmarks. In Proceedings of the 12th International Symposium on Combinatorial Search, SoCS 2019, 2019.

[81] Pavel Surynek. An Optimization Variant of Multi-Robot Path Planning is Intractable Introduction and Motivation. Technical report, Association for the Advancement of Artificial Intelligence, 2010. www.aaai.org.

[82] Pavel Surynek, Ariel Felner, Roni Stern, and Eli Boyarski. An Empirical Comparison of the Hardness of Multi-agent Path Finding under the Makespan and the Sum of Costs Objectives. Ninth Annual Symposium on Combinatorial Search, 7(1), 7 2016. www.aaai.org.

[83] Diego Alonso Tabares and Felix Mora-Camino. Aircraft ground handling: analysis for automation. In 17th AIAA Aviation Technology, Integration, and Operations Conference, page 3425, 2017.

[84] Diego Alonso Tabares and Felix Mora-Camino. Aircraft ground operations: steps towards automation. CEAS Aeronautical Journal, 10(3):965–974, 2019.

[85] Jiahai Wang, Ying Zhou, Yong Wang, Jun Zhang, C. L.Philip Chen, and Zibin Zheng. Multi-objective Vehicle Routing Problems with Simultaneous Delivery and Pickup and Time Windows: Formulation, Instances, and Algorithms. IEEE Transactions on Cybernetics, 46(3):582–594, 3 2016. ISSN 21682267. doi: 10.1109/TCYB.2015.2409837.

[86] Ko-Hsin Cindy Wang and Adi Botea. MAPP: a Scalable Multi-Agent Path Planning Algorithm with Tractability and Completeness Guarantees. Journal of Artificial Intelligence Research, 42: 55–90, 2011.

[87] Peter R. Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In AI Magazine, volume 29, 2008.

[88] Jingjin Yu and Steven M Lavalle. Structure and intractability of optimal multi-robot path planning on graphs *. Technical report, Association for the Advancement of Artificial Intelligence, 2013. www.aaai.org.

[89] Haifei Zhang, Hongwei Ge, Jinlong Yang, and Yubing Tong. Review of Vehicle Routing Problems: Models, Classification and Solving Algorithms. Archives of Computational Methods in Engineering, 29(1):195–221, 1 2022. ISSN 18861784. doi: 10.1007/s11831-021-09574-x.

[90] Lihua Zhu, Xianghong Cheng, and Fuh Gwo Yuan. A 3D collision avoidance strategy for UAV with physical constraints. Measurement: Journal of the International Measurement Confederation, 77, 2016. ISSN 02632241. doi: 10.1016/j.measurement.2015.09.006.