Delft University of Technology

Multi-agent Planning and Coordination for Automated Aircraft Ground Handling

Master of Science Thesis

Szu-Tung Chen

TU Delft

Multi-agent Planning and Coordination for Automated Aircraft Ground Handling

Master of Science Thesis

by

Szu-Tung Chen

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday August 11th, 2022.

Student number:	5200474	
Project duration:	September 2021 – August 2022	
Thesis committee:	Dr.ir. B.F. Lopes Dos Santos	(TU Delft, Chairman)
	Dr. O.A. Sharpanskykh	(TU Delft, Supervisor)
	Dr. G. Ermis	(TU Delft, Additional)
	Ir. O. Stroosma	(TU Delft, Examiner)

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

TU Delft

Acknowledgements

It has come to this day. This thesis marks the end of my master's study at TU Delft, and it also marks the end of my 19-year student life. I am very grateful for all the people who made my journey at TU Delft possible.

First of all, I would like to thank my supervisor Dr. Alexei Sharpanskykh, who guided me through the entire thesis project. During our meetings, he could always capture what I was trying to express and gave critical comments. Alexei never fails to amaze me by the fact that he always knows what I am doing and understands even the tiniest detail in my research. Next, I would like to thank Dr. Gulcin Ermis for helping me with the optimization of the model and for reviewing my research paper.

My studies at TU Delft would not be possible without the support of my family. I am grateful to my parents who trust me a lot and support me. Thank you mom for spending lots of time in video and audio calls with me whenever I felt bored and lonely during the lockdown. Thank you dad for always supporting me. Thank you, grandma, you never failed to remind me to eat well and sleep well. The thought of being able to go home and spend some time with you has always been one of my greatest motivations to complete my studies. I am also very thankful for my sister for accompanying my parents and grandparents when I am not at home.

Moreover, I would like to thank Melissa Oremans, my good friend in ATO, for working with me on several courses and hanging out with me sometimes. You let me feel I belong here. I would also like to thank my Friday-gathering friends for always being there for each other and letting me feel at home while being abroad. We start the Delft journey together, and now our even-more-exciting journey is about to begin. Last but not least, a special thanks to Ying-Chuan Ni, my boyfriend, who never stops supporting me and cheering me up. Without your accompany and support, this journey would be much more different and difficult. Thank you, Chuan.

If I look back to 2 years ago – when I have to choose between several graduate schools, I feel really happy and grateful that I've followed my inner voice and chose to pursue my interest to study air transport and operation at TU Delft. I wish I can remember how brave I was then, and keep following my inner voice in my life.

Szu-Tung Chen
Delft, July 2022

Contents

List of Figures

List of Tables

List of Abbreviations

A-CDM	Airport-Collaborative Decision Making
ACO	Ant colony optimization
ASP	Answer Set Programming
ATC	Air Traffic Control
ATO	Air Transport and Operations
BCBS	bounded conflict based search
BCO	Bee colony optimization
BP	bypassing conflicts
CA	combinatorial auctions
CA*	Cooperative A*
CBAA	consensus-based auction algorithm
CBBA	consensus-based bundle algorithm
CBM	Conflict-Based Min-Cost-Flow
CBS	Conflict Based Search
CBS-TA	CBS with task allocation
CBSw/P	conflict-based search with priority
CCBS	continuous-time conflict-based search
CCBS-kc	CCBS with kinematic constraints
CO-WHCA*	Conflict Oriented WHCA*
CO-WHCA*P	Conflict Oriented WHCA* with prioritization
CSP	Constraint Satisfaction Problem
CT	constraint tree
DAO	Dragon Age Origins
DP	Dynamic Programming
ECBS	enhanced conflict based search
ECBS-CT	enhanced conflict based search with continuous time
ECBS-TA	enhanced conflict based search with task allocation
EOBT	Estimated Off-Block Time
EPEA*	Enhanced Partial Expansion A*
FOD	Foreign Object Debris

GA	genetic algorithm
GCBS	Greedy conflict base search
GPU	Ground Power Unit
GSE	Ground support equipment
HCA*	Hierarchical Cooperative A*
ICBS	improved conflict base search
ICT	increasing cost tree
ICTS	Increasing Cost Tree Search
ID	Independence Detection
ILP	integer linear programming
KPI	key performance indicator
LA-MAPF	multi-agent path finding for large agents
LP	linearing programming
MA-CBS	meta-agent conflict based search
MAMP	Multi-Agent Motion Planning
MAPF	Multi-agent Path Finding
MATA	Multi-agent Task Allocation
MC-CBS	multi-constraint conflict based search
MR	merge and restart
OD	Operation Decomposition
PBB	Passenger Boarding Bridge
PBS	Priority Based Search
PC	Prioritizing conflicts
PCA	Pre-conditioned Air Unit
PT	priority tree
pTeSSB	probabilistic Temporal Sequential Single Bundle
pTeSSI	probabilistic Temporal Sequential Single Item
SAT	Boolean satisfiability
SBB	sequential bundle bid
SIPP	Safe Interval Path Planning
SIPPwRT	SIPP with reservation table
SSC	sequential single cluster
SSI	sequential single item
STN	simple temperol network

STNU	simple temporal network with uncertainty
TA	task allocation
TAPF	target-assignment and path finding
TeSSI	Temporal Sequential Single Item
TOBT	Target Off Block Time
TPG	temporal plan graph
TRT	trunaround time
TSAT	Target Start Up Approval Time
TSP	Traveling Salesman Problem
ULD	Unit Load Device
VRP	Vehicle Routing Problem
WHCA*	Windowed Hierarchical Cooperative A*

I

Scientific Paper

Multi-agent Planning and Coordination for Automated Aircraft Ground Handling

Szu-Tung Chen*, Alexei Sharpanskykh†, Gülçin Ermiş‡

Delft University of Technology, Delft, the Netherlands

Abstract

Inspired by the vision of fully autonomous airside operations at Schiphol airport, this study aims to contribute to the short-term goal of automated aircraft ground handling. In this research, we design and evaluate a multi-agent system for planning of automated ground handling. There are two main components in the system, task allocation optimization, and multi-agent path planning. To allocate tasks to ground support equipment (GSE) vehicles, an auction mechanism inspired by temporal sequential single item (TeSSI) auction is proposed. Ground handling tasks scheduling for GSE vehicles is modeled as several single-vehicle pickup and delivery optimization problems (SPDP), and the values of the objective functions are applied to generate bids for GSE vehicle agents in the auction. Moreover, Prioritized Safe Interval Path Planning for large agents (LA-SIPP) is used to plan collision-free paths for GSE vehicle agents in the model to execute tasks. Experimental studies have shown that the system is able to perform task allocation and path planning of ground handling tasks for flights in 3 aircraft stands within a 4-hour time in a reasonable computational time. Moreover, the model is capable to replan the tasks for agents when disruption happens. Applying the lowest possible numbers of vehicles used in the current operation, the model can always reach success allocation and path planning rates higher than 81% and 98%, respectively.

1 Introduction

At Amsterdam Airport Schiphol, Royal Schiphol Group aims at achieving fully-autonomous airside operations by 2050. One of the anticipated implementation is the automation of ground handling processes. The automation will help to utilize the resources more effectively, shortening the traveling distances and time, and thus helping to decrease the emission produced by the aviation sector. The automation of aircraft ground handling has to be introduced gradually. In this study, we focus on the automation goals to be achieved in the near future, automated docking of ground support equipment (GSE) vehicles. To achieve automation, dedicated procedures and technology need to be developed. This work contributes toward the procedure development. In particular, we propose an approach for planning and coordination for automated aircraft ground handling operations. This approach is based on the agent-based system modeling paradigm.

Agent-based modeling is a promising approach to modeling complex systems. Agent-based modeling uses a bottom-up approach, in which the system can be defined from the perspective of heterogeneous autonomous entities. It can capture various goals and interests of the involved agents, as well as the interaction and the interdependence between them. In the operation of aircraft ground handling, there are multiple stakeholders involved, including the airline, airport operator, ground handling service companies, and air traffic control. The stakeholders have different interests and goals. At the same time, their behaviors are highly dependent on other involved agents. Thus, intensive communication and cooperation are necessary for ground handling operations. Some aviation-related applications include agent-based delay management in autonomous taxing [1] and resilient airport surface movement [2]. However, in the area of aircraft ground handling, little research exists that has applied agent-based modeling approaches. Based on the characteristic of ground handling operations, we believe that agent-based modeling is an appropriate approach for the automation of aircraft ground handling.

This thesis aims to design and evaluate a multi-agent system model for automated aircraft ground handling. Two important capabilities are implemented in the model. First, a task allocation optimization mechanism is designed to optimally allocate the ground handling tasks to ground support equipment vehicles. Second, to plan collision-free paths for the GSE vehicles to execute these tasks, an adapted multi-agent path finding (MAPF) algorithm is implemented. The system can better contribute to the automation of aircraft ground handling if it can be applied online. As ground handling processes are operated in dynamic environments, in realistic

*MSc Student, Faculty of Aerospace Engineering, Delft University of Technology

†Assistant Professor, Faculty of Aerospace Engineering, Delft University of Technology

‡Post-doctoral Researcher, Faculty of Aerospace Engineering, Delft University of Technology

operations, lots of uncertainties exist and disruptions occur often. Replannings are always necessary for such circumstances. It is therefore important for our system to have a reasonable computational complexity. Honig et al. pointed out that task assignment and path planning are conducted separately in most works [3]. Although the solution quality may be compromised by applying such approaches, the efficiency is much better and the solution time is reasonable for real-world problems. Moreover, in automated aircraft ground handling, temporal and operational constraints make the problem of solving task allocation and path planning simultaneously more complex. For these reasons, task allocation and path planning of the ground handling tasks will be solved in separate stages in our model.

There are multiple stakeholders involved in the automation of aircraft ground handling. To feasibly assign the ground handling tasks to GSE vehicles such that a global objective function is optimized, centralized approaches are often inefficient due to the computational time and communication limits. On the other hand, auction mechanisms are considered suitable frameworks for task allocation of automated aircraft ground handling. Throughout the auction processes, bids of agents can capture the individual goals and utilities of agents under various circumstances. Auction-based multi-agent task allocation solvers can also efficiently perform with different team objectives. To deal with various kinds of constraints in ground handling, an adapted temporal sequential single item (TeSSI) auction is developed for solving the task allocation problem at the first stage of our model. Bids of agents for the TeSSI auction are the costs for them to perform the ground handling tasks, which are generated by modeling the task scheduling as single-vehicle pickup and delivery optimization problems (SPDP). For the path planning of agents in the second stage of the model, priority based algorithms are selected to be implemented. Priorities of different types of ground handling vehicles are critical in current aircraft ground handling operations, and continuations of some of the current operations would also be important to gradually move towards automated aircraft ground handling. In the model, prioritized Safe Interval Path Planning (SIPP) is implemented. SIPP for large agents (LA-SIPP) is implemented as the single agent path planning algorithm in the model. Experimental analyses are conducted to evaluate the performance of the model. We test task allocation in terms of makespans, task allocation rates, and computational time in different scenarios. The implemented path planning algorithm is also tested by exploring the planning path length and duration. To further validate the model, sensitivity analyses are performed by varying the number of ground handling vehicles.

This paper is structured as follows. First, the assumptions regarding ground handling operations and model specifications are introduced in section 2. Then, the task allocation mechanism using auctions is discussed in section 3. In section 4, the utilized MAPF algorithm, prioritized SIPP, is discussed. Furthermore, the performed experimental analyses are presented in section 5. Finally, section 7 presents the conclusion of the work.

2 Model Formulation

In this research, a multi-agent system model for automated aircraft ground handling is formulated. In this section, we first introduce the assumptions that are made based on current ground handling operations. Considering the assumptions, we simplify realistic aircraft ground handling environment into grid maps. Also, the solution approaches utilized are briefly introduced in this section. Moreover, the agents involved in the system, including their properties and interactions are specified.

2.1 Assumptions

2.1.1 Ground handling activities

The research focused on the ground handling operations of the aircraft used for short-haul flights. A single type of aircraft, Boeing 737-800 aircraft is considered in the model. Only the ground handling activities outside the aircraft cabin are considered, among them: aircraft refueling, catering galleys unloading and loading, baggage unloading and loading, water servicing, and toilet servicing. Refueling and charging of the ground handling vehicles, seasonal operations such as de-icing are not included in this study.

2.1.2 Infrastructure

Aircraft stands (also referred to as aircraft bays or bays) are the parking spots for aircraft to park between flights. Regarding the infrastructure of the aircraft stands, it is assumed that all aircraft stands are equipped with Passenger Boarding Bridges (PBB), and all passengers are able to board the aircraft via PBBs. Also, all aircraft stands have an underground fueling system so that only small fuel filling vehicles are needed.

Table 1: GSE resources for the ground handling of an aircraft operating short-haul flights with recommended priorities. Adapted from [4] and internal documents.

GSE resources	Numbers	Priority
Fuel filling vehicle (RE)	1	1
Catering truck (CA)	1	2
Baggage vehicle (BA)	1	3
Water service truck (WA)	1	4
Toilet service truck (WC)	1	5

2.1.3 Operation

It is assumed that all Ground Service Equipment are shared by the aircraft stands in the same pier. The same type of ground handling vehicles are assumed to be homogeneous for simplicity. Numbers of different type of GSE vehicles required to serve one flight and their priorities are listed in Table 1. In case of conflicts between movements of GSE vehicles, the listed order is used to set the priority. There is no bound on the number of personnel available to support the operation of GSE.

Movements of aircraft are not considered in the path planning stage of the model. Thus, we perform path planning only for the GSE vehicles. Taxiing and towing of aircraft are not included in the scope of this research. The path planning of PBB is also not considered. For safety reasons, ground handling vehicles have to enter or exit the ramp through specified traffic lanes.

2.1.4 Task duration and time windows

A general schedule of the ground handling operation of a B737-800 aircraft is shown in Figure 1. However, the time chart is only a guideline for various turnaround operations. The real execution time of the ground handling tasks is affected by different factors. In our model, we set time windows for every type of ground handling task. The execution duration of ground handling tasks should lie in the specified time windows. Referring to the chart in Figure 1, duration and time windows of different types of ground handling tasks can be defined, as shown in Figure 2.

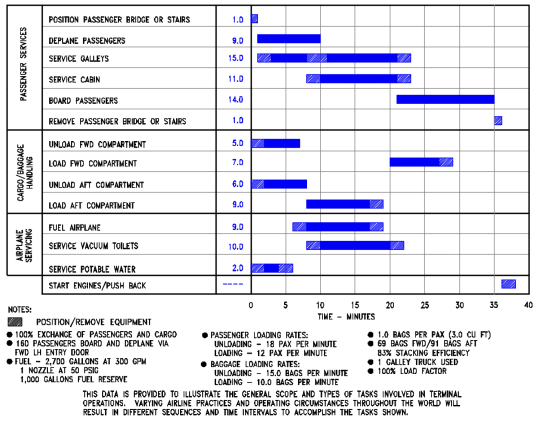


Figure 1: B737-800 turnaround time chart [5].

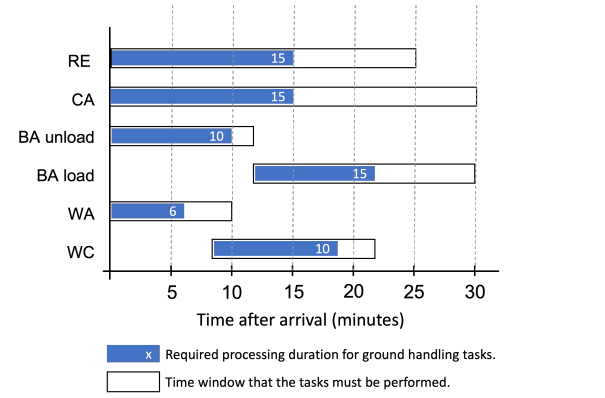


Figure 2: Processing times and time window restrictions for ground handling tasks.

2.2 Environment Specification

In the multi-agent system, we model an airside environment, including vehicle depots, a service road, and several aircraft stands. All modeled elements are represented by grid maps with obstacles. Consulting the typical GSE vehicle layouts on an aircraft stand, a 16×16-grid (of grid size 4 m) or a 64×64-grid (of grid size 1 m) environment with 15-20% obstacles can be implemented. The environment of aircraft stands modelled by 16×16 grids and 64×64 grids, together with the locations for performing different ground handling tasks, are illustrated in Figure 3 and Figure 4.

Depending on the grid size used for the aircraft stands, the service road environment is modeled as open grids with the same grid size as the aircraft stands. Directional constraints are applied on the grids of the service road. The constraints make sure that the GSE vehicles always drive on the right side of the road. The

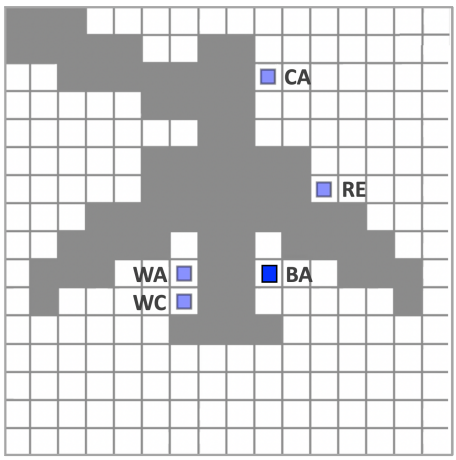


Figure 3: Environment of an aircraft stand modeled by 16×16 grids.

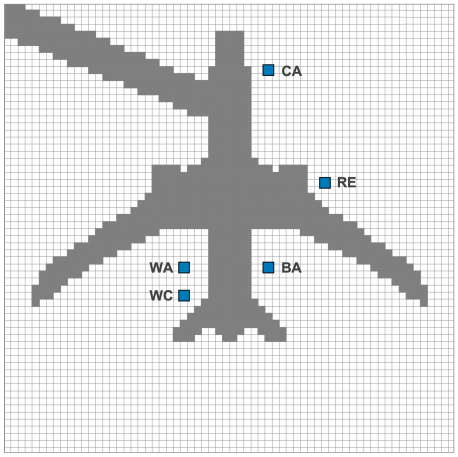


Figure 4: Environment of an aircraft stand modeled by 64×64 grids.

vehicle depots are represented by some grids connected to the service road. An example modeling environment is shown in Figure 5.

We first built a basic path planning model, using 16×16 grids maps to model the aircraft stands. In this basic model, GSE vehicle agents are represented by rectangular shapes that only occupy one grid. To better represent the different sizes of GSE vehicles, agents with different sizes larger than unit grid can be used. In the extended model, we use 64×64-grid maps to model the aircraft stands. In such a modeling environment, GSE vehicle agents can be represented by different sizes of squared shapes around reference points. Transformation of agents, including rotation, is not considered in our model.

To limit the model complexity and the required computational time, paths on the service road and paths on different aircraft stands are planned separately. Also, as mentioned in the previous section, GSE vehicle agents need to enter or exit the aircraft stands via corresponding bay entrances and exits. Therefore, paths are planned between the vehicle bases and the entrance/exit of aircraft stands, between the entrance/exit of aircraft stands and the working locations beside the aircraft, and between the exit of aircraft stands and the entrance of other aircraft stands.

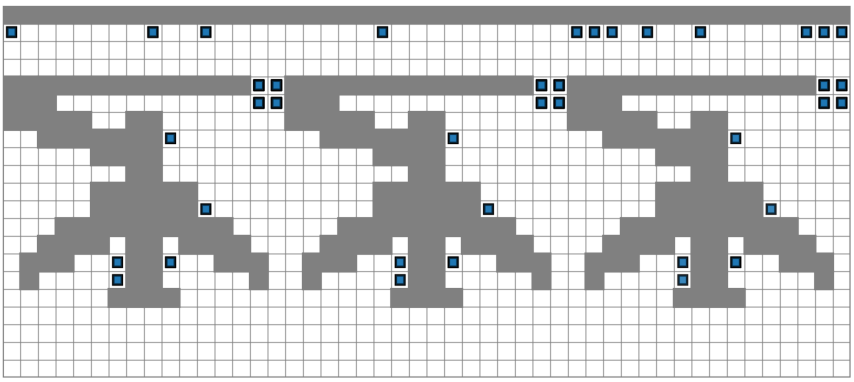


Figure 5: An instance of modeling environment.

2.3 Agent Specifications

In this section, we introduce the agents that are included in the multi-agent system for automated ground handling. There are four types of agents: task allocation agents, GSE vehicle agents, path planning coordination agents, and environment agents.

2.3.1 Task allocation agents

Auction is applied as the task allocation mechanism in this research. The task allocation agents are modeled as the auctioneers of the auctions. Every type of ground handling task is allocated separately, so there is one task allocation agent for each type of ground handling task. The agents have the following properties:

- *Task announcement property*: Given the flight schedule within the model planning window, agents first formulate different types of ground handling tasks that have to be performed for the flights. Then, as auctioneers, the task allocation agents announce the tasks to the bidders, the GSE vehicle agents. The auction has multiple rounds and one task is announced in each round. Information about tasks including their locations, duration, and time windows is declared at this stage.

- *Winner determination property*: After announcing the task in every round of the auction, GSE vehicle agents bid on the task and send their bids to the task allocation agents (the auctioneer). The task allocation agent determines the winner of the task depending on the agents' bids.

- *Task allocation property*: After determining the winning agent of a task, the task allocation agent announces the winner and allocates the task to the GSE vehicle agent.

2.3.2 GSE vehicle agents

The GSE vehicle agents are involved in both task allocation and path planning mechanisms of the model. Each GSE vehicle is modeled as an agent. The agents are classified by their type of ground handling tasks. In the setup of our experiments, there are two available GSE vehicle agents for which each type of ground handling task except for baggage handling four available GSE vehicle agents are utilized. The GSE vehicle agents have the following properties.

- *Bidding property*: The GSE vehicles agents are bidders in the task allocation auctions of their type of ground handling task. After the auctioneer announces the task to bid in each round of the auction, every bidder bids on the task. The bidder then sends its bid to the auctioneer.

- *Scheduling property*: If the GSE vehicle agent is chosen as the winner of a ground handling task, the agent adds the task into its schedule. Based on its schedule, a GSE vehicle agent can plan trip itineraries including the locations it needs to visit.

- *Path planning property*: Given the trip itineraries, maps of the environment, and continuous periods without collisions for every grids on the map (safe intervals), the GSE vehicle agents are able to plan their paths without colliding with other agents or obstacles in the environment. Paths of individual trips in the agents' itineraries are planned separately.

- *Path integration property*: As paths of different trips are planned individually on separate maps, the GSE vehicle agents have to integrate their paths before moving. Paths that belong to different environments (maps) have to be spacially integrated into one route. Temporally, the end time of the previous trip may not fit the start time of the succeeding trip. Therefore, wait motions have to be included in the integrated route.

- *Motion property*: Once the paths are planned and integrated, the GSE vehicles can start moving in the modeling environment.

2.3.3 Path planning coordination agent

The path planning coordination agent manages the priority order of agents.

- *Priority management property*: The path planning agent manages the priority order of agents. Agents are first sorted based on their types of ground handling tasks, then they are sorted by their departure time from the depot. The applied priority order of agents is listed in Table 1.

2.3.4 Environment agents

The environment contains two kinds of maps: service road maps and bay maps. There are a service road map and several bay maps in the model. For each map the corresponding map agent is defined.

- *Safe interval management property*: The map agents keep tracks of the safe intervals on their map. Every time a path is planned on a map by a GSE vehicle agent, the path is sent to the corresponding map agent. The safe intervals of the cells on the map are then updated.

3 Task Allocation Mechanisms

Task allocation is one of the main components of automated aircraft ground handling. To complete various kinds of ground handling tasks of all flights on time, ground handling tasks must be allocated to GSE vehicles using a effective and efficient procedure that minimizes the completion times, while respecting the capacity, connectivity, and temporal constraints. In this research, an auction mechanism inspired by temporal sequential single-item (TeSSI) auction [6] is proposed to allocate tasks to GSE vehicles. The distinctive part of the proposed procedure from TeSSI is that the executions of ground handling tasks in schedules of GSE vehicles are modeled as single vehicle pickup and delivery optimization problems (SPDP) [7][8] to generate the bids for the auction. Bundling of tasks is also included in the model to speed up the allocation process. As the required GSE vehicles for different types of ground handling tasks are different, each type of ground handling task is allocated separately.

3.1 Adapted TeSSI Auction for ground handling task allocation

Within the planning window and planning scope of the model, different types of ground handling tasks for flights are allocated separately in several adapted temporal sequential single item (TeSSI) auctions. TeSSI auctions are extensions of single sequential item auctions introduced by Nunes and Gini [6]. The strength of TeSSI is its ability to handle temporal constraints. Time windows are used to indicate the time intervals within which tasks have to be processed.

Consider a set of agents $A = \{a_1, a_2, ..., a_n\}$, representing the available ground handling vehicles, and a set of ground handling tasks $T = \{t_1, t_2, ..., t_m\}$. Each task t_i has its earliest possible start time Est_i and its latest start time Lst_i, where $Est_i \leq Lst_i$. The duration of the task is denoted as Dur_i and the latest finish time is Lft_i, with $Lst_i + Dur_i = Lft_i$. The interval $[Est_i, Lft_i]$ defines the time interval in which the task has to be performed.

In the auction, tasks are allocated independently in multiple rounds. In each round of the auction, all participating agents bid on an unallocated task. The agents evaluate the cost of committing to the task-to-bid, taking into account their current schedule. After generating bids on tasks, the agents send their bids to the auctioneer. Once the auctioneer receives all the bids, the auctioneer performs winner determination and allocates the task to the agent who bids the best price. If no agent bids on a task, resulting in the task cannot being assigned, the task is removed from the set T. The auction continues until all the tasks in T have been allocated or discarded. The allocation is done in a hill-climbing way. Although optimal allocation is not guaranteed using such methods, good solutions can usually be found.

In the original TeSSI auction, agents maintain the temporal consistency of their allocated tasks using simple temporal networks (STN). In this research, we model the execution of ground handling tasks as pickup and delivery operations. Therefore, STN for agents is substituted with single agent pickup and delivery optimization models, which is discussed in section 3.2. We refer to the new optimization based TeSSI auction approach as Adapted TeSSI auction.

3.1.1 Bidding rules

Unlike common auctions, auction mechanisms used for task allocations are usually cooperative auctions. It is crucial in cooperative auctions to establish bidding rules that optimize the problem objectives. In the model, we consider two different team objectives. The first objective is to minimize the maximum completion time over all tasks allocated to a team of agents (also known as *makespan*). Minimizing *makespan* is often referred to as MINIMAX and it is commonly used in auction-based routing literature [9]. Another objective we aim to minimize is the total traveling time of all GSE vehicles to complete all tasks in the task set. It is a variant of the objective MINISUM [9]. We use the term *sum-T* to refer to the objective in this paper. Let $F = \{F_1, F_2, ..., F_n\}$ be the final allocation of tasks, where F_i contains all allocated tasks of agent a_i. C_{max_i} denotes the *makespan* for agent a_i to complete all allocated tasks in F_i. Also, TT_i represents the minimum traveling time that agent a_i must spend to perform its allocated tasks F_i. The objectives $obj(F)$ to be minimized can be formed as follows:

- *makespan*: $obj(F) = \max_i[C_{max_i}(a_i, F_i)]$
- *sum-T*: $obj(F) = \sum_i TT_i(a_i, F_i)$

Consider t as the task-to-bid in the current round of auction, and $S = \{S_1, S_2, ..., S_n\}$ as the current partial allocation of the tasks. Depending on the objective, the following bidding rules should be applied:

- bid for objective *makespan* – *makespan*: $C_{max_i}(a_i, S_i \cup t)$. Agent a_i bids the new minimum makespan of its tasks after including task t in its plan. Note that the maximum agent path cost before allocating t can be neglected since this is a system-level variable.
- bid for objective *sum-T* – *marginal-sum-T*: $TT(a_i, S_i \cup t) - TT(a_i, S_i)$. Agent a_i bids the marginal increase of traveling time incurring in adding t to its plan.

3.1.2 Adapted TeSSI auction

The procedure of TeSSI auction for the auctioneer is shown in Algorithm 1. The inputs of the algorithm contain a set of agents, A, and a set of tasks, T, that are to be allocated to agents. Task set, $T_{unallocated}$, contains the tasks that cannot be allocated. In the beginning, the task set $T_{unallocated}$ is an empty set. For each task t in T, every agent $a \in A$ evaluates the task using the function *a.evaluateTask*. According to the applied bidding rule, each agent a generates a bid on the task and a corresponding schedule (if the task can be included in its schedule without violating any constraints). The generated bids for task t from agents are compared to the current lowest bid *lowestBid*. If the minimum bid is lower than the existing lowest bid, the current lowest bid replaces the existing lowest bid. The bidding agent a and its generated schedule *tempSchedule* are saved. Also, the task t is marked as *assigned*. After all the agents have tried to bid on the task t, it is allocated to the winning agent *winner*. The schedule of the agent *winner* is updated. On the other hand, if task t is not assigned to any of the agents, it is moved into the unallocated task set $T_{unallocated}$. The auction procedure continues until all the tasks in T have been allocated or moved to $T_{unallocated}$. The outputs of the algorithm are sets of assigned tasks for all agents $a \in A$, and the final schedules $Schedule_a$, $a \in A$.

Algorithm 1 TeSSI auction for the auctioneer

Input:
  A: Set of agents
  T: Set of tasks to be allocated
Output:
  S_a: Sets of assigned tasks for agents a, $a \in A$
  $Schedule_a$: Final schedules $Schedule_a$ for agents, $a \in A$
$T_{unallocated} = \emptyset$;
for $t \in T$ **do**
  $lowestBid = \infty$
  $assigned = False$
  for $a \in A$ **do**
    $Bid, tempSchedule = a.evaluateTask(t)$
    if $Bid < lowestBid$ **then**
      $winner = a$
      $lowestBid = Bid$
      $winnerSchedule = tempSchedule$
      $assigned = True$
    end if
  end for
  if $assigned = True$ **then**
    $S_{winner} = S_{winner} \cup \{t\}$ {update the set of assigned tasks for the winning agent}
    $Schedule_{winner} = winnerSchedule$ {update the schedule of the winning agent}
  else
    $T_{unallocated} = T_{unallocated} \cup \{t\}$
  end if
end for

Algorithm 2 presents the *evaluateTask* procedure for agents. The inputs are the set of assigned tasks for the agent, S_a, and the task to be evaluated, t. The considered task set includes the set of tasks in the existing partial schedule and the candidate task, $T_{evalSet}$. To apply the bidding rule of the auction, the task set will be optimized in a single vehicle pickup and delivery optimization model, which is discussed in detail in section 3.2. For every pickup and delivery optimization, a solution time limit of 5 seconds is applied. If the instance is not infeasible and is able to be solved within the time limit, the optimizer returns the objective value and the optimized schedule of the set of tasks $T_{evalSet}$. The objective value is the *Bid* that the agent bids on task t. The procedure *evaluateTask* generates the value of the bid and the corresponding schedule as output for any agent.

If time limit for optimization is reached, the agent bids the feasible solution obtained at the end of the time limit, which is not optimized. In this case, the generated schedule is not optimal either. As there are other completing agents, a non-optimized bid decreases the chance that the task will be assigned to the agent. Further, if no feasible solution is able to be found within the 5-second time limit, an infinite number is used as the agent's bid. In this case, the agent also returns an empty schedule. This also means that the agent withdraws itself from the current round of the auction.

Algorithm 2 Procedure *evaluateTask* for agent a

Input:
  S_a: Set of assigned tasks for agent a, $a \in A$
  t: Task to be evaluated
Output
  Bid: Bid of task t for agent a, $a \in A$
  $Schedule_a$: Schedule of agent a considering task t, $a \in A$
$T_{evalSet} = S_a \cup \{t\}$
$objective$, $Schedule_a = optimizerSPDP(T_{evalSet}, a)$
if objective exists **then**
  $Bid = objective$
else
  $Bid = \infty$
end if

3.1.3 Handling uncertainties

In real-world aircraft ground handling operations, the duration of executing a certain ground handling task is not fixed. For GSE vehicles, the travel time between their depots and locations on aircraft stands varies, depending on the airside traffic congestion. To deal with the temporal uncertainty, Rizzo proposed probabilistic Temporal Sequential Single Item (pTeSSI) auction [10], a variant of TeSSI. Inspired by pTeSSI, uncertain vehicle traveling time is incorporated in the auction mechanism of the model.

To successfully plan the paths for agents, possible traveling delays and waiting times due to conflicts should already be taken into account in the task allocation phase. Considering a constant speed of agents, the traveling time between two airside locations is defined as:

$$distance\ buffer\ coefficient \times shortest\ distance \div speed \tag{1}$$

The distance buffer coefficient is a normally distributed random variable with a mean value larger than one. The distances between airside locations are important parameters used in the bid generation of agents. A mean value of 1.4 and a standard deviation of 0.2 are used in the model. According to our experience, using 1.4 as the mean value of the distance buffer coefficient in task allocation gives the best result in path planning. At the same time, different standard deviation values are tried but they do not cause apparent differences.

3.2 Pickup and Delivery Optimization

To generate a bid on a certain ground handling task in the task allocation process of automated ground handling, an agent needs to evaluate the task-to-bid considering its existing schedule. This can be done using an optimization solver that optimizes the agent's schedule including the existing tasks and the task-to-bid. In this study, an optimization solver is developed by modeling the single agent task scheduling problem as single vehicle pickup and delivery problem with capacity constraints. Single vehicle pickup and delivery problem with capacity constraints is a variant of Vehicle Routing Problems with Time Windows and Pickups and Deliveries.

3.2.1 Problem definition

The route of a GSE vehicle needs to be optimized by determining the pick-up times and delivery times of the considered tasks. To complete the processing of its allocated tasks, a GSE vehicle has to visit a set of pick up and delivery points at different locations of aircraft stands and the vehicle depot, starting and finishing its service within specific time windows at the pickup and delivery locations. GSE vehicle has a loading capacity. While traveling between locations outside and inside aircraft stands, it has to pass through specific entrance and exit locations of different aircraft stands. The main goal is to minimize the makespan or total traveling time of the GSE vehicle by scheduling a set of tasks to bid on the candidate task. The problem is a single agent scheduling problem with earliest start time and and latest completion time constraints. However, the required time to complete a task is not only dependent on the processing time. It is also affected by the positions of GSE vehicles and their servicing points, and the routes that need to be taken between loading and unloading points, which makes the problem more complex. Therefore, a novel approach is presented in this study by re-designing the problem as a vehicle routing problem with time windows, pickups and deliveries. To convert the GSE vehicle scheduling problem to the vehicle routing problem with pickups and deliveries, ground handling tasks are redefined by being split into several pickup and delivery sub-tasks that occurs at various loading/unloading locations and service points. The paths between the locations are generated by also considering the compulsory exit and entrance positions in addition to the locations of the actual nodes, which are also taken into account

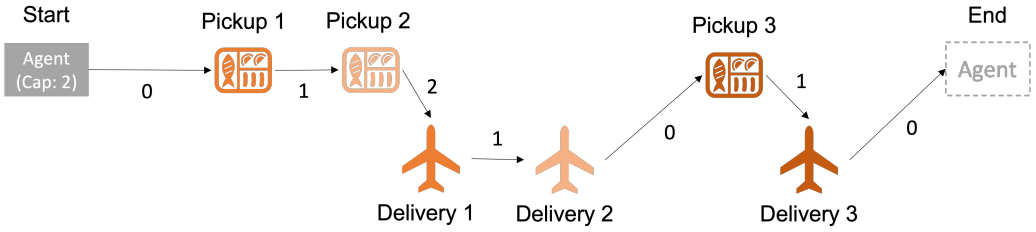
Table 2: Description of pickup and delivery tasks for different ground handling tasks and the capacities of corresponding ground handling vehicles.

Task	Pickup Location	Pickup sub-task	Delivery Location	Delivery sub-task	Capacity
RE	Terminal	N.A.	Bay	Use the underground refueling system	inf.
CA	Terminal	Pickup the food for flights	Bay	Empty the used catering galley on the aircraft, and load the new catering galleys to the aircraft.	2
BA unload	Bay	Unload baggage from the aircraft	Terminal	Unload the baggage vehicle	1
BA load	Terminal	Pickup the baggage from the terminal	Bay	Load the baggage to the aircraft	1
WA	Bay	Fill-up the water tank of the truck	Bay	Fill the aircraft water tank	3
WC	Bay	Drain the waste from the aircraft lavatory	Terminal	Drain the waste water from the truck	3

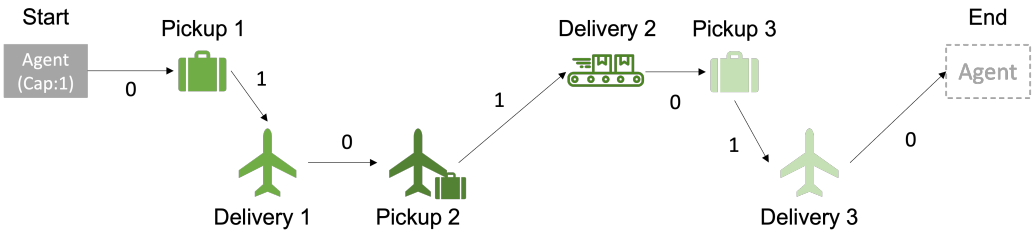
RE: refueling; CA: catering service; BA: baggage handling; WA: water service; WC: toilet service.

in the computation of traveling time between locations. These paths serve as single links or edges between the task nodes in the proposed design.

For outbound flights, the loading of resources to vehicles is set as a pickup task, while loading or supplying the resources to the aircraft represent the delivery task. Two examples for pickup and delivery representation of ground handling tasks are shown in Figure 6. For the catering service task in Figure 6(a), loading the food from the terminal to the GSE vehicle and loading the food from GSE vehicle to the aircraft represent the pick up and delivery sub-tasks, respectively. For the baggage handling task in Figure 6(b), loading the baggage from terminal and from aircraft to the GSE vehicle are defined as pick-up sub-tasks, whereas loading the baggage from GSE vehicle to the aircraft and unloading the baggage from GSE vehicle to be placed on the conveyor belt are modeled as delivery sub-tasks. The numerical values next to the links connecting pickup and delivery nodes denote the load of the vehicle while traveling between two nodes. The definition of pick-up and delivery sub-tasks for different ground handling tasks and the capacity values of corresponding ground handling vehicles are given in Table 2.



(a) Catering service



(b) Baggage handling

Figure 6: Schematic of the pickup and delivery representation of aircraft ground handling tasks.

Apart from the makespan, various objective functions can be used to generate the bidding value in the auction model. Two different objectives, *makespan* and *sum-T*, are used for bidding in the auction models in this work. Proposed pickup and delivery optimization model is used to generate these objective values for each agent, thus being a sub-problem of the higher level auction model. When the bidding strategy is based on makespan, each agent makes an offer with the minimum makespan of the GSE vehicle that is obtained by solving the pickup and delivery model, using the *makespan* objective, over the set of previously assigned tasks and including the candidate task. While in the *sum-T* case, agent offers the minimum *marginal-sum-T* by solving the model using the *sum-T* objective which includes only the traveling time of the GSE vehicle.

Data related to the set of tasks to be evaluated and optimized contains the following information: task identity numbers i, resource (pickup) locations P_i, task (delivery) locations D_i, earliest pickup time Est_{P_i}, latest pickup time Lst_{P_i}, earliest delivery time Est_{D_i}, latest delivery time Lst_{D_i}, pickup duration Dur_{P_i}, and delivery duration Dur_{D_i} of the tasks. Pickup and delivery locations are either the depot of the vehicle or the

Table 3: An instance of the data for a set of refueling tasks.

id	pickup location	delivery location	pickup earliest time	pickup latest time	delivery earliest time	delivery latest time	pickup duration	delivery duration
0	T1	B0	0	inf.	0	720	10	780
4	T1	B1	0	inf.	3900	4620	10	780
5	T1	B1	0	inf.	6900	7620	10	780
8	T1	B2	0	inf.	900	1620	10	780
9	T1	B2	0	inf.	6600	7320	10	780
10	T1	B2	0	inf.	9300	10020	10	780

T: GSE vehicle base/depot/terminal; B: Task locations on aircraft stands. The values of delivery duration are set based on Figure 2. All time values are in seconds.

locations of certain ground handling tasks on aircraft stands. In our model, pickup and delivery locations and the distances in between are defined based on realistic airport airside operations. Paths between locations and their traveling time are defined and computed as mentioned in section 3.2.2. Also, uncertainty of traveling times of vehicles, as mentioned in section 3.1.3, is introduced into the model.

An illustration of the depot and the aircraft stand locations are shown in Figure 7. Also, an instance of the data of a set of refueling tasks are provided in Table 3. In our model, the focus of the ground handling task executions is on the operations that are performed on the aircraft stands. In this focused procedure, in some cases delivery sub-tasks, in other cases pick-up sub-tasks are critical due to the strict time windows constraints as well as being the main task rather than a helper task such as collecting or delivering an empty vehicle at the depot. In the example of refueling tasks in Table 3, the focus of refueling tasks is on the delivery sub-tasks. Pickup sub-tasks do not have time window constraints. The duration of a pickup sub-task is set to a significantly small constant.

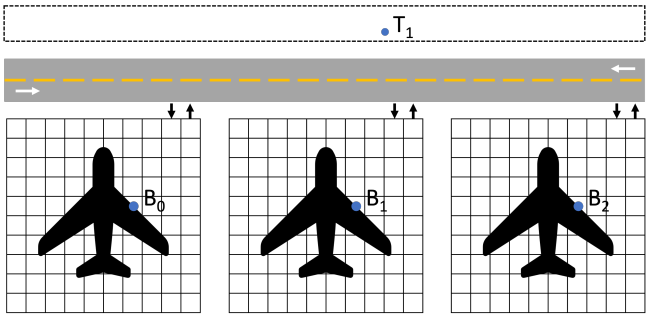


Figure 7: Illustration of the depot and aircraft stands with pickup and delivery locations.

3.2.2 Optimization model

Parragh et al. have performed a comprehensive survey on pickup and delivery problems [7][8]. They considered transportation between pickup and delivery locations (transportation between customers) as Vehicle Routing Problems with Pickups and Deliveries (VRPPD). In a scenario that transportation requests are associated with certain origins and destinations, the pickup and delivery points are paired. The execution of ground aircraft handling tasks lies in this category. The mathematical model of the single vehicle variant of pickup and delivery problem, the single vehicle pickup and delivery problem (SPDP) is presented in the survey. SPDP is used to generate the value of bids for agents in auctions. Mathematical model of the ground handling problem, which we design by adapting from the mathematical model of SPDP [7][8] is given as follows:

Parameters The parameters of the model are listed as follows:

n: number of pickup vertices.

$\tilde{n}$: number of delivery vertices.

P: set of pickup vertices, $P = \{1, ..., n\}$

D: set of delivery vertices, $D = \{n + 1, ..., n + \tilde{n}\}$.

A virtual start node 0 and a virtual end node $n+\tilde{n}+1$ are also added into the original graph. The problem is modeled on a complete graph $G = (V, A)$, where V is the set of all vertices, and A is the set of all arcs, defined as follows:

V: the set of all vertices, $V = \{0, n+\tilde{n}+1\} \cup P \cup D$.

A: the set of all arcs, $A = \{(i,j) : i,j \in V, i \neq n+\tilde{n}+1, j \neq 0, i \neq j\}$.

q_i: demand/supply at vertex i; pickup vertices have positive q_i values, while delivery vertices have negative values. The start vertex 0 and the end vertex $n+\tilde{n}+1$ have zero supply/demand values, $q_0 = q_{n+\tilde{n}+1} = 0$.

e_i: earliest time to begin service at vertex i.

l_i: latest time to complete service at vertex i.

d_i: service duration at vertex i.

t_{ij}: travel time from vertex i to vertex j.

C: capacity of the ground handling vehicle (depending on its type of ground handling task).

S: the set of sets of vertices that the the pickup or delivery tasks are executed on the same aircraft stand, $S = \{s_1, s_2, ..., s_m\}$.

s_m: the m^{th} set of vertices in which the pickup or delivery tasks are executed on the same aircraft stand, $s_m = \{i : i \in P \cup D\}, s_m \in S$.

In airport environment, GSE vehicles require to enter or exit aircraft stands via specified bay entrances and exits. The travel distance is found by computing the length of the path from vertex i to vertex j, which also passes through entrance or exit points in between. For simplicity, we assume that GSE vehicle agents move with unit speed. Therefore, the travel distance and travel time between nodes are equivalent. To align with the design of the path planning algorithm, distances used here are not Euclidean but Manhattan distances.

Decision variables The decision variables of the model are defined as follows:

x_{ij}: binary decision variable that is equal to 1 if the arc (i,j) is selected for the vehicle's route, or 0 otherwise.

Q_i: load of the vehicle after leaving vertex.

B_i: beginning time of service at vertex.

Objectives The objective of the optimization problem depends on the bidding rule of the auction that is used to allocate the ground handling tasks. The bidding rule of the auction can be *makespan* or *sum-T*. Bidding *makespan* means that the agent bids the total time span of its schedule including its current allocated tasks and the task-to-bid; while bidding *sum-T* means that the vehicle bids the extra travel time if the task-to-bid is included in its current schedule. Below we formulate the objective function for the SPDP based on two bidding rules.

Bid *makespan*:

$$minimize \quad B_{n+\tilde{n}+1} - B_0 \tag{2}$$

Bid *sum-T*:

$$minimize \quad \sum_{(i,j)\in A} t_{ij} x_{ij} \tag{3}$$

Note that the objective (3) is the total travel time of the vehicle. The marginal travel time is the required extra travel time if the task-to-bid is included in its current schedule. The value of the bid used in the TeSSI auction should be the value of the objective (3) minus the total travel time of the agent's current schedule.

Constraints The problem is subject to the following constraints:

$$\sum_{i:(i,j)\in A} x_{ij} = 1, \quad \forall j \in V \setminus \{0\}. \tag{4}$$

$$\sum_{j:(i,j)\in A} x_{ij} = 1, \quad \forall i \in V \setminus \{n+\tilde{n}+1\}. \tag{5}$$

$$M(1 - x_{ij}) + Q_j \geq Q_i + q_j, \quad \forall (i,j) \in A. \tag{6}$$

$$max\{0, q_i\} \leq Q_i \leq min\{C, C + q_i\}, \quad \forall i \in V. \tag{7}$$

$$B_i \leq B_{n+i}, \quad \forall i \in P. \tag{8}$$

$$M(1 - x_{ij}) + B_j \geq B_i + d_i + t_{ij}, \quad \forall (i,j) \in A. \tag{9}$$

$$e_i \leq B_i, \quad \forall i \in V. \tag{10}$$

$$B_i + d_i \leq l_i, \quad \forall i \in V. \tag{11}$$

$$x_{ij} = 0, \quad \forall i \in s_m \ \ \forall j \in s_m \ \ \forall s_m \in S. \tag{12}$$

Constraints (4) and (5) ensure that every vertex is visited exactly once. Constraint (6) is a big-M constraint which ensures that the load of the vehicle is updated after visiting a pick-up or delivery node. Constraint (7) makes sure the vehicle's load is not greater than its capacity. In SPDP, pickup and delivery vertices are paired. Therefore, the number of pickup vertices n equals the number of delivery vertices $\tilde{n}$. The pickup vertices are indexed by $i = 1, ...n$ and the corresponding delivery points are indexed as $n + i$. Using start time variables, constraint (8) ensures that for every pickup-delivery pair, the pickup vertex is visited before the delivery vertex. The big-M constraint in (9) ensures that the start time of the task at vertex j is greater than or equal to the completion time of task at vertex i plus the travel time from i to j, if vertex j is visited right after vertex i. Finally, (10) and (11) are added to ensure that the start time and end time of the service are within certain time windows. It is worth noting that by introducing constraints (9), (10), and (11), subtours of vertices are eliminated. Typical subtour elimination constraints can be omitted. Constraint (12) is a special constraint implemented to fit the need of the path planning part in this research. If the pick up and delivery tasks of an aircraft are to be consecutively processed by the same GSE vehicle in the same stand, there exists a waiting time after the completion of the preceding task due to the earliest start time constraint of the successive task. In path planning, having an agent occupying spaces on the aircraft stand for a long period can cause some problems. For example, the agent can block the necessary path of other agents with lower priority, or it can stop the other agent from entering the pickup or delivery location it occupies. Therefore, (12) forbids the consecutive processing of tasks if they belong to the same aircraft. There might be different types of path planning methods where collisions are dealt with using different strategies. In these cases, constraint (12) can be omitted.

By optimization of single vehicle pickup and delivery, the schedule for the agent is constructed. A route of a GSE vehicle is a tour of multiple pick up and delivery locations, starting and ending at its depot at the terminal without visiting the depot in between. The values of the decision variables x_{ij} are used in generating the route or task sequence for the agent. The values of start time variables B_i are the start times of tasks or services at the nodes from which start times of ground handling tasks on an agent can be extracted. The final schedules of agents are formed based on the iterative calls of optimization solver during the TessI auction, till the last candidate is allocated to an agent. These schedules and routes of agents are important inputs that will later be used in the path planning part of the model.

3.3 Bundling

In TeSSI auctions, tasks are allocated one by one in multiple rounds. The allocation process can be lengthy if there are large numbers of tasks to be allocated. To accelerate the auctioning process, tasks can be bundled before allocation. Agents evaluate the cost of performing all tasks in a bundle and generate a single bid on the bundle. All tasks in the bundle are then allocated to the same agent with the best bid in the same round of auction. Tasks of the same type of ground handling operation, which are also the tasks to be allocated in the same auction, can be combined into bundles. Regardless of the time constraints, possible bundles are all subsets

of the tasks. However, checking all possible bundles is not applicable in real time due to the huge amount of computational time. Rather than exploring all possible combinations, Gansterer and Hartl [11] have defined the bundle generation problem (BuGP) to generate attractive bundles. The attractive bundles are non-overlapping bundles covering all tasks to be assigned. In this research, we adopt the idea of BuGP which generates non-overlapping bundles. Instead of applying genetic algorithms, which is proposed in the research of Gansterer and Hartl [11], we use a heuristic method to generate the bundles.

3.3.1 Bundle generation method

The bundle generation heuristic is discussed in detail in the appendix of the thesis report. Before bungle generation starts, two parameters have to be specified. First, the parameter *bundle sizes* represents the number of tasks that are desired in a bundle. Considering the number of ground handling tasks to be handled within the planning window of our model, we use the bundles sizes of two and three in the model. Also, it may not be profitable to assign all tasks into bundles. The parameter *bundle rate* indicates the percentage of tasks to be assigned to bundles. A bundle rate of 0.5 is applied in the model.

3.3.2 Available bundles and bundle values

At the time of bundling, it is not known which agent will be allocated to the bundle. Thus, no information on agents' schedules and the possible task execution time exists. To make sure that all tasks in a bundle can be executed successfully, the availability of the bundle has to be checked. The principles of functions *isAvailable* and *makeBundleValue*, which are used in *bundle generation heuristics* to evaluate if a bundle of tasks is available and to assign bundle values are described as follows:

isAvailable All permutations of tasks in the bundle are tried while checking the availability of the bundle. For all successive task pairs in a bundle, we assume that the previous task is completed at the latest completion time defined by its time window. Feasibility of executing the latter task on time provided that the previous tasks are completed before their lastest completion time is checked. Let $[start_1, end_1]$ be the time window of the first task in the pair and $[start_2, end_2]$ be the time window of the successive task. Dur_1 and Dur_2 denote the duration of the two tasks, and t_{12} represents the required travel time between two tasks. If the following condition applies, the bundle of tasks is considered an available bundle.

$$end_1 + Dur_1 + t_{12} <= end_2 - Dur_2 \tag{13}$$

makeBundleValue To determine which tasks should be combined, *bundle values* are used. Bundles with time and/or space proximity have smaller bundle values. Bundles of tasks with smaller bundle values are preferred, and they are more likely to be officially combined into task bundles for allocation. For a bundle with ordered tasks, the bundle value is defined as the sum of the travel time and the time span from the start time of first window to the end time of second window between all successive task pairs. Let $[start_i, end_i]$ be the time window of the first task in the pair and $[start_j, end_j]$ be the time window of the successing task. Also, let t_{ij} represent the travel time between two tasks. The bundle value is computed as

$$(\sum_{i,j \in A \text{ and } x_{ij}=1} t_{ij} + end_j - start_i)/\text{number of tasks in the bundle} \tag{14}$$

During bundle generation, task permutation with the lowest bundle value over all permutation is used as the *order* of the set of tasks. The best permutation and the corresponding bundle value are saved to be used in the bundle generation heuristics. The process of generating bundle values depicted in (14) is used in a function *makeBundleValue*, which is employed by bundle generation heuristics shown in the appendix of the thesis report.

3.3.3 Re-allocation

In the auctions, all tasks in the bundle are allocated in the same round of auction. In any round of the auction, if a task bundle cannot be assign to any agent involved in the auction, the task bundle is split into individual tasks. The tasks are added to the set of tasks to be allocated, and they are then re-allocated separately in different rounds of the TeSSI auction. By splitting bundles that can not be allocated, the approach can help to enhance the allocation rates for ground handling tasks. However, because of the order of allocation and other task bundles in the auction, this approach cannot guarantee the tasks that are originally able to be allocated separately in a model without task bundles can still be allocated after being split from a task bundle.

The aim of bundling is to shorten the task allocation process in TeSSI auctions. With the help of bundling tasks, the number of rounds needed in an auction is reduced to the number of bundles (including single-task bundles) to be allocated in the auction. However, the process of bundle generation itself takes time. Combining tasks into bundles can also cause deviation from optimal solutions, as the tasks in the same bundle are forced to be assigned to the same agent. The analysis of applying bundles, including the decrease in computation time and the loss in solution quality, are discussed in section 5.2.2.

3.4 Replanning

Airport ground handling is operated in a dynamic environment. Lots of uncertainties exist and disruptions occur often. For example, removing the baggage of a no-show passenger can delay the baggage loading operation. Also, malfunctions of GSE vehicles are common reasons for delayed ground handling. These disruptions often happen during the handling of the operation, and they cannot be taken into account beforehand in the task allocation phase. To tackle operation disruptions, the design of the multi-agent system for automated ground handling must be flexible and resilient. Possible re-allocation of ground handling tasks should be considered in the design of the system. In this research, a replanning function is used to respond to disruptions.

Delays of ground handling tasks and malfunctions of GSE vehicles are interpreted as prolonged task execution times. Because the disruptions are unpredictable, the resulting prolonged task duration may exceed the time windows of the ground handling tasks. To successfully reallocate the tasks which had not started to be processed at the time of disruption, several adjustments are proposed. The allocation of tasks whose pickup sub-tasks were already completed before the disruption is kept as planned. Also, the latest completion time constraint is removed for the disrupted task. Then, TeSSI auction is repeated with agents who already contain partial sequences of tasks. From agent's point of view, while generating bids using the pickup and delivery optimizer, the pickup and delivery times and the node connections set before the time of the disruption remains the same.

We perform the experiments simulating a set of disruptions of ground handling tasks. The setup of the experiment and the analysis of the replanning function in our model is presented in section 5.2.3.

4 Path Planning Algorithm and Modeling Methods

After the allocation of ground handling tasks, paths for GSE vehicle agents have to be planned. The planned paths for multiple agents should allow the agents to travel from their start locations to goal locations simultaneously without collision with other agents or obstacles in the environment. We modeled the path planning of GSE vehicle agents as Multi-Agent Path Finding (MAPF) problem. In this research, Safe Interval Path Planning (SIPP) [12] is applied as the path planning algorithm. The environment of the model is set up as grids with obstacles. In realistic ground handling operations, the GSE vehicles have different shapes and sizes. To better model the vehicle movements considering the geometry of vehicles, different sizes of agents are considered. We only used squares of different sizes to model the various sizes of GSE vehicle agents for simplicity of the model. Further, to perform the path planning for agents whose sizes are larger than unit grid, we implement LA-SIPP, a variant of SIPP that is able to handle agents with larger sizes.

4.1 Prioritized SIPP

Safe Interval Path Planning (SIPP) is a single agent path finding algorithm proposed by Phillips and Likhachev [12]. SIPP plans a path for a single agent and it considers other agents as dynamic obstacles. When applied to multi-agent problems, prioritized SIPP is used and it takes pre-defined priority order as input. It transforms the paths of agents with higher priority into (non-)safe intervals for cells on the map and stores them in a reservation table. Safe Interval Path Planning with reservation table (SIPPwRT), a generalization of SIPP introduced by Ma et al. [13] introduced a similar algorithm. In this research, we implemented a prioritized SIPP algorithm that is inspired by SIPP [12] and SIPPwRT [13].

4.1.1 Prioritized SIPP in the model

Based on the allocation result of ground handling tasks, trip itineraries for GSE vehicle agents are formed. The trip itinerary shows the start location, the intermediate stops, and the final destination for an agent. For example, a trip itinerary for a refueling truck agent might be: *depot* → *bay 1 entrance* → *refueling location at bay 1* → *bay 1 exit* → *depot*. Every trip segment in the itinerary belongs to a separate part in the modeling environment, either the service road environment or one of the bay environments. Every trip segment can be modeled as a path finding problem for the agent. To plan paths for all involved GSE vehicle agents, the trip itineraries are

ordered by the type of GSE vehicle used, which is shown in Table 1. Within the same type of GSE vehicle, the trips are sorted by their start times.

Algorithm 3 shows the procedure of prioritized SIPP in the model. The inputs of the algorithm are the set of agents $A = \{a_1, a_2, ...a_n\}$, the trip itineraries G_a for agents, the map of the service road and the aircraft stand, and the safe intervals of the grids on the service road and the aircraft stand. According to the priority order for different types of GSE vehicles list in Table 1, paths for agents are planned. Paths in the itinerary of the same agent are planned in sequential order. The paths either belong to the service road environment or one of the aircraft stands. Corresponding maps and safe intervals are used as the input for SIPP. Once a path is planned, it is stored in P_a, and the safe intervals of the grids in the corresponding environment are updated. The way that safe intervals are updated is discussed in the appendix of this research paper. The outputs of the model are sets of paths P_a for agent a.

Algorithm 3 Prioritized SIPP

Input:
  A: Set of agents
  G_a: Sets of trip segments (itinerary) for agents a, $a \in A$
  M_{road}: Map of the service road
  M_{bay}: Map of the aircraft stand (bay). It is identical for all bays
  S_{road}: Safe intervals of grids on the service road
  S_{bay_b}: Safe intervals of grids on the b^{th} aircraft stand (bay)
Output
  P_a: Sets of paths for agent a, $a \in A$
Initialize S_{road}, S_{bay}
for $a \in A$ **do**
  for *segment* $\in G_a$ **do**
    if *segment* belongs to the service road **then**
      $path = SIPP(segment\ start, segment\ goal, M_{road}, S_{road})$
      add *path* to P_a
      $S_{road} = updateSafeInterval(path, M_{road}, S_{road})$
    else if *segment* belongs to an aircraft stand **then**
      $b =$ id of the aircraft stand
      $path = SIPP(segment\ start, segment\ goal, M_{bay}, S_{bay_b})$
      add *path* to P_a
      $S_{bay} = updateSafeInterval(path, M_{bay}, S_{bay})$
    end if
  end for
end for

4.1.2 SIPP Algorithm

For a single trip segment of an agent, SIPP is used as the path planning algorithm. The strength of SIPP is that it utilizes the concept of safe intervals. A safe interval is defined as a continuous period for a configuration of agent (positions; grids occupied by the agent), during which there is no collision. One timestep prior and one timestep after the safe interval, the configuration is in collision with dynamic obstacles. The use of safe intervals significantly reduced the computation time, making SIPP suitable for online real-world applications.

SIPP applies a modified A* search, which is similar to usual A* except for the way it gets the successors of a state and updates the time intervals for states. Instead of searching agent configurations and time steps, SIPP searches in a state space consisting of pairs of agent configurations and intervals. Similar to A*, g-values representing the current costs of the states from the start and h-values representing the heuristic costs of the states to the final destination are used in the search. Also, in the SIPP algorithm, a function *possibleMoves* is designed to return the possible motions from states. The resulting configurations of the motion are checked for their safe intervals. For each of the safe intervals in the new configuration, a successor with the earliest possible arrival time without collision is generated. Function *getSuccessor* is designed to generate successors of given states.

Readers are referred to the research of Phillips and Likhachev [12] for a detailed explanation of the SIPP algorithm. To better fit our need for path planning for GSE vehicle agents, we modified function *possibleMoves* and function *getSuccessor* in the original SIPP in our model. In the appendix of this research paper, we discuss the two functions which differentiate our proposed method from the original SIPP algorithm.

4.2 Prioritized SIPP for Large Agents

In real-world aircraft ground handling, GSE vehicles have different geometry shapes. In the extended version of the model, agents with different sizes are also included. They are modeled as squared shapes with sizes larger than one grid. To deal with such agents, some adjustments to SIPP must be conducted. Ma et al. proposed the concept of time offsets in the SIPPwRT algorithm [13] that is able to deal with circular agents of different sizes. In this research, we apply similar ideas and generalize them to agents with arbitrary shapes. The algorithm is called Large Agents Safe Interval Path Planning (LA-SIPP).

In LA-SIPP, the search for state space is similar to that of the original SIPP. Elements of LA-SIPP that differ from SIPP are the generation of successors and the way that the safe intervals are updated. Below we discuss the generation of successors in LA-SIPP. Algorithm 4 shows the procedure of successor generation in LA-SIPP. The inputs of the function are the current state s, the map used for path planning M, and the corresponding safe intervals S. In the beginning, *successors* is set to be an empty set. Then, for all possible movements, the configuration of agent *cfg* results from the movement *move* is checked. If the resulting configuration *cfg* collides with the static obstacle on the map M, the current *move* is discarded and the algorithm continues to search the next *move*. If no collision happens, the earliest and the latest time that the agent can reach the configuration *cfg* are generated. The earliest time *start* is the timestep of the current state s plus the required time to perform the movement *move*, while the latest time *end* is the end time of the current safe interval plus the required time of *move*. The accessible interval of *cfg* is first set to be an interval from time step zero to infinity. Then, a loop is ran for all grids in *cfg*. The accessible intervals are updated as the intersection of the interval $[start, end]$, the safe intervals of the grid $S(grid)$, and the current accessible intervals. The resulting accessible intervals are checked after all grids in *cfg* are explored. If the accessible interval is an empty set, the algorithm continues to search for the next possible *move*. Finally, the states of configuration *cfg* with all the intervals in accessible intervals are generated. The generated state is added to the *successor*. The output of the function is a set of successor states.

Algorithm 4 Function *getSuccessors-LA*

Input:
  s: The current state
  M: The map used for path planning
  S: Safe intervals of grids on the map
Output
  successors: successors of state s
$successors = \emptyset$
for $move \in$ possible movements **do**
  $cfg = cfg(s, move)$ {configuration resulting from $move$}
  if agent in *cfg* collides with static obstacle on map M **then**
    Continue
  end if
  $start = timestep(s)$+required time of *move* {the earliest time the agent can reach *cfg*}
  $end = max(\mathrm{S(s)})$ +required time of *move* {the latest time the agent can reach *cfg*}
  accessible intervals(cfg) = $[[0, \infty]]$
  for $grid \in cfg$ **do**
    accessible intervals(cfg) = $[start, end] \cap S(grid) \cap$accessible intervals($cfg$)
  end for
  if accessible intervals(cfg) = $\emptyset$ **then**
    Continue
  end if
  for $i \in$accessible intervals **do**
    $t =$ start time of i
    $s' =$ state of configuration *cfg* with interval i and time t
    $successors = successors \cup s'$
  end for
end for

5 Experimental Analysis

To analyze the behavior of the designed multi-agent system for automated aircraft ground handling and to investigate the performance of the model, experiments are performed. Experimental design is based on the

assumptions discussed in section 2.1 and the model environment specifications presented in section 2.2. For the task allocation experiments, the size of aircraft stand map is not seen as a critical parameter, thus the basic model is used. Path planning experiments are also performed, using both the basic model and the extension model. Moreover, a sensitivity analysis section is included.

5.1 Experimental Setup

For all experiments, the following settings remain unchanged unless specified. A model with three aircraft stands and a service road is used. Flight schedules of the aircraft stands are generated in a planning window of four hours. The turnaround times of flights are X minutes, where $X \sim U(40, 50)$. The turnarounds of the following flights start immediately after the turnaround time of the previous flights end. The task duration and the time windows that the ground handling tasks must be performed are identical for all flights. Figure 2 shows the time windows and task duration of ground handling tasks. The time windows are calculated from the arrival time of flights.

Different types of ground handling tasks of the flights, including refueling, catering, baggage handling, water servicing, and toilet servicing, are allocated separately to limited numbers of ground handling vehicles. The bidding rule of minimizing *makespan* is used. In experiments with a three-aircraft-stand model, there are two available vehicles for each type of ground handling task except baggage handling. For baggage handling, four available vehicles are used because both unloading and loading operations are needed for a flight. The depots of the GSE vehicles are located at randomly selected grids connected to the service road on the maps. The distance buffer coefficient handling uncertainties is Y, where $Y \sim N(1.4, 0.2^2)$.

For all pickup and delivery optimization for agents during the bidding phase, Gurobi 9.5.0 was used to perform the optimization. All simulations are conducted on an 8-core Apple M1 chip with 8GB RAM. For every pickup and delivery optimization, a solution time limit of 5 seconds is applied.

5.2 Task Allocation Experiments

To evaluate the task allocation mechanism, we run experiments to test our multi-agent system model for automated ground handling. Experiments are designed to investigate the model performance for different traffic demand scenarios and task allocation objectives. The performances of the task bundling and replanning are also evaluated by the experiments.

For each experiment in the following, 100 simulations are performed. The discussions of chosen numbers of simulations to perform can be found in the appendix of the thesis report. In Table 4, 5, and 6, for rows showing results of different types of ground handling tasks, the numbers are median values retrieved from the simulations. For the rows showing overall model performances, mean values of all types of ground handling tasks are used in the columns of makespan and allocation rates, while in the columns of computational time, the summations of computational times for different types of tasks is used. To test the statistical significance, we use the non-parameteric Wilcoxon signed-rank test as the values retrieved from experiments are paired but not normally distributed. The results are reported in p-values, which are shown in Table 4, 5, and 6, and 7. Also, the effect sizes of all pairs of experiments are reported in Vargha-Delaney A-values. The A-values reveal the probability that the experimental values (makespans, allocation rates, or computational times) in one data set is larger then the values in another set of data. For comparisons that lead to statistically significant ($p < 0.05$) or large effect sizes ($A > 0.71$ or $A < 0.29$), the p-values or A-values are shown in bold.

5.2.1 Experiment A: Traffic demand scenarios

In this experiment, scenarios of different traffic demands are tested. Three different flight schedules are used in this experiment. We use turnaround times of 50 to 60 minutes, 40 to 50 minutes, and 30 to 40 minutes to simulate the traffic demand of offpeak hours, normal hours, and busy (peak) hours in an airport terminal. A longer turnaround time does not necessarily mean that the ground handling tasks takes longer to perform. It is used to represent longer waiting time between the arrival of two successive flights because the task duration and the time windows of ground handling tasks are consistent among all flights. For a three-aircraft-stand model with a four-hour planning windows, there are in total 13, 16, and 20 flights in the offpeak, normal, and busy hours schedule, respectively.

We investigate the makespan, the task allocation rates, and the computational time of the model. Especially, the performances of the model's makespan in three different traffic demand scenarios are what we would like to further explore. Higher *makespan* are expected to be found for the busy hour schedule as there are more numbers of ground handling tasks to perform. Therefore, we hypothesize that:

- $H_{A1.1}$: The makespan for the normal hours schedule is higher than that of the offpeak hours schedule.
- $H_{A1.2}$: The makespan for the busy hours schedule is higher than that of the normal hours schedule.

Table 4: Comparison of the simulation results of different traffic demand scenarios.

scenario	makespan (min)	task allocation rates	computational time (s)
offpeak	153.81	0.9846	38.89
normal	213.40	0.9375	85.59
busy	200.79	0.8500	192.38
offpeak-normal p	**3.90E-18**	**1.41E-18**	**3.90E-18**
normal-busy p	**5.27E-18**	**3.25E-18**	**3.90E-18**
offpeak-normal A	**0.0000**	**1.0000**	**0.0000**
normal-busy A	**0.9852**	**0.7143**	**0.0000**

The makespan, the task allocation rates, and the computational time of the model in different scenarios are shown in Table 4. To distinguish the individual distributions between data sets, we test statistical significance and effect sizes of both the offpeak-normal traffic demand pair and the normal-busy pair. Instead of showing results for individual ground handling tasks, Table 4 only shows results of the overall performance of the model due to limited space. The complete results can be found in the appendix of the thesis report.

As there are more flights and thus more ground handling tasks in the normal scenario than in the offpeak scenario, the makespan of the normal scenario is higher than that of the offpeak scenario as expected. The p-value comparing the two data sets confirms that $H_{A1.1}$ is supported. However, with even more flights, the makespan of busy scenario is proven to be lower than that of the normal scenario. The result is possibly related to the lower task allocation rate in the busy scenario. As a result, $H_{A1.2}$ is rejected. In our model, the flight schedule of the normal scenario is not a subset of the busy scenario schedule. Not all allocated tasks in the normal scenario can be allocated in the busy scenario. Some GSE vehicle agents may be allocated with more tasks in the normal scenario than in the busy scenario. The makespan in the normal scenario is thus higher than that of in the busy scenerio.

Task allocation rates in the three scenarios are also compared. Table 4 shows that scenarios with busier schedule have lower task allocation rates. The result is as anticipated since the number of flights increases in different traffic demand scenarios, while the numbers of available GSE vehicles remain the same. Lastly, we compare the computational time in the three models, as shown in the last column in Table 4. The comparison also tells about how models scale while increasing the number of flights (and thus the number of ground handling tasks). For the offpeak traffic demand scenario, the task allocation for the ground handling tasks for all flights in the 4-hour planning window takes around 38 seconds. The required allocation time doubles when the numbers of flights increases from 13 to 16 in the normal traffic demand scenario, and it doubles again when the number of flights increases from 16 to 20 in the busy scenario. As the numbers of flights used in the models are much smaller than realistic scenarios, the allocation mechanism or the utilized optimizer still need some improvement to fit the need of online applications.

5.2.2 Experiment B: Bundling

To possibly shorten the task allocation procedures, bundles of tasks are assigned using TeSSI auctions. In this experiment, the bundling mechanism is tested and evaluated. The maximum vehicle makespan, task allocation rates, and the model computational time are reported.

We expect higher vehicle makespan for the model with bundled tasks. In the model with task bundling, tasks in the same bundle cannot be allocated to different agents. This may effect the optimization of the task allocation process and cause some increments in the makespan. Still, bundles are formed using the bundle generation heuristics, which aims at making good bundles, taking temporal and spatial proximity among tasks into account. Thus, it is expected that the makespan for the model with bundled tasks is higher than that with un-bundled tasks only by a small amount. Regarding the allocation rate, we expect the model with bundled tasks has lower task allocation rates than the one without. Although tasks bundles that cannot be assigned to any agents are broken down into individual tasks in the model, the separated individual tasks are allocated in the last few rounds of the auction. Agents with allocated bundled task sets have less flexibility in their schedule. Therefore, the remaining separated individual tasks have less opportunities to be allocated. Most importantly, the computational time of model with bundled tasks is expected to be lower as task bundles help reduce the number of auction rounds. Computational times of models with and without bundles are important performance indicators to evaluate the mechanism of task bundling. The above-mentioned expectations can be expressed in the hypotheses listed below.

- H_{B1}: The maximum vehicle makespan for the model with task bundles is higher than that of the model without task bundles.

Table 5: Comparison of the simulation results for bundling.

	makespan (min)				task allocation rates				computational time (s)			
task	no bundle	bundle	p	A	no bundle	bundle	p	A	no bundle	bundle	p	A
RE	202.62	237.87	**3.89E-18**	**0.0072**	0.8750	0.8125	**2.86E-19**	**0.9750**	20.29	52.78	**3.90E-18**	**0.0000**
CA	220.47	224.14	**3.88E-18**	**0.0000**	1.0000	1.0000	**1.97E-09**	0.3200	15.08	22.02	**3.90E-18**	**0.0017**
BA	221.05	241.88	**7.30E-18**	**0.0600**	1.0000	1.0000	**2.44E-02**	0.5300	1.32	3.55	**3.90E-18**	**0.0000**
WA	199.97	218.82	**9.04E-16**	**0.1779**	0.8750	1.0000	**9.91E-19**	**0.0318**	32.20	36.25	**4.02E-18**	**0.0049**
WC	222.88	229.38	**3.88E-18**	**0.0000**	0.9375	0.8125	**1.73E-19**	**0.9906**	16.70	40.10	**3.90E-18**	**0.0000**
overall	213.40	230.42	**3.90E-18**	**0.0019**	0.9375	0.9250	**1.47E-10**	**0.7773**	85.59	154.69	**3.90E-18**	**0.0000**

- H_{B2}: The task allocation rates for the model with task bundles are lower than that of the model without task bundles.
- H_{B3}: The computational time for the model with task bundles is lower than that of the model without task bundles.

Table 5 shows a comprehensive comparison of simulation results of bundling. We test H_{B1} by comparing the makespans of model with and without utilizing bundles. The p-values for all ground handling tasks show significant differences between the values of the two groups, therefore, H_{B1} is supported. The Vargha-Delaney A-values show that the increments in vehicle makespans caused by bundling are considerable. This result implies that although the bundle generation heuristics take temporal and spatial proximity among tasks into account, the bundling results are not ideal as they affect the overall optimality of the auction. It is also possible that pickup and delivery optimization takes longer time to solve for some of the bundled tasks. In these cases, optimization process is interrupted and the agent loses the opportunity to bid on the current task bundle.

Regarding the task allocation rates, H_{B2} is tested by comparing the allocation rates of the two groups. The p-values show that the allocation rates of the two models have significant differences. For the refueling and the toilet servicing tasks, as expected, the task allocation rates with bundles are smaller. H_{B2} is supported for these two tasks. For the catering and baggage handling tasks, the medians of the task allocation rates are equal to 1 for both tasks, and the Vargha-Delaney A-values indicate that the effect sizes are small. For these tasks, the solution times are also shorter than that of the other tasks. If most of the bids can be generated in time using SPDP optimization, better task allocation rates can be reached. Lastly, for the water servicing task, the allocation rate is higher in the model with bundles. Thus, H_{B2} is rejected for the water servicing task.

We compare the computational time of the model with and without bundling tasks to test H_{B3}. The computational time column "bundle" shows the total computational time of task allocation and bundle generations. The p-values reveal that for all ground handling tasks, there are significant differences in the computational times of the two models. However, the difference is not in the trend we expected. The Vargha-Delaney A-values also reveal large effect sizes between data from the two groups. In the appendix of this research paper, the detailed task allocation computational time and the bundle generation time of the model are presented. The great increases in computational times are not caused by the generation of bundles. It is possible that the pickup and delivery optimizer has some difficulties dealing with tasks in the same bundles while optimizing schedules for agents. On the other hand, by generation of bundles, some of the time windows might have become tighter, which were originally generating cuts on the solution space and converging faster. For this reason the bundled version could be faster. However, this effect is not observed in our experiment.

After testing the three hypotheses listed above, we found that the performance of bundling in the multi-agent system for automated ground handling is far from what we have expected. In order to help shorten the allocation process of the TeSSI auctions, the design of the bundling mechanism still needs some improvements.

5.2.3 Experiment C: Replanning

To tackle disruptions happening during the ground handling operations, a replanning function is implemented. In this experiment, we evaluate the performance of the replanning function by simulating two scenarios with the same disruptions. In one of the simulation sets, the allocation of tasks remains unchanged. The completion time constraints of all tasks that are not performed or partly performed at the time of the disruption are prolonged by the length of delay. In another set, the replanning function is activated. The function works as described in section 3.4. The following settings are used for the simulation scenario. For every ground handling task, the probability of being disrupted is 20%. When a disruption happens, the processing duration of the chosen ground handling task will be increased by 20%. We examine the makespan in both scenarios.

When a disruption happens, the replanning function is expected to more efficiently allocate the un-completed tasks to GSE vehicle agents given their limited availability. A disrupted and delayed task in a GSE vehicle's schedule may cause more delays on the following scheduled tasks of the vehicle. Therefore, the makespan of the simulation set without replanning is expected to be higher than the one with activated replanning function. In

Table 6: Comparison of the simulation results of test scenarios with and without disruptions.

	makespan (min)				task allocation rates			
task	replan	no replan	p	A	replan	no replan	p	A
RE	221.68	199.72	**2.01E-15**	**0.8186**	0.9375	0.8750	**2.78E-13**	**0.8650**
CA	220.73	223.63	**1.08E-01**	0.4036	0.9375	1.0000	**2.20E-08**	**0.2720**
BA	233.75	225.02	**8.63E-18**	**0.9902**	1.0000	1.0000	**6.33E-05**	0.4200
WA	200.88	200.12	**1.12E-07**	**0.7836**	1.0000	0.8750	**2.73E-19**	**0.9888**
WC	222.95	219.38	**4.12E-18**	**0.9887**	0.9375	0.9375	1.32E-01	0.5247
overall	220.00	213.57	**1.37E-17**	**0.9605**	0.9625	0.9375	**8.53E-17**	**0.9344**

the simulation set without a replanning function activated, we do not redo the task allocation so the allocation remains the same as in the scenario of no disruption happening. On the other hand, in the simulation set with an activated replanning function, part of the allocation is re-performed when disruptions happen. The replanning phase gives opportunities to the tasks that cannot be allocated in the beginning (due to limited computational time or the order of tasks in the auction) a second chance to be bid by agents. Thus, the task allocation rates in the model with replanning are expected to be higher. The hypotheses based on the above discussions are formulated below.

- H_{C1}: The makespan of the simulation set with an activated replanning function is lower than the one without an activated replanning function.
- H_{C2}: The tasks allocation rate of the simulation set with an activated replanning function is higher than the one without an activated replanning function.

The comparison of the simulation results of test scenarios with and without the replanning function is shown in Table 6. We test H_{C1} by comparing the makespan of the two scenarios. For all ground handling tasks except catering, makespans in the replanning scenario are higher. For the catering task, the makespans are lower in the replanning scenario. However, the p-value indicates that there is no significant difference between makespan of models with and without replanning. Therefore, H_{C1} cannot be supported for all ground handling tasks. The results are not expected, and they may be affected by the higher allocation rates in the replanning scenario. Except for the catering tasks, the task allocation rates for all ground handling tasks in the replanning scenario are higher or equal to the allocation rates in the scenario without replanning. The other reason that the makespan in the model with replanning are not shortened is because there are only 20% of tasks being delayed, and the delayed tasks are only prolonged by 20%. If a delayed task lies in the middle of an agent's schedule without effecting the following tasks in the agent's schedule, the makespan of the agent in the model without replanning will not change. In the model with replanning, parts of the allocation of the ground handling tasks are fixed. The remaining tasks are allocated in different orders comparing to the initial order of auction. The different orders of allocation cause different results of agents' makespan. Also, the fixed pickup and delivery connections may hinder the schedule of tasks in agents' schedules to be optimized, resulting in worse makespan for agents.

Hypothesis H_{C2} is tested by comparing the allocation rates of simulations sets with and without an activated replanning function. For all ground handling tasks except the catering task, the allocate rates with replanning are higher than the ones without. Once a disruption happens, part of the tasks have to be reallocated. We found that the reallocation creates opportunities for tasks to change their orders in the auctioning task set. Also, a task might not be able to be allocated in the first round of allocation since the computational time of bid generations for agents of involving the task exceed the limit. In the re-allocaiton rounds, these tasks have opportunities to be involved in different schedules and to be optimized again. Hypothesis H_{C2} is thus supported for all ground handling tasks except the catering task.

5.2.4 Exploring the task allocation objectives

In the adapted TeSSI auction for ground handling task allocation, different bidding rules can be applied. Two common objectives to be minimizes are *makespan* and *sum-T*, as mentioned in section 3.1.1. In this experiment, we run simulations of task allocation with the two bidding rules. We would like to know the performance of the model regarding the task allocation rates and the computational time using the two different bidding rules.

The comparison of the simulation results using the objective *makespan* and *sum-T* is shown in Table 7. There are large differences in the task allocation rates and the computational time of the model. Except for the catering and baggage handling tasks, the allocation rates of the model using the objective *makespan* is smaller than using the other objective *sum-T*. Using different objectives, the agents bid different values on the same tasks, resulting different task allocation. The tasks allocated in earlier rounds may affect the tasks in later

Table 7: Simulation results of different objectives.

	task allocation rates		computational time (s)	
task	makespan	sum-T	makespan	sum-T
RE	0.8750	1.0000	20.29	4.06
CA	1.0000	1.0000	15.08	23.78
BA	1.0000	1.0000	1.32	1.48
WA	0.8750	1.0000	32.20	21.40
WC	0.9375	1.0000	16.70	13.37
overall	0.9375	1.0000	85.59	64.10

rounds being allocated. The other reason that the task allocation rates are smaller using objective *makespan* can be associated with the computational time. If no solution of the SPDP optimization problem containing a certain ground handling task can be found within the 5-second time limit, the optimization will be aborted. As the agent's bid cannot be generated, the agent loses its opportunity to bid on the task. In such situations, the task allocation rates can indirectly be affected.

There are also large differences in the model performances in terms of computational time. Computational times for task allocation of different ground handling tasks are quite distinct. For example, the computational time for baggage handling is significantly lower then computational time for other ground handling tasks. As the constraint related to unit capacity of baggage handling generates a strong cut on the solution space since the alternative links and nodes are highly reduced compared to non-unit capacity constraints. Therefore, the optimization algorithm only needs to explore a limited solution space, resulting in lower computational time. For the same types of ground handling tasks, there is considerable difference between the computational times when the model is run using different objectives. Moreover, computational times for running the pick up and delivery optimization for different ground handling tasks are variable. For the catering and baggage handling tasks, the computational time of allocations are comparatively lower than other ground handling tasks, using the objective *makespan*. The models behave oppositely for other types of ground handling tasks.

5.3 Path planning experiments

To evaluate the algorithm of path planning, experimental analyses are performed. Keeping all settings in section 5.2 unchanged, we utilized the result of task allocation as input to plan paths for the GSE vehicles. We run the path planning simulation in two modeling environments, the basic path planning model and the extension model, as introduced in section 2.2. In the basic model, agents are assumed to have the size of unit grid. Agents move one cell per time step, which is equivalent to a speed of $4m/s$ in our environment settings. On the other hand, in the extension model, agents have sizes larger than unit grid. For simplicity, we assume all the modelled GSE agents are all 3×3 grids in size. This assumption can of course be lifted by replacing the agents' size with various values. Agents in the extension model also move a cell every time step, which is also equivalent to a speed of $4m/s$.

We investigate the length and duration of paths for the GSE vehicle agents in both models. Also, the shortest possible path length and duration without considering conflicts with other agents are listed for comparison. For both models, the mean values of the average path length and duration in the performed simulations are listed in Table 8. The path duration are longer than the required traveling times for agents based on the presented path lengths for both models. As the speed of agents are constant throughout the journey and for all agents, the results show that the agents sometimes stop and wait on their route to avoid conflicts with other agents. For the basic model and the extension model, respectively, the lengths of paths are 1.0012 and 1.0014 times the shortest path length. The two numbers reveals that agents' traveling distances are barely longer than the shortest possible distance without considering other agents. Utilizing prioritized SIPP, an algorithm that cannot guarantee optimality, the obtained results show that the path planning algorithm is able to generate good solutions in our problem settings. The path duration are 1.28 and 1.39 times the shortest duration. Comparing with the ratio of (actual) path length to shortest path length, the ratio of (actual) path duration to shortest duration are much higher. This result discloses that agents tend to stop and wait on the path while encountering conflicts instead of taking detours. This behavior can be interpreted as follows: Substitution paths might not exist in the environment. Also, it is possible that the obstacles (agents with higher priorities) only occupy the paths for a short period, so agents are willing to wait rather than detour. Note that the shortest path lengths and durations are not achievable in scenarios with multiple agents. They are only listed in the table to show the qualities of planned paths using prioritized SIPP.

In Table 8, we also show the average delay per agent and the average delay per task. For both models, the average delays per agent are higher than the average delay per task. The result indicates that agents are assigned multiple tasks, and delays happen on multiple tasks of agents. The results of agent on time rate and

Table 8: Performance indicators of path planning in the basic and extension model.

performance indicators	basic model	extension model
path length (m)	400.63	372.76
shortest path length (m)	400.15	372.23
path duration (s)	128.45	129.09
shortest path duration (s)	100.04	93.06
average delay per agent(s)	22.28	37.96
average delay per task (s)	15.49	27.66
agent ontime rate	0.94	0.93
task ontime rate	0.96	0.95
agent success rate	1.00	0.98
computational time (s)	5.11	86.83

task on time rate also have the same trend. The average task delays, being 15 and 28 seconds for the basic model and the extension model, respectively, are both within acceptable ranges for online usage. The high on time rates are also considered quite sufficient in real-world cases.

The algorithm used for path planning, prioritized SIPP, is a decoupled approach. As paths for agents are planned one after another based on the priority order, complete solutions are not guaranteed. The success rates that agents can successfully find paths in the modeling environment are also listed in Table 8. Although the success rate is not 1 in the extension model, it is still quite high. Moreover, only a single priority order is applied in our model. If different priority orders are tried when the path planning failed, the success rate for agents can be further increased. The last row in Table 8 presents the computational time for both models.

5.3.1 Analysis on heat maps

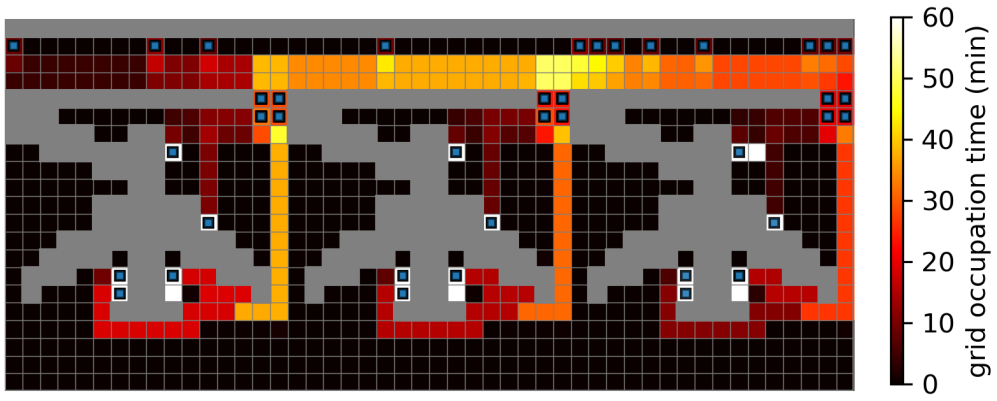


Figure 8: Heat map of planned paths in the basic model. The blue squares show the depots of GSE vehicles, the entrances and exits of bays, and the locations for ground handling task execution. The color bar besides the figure shows the average time that the grids are occupied over 100 simulations.

The path planning result for GSE vehicles can also be investigated using heatmaps. Figure 8 and Figure 9 show the heat map of the planned paths in both models. One can observe from the heat maps that the locations the ground handling tasks take place are where the GSE vehicle agents stay the longest. The intersections between the service road and bay areas are also places that are occupied most. Also, the middle section of the service road connecting the two bay entrances on the left is an area with busy traffic. However, the figure only shows the frequency that grids are occupied by agents, and it does not show the waiting time for agents. Hence, we can not be sure whether the grids with higher values are bottlenecks on the map that cause congestion or not. Moreover, from the grids with brighter colors on the map, we can tell that most agents take the shortest possible routes instead of taking detours. This result help us to confirm the discussion we derived from Table 8 in the previous paragraphs.

Interestingly, some grids next to the task locations, especially the baggage handling locations, also have high values in the heat map. This phenomenon can be explained as follows: It is possible that a baggage loading vehicle arrives at the task location before the baggage unloading operation of the same aircraft is complete, so the early-arrived vehicle wait beside the task execution location before it can enter the cell. For other types

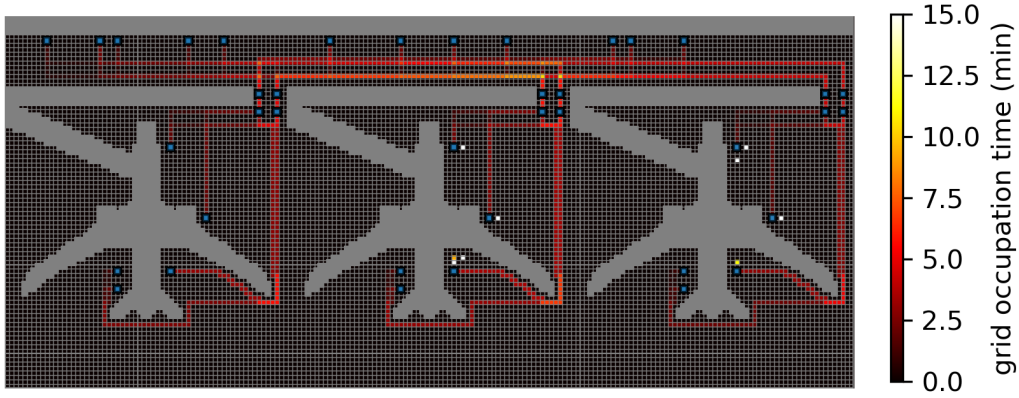


Figure 9: Heat map of planned paths in the extension model. The blue squares show the depots of GSE vehicles, the entrances and exits of bays, and the locations for ground handling task execution. The color bar besides the figure shows the average time that the grids are occupied over 100 simulations.

of ground handling tasks, it is possible that agents assigned with the ground handling tasks of the next flights are already waiting aside. The objective of task allocation, *makespan*, only concerns about the starting time of the first task and the finishing time of the last task in the model. Thus, the times that agents depart for other tasks do not affect the objective. These circumstances can possibly cause some congestion on aircraft stands. To improve the matter, other objective functions can be designed. On the other hand, if the task allocation objectives are kept, the starting time of agents' trip itineraries can be adjusted to deal with agents' idle time on aircraft stands.

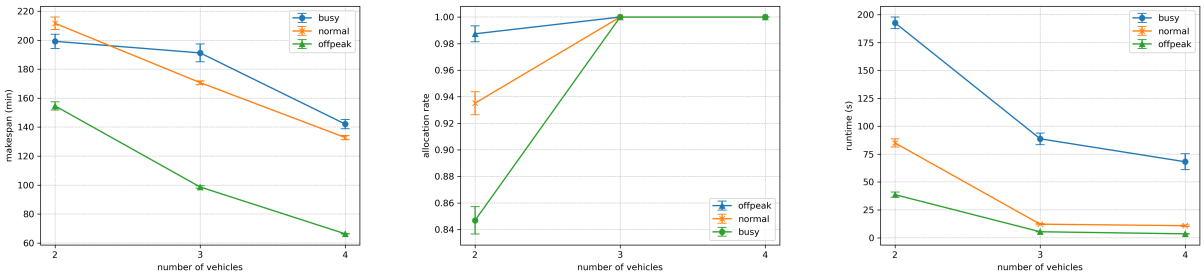
5.4 Sensitivity Analysis

In the previous experiments, the numbers of available ground handling vehicles are fixed. Consulting experts in ground handling operations, in the previous experiments, we used the lowest possible numbers of ground handling vehicles that are able to perform the tasks for all flights within the planning window. The numbers of different types of GSE vehicles represent the limited resources in task allocation. The more the resources are available, the more flexible the allocation is. Therefore, the numbers of available GSE vehicles are considered important factors in the allocation of ground handling tasks.

5.4.1 Analysis on numbers of available GSE vehicles

Changing the number of GSE vehicles, we investigate the three main performance indicators: makespan, task allocation rate, and computational time in different traffic demand scenarios.

Figure 10 shows the simulation results when we change the number of available GSE vehicles. The available number of vehicles for the refueling, catering, water servicing, and toilet servicing is set to be 2, 3, or 4 for each task. The numbers of vehicles used for baggage handling are 4, 6, or 8 in the simulation. The number of vehicles is doubled because both unloading and loading operations are needed for a flight.



(a) Makespan (b) Allocation rate (c) Computational time

Figure 10: Performance indicators of the model in three traffic demand scenarios by varying the number of available vehicles.

Figure 10(a) shows the makespan of different traffic demand scenarios by changing the number of available vehicles. When the number of vehicles increases, makespan of the model decreases, as a higher number of vehicles can share the same set of ground handling tasks. We can also observe that the busier the traffic demand scenario is, the higher the makespan. The only exception occurs when the number of vehicles is 2. This is related to the allocation rates when the number of vehicles is restricted. Among all traffic demand scenarios, the highest number of flights, and thus the highest number of ground handling tasks exist in the busy scenario. The restricted resources do not allow all ground handling tasks to be successfully allocated, resulting in a lower allocation rate. This case is also shown in Figure 10(b). When the available number of vehicles is 2, not all ground handling tasks can be allocated. The scenarios with fewer flights have higher allocations rates. When the number of available vehicles increases to 3, in all scenarios, all ground handling tasks can be allocated.

We also examine the computational time the model takes to complete the allocation, as the solution time is a crucial parameter in evaluating whether the system is suitable for online applications. In Figure 10(c), it is obvious that the computational time for the busy scenario is longer than the computational time in the other two scenarios. A similar result is also shown in section 5.2.1. Also, we can also observe that in the same traffic demand scenario, the computational time decreases when the number of available vehicles increases. The solution time does not rise when there are more agents involved in the auction, generating more bids. Instead, more available vehicles result in fewer tasks in one vehicle's schedule, and fewer numbers of tasks greatly save the SPDP optimization time for an agent to generate bids. This circumstance is especially evident in the busy scenario. The analysis demonstrates that our model is quite sensitive to the number of available GSE vehicles.

5.4.2 Analysis on task processing duration

The other factors we believed to have some affect in task allocation is the execution duration of tasks. In current ground handling operations, the duration of most types of tasks are close to the ones presented in Figure 1. Task duration for most ground handling tasks do not vary a lot except the refueling task. The refueling time of a flight greatly depends on the destination of the flight and whether the aircraft carries fuels for the return flight. For example, for European flights departing from Amsterdam, the refueling duration ranges from 10 to 25 minutes. Because the refueling tasks are the ones that have most distinct task durations, task allocation of refueling is investigated in the sensitivity analysis.

We run simulations in different traffic demand scenarios, using the offpeak, normal, and busy flight schedules. Available numbers of refueling vehicles are set to be 2, 3, and 4. The refueling duration varies from 10 to 25 minutes. For each vehicle-duration combination, 100 simulations are run. The bidding rule *makespan* of task allocation is applied. The result is shown in Figure 11.

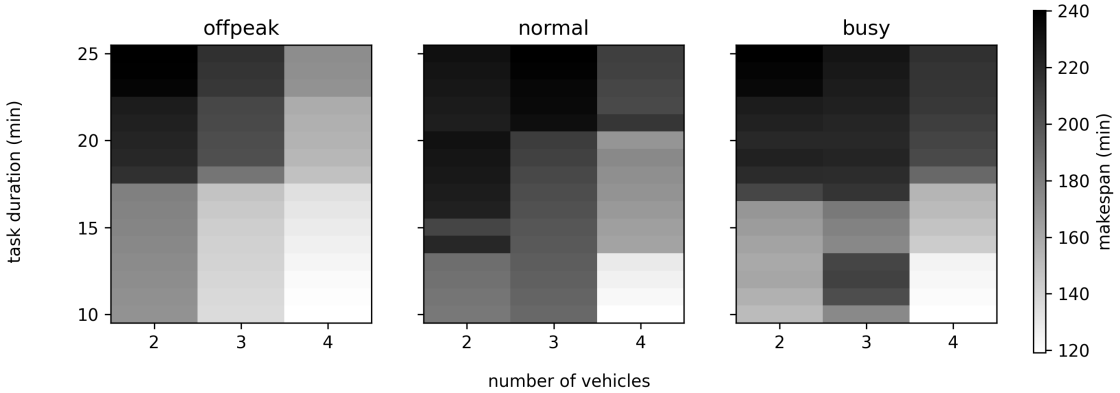


Figure 11: Contour plot of maximum vehicle makespan with the *makespan* bidding rule for three traffic demand scenarios, with varying numbers of available vehicles and task duration.

The colorbar in Figure 11 shows the maximum vehicle makespan in the models. Within the same scenario, the makespan values tend to be higher with less number of available vehicles. This is reasonable as more tasks are allocated to every vehicle. Also, as the task duration grows, the makespan for vehicles increase. However, the increments are gradual when there are more numbers of vehicles. When the amount of vehicles are limited, makespan for vehicles encounter more sudden changes. We conjecture that when there are only two available vehicles, the resources are quite restricted. As there are only few possible feasible allocations, the allocation does not change much when the task duration is extended. Also, the makespan for vehicles only depend on the vehicles' departing time for the first tasks in their schedule and the returning time after completing the last

tasks. Therefore, the makespan of vehicles also do not vary a lot. When the amount of resources are sufficient, the allocations are more flexible, and they can be better optimized based on the varying task duration. The makespan of vehicles also change depending on the allocation.

Moreover, one can observe from the three plots that in general, busier traffic demand scenarios lead to longer makespan for vehicles. This is not the case for the busy scenario with two vehicles. In such case, it is possible that some tasks cannot be allocated. Lower task allocation rates lead to fewer tasks in schedules of agents. Hence, makespan of vehicles in such cases are not higher than the ones in the offpeak and normal scenarios.

6 Discussion

A multi-agent system for automated aircraft ground handling is designed in this research. In this section, we discuss the obtained results from the experiments conducted and the methods proposed in this model.

We conducted experiments to test the model using different traffic demand scenarios. We also tested the model with task bundles and with a replanning function. Furthermore, different task allocation objectives are explored. Based on the result of the makespan values in Table 4 and 5, the outcome is observed to be supporting the hypotheses. Also, the task allocation rates in Table 4, 5, 6, 7 are in general very close to 1.00. Most allocation rates are higher than 0.875, except one of them being 0.81. However, the required model computational time in the experiments is not as good. The bundling mechanism and the replanning function do not behave as expected either. We believe that part of the reasons for the unwanted results is caused by longer computational time in some experiment scenarios. After a thorough inspection of the model, it turns out that the bids generation using SPDP optimization models takes the longest time. For some instances, it is difficult to find the optimal schedules. This may be improved by formulating the optimization problem in other ways or applying different bid generation methods. Still, by setting an upper bound on the solution time of the optimization, the required computational times for the experiments are within acceptable ranges. Apart from the bids generation time, we believe the use of adapted TeSSI auction is helpful for task allocation of ground handling tasks. The algorithm does not only allow the involved agents to generate bids that represent their preference for tasks, but it can also complete proper allocations in a reasonable time.

Experiments were also conducted on the implemented path planning algorithm. The resulting path lengths and duration are quite close to the shortest possible value without considering other agents. This result indicates that in our model, the application of decoupled approaches can produce results close to optimal ones. Moreover, in the extension model, agents are modeled in different sizes larger than a unit grid. This application helps us to more realistically model the sizes of GSE vehicles. However, the modeling of shapes of agents can still be improved as we only used square shapes. Kinematics of agents, including rotational motions, acceleration, and deceleration are not included in the model. Different agent shapes and agent kinematics can be considered in future developments.

The sensitivity analyses demonstrate that our model is quite sensitive to the number of available GSE vehicles. This result partly reflects the situation in current ground handling operations. In current operations, the most limited resource is the availability of personnel. As limited personnel can only operate a limited number of GSE vehicles, the small fleet size utilized in our model can also reflect the personnel shortage in current ground handling operations. Obviously, with additional resources, the ground handling tasks can be performed better. However, for most aircraft ground handling operators, increasing the amount of available personnel or GSE vehicles can be difficult. Therefore, applying a proper task allocation approach that can efficiently utilize the limited resources considering multiple stakeholders is crucial for aircraft ground handling operators. Centralized optimization can help allocate limited resources efficiently, while decentralized approaches are better mechanisms to capture the interest of multiple stakeholders. Other than using pure centralized or decentralized approaches, the method proposed in our model combines a decentralized task allocation mechanism, TeSSI auction, with centralized local optimization, the SPDP optimization. The proposed approach in this research responds well to this need.

In this research, we focus on the ground handling tasks of flights. The relationship between ground handling task delays and flight delays are not explored in our work. In the experimental research regarding testing the replanning function, we only investigate the task allocation rates after replanning, but not the flight delays caused by the interrupted tasks. It would be beneficial if the flight delays are first estimated before the system re-allocates all the remaining ground handling tasks accordingly. Also, because path planning for agents is performed after task allocation, situations and congestion happening during traveling are unknown in the task allocation phase. Thus, it is not possible to estimate the exact delay of the following tasks if the previous task is delayed. Better explorations on the connection between interruptions of ground handling tasks and flight delays can be considered in future works.

7 Conclusion

In this research, we designed a multi-agent system for automated aircraft ground handling. The system has the capability of task allocation optimization and path planning for multi-agents. As there are multiple stakeholders involved in aircraft ground handling operations, an auction mechanism, adapted TeSSI auction is implemented to perform the task allocation. Agents use the costs for them to perform the ground handling tasks as the bids for the TeSSI auction. The bids are generated by modeling the task scheduling for agents as single-vehicle pickup and delivery optimization problems. Moreover, a bundling mechanism is proposed in the hope of saving allocation time by making individual ground handling tasks into task bundles. To deal with the disruptions that often happen during the ground handling operations, a replanning function is also included in the model. When a disruption happens, the replanning function is able to reallocate the tasks that have not started to be processed. For the path planning algorithm in the model, Prioritized SIPP is applied. Due to the common use of vehicle priorities in airside environments, decoupled path planning approaches with priorities are appropriate approaches to be implemented in the model. To better model the different sizes of the realistic ground handling vehicles, LA-SIPP is implemented in the model.

Experiments on task allocation and path planning of the model are conducted to evaluate the performance of the proposed model. Overall, the experimental results of applying different traffic demand scenarios in the model support the formed hypotheses. However, the bundling mechanism and the replanning function do not behave as expected. One of the reasons for the unwanted results might be the long optimization time in the SPDP optimization. Still, by setting limitations on the optimization time, the model is able to allocate the ground handling tasks for the designed time window and the number of flights within reasonable computational time. Combining a decentralized TeSSI auction and centralized local SPDP optimizations, the proposed model can efficiently utilize the limited resources considering multiple stakeholders for aircraft ground handling.

Furthermore, allocation of ground handling tasks and path planning for GSE vehicles are performed separately in the model. Uncertain events, including delays caused by congestion, are not taken into account during the task allocation phase, making the allocation less efficient. Although we implemented a replanning function that can deal with delayed ground handling tasks, the relationship between delays of ground handling tasks and flight delays is not explored in this study.

To formulate a brand-new multi-agent system model for automated aircraft ground handling, in this research, we made some assumptions based on current ground handling operations, as introduced in section 2.1. To better model the ground handling operations and develop a system that can be applied to real operations, more details can be taken into account. Moreover, we focused on the allocation of ground handling tasks in this study. In the path planning algorithm we implemented in this study, LA-SIPP, square-shaped agents with different sizes are used. The applied geometries of agents can still be improved to better model the realistic operation environment in ground handling. The kinematic of agents can also be improved by considering the rotational motions and acceleration of agents. More sophisticated geometry and kinematic of agents are thus recommended for future works.

References

[1] Joost Soomers. Agent-based delay management in autonomous engine-off taxi systems. 2022.

[2] K. Fines. Agent-based distributed planning and coordination for resilient airport surface movement operations. 2019.

[3] Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph Durham, and Nora Ayanian. Conflict-based search with optimal task assignment. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2018.

[4] 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.

[5] Boeing Commercial Airplanes. 737: Airplane characteristics for airport planning, sept. 2013. Technical report, D6-58325-6, 2013.

[6] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[7] 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, to appear: Journal für Betriebswirtschaft*, 2007.

[8] Sophie N Parragh, Karl F Doerner, and Richard F Hartl. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51, 2008.

[9] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J Kleywegt, Sven Koenig, Craig A Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, volume 5, pages 343–350. Rome, Italy, 2005.

[10] Paolo Rizzo. Auction-based task allocation for online pickup and delivery problems with time constraints and uncertainty. 2021.

[11] Margaretha Gansterer and Richard F Hartl. Centralized bundle generation in auction-based collaborative transportation. *Or Spectrum*, 40(3):613–635, 2018.

[12] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5628–5635. IEEE, 2011.

[13] Hang Ma, Wolfgang Hönig, TK Satish Kumar, Nora Ayanian, and Sven Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7651–7658, 2019.

II

Literature Study
previously graded under AE4020

1

Introduction

At Amsterdam Airport Schiphol, Royal Schiphol Group developed a vision to operate the most sustainable airport by 2050. The aim is to achieve fully-autonomous airside operations by 2050 [97][81]. The daily airside operations are anticipated to have changed significantly. All vehicles will be replaced by interconnected, autonomous, and emission-free vehicles. All associated ground handling processes will also be automated, including automated baggage and passenger transport, autonomous passenger boarding bridges, and autonomous towing trucks.

The automation of ground handling processes will help to utilize the resources more effectively, shortening the traveling distances and time, and thus help lower the emissions produced by the aviation sector [80]. With the help of automation, necessary manpower can be greatly decreased. Also, automated aircraft ground handling operations decrease the human factors involved in the processes. The safety and security of the airport can be improved by diminishing possible human errors and eliminating potential related threats [3].

The automation of aircraft ground handling operations is be steered by intelligence technologies. Artificial intelligence can bring data-driven insights into key airside processes. Agent-based modeling can be used to model the operation of aircraft ground handling. A multi-agent system consists of multiple intelligence decision-making agents which interact in the same environment to achieve goals. As different stakeholders are involved in the ground handling operation, agent-based modeling techniques well capture various goals and interests of the involved agents [86].

Automated aircraft ground handling has to be introduced to the current operations gradually. Therefore, in this research, we focus on the automation goals to be achieved in the near future. This thesis aims to design and evaluate a multi-agent control system model for the automation of aircraft ground handling operations. There are two main goals we would like to achieve in our model. First, an automated task allocation optimization mechanism will be designed to optimally allocate the ground handling tasks to ground support equipment (GSE) vehicles, complying with temporal and operational constraints. Second, planning optimized collision-free paths of the GSE vehicles will be performed by implementing an adapted multi-agent path finding (MAPF) algorithm. Both task allocation and path planning take realistic assumptions into account in the model. To limit the computational time, task allocation and path planning for GSE vehicles will be performed in a two-stage model in our research. After the model is developed, it will be evaluated by simulation experiments and different statistical tests. The scope of the model to be developed in this research contains flights of an aircraft pier. An example aircraft pier for short-haul flights, pier B of Amsterdam Airport Schiphol, will be used as a case study for this research.

This literature study report is organized as follows: In chapter 2, current aircraft ground handling operations, the modeling and optimizing approaches, and the opportunities of automated aircraft ground handling are presented. chapter 3 contains the relevant task allocations approaches and different multi-agent task allocation (MATA) algorithms. In chapter 4, various multi-agent path finding approaches are explored and discussed. Finally, in chapter 5, the research proposal is presented.

2

Aircraft Ground Handling Operations

In this chapter, an overview of aircraft ground handling operations is provided. The current operation, including the ground handling procedures, the required equipment, and the related infrastructure are introduced in section 2.1. In section 2.2, the Airport Collaborative Decision Making (A-CDM) system will be discussed. Current approaches that are used to model aircraft ground handling operations are presented in section 2.3. After reviewing conventional operation approaches and modeling techniques, in section 2.4, we discuss the opportunities of automated aircraft ground handling. Finally, in section 2.5, current gaps of developments of the automation are deliberated.

2.1. Current Aircraft Ground Handling Operations

This section presents the current aircraft ground handling operations. Only the ground handling activities at the aircraft stands are considered. Aircraft stands are areas of an airport where aircraft park between flights for passenger boarding and deplaning, cargo loading and unloading, refueling, and other cabin services. Some typical ground handling tasks are listed below, divided by the location they are carried inside or outside the cabin. In this study, only activities outside the aircraft cabin are considered.

- Activities inside the aircraft cabin:
  - Passenger deplaning and boarding
  - Cabin cleaning and preparation
  - Catering preparations
  - Safety and security checks
- Activities outside the aircraft cabin:
  - Cargo unloading and loading
  - Catering galleys unloading and loading
  - Connecting and disconnecting Ground Power Unit (GPU)
  - Connecting and disconnecting Pre- Conditioned Air unit (PCA)
  - Refuelling
  - Water and toilet servicing
  - Connecting and disconnecting passengers boarding stairs or Passenger Boarding Bridge (PBB)

2.1.1. Procedure

General ground handling procedures are introduced in this subsection [1]. The general sequence of the ground handling activities is shown in Figure 2.1. Before the aircraft touches down, the ground handlers are already

[1] Part of the procedures described in this section are retrieved from internal documents of case studies of Amsterdam Airport Schiphol, under the project "Safety of Turnaround Operations".

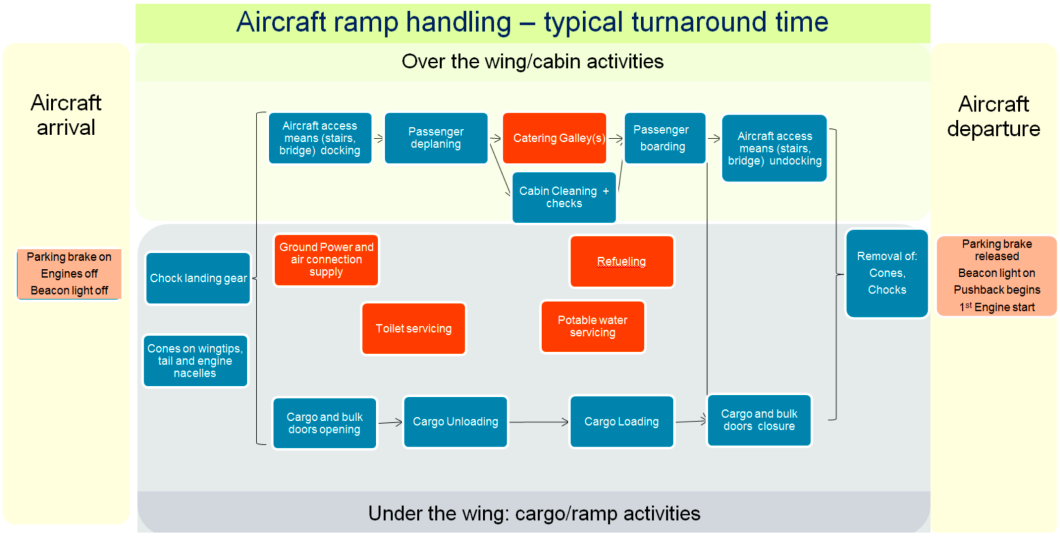


Figure 2.1: Typical turnaround time-task sequence. Retrieved from [3].

preparing for the arriving flights [67]. The team of ground handlers check if all the equipment needed is available, and be parked or placed clear of the aircraft's maneuvering area. Also, they check if the parking stand is clean of Foreign Object Debris (FOD), debris that may damage the aircraft or equipment.

The parking guidance is turned on if the parking stand is clear. The aircraft can then enter the parking stand. After the parking brakes are set, engines are shot down, and the beacon lights are turned off, the aircraft's gear will be choked by placing blocks around the aircraft's wheel. Pylons are also placed under the wingtips, the tail, and around the engine nacelles. This time point is noted as the aircraft's in-block time (Milestone 7 of A-CDM, which will be discussed in section 2.2). The ground handling activities officially start from this time point.

Shortly after the wheel blocks are placed, GPU will be connected to supply power to the aircraft while the engines are turned off. PCA will also be connected to provide air conditioning of the cabin if necessary. The passenger boarding bridges or stairs are docked for passenger deplaning. While the passengers head to the terminal, the cargo and bulk doors are opened, preparing for unloading the cargo. Conveyor belt vehicles or cargo loaders are used to transport the baggage and cargo to the dollies of Unit Load Devices (ULD) or cargo pallets. Due to the weight balance of the aircraft, sometimes the apt cargo needs to be unloaded first before unloading the forward compartment [3]. Cargo loading is done in similar manners. Loading of the aircraft is carefully calculated by the airline's planning department to ensure the aircraft's center of gravity is within a certain safety range.

Portable water and toilet servicing are performed under the tail of the aircraft. Pipes are connected to the aircraft and the content in the waste tank is emptied into a vehicle. Also, portable water is replenished. Certain hygienic standards have to be complied [82]. At the same time, catering trucks dock on the right-hand side of the aircraft. They are responsible for delivering food trolleys to the catering galleys and removing trolleys and waste from the previous flight.

Refueling takes place under the aircraft's wings. Depending on the infrastructure of the aircraft stands, fuel hydrant systems (with special carts or small trucks) or large fuel trucks can be used. Aircraft operators or airlines have different procedures for refueling. In many cases, refueling is not allowed with passengers onboard, thus making refueling a critical path for aircraft ground handling. However, in some cases with tight aircraft turnaround time, refueling can be done parallel with passenger deplaning or boarding under certain safety measures.

After passenger boarding is completed, aircraft's doors are closed, and passenger boarding bridges or stairs are removed. Cargo doors are closed after all the cargo and baggage are loaded. The pre-departure-

safety check is performed by a ground handler, making sure that the stand is clear of FOD, and all aircraft doors are closed properly. Subsequently, pylons and wheel blocks are removed by ground handlers, parking brakes are released and beacon lights are turned on by the pilot. The aircraft is now ready for push back from the parking stand. This time point is noted as the aircraft's off-block time (Milestone 15 of A-CDM, which will be discussed in section 2.2).

It is worth noting that not all the above-mentioned ground handling tasks will be carried out at each aircraft turnaround. At outstations, aircraft operators may shorten the ground handling procedures by omitting some ground handling activities. For example, aircraft bringing sufficient fuel for a round trip may not need refueling at outstations. Some activities may not be done depending on the equipment or infrastructure available at the airport. Further, the first and last flights of the day do not need tasks related to arrival and departure, respectively.

The turnaround time (TRT) can be defined as the time between the aircraft is on-chocks until it is off-chocks. The TRT duration highly depends on different scenarios and parameters. An example of an aircraft turnaround time chart is shown in Figure 2.2. The Boeing 737-800 aircraft is used as a demonstration here.

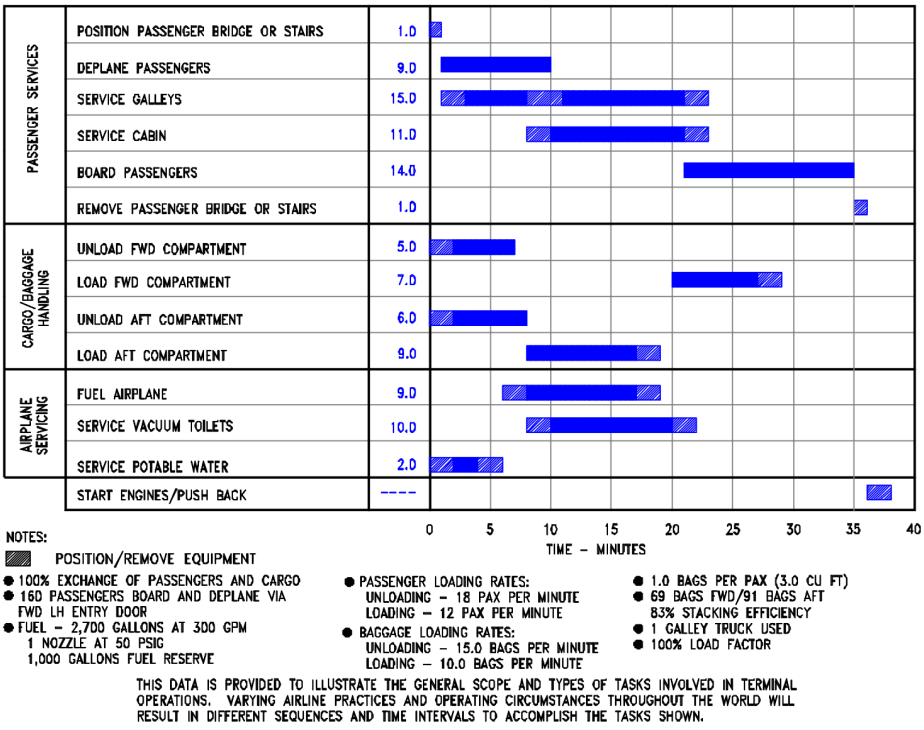


Figure 2.2: B737-800 turnaround time chart. Retrieved from [2].

It is essential to note that even with the same aircraft type, TRT is not a constant. Procedures in the schedule and the duration of the task can vary, and thus influence the global TRT. Some possible influential factors mentioned in [3] are discussed in the following. The passenger boarding and disembarking processes are critical paths that significantly affect TRT [82]. Aircraft cabin layout, including cabin classes and the number of seats, and load factors of aircraft are important factors that determine the required passenger boarding and disembarking time. Also, in short-haul flights, the passenger processes affect the total TRT more than the passenger processes in long-haul flights. From the aircraft ground handling point of view, related infrastructure and the ramp layouts, including facilities at gates or remote stands are also decisive factors. For example, at the aircraft stands with underground piping systems, the time needed for refueling can be shorter. Together with the above-mentioned facilities, the numbers of available Ground Support Equipment (GSE) and personnel are also key factors. Besides the hardware, some organizational factors matter. The efficiencies of ground handlers affect the time of various ground handling operations. Airline and ground handling policies,

for example, the time that passengers are required to be present at gates, influence the process of passenger boarding substantially. Some uncertainties like weather are also involved. For example, no ground handling activities can be performed during thunderstorms due to safety reasons.

2.1.2. Ground Support Equipment

Necessary Ground Support Equipment in the ground handling activities is discussed in this section. Within the scope of our study, only the GSE which is related to ground handling activities outside the aircraft is involved.

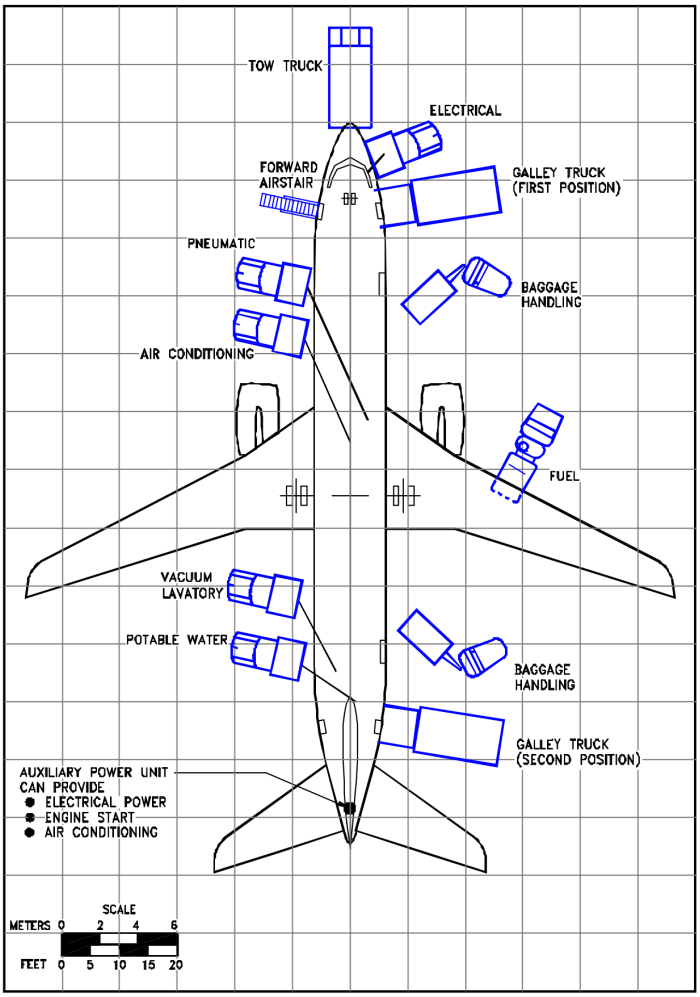


Figure 2.3: Typical aircraft ground handing service layout. Retrieved from [2].

- Passenger boarding bridges: Passenger boarding bridges are GSE that have specifically reserved routes while connecting and disconnecting themselves to the aircraft.
- Catering trucks: Catering trucks have several positions to connect to the aircraft. They usually work on the right side of the aircraft.
- Refuelling/hydrant vehicles: Refuelling trucks work beside or under aircraft's wings, and are the only GSE that can travel under the wings. Depending on the infrastructure of the airport, refueling vehicles can be large fuel trucks or smaller hydrant carts. Unimpeded routes must be reserved for refueling vehicles to exit in case of emergency. Thus, they usually have higher priorities in conflicts resolution.

Table 2.1: GSE resources for the ground handling of a short-haul flight and GSE sharing strategies with recommended priorities. Adapted from [3] in internal documents.

Ground Handling Equipment	Numbers	Shared Resources Strategy	Priority
Passenger boarding bridge	1	Specific to aircraft stand	2
Tractor with GPU	1	Shared with adjacent stands/zones	8
Fuel filling vehicle	1	Shared with all the airport	1
Water service truck	1	Shared with all the airport	5
Toilet service truck	1	Shared with all the airport	6
Catering truck	1-2	Shared with all the airport	3
Belt loaders	2	Shared with adjacent stands/zones	4
Baggage tractors	2	Shared with all the airport	7
Baggage carts	6-8	Shared with all the airport	9

- Portable water trucks: Portable water trucks usually work at the left aft of the aircraft where fewer conflicts happen.
- Toilet service trucks: Toilet water trucks usually work at the left aft of the aircraft where fewer conflicts happen. Although there are few conflicts between water trucks and toilet trucks, certain hygienic regulations have to be followed. For instance, water service and toilet service cannot be performed at the same time.
- Towing vehicles: Towing vehicles stay in front of the aircraft when other ground handling activities are being performed. Although the towing of aircraft is an important ground handling operation outside the aircraft, it is not considered in this study as our research focuses on the ground handling tasks done after the aircraft arrives in block and before its push-back.
- Luggage trucks: Luggage trucks include conveyor belt vehicles, cargo loaders, tow tractors, ULDs, and cargo pallets. The size and shape vary a lot among different luggage trucks. They also have the most complicated trajectories. A rule of thumb is that they always leave the aircraft on the left.
- GPU mobile: GPUs usually work at the front of the aircraft.

Table 2.1 listed the number of GSE typically need for handling a short-haul flight. The table also shows the strategy that the GSE resources are shared. Further, in case of collisions, while performing ground handling operations, the rightmost column shows the typical priority order of GSE. Vehicles with lower priorities have to give way to the ones with higher priorities.

2.1.3. Infrastructure

In this section, infrastructures used for aircraft ground handling, mostly aprons and aircraft stands, are discussed. Amsterdam Airport Schiphol (AAS) is used as a case study in our research.

Aprons and gates

Amsterdam Airport Schiphol (AAS) has 91 connected gates, located across 7 piers [78], as shown in Figure 2.4.

The aircraft ground handling operations of a single pier will be focused on in this study. Using AAS as a case study, the feature of a pier at AAS, including the configuration, the number of aircraft stands, parts of the infrastructure, and the data of operating flights will be modeled under some assumptions.

Consider a pier as linear parking stands layout with two service roads, as shown in Figure 2.5. While traveling on the service roads, the maximum allowed speed of ground handling vehicles is 30km/h [79]. A 6km/h speed is set for vehicles on the aircraft stands, and the speed limit is 0.8km/h while docking the vehicles to the final positions that are in contact with the aircraft [3]. Note that in Europe, it is usual to have only a front service road, while in the US, having a service road only at the back of the aircraft stands is common. In AAS, service roads are only in the front of aircraft stands.

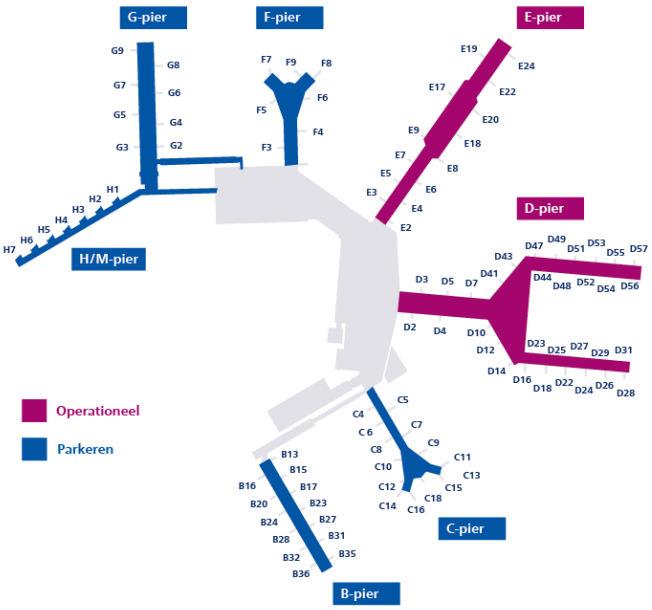


Figure 2.4: Overview of the piers of Amsterdam Schiphol Airport. Retrieved from [75].

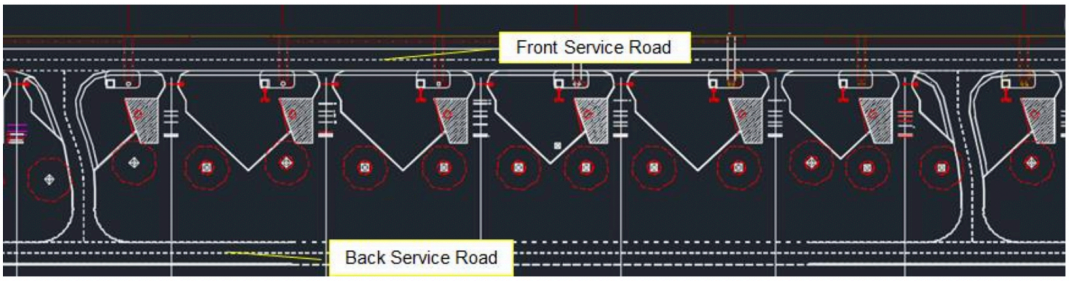


Figure 2.5: Typical airport parking layout. Retrieved from [3].

Aircraft stands

Aircraft stands are the parking spots for aircraft to park between flights. Aircraft ground handling operations also take place at aircraft stands. They should be designed so that their layouts do not block the flow of aircraft and GSEs, and they should be properly connected to the supporting taxiway system [46]. An overview of a typical aircraft stand layout and the signage used is shown in Figure 2.6.

Number 2 marks the passenger bridge maneuvering area. This area is specially used for the movement of the passenger boarding bridge, and it should not be used by other GSEs. Number 6 shows the aircraft clearance lines, indicating the boundaries on the sides of the aircraft, which ensure that aircraft have enough space to enter and depart from an aircraft stand. Number 7 shows the red clearance line, which is a 60-cm-wide line indicating the border between the aircraft stands and the taxiways. The area marked off between the aircraft clearance lines and the red clearance line is the aircraft ground handling area, where the ground handling service can be performed. Importantly, number 9 marks the entrance and exit of the aircraft stand. All ground handling vehicles must enter or exit the stand via number 9. Usually, ground handling vehicles have to be in place 5 minutes before the aircraft arrive. This entrance/exit area also serves as the holding point for the vehicle. Also, the area needs to be clear while performing ground handling tasks, because the area is intended as an escape route.

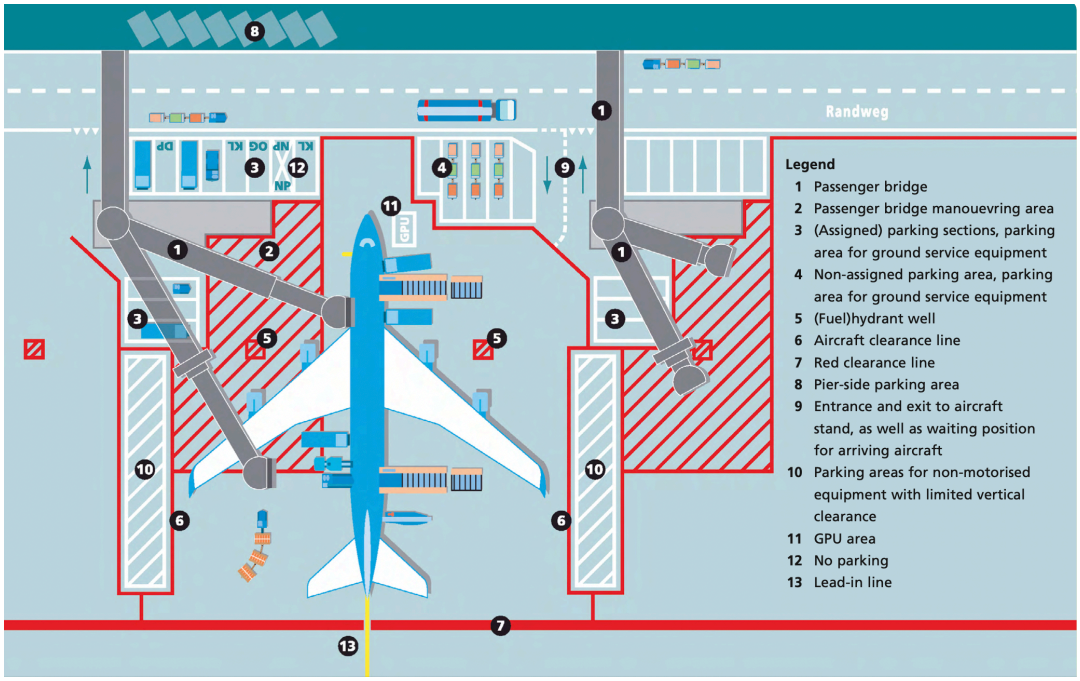


Figure 2.6: Overview of an aircraft stand layout and the signage used. Retrieved from [76].

2.2. Airport Collaborative Decision Management

In this section, the concept of Airport Collaborative Decision Making (A-CDM) is introduced. Part of the contents presented in this section are adapted from [95], [28] and [77].

A-CDM is a concept that originated from Eurocontrol [2]. By optimizing resource utilization and improving air traffic predictability, A-CDM aims at improving the efficiency and resilience of airport operations. This is achieved by transparently and collaboratively exchanging timely information between A-CDM stakeholders. The stakeholders include airlines, ground handlers, air traffic controllers, and airports. The timely information especially focuses on aircraft pre-departure an turn-around processes [28].

One of the most important implementations of A-CDM is the milestone approach. It captured the significant events of a flight, including the initial planning phase, the approach and arrival, the turnaround process, and its final departure. The events are symbolized as milestones. A completed milestone will trigger the decision-making processes for succeeding events. It also influences the later progress of the flight and their prediction accuracy [77]. The recommended milestones by Eurocontrol are presented in Figure 2.7 and elaborated in Table 2.2.

Milestones 7, 8, 9, 11, 12, and 15 are relevant for this study. Interpretation of the milestones, as well as the possible utilization of the timely information derived from the milestones, are presented in the following:

- Milestone 7, In-block: The time that an aircraft arrives in-block. In our study, this milestone also represents the ground handling simulation of a specific aircraft/bay area begins. The simulation can also begin earlier that some ground handling vehicles arrive at the bay in advance to prepare the ground handling tasks. Milestone 7 (or some earlier time point) can serve as the start time of the duration that some ground handling vehicles are being assigned to the specific aircraft/ bay area.
- Milestone 8, Ground handling starts: The time of milestone 8 is usually identical to that of milestone 7. However, due to limited vehicle capacity, in some rare cases, ground handling might not be able to start

[2] The European Organisation for the Safety of Air Navigation

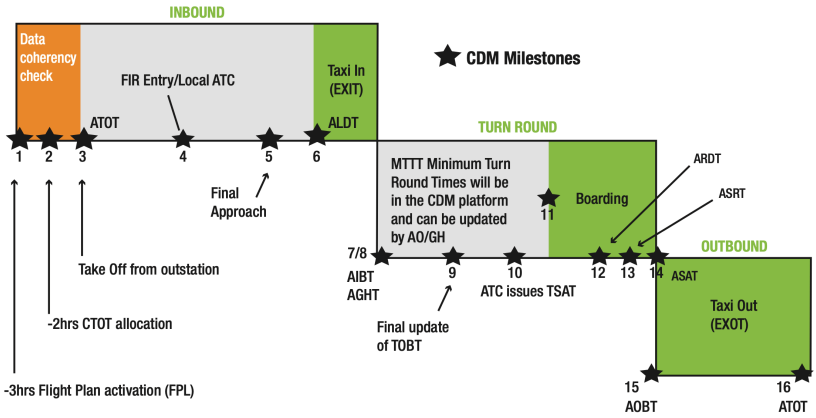


Figure 2.7: Recommended milestones in the A-CDM manual. Retrieved from [95].

Table 2.2: Recommended milestones in the A-CDM manual. Retrieved from [95].

number	milestones
1	ATC (Air Traffic Control) flight plan activation
2	Estimated Off-Block Time (EOBT) -2hr
3	Take off from outstation
4	Local radar update
5	Fianl approach
6	Landing
7	In-block
8	Ground handling starts
9	Target Off Block Time (TOBT) update prior to TSAT
10	Target Start Up Approval Time (TSAT) issue
11	Boarding starts
12	Aircraft ready
13	Start up request
14	Start up approved
15	Off-block
16	Take off

immediately when the aircraft is in-block. Milestone 8 is thus necessary to be considered separately from milestone 7.

- Milestone 9, TOBT update prior to TSAT: TOBT is the time that the ground handler estimates that all the ground handling of an aircraft will be completed. This means, at the time, the aircraft's doors will be closed, the passenger boarding bridge and all the ground handling equipment will be removed. The aircraft will be ready to start up or push back immediately upon approval.

- Milestone 11, Boarding starts: Milestone 11 represents the time the passenger boarding process starts. Although the time point is not directly relevant to this study, as the passenger processes are not considered, the boarding starting time can serve as a reference to estimate the boarding complete time. For example, the boarding starting time pluses a constant duration of the boarding process indicates a predicted time that boarding has completed. Passenger boarding bridge(s) can be removed after boarding has been completed.

- Milestone 12, Aircraft ready: The aircraft ready time indicates the time that all doors of the aircraft are closed, boarding bridge or stairs removed, push back vehicle connected, and the aircraft is ready to push back upon approval. In our study, this can be the time point that the simulation of the particular aircraft/ bay area stops.

- Milestone 15, Off-block: Milestone 15 is the time that the aircraft actually starts pushing back and leaves the parking position. Once the bay area is cleared, obstacles representing the aircraft can be removed in the simulation.

The timely information of the milestone retrieved from the current A-CDM operational data of Schiphol airport can be viewed as reference input parameters for this study. Further, the present data can also be utilized as a datum while evaluating the developed model of this study.

2.3. Current Approaches of Modeling Aircraft Ground Handling Operations

In the previous sections, current ground handling operations, including necessary GSEs, the procedures, and the related infrastructures, are presented. Schmidt has provided an introduction to aircraft turnaround operations and reviewed related simulations [82]. In this section, we briefly present the current approaches that are used to model and optimize aircraft ground handling operations in literature. Different approaches include linear programming, network analysis, stochastic approaches, and other approaches.

2.3.1. Linear programming

Linear programming (LP) is an optimization technique. With a set of linear constraints, LP aims to optimize a linear objective function by determining the values of a set of decision variables. Linear programming is a useful approach to efficiently allocate resources. The advantage of LP is that it always returns optimal solutions. However, for realistic problems, it requires considerable computational time. The other downside of LP is that it solves the problems in a centralized way. It is thus more difficult to capture the interactions between agents and to consider the goals of individual agents.

Example applications of LP in aircraft ground handling are discussed below. Aircraft turnaround is a critical process in airports that significantly affect the punctuality of flights. The operations involved in the turnaround process are interrelated [65]. To efficiently utilize the available resources, Padrón et al. handled the problem in a global perspective through optimizing the goal with two objectives [66]. The first objective is to minimize the waiting time before an operation starts and the total reduction of corresponding time windows, while the second objective is to minimize the total duration of the turnarounds. The authors proposed an approach for scheduling ground-handling vehicles called Sequence Iterative Method [66]. They tested the approach using real data from two Spanish airports. The results showed that different solutions representing the trade-off between two objectives were found. The results can thus be utilized to modify the order of the vehicles in the turnaround schedules. To limit the required computational time, Padrón et al. further developed an enhanced method, called improved Sequence Iterative Method [65]. The novel methodology first explores the solution space using a fast heuristic. Later, it focuses on the most promising solutions, strengthening the search close to the Pareto frontier. The two-step approach was proofed to significantly reduce the computational time.

2.3.2. Network analysis

Networks refer to structures representing the relationship of groups of objects. Nodes and edges are included in network structures. Simple Temporal Networks (STN) [18] is a network analysis framework that is used to handle temporal constraints. Constraints in STN are in the form of $x - y \in [a, b]$, where x and y refer to two time-points and a and b represent the lower bound and the upper bound of the time interval, respectively [70]. STN is suitable to be applied to a serial of tasks with certain orders and some temporal constraints. It is also commonly used in the area of aircraft ground handling.

In order to make the aircraft ready at the scheduled off-block time, Vagner et al. [47] used STN to optimize the ground handling processes. Optimal timetables of ground handling activities for the Boeing 737-300 aircraft are created. Leeuwen and Witteveen have also used STN to model the turnaround processes [99]. Moreover, the authors pointed out that the global plans provided by STN have several disadvantages as vehicle agents have different individual goals. Leeuwen and Witteveen utilized temporal decoupling [45] to generate a set of temporal plans that can be scheduled independently. They proofed that even if every agent arranges their own schedule and resources independently, the overall solution is still feasible.

2.3.3. Stochastic process

Combining historical data, stochastic processes are widely used in modeling aircraft ground handling operations. Stochastic models like Markov chains have been utilized for modeling. Monte Carlo simulation is also a common approach to simulate the various duration of aircraft turnaround operations.

An early research of Wu et al. used a Markovian type simulation model to simulate uncertainties of aircraft ground handling operations [101]. The authors showed the inherent schedule punctuality is influenced by the amount of scheduled buffer time and the operational efficiency of ground handling activities. Later, Fricke et al. [32] performed a large field study at German airports. They discussed the reasons for uncertain turnaround process time. Monte Carlo simulations are conducted to learn the level of reliability in the turnaround operations at that time and to show the potential for improving TRT reliability through aircraft interface optimization. Recently, Sheibani [87] considered critical path analysis and Monte Carlo simulation of aircraft ground services during the turnaround. The simulation would help to estimate the aircraft turnaround time considering its type and load accurately, further improving the efficiency of resource assignment. A case study of a long-range wide-body twin-engine jet aircraft is presented in the study.

2.3.4. Other approaches

Other approaches that are used to optimize aircraft ground handling include some suggestions of the hardware and infrastructure and the proposal of new systems. Schmidt et al. [83] proposed some novel concepts to reduce the required sources and time based on clustering aircraft interfaces, such as doors and service panels. The novel concepts they proposed are mainly some minor improvements to the aircraft cabin. For example, relocating and installing wider passenger doors, merging two galleys, and separating galleys from the aircraft entrance area. The proposed concepts showed large improvements in turnaround time. Early research of Wu [100] developed an Aircraft Turnaround Monitoring System (ATMS) to collect aircraft turnaround operational data on a real-time basis. Utilizing the collected data, the system is also used to monitor aircraft turnaround processes in real-time. Timestamps of passenger, cargo, engineering, and catering are provided by different stakeholders and are shared on the system. ATMS can be viewed as a predecessor of A-CDM.

2.3.5. Agent-based modeling

Up to this point, we reviewed the current approaches used to model and optimize aircraft ground handling operations. Although different approaches have been utilized to deal with problems of aircraft ground handling, little literature has utilized agent-based modeling in the area of aircraft ground handling. Agent-based modeling is commonly used to model complex systems which contain many interactions. Other than a top-down approach, agent-based modeling uses a bottom-up approach, in which the system can be defined from the perspective of heterogeneous autonomous entities. Despite there are few applications of agent-based modeling in the area of aircraft ground handling, this modeling technique has been used to model various aerospace contexts. Some examples include modelling (non-)autonomous taxi operations [7] and autonomous baggage handling systems [26]. As discussed by Sharpanskykh, agent-based modeling is one of the most promising approaches to model and evaluate different air transport systems [86]. Therefore, in our study, we apply agent-based modeling techniques to construct and evaluate a model for automated aircraft ground handling.

2.4. Opportunities for Automated Aircraft Ground Handling

After reviewing current operations of aircraft ground handling and possible modeling approaches, in this section, we discuss the opportunities for automation of aircraft ground handling. A thorough analysis of automation for aircraft ground handling can be found in Tabares' study [3]. Below we introduce some general conditions, challenges, and goals at different stages.

2.4.1. General conditions

Current aircraft ground handling operations at airports, as described in section 2.1, do not have major changes in operation since the beginning of commercial flights [3]. In the past decades, technology has progressed significantly, making various possibilities for the automation of aircraft ground handling. Some general favor-

able conditions for aircraft ground handling automated retrieved from Tabares' study [3] and other literature are introduced below.

Similar aircraft configurations

The configuration of most modern civil aircraft with more than 100 seats is very similar in the perspective of ground handling. The fuselages of aircraft are tube-shaped with at least two decks for passenger cabin and cargo. Low wings are attached to the fuselage with positive swept angles, and 2 (or 4) engines are hanging below the wings. The landing gears have a tricycle configuration with one nose gear and multiple sets of main gears [3].

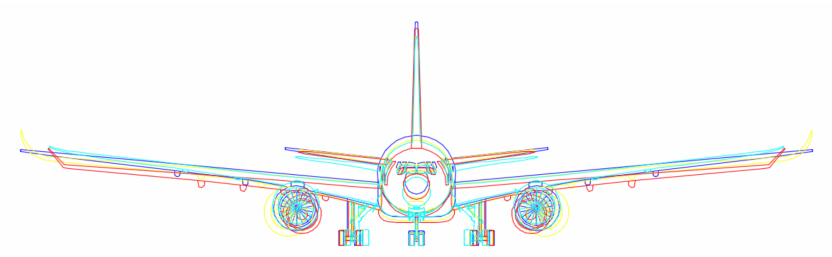


Figure 2.8: Comparison of configurations of different aircraft, with A350-900 in yellow, A330-200 in red, 777-300ER in blue, and 787-10 in cyan. Retrieved from [3].

Common interface locations

The aircraft servicing interfaces follow some standards and are often placed at common areas. For example, the connectors of electricity are usually at the front, while the refueling interfaces are under the wings and located halfway to the fuselages. Figure 2.9 shows the common locations of aircraft service points. Also, Schmidt et al. [83] investigated the interfaces used in ground handling of 63 different types of aircraft. They clustered the interface with similar functions using normalized aircraft length and half-wingspan. The investigated interfaces include passenger and cargo door position, service panels of portable and waste water, PCA interfaces, GPU interfaces, and fuel connectors. For example, Figure 2.10 shows the service panel locations of GPU and refueling. Schmidt et al. mentioned that although a standardized location for aircraft system interface is not presented, the cluster analysis of aircraft in and out of production revealed a trend of shifting of the interfaces towards common locations.

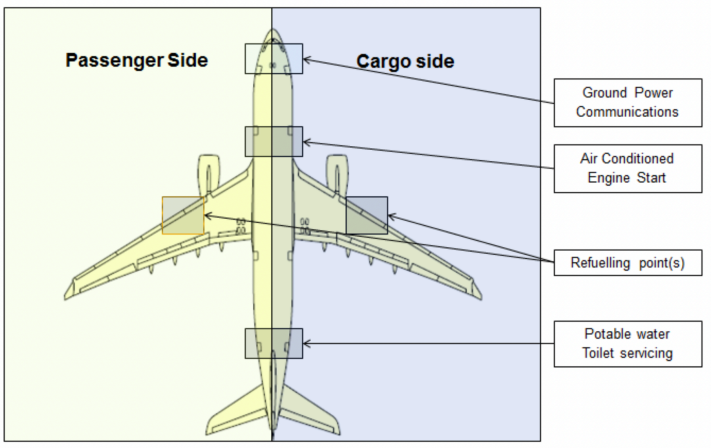


Figure 2.9: Common locations of aircraft servicing points. Retrieved from [3].

Particularly, Tabares mentioned that the aircraft doors and their usages are standardized [3]. Doors on the forward port side for passenger boarding / deplaning, while doors on the starboard side and aft port side are for catering and cleaning tasks. Cargo doors are located nearby the wings, and the bulk doors are always at the aircraft rear part.

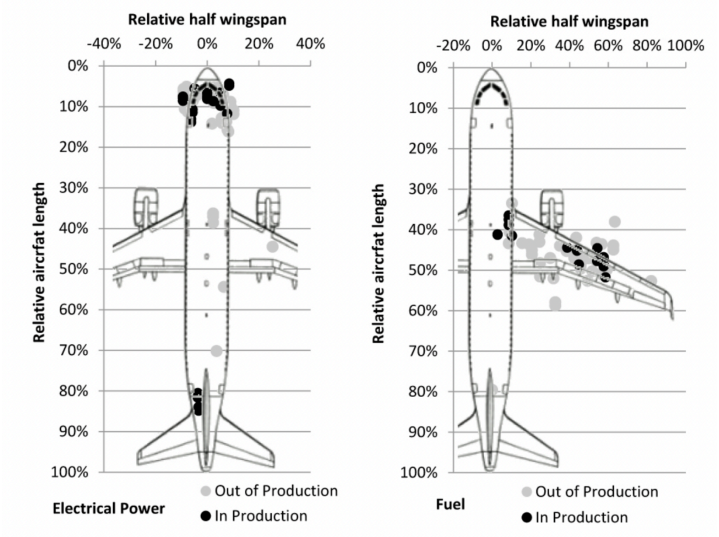


Figure 2.10: Service panel locations of GPU and fuel. Retrieved from [83].

Manpower availability

In certain cities with high living expenses, hiring ramp workers with limited budgets is difficult. Also, operator ergonomic, in both health and safety aspects, is becoming more valued. Airports in Scandinavia first implemented load-lifting limits, and the measure is slowing but steadily spreading out to airports in other regions [3].

Available technology

As the technology of robotics and autonomous vehicles become available and affordable, there are possible developments of automated GSE. Aircraft stands are mostly flat, having limited numbers of potential events performed by known actors. Thus, aircraft stands are considered to be homogeneous and controllable environments. Moreover, the aircraft couplings for GSE connections are standardized, making the automation of coupling connections possible in the future [3].

Safety and security challenges

Safety and security issues of airports and aircraft are mostly human-related. Controlled and automated aircraft ground handling operations help to reduce the human factors and thus diminish the possible human errors. In terms of security, the fewer people around the aircraft, the fewer potential threats [3].

2.4.2. Challenges

Despite the favorable general conditions presented in subsection 2.4.1 support the automation of aircraft ground handling, some challenges for the implementation exist. Tabares has noted that the main objection includes three aspects: aircraft-related conditions, operational restrictions, and scientific limitations [3]. The recognized challenges are discussed below.

Aircraft-related conditions

For smaller aircraft like B737, the volume of cargo cabins is not large enough to receive some cargo containers and pallets like ULD. Without accepting the cargo containers that can be handled by the conveyance systems, baggage loading and unloading have to be performed manually. The other hindrance of automation is automated cabin doors opening. At present, remote door opening or door opening by robots are not certified. Even if the operation is technically feasible, current certification rules prevent this possible operation [3].

Operational restrictions

There are multiple stakeholders involved in aircraft ground handling operations. Each stakeholder may have different goals that can cause potential conflicts. Also, while gradually introducing the automation of ground handling, manual and automated operations will exist in the airport environments at the same time. This mix-mode complicates the operating environment and is a challenge for automation. The current government rules and industrial attitude also affect the progress of automation [3].

Scientific limitations

Even if the technology of robotics and autonomous vehicles is more and more mature, some scientific limitations still exist. First, the real-time control of heterogeneous GSE vehicles is a challenge. Using non-linear or hybrid control theory, multi-dimensional guidance of GSE vehicles must be implemented. Also, conflict-free trajectories have to be planned between GSE vehicles, including responses for incidents and disruptions [3]. Modeling the dynamic ground handling system, with a coordination and communication mechanism is a challenge, and, to our best knowledge, is not yet exist in literature. In this research, we aim to construct a model for automated aircraft ground handling, including task allocation and path planning for GSE vehicles.

2.4.3. Goals at different stages

Automated aircraft ground handling cannot be done overnight. The automation have to be introduced to the current operations gradually. Based on the general conditions and the challenges discussed above, goals of automation of aircraft ground handling at different stages can be set. Some automation goals mentioned by Tabares [3] are introduced below.

Short-term goals

Automated docking of GSE to aircraft is one of the possible short-term progress. To achieve the automation of docking, targets around aircraft doors should be designed to assist the positioning of GSE vehicles. Also, collision-free paths for the GSE vehicles should be planned. The design and implementation of possible path planning algorithms for the GSE vehicles are one of the main topics in this research.

Mid-term goals

In the mid-term, the progress of automation focuses on the automation of ground handling activities that are currently performed manually. The midterm goals include door opening and panel access remotely or by robots, as well as automatic connection of system and utilities. Moreover, the implementation of automated cargo loading systems inside the aircraft is part of the goals in the mid-term.

Long-term goals

In the long term, different cabin supplies are expected. Possible approaches include modulation of aircraft cabin components. For example, exchanging the catering supplies with full galleys instead of individual trolleys. Moreover, automated supply of utilities (electricity, water, and air) to aircraft beyond the current aircraft interface locations is a goal that can only be reached in the long term.

2.5. Gaps in automated aircraft ground handling

Up to this point, current aircraft ground handling operations, current modeling approaches, and the opportunities of automated aircraft ground handling are discussed. In this section, we identified some gaps in automated aircraft ground handling. We also discuss the research gaps that our study is trying to fulfill.

The automation of aircraft ground handling operations is a novel concept that has not been completely developed and implemented. In terms of hardware, robotics and autonomous vehicles technologies have been developed to fulfill the requirement of automation. On the other hand, the required software system supporting the automated operation, as well as the transformation from current systems to automated operations, are still under development. The software for automation requires data and intelligence at the base of its operation. The utilized intelligence includes real-time data insights, adaptive routing, and integrated planning and forecasting. Also, to transit from the current states to fully autonomous ground handling operations, lots of executing procedures and steps still need to be defined. Many organizational aspects of automation are still uncertain in the present scheme [97].

Our research focuses on developing the intelligence system needed for automated aircraft ground handling. Real-time data-driven planning and coordination for GSE vehicles will be performed in our model. As the initial steps of automation, automated task allocation and path planning for GSE vehicles will be included in our model. The planning and coordination are achieved by a multi-agent system. The system automatically allocates ground handling tasks to GSE vehicles, complying with temporal and operational constraints. After the tasks to be performed and the corresponding execution time windows are assigned to the GSE vehicles, trajectories of GSE vehicles can be planned. Collision-free paths with allocated time windows need to be found for GSE vehicles. The task allocation and path planning for ground handling tasks of flights can

be performed in real-time with pre-defined time windows. The time windows can be aligned with the rush hours of inbound and outbound traffic at Schiphol.

3

Task Allocation

One of the main features of automated aircraft ground handling is the task allocation capabilities, in which the GSE vehicles must be assigned to the ground handling tasks. In this chapter, different task allocation approaches are reviewed. In section 3.1, an overview of different classes of task allocation solvers is presented. After that, in section 3.2, we compared the discussed classes of task allocation solvers and identify the most applicable one for the task allocation of automated aircraft ground handling. Lastly, in section 3.3, the most applicable solution class, auctions, are discussed in more detail.

3.1. Overview of solution techniques

The purpose of task allocation is to feasibly assign a set of tasks to an agent such that a global objective function is optimized [22]. From centralized solvers to decentralized approaches, some common classes of task allocation solvers are operation research, evolutionary and swarm algorithms, auction-based multi-agent systems, and game-theoretic approaches. In this section, we review different classes of task allocation solvers. In each class, prominent approaches are presented and discussed.

3.1.1. Typical optimization approaches

Task allocation of aircraft ground handling operation can be modeled as the assignment problem, which is one of the fundamental combinational optimization problems in operation research. Assignment problems consist of a number of resources and tasks. The resources have to be assigned to tasks on a one-to-one basis. Every resource-task combination incurs some cost, and the objective of the assignment problem is to minimize the total incurred cost. Modeling the task allocation of automated ground handling as an assignment problem, the GSE vehicles are resources, and the ground handling tasks are tasks to be assigned. At any time point, each GSE vehicle can only perform one ground handling task. Also, a ground handling task can only be carried out by one GSE vehicle.

The assignment problem is well-studied in the field of operation research. Kuhn-Munkres Hungarian algorithm was early developed by Kuhn [52] to solve the assignment problems in $O(r\,t^2)$ time, in which r is the number of agents and t is the number of tasks. Furthermore, some variants of assignment problems are well-studied in operation research, including the traveling salesman problem (TSP) and its generalization, the vehicle routing problem (VRP). Algorithms like the simplex method, branch and bound, and dynamic programming (DP) can be used to solve these problems.

The typical optimization approaches can provide optimal and complete solutions to the problems. However, these kinds of global optimization methods require long computation times. They are thus less scalable and applicable to large practical problems. Also, because the allocation of tasks is performed all at once, it is not possible to flexibly and dynamically adjust the cost of a resource-task combination based on current allocation progress or changes that happen during the allocation.

3.1.2. Evolutionary and swarm algorithms

Evolutionary and swarm algorithms, including genetic algorithm (GA) [35][40] , ant colony optimization algorithm (ACO) [23][24], and bee colony optimization algorithm (BCO) [96] are nature-inspired algorithm that can solve optimization problems with shorter computational time.

A famous kind of evolutionary algorithm is the genetic algorithm [35][40]. GA was inspired by the Darwinian theory of evolution, in which the fitter creature survive and pass on their genes. GA is a population-based algorithm. The main operators of GA are selection, crossover, and mutation [60]. There are numerous variants of genetic algorithms studied in literature. Below we briefly describe how the genetic algorithm works.

Genetic algorithm [60][59]

Four phases are included in the genetic algorithm.

1. **initial population**: GA starts with a random population. Each individual in the population represents a solution to the problem to be solved. Every set of solutions represents a chromosome, and each parameter in the solution set is a gene. A fitness function is used to evaluate the fitness of each individual in the population.
2. **selection**: With probabilities proportional to the fitness scores, some individuals are selected as parents. The mechanism simulates the natural selection in which the fittest individual survives.
3. **crossover**: After the selection process, the survived individuals are mated to reproduce the new generation. One or two *crossover point(s)* is chosen randomly. The chromosomes of two parent solutions are swapped before, after, or between the crossover point(s), as shown in 3.1.
4. **mutation**: The mutation operator maintains the diversity of the population and prevents the solutions from being trapped in local optima. After child solutions are generated, one or multiple genes are altered. Mutation can be done by flipping some of the genes in the chromosome or simply changing some genes randomly. Mutation happens with low probabilities in GA.

By repeating the operations mentioned above, the algorithm keeps generating new generations, and the entire population becomes better generation by generation. The algorithm terminates if the population has converged. That is, the parents do not produce offspring that are significantly different from the previous generation. The best solution in the remaining population is returned as the best approximation of the optimal solution of the given problem.

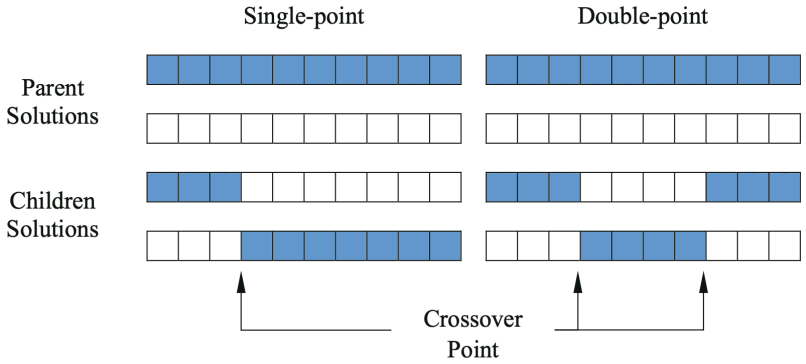


Figure 3.1: Two popular crossover techniques in GA. Retrieved from [60].

GA has been widely used in literature to solve optimization problems including assignment problems. Chu et al. [12] have presented a GA-based heuristic to solve the generalized assignment problem. Different GA variants have also been proposed to deal with the gate assignment problem [44][62]. Also, genetic

algorithm has a wide range of application in airport operations, including optimization of airport ground operations [34], scheduling for baggage transport vehicles [37], ground traffic optimization [36] and finding optimal locations of airport fire stations [98].

Ant colony optimization and bee colony optimization are well-known swarm intelligence techniques in literature. The main inspiration of ACO is the concept of stigmergy, and BCO is inspired by the foraging behavior of honeybees. Both ACO and BCO have been proven to solve difficult assignment problems with their ability to fast converge to high-quality solutions. Some assignment problems, for example, the gate assignment problem have been solved by Zhao and Cheng using ACO [105]. Deng et al. proposed an improved ant colony optimization (ICMPACO) algorithm [20] that can balance the convergence speed and solution diversity. In their experiments, Deng et al. verified the performance of ICMPACO by solving TSP and the gate assignment problem. Furthermore, BCO has also been applied to solve gate assignment problems. Dell'Orco et al. have developed a metaheuristic algorithm that is based on the Fuzzy Bee Colony Optimization [19].

3.1.3. Auction

Centralized approaches are often inefficient for distributed systems due to computational time and communication limits. Therefore, auctions are studied by researchers to better solve the task allocation problems [51]. Auction is a kind of market mechanism. It runs under specific rules that determine the allocation of the goods and the price [61][51]. Auctions have been studied extensively in the field of artificial intelligence and multi-agent task allocation. An early work applying auctions into distributed problems is the contract net protocol [89]. Other examples of utilizing auction in task allocation include allocating roles to robots in RoboSoccer [31], task to ambulance teams, fire bridges and police force in RoboCup Rescue [63], and tasks allocation for online pickup and delivery problems [71].

Multi-agent task allocation problems can be modeled as auction-based agent coordination systems. In the system, agents are bidders, and the tasks to be allocated are items to be auctioned. All agents bid their costs on the tasks. Then, the agent with the lowest bid on a task wins that task. Also, there is an auctioneer which is responsible for determining the winner and awarding the winning tasks to the bidders. Different from auctions used in economics, auction mechanisms used for task allocations are usually cooperative auctions. Critical problems in cooperative auctions for task allocation are establishing the bidding rules and task scheduling rules that optimize the desired problem objectives. On the other hand, in economics, bidders are often competitive, focusing on strategic behavior and possible collusion. Still, some insights from economic can be employed in auction-based coordination systems.

3.1.4. Game-theoretic approaches

Another approach that formulates the task allocation problem is the game-theoretic approach. Agents are modeled as independent decision-makers that attempt to maximize their own utility in the game. In game theory settings, agents are often competitive and are unwilling to share relative information. Thus, the decisions of agents are made based on their local information and their expectations of other agents. Game theory approaches have been applied in various competitive problems. In recent studies, it has also been implemented in some cooperative settings.

3.2. Comparison of solver techniques

In this section, a general comparison of different task allocation solution classes is presented. Based on the characteristic of the solver classes discussed in section 3.1, we evaluate the classes of solvers using the following criteria:

- **Robustness**: Robustness of a solution class includes the success rate of the solver, the ability to deal with uncertainties, additional constraints, or failures.

- **Solution quality**: Solution quality reflects the optimality and completeness of the solver. That is, the obtained total cost of the allocation problems and the percentage of successfully assigned tasks.

- **Scalability**: Scalability measures the extent that the approach is able to maintain its computational efficiency as the scale of the problem expands, such as increasing the number of agents, tasks, and constraints.
- **Efficiency**: Efficiency reflects the required runtime of the approach to return feasible solutions.
- **Complexity**: Complexity evaluates how easy it is to implement the related algorithm and to extend previous works. Note that the complexity here does not refer to the time complexity of the algorithm.
- **Flexibility**: Flexibility represents how easy it is to modify the model with varying objectives, characteristics of agents, or task constraints.
- **Applicability**: Applicability refers to how well the approach is suited for implementation in task allocation for automated aircraft ground handling. The measure includes the similarities between the characteristics of agents and the team objective.

The comparison is presented in Table 3.1. For applications in task allocation of automated aircraft ground handling, the criteria listed in the left-most column are not equally-weighted. Therefore, instead of presenting quantitative scores, marks like ✓, Δ, and ✗are given.

Table 3.1: Comparison of different task allocation solution classes.

	Typical optimization approaches	Evolutionary and swam algorithms	Auction mechanisms	Game theoretic approaches
Robustness	✗	✗	✓	✓
Solution quality	✓	Δ	Δ	Δ
Scalability	✗	Δ	✓	✓
Efficiency	✗	Δ	✓	Δ
Complexibility	✓	✗	Δ	Δ
Flexibility	✗	Δ	✓	✓
Applicability	✓	Δ	✓	✗

Overall, the class of auction mechanisms seems like the most appropriate approach. The performance of auctions is good or acceptable in every single criterion. Auction mechanisms are robust task allocation approaches since the uncertainties can be included in agents' bids. Centralized approaches like operation research or swarm algorithms are not as robust because the whole problem has to be solved again if the team objective or the incurred cost of task-agent combinations change. In terms of solution quality, auction usually provides suboptimal solutions. However, the solution quality is still fairly good. Even in some cases, sub-optimality can be guaranteed [53]. While the solution quality of auction mechanisms is not as good as the typical optimization approaches, the scalability and efficiency of auctions outperform that of typical optimization approaches. The long computational time of typical optimization while dealing with large-scale problems makes it difficult to be implemented in real-world applications. Evolutionary and swam algorithms have shorter computational time, but they still need great amounts of generations to stabilize. Regarding the criteria of complexity, the implementation of auctions is harder than that of typical optimization approaches, but it is still manageable. Also, the flexibility of the auction mechanisms is better than other approaches. Lastly, the applicability of auctions to be implemented for task allocation of automated aircraft ground handling is high. Agents' bids can capture the individual goals of agents under various circumstances throughout the auction process. Auction-based multi-agent task allocation solvers can also efficiently work with different team objectives. To sum up, the solution class of auction mechanisms is the most suitable framework for task allocation of automated aircraft ground handling.

3.3. Auction-based multi-agent task allocation

In the rest of this chapter, we focus on auction mechanisms. Auction-based multi-agent task allocation solvers are presented and discussed in this section. Compared to centralized allocation methods, auctions have the advantage of distributing the computation of bids to the involved agents. Auctions applied in task allocation problems are most likely multi-unit auctions of heterogeneous items, as there are usually multiple non-identical tasks to be assigned. Multiple tasks can be allocated in parallel auctions, sequential auctions, or

a single combinatorial auction. In parallel auctions, tasks are allocated independently in multiple simultaneous single-round auctions, one for each task. Synergies among tasks are not taken into account, resulting in poor solution qualities. Another well-known class of multi-unit auctions is Sequential Single Item (SSI) auctions [51]. Tasks are allocated in multiple auction rounds in SSI auctions. SSI auctions take parts of synergies among tasks into account, which can provide bounded suboptimal solutions [53]. Combinatorial auctions (CA) allocate tasks in a single round auction. Each bidder bids on all sets of tasks. Optimal solutions can be found as synergies among tasks are fully taken into account. However, in combinatorial auctions, bids generated are exponential to the number of tasks to be assigned. Thus, this kind of auction is not applicable to large online problems because of the high computational costs and communicational demands. A trade-off option between SSI and CA is sequential single cluster (SSC) auction [51][39]. In SSC, agents are allowed to bid on some subsets of tasks. Compared to SSI, more synergies are captured in SSC, while the required computational time can be significantly reduced.

The above mentioned algorithms use a specified central agent as the auctioneer. Still, the role of the auctioneer can vary. It can be different agents while solving the problem. In real-world applications, having all agents communicate to a central auctioneer may be impractical because of limited communication abilities. Decentralized algorithms that consist of multiple local auctions with auctioneers acted by different agents can be possible solutions. Consensus-based auction algorithm (CBAA) and its multi-assignment extension, consensus-based bundle algorithm (CBBA), are proposed by Brunet et al. [11]. The two algorithms are used to solve static allocation problems using a consensus routine based on local communication, allowing agents to converge on conflict-free allocations independently.

For applications in aircraft ground handling, parallel auctions are inappropriate because of their poor solution quality. Taking partial synergies among tasks into account, SSI, SSC and their variants are possible solution approaches. They are both efficient algorithms suitable for online use, at the same time, they can provide solutions with rather good qualities. Extensions of SSI taking temporal constraints and task duration uncertainty into account will be addressed in the following sections. These extensions captured the characteristics of automated aircraft ground handling well, making them strong candidates for this study. In subsection 3.3.1, we discuss SSI and its extensions. In subsection 3.3.2, SSC and its variants will be further discussed. Considering the full synergies among tasks, combinatorial auctions can provide optimal solutions but they required large amounts of computational efforts. Thus, CA is an ideal solving approach only when the size of problems is limited. Detailed discussions of CA will also be presented in subsection 3.3.3. On the other hand, approaches using multiple local auctions, namely CBAA and CBAA, allocate tasks without a central coordinator. For ground handling operations that struggle to reach punctuality, task allocation with local auctions may cause inconsistent situational awareness among agents. These approaches can return solutions far from optimal, which is not suitable for applications in automated aircraft ground handling operations. Therefore, CBAA and CBAA will not be discussed further in this literature study.

3.3.1. Sequential Single Item (SSI) auctions

In this section, Sequential Single Item (SSI) auction and its variants are discussed.

SSI

In Sequential Single Item (SSI) auction, tasks are allocated independently in a multi-round auction. In each round, every agent bids on all unallocated tasks. The auctioneer then performs winner determination and allocates a single task with the best bid value to the agent bids the price. Early research of Dias and Stentz [21] has shown that SSI auctions can efficiently handle multi-agent task allocation problems.

Lagoudakis et al. [53] were the first to perform theoretical studies on SSI auctions for a multi-agent coordination problem, multi-robot routing. They analyzed the bidding rules for three different objectives. Rizzo rephrased the notations used in the objectives and bidding rules of Lagoudakis et al. in the literature study of his master thesis [71]. The notations in the following descriptions ally with the ones used by Rizzo. Consider a multi-robot routing problem as an allocation problem with a set of agents $A = \{a_1, a_2, ..., a_n\}$, a set of targets $T = \{t_1, t_2, ..., t_m\}$, and their locations. Let i, j be two of the locations, $c(i, j)$ denotes the cost of travelling between location i and j. The traveling costs are assumed to be symmetric, that is, $c(i, j) = c(j, i)$, are the same for all robots, and fulfill triangular inequalities. Assume $P = \{P_1, P_2, ..., P_n\}$ is a partition of T, in which the tasks

in P_i are allocated to agent a_i. Also, two functions are defined. Function $APC(a_j, P_j)$ returns the minimum agent path cost for agent a_j by travelling to all targets in P_j. Similarly, function $CTPC(a_j, P_j)$ returns the minimum cumulative agent path cost for all agents a_j travelling to their assigned targets P_j. Lagoudakis et al. studied the multi-robot routing problem as task allocation problems under three objectives:

- MINISUM: minimize the sum of path costs over all agents. That is, $min_P \sum_j APC(a_j, P_j)$.
- MINIMAX: minimize the maximum path cost over all agents. That is, $min_P max_j APC(a_j, P_j)$. This objective is also known as minimizing the *makespan*.
- MINIAVE: minimize the average path cost over all agents. That is, $min_P \frac{1}{m} \sum_j CTPC(a_j, P_j)$.

Lagoudakis et al. proposed several bidding rules for agents that help to achieve the above optimization objectives [53]. Let $S = \{S_1, S_2, ..., S_n\}$ be the current partial allocation of the targets and t be an unallocated task. Beginning the next round of the auction and putting task t on the auction, each agent a_i should bid the difference in performance for the given objective between the current allocation and the allocation of adding t to its current allocated task set. It is important to note that the allocation approach is done in a hill-climbing way. Good solutions can usually be found, but optimality is not guaranteed. Depending on the objective, the following bidding rules should be applied:

- BID-MINISUM: $APC(a_i, S_i \cup t) - APC(a_i, S_i)$. Agent a_i bids the marginal increase of path cost incurring in adding t to its plan.
- BID-MINIMAX: $APC(a_i, S_i \cup t)$. Agent a_i bids the new path cost containing t in its plan. Note that the term of the maximum agent path cost before allocating t can be neglected. This term is a system-level variable, and it is the same among all agents.
- BID-MINIAVE: $CTPC(a_i, S_i \cup t) - CTPC(a_i, S_i)$. Agent a_i bids on the increase of the cumulative path costs over all agents incurring in adding t to its plan. The factor $1/m$ can be omitted since it is identical for all agents.

For real-time applications, the agents must generate the bids efficiently. Computing *APC* and *CTPC* requires agents to generate the optimal path for visiting all their assigned targets. The computation is NP-hard. To deal with this barrier, Lagoudakis et al. proposed an insertion heuristic method to approximate the optimal paths [53]. Each agent checks the possible insertion points of adding the new task t into their original path. Then, each agent uses the insertion point that leads to the least increase of path costs. The authors also provide a theoretical assessment of the performance achieved by using the above described bidding rules and the proposed heuristic. The upper and the lower bounds of the ratios between the obtained cost to the cost of optimal allocations are provided in the original research paper of Lagoudakis et al.[53]. The suboptimality bounds still hold when utilizing other heuristics to approximate the optimal paths of agents, as long as the approximation is no worse than the insertion heuristics proposed in the paper.

TeSSI

Temporal sequential single-item (TeSSI) auction is an extension of SSI auction introduced by Nunes and Gini [64]. In TeSSI, temporal constraints indicating the time that the task must be done are expressed as time windows. Time windows of the tasks to be allocated do not have any restrictions, and they can overlap. The agents keep track of their allocated tasks using simple temporal networks (STN).

Task allocation complying with the temporal constraints and minimize a cost function, for example the objective MINIMAX (that is, minimizing the *makespan*), is being investigated. Like in SSI [53], consider a set of agents $A = \{a_1, a_2, ..., a_n\}$ and a set of tasks $T = \{t_1, t_2, ..., t_m\}$. Each task t_i has its earliest possible start time $E_{S_{t_i}}$ and its latest start time $L_{S_{t_i}}$, where $E_{S_{t_i}} \leq L_{S_{t_i}}$. The duration of the task is denoted as DUR_{t_i} and the latest finish time is $L_{F_{t_i}}$, with $L_{S_{t_i}} + DUR_{t_i} = L_{F_{t_i}}$. The interval $[E_{S_{t_i}}, L_{S_{t_i}}]$ defines the time interval in which the task has to be done.

In TeSSI, the procedure of the auction follows the same logic as the SSI auctions. The auction starts when the auctioneer announces the tasks for bidding. In each round of the auction, every agent bids their cost on the tasks. The agents take into account their current schedule. The auctioneer receives bids from the agents

and allocates the task with the lowest bid to the corresponding agent. If no agent bids on a task, resulting in the task cannot being assigned, the task is removed from the set T and is added to the unallocated task set $T_{unalloc}$. The auction continues until the set of tasks T becomes empty.

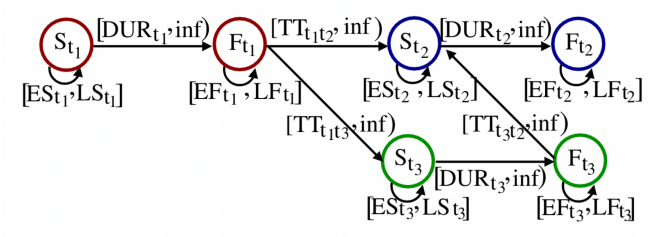


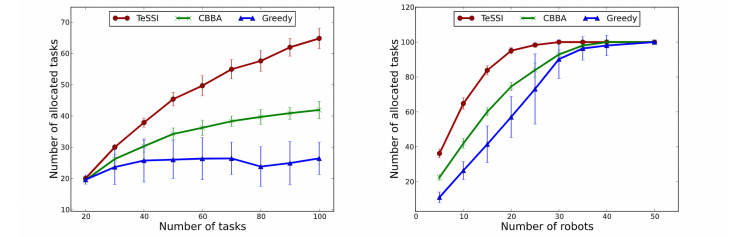
Figure 3.2: Example STN of three tasks. Retrieved from [64].

When an agent bids on a task, it takes its current schedule, which is the STN it kept, into account. An example three task STN is provided in Figure 3.2. Every task t_i is represented by the starting time point S_{t_i} and the ending time point F_{t_i}. A reference origin with time 0 is also added to the STN, although it is not shown in Figure 3.2. There are two main constraints incorporated in the STN, the duration constraints and the travel time constraints. The duration constraints imply that the start time of a task cannot be later than the end time of the task, that is, $F_{t_i} - S_{t_i} \in [DUR_{t_i}, \infty]$. The travel time constraints state that an agent can only reach its next task t_i after finishing its previous task t'_i, that is, $S_{t_i} - F'_{t_i} \in [TT_{t'_i,t_i}, \infty]$, where $TT_{t'_i,t_i}$ represents the travel time between the two locations. Agents only bid on a task if the related constraints are met. While computing a bid for a task, agents try to insert the task into all the possible segments in its schedule without violating the constraints, and it selects the point that minimizes the additional cost of containing the task in its schedule. The agent then bids the corresponding cost. Considering the team objective MINIMAX, the bidding rules of the agents are to bid the resulting *makespan* of the agents' schedule. Once a task is added to an agent's schedule, the STN of the agent is propagated using the Floyd-Warshall algorithm.

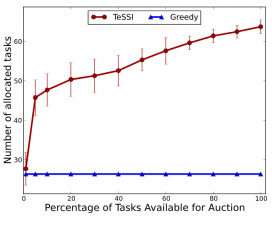
TeSSI works well both when all the tasks to be assigned are known upfront or are introduced dynamically. Consider an allocation problem with n agents and m tasks to be allocated, the complexity of TeSSI is $O(nm^2)$. In the bidding phases, the complexity of the propagation of STN is $O(n^3)$. Nunes and Gini have performed some experiments, evaluating the performance of TeSSI in simulations compared to a version of Consensus Based Bundle Algorithm (CBBA) that handles time windows, and the greedy algorithm. Each agent in the simulation bids the *makespan*. First, an experiment of the offline case is carried out. The results are shown in Figure 3.3a. Figure 3.3b shows the result of an experiment of the online case, in which tasks are dynamically introduced to the system in batches with the size of 1,5,10,20, ..., 100. Results in both experiments show the TeSSI outperforms CBBA and Greedy. In the online case, the performance of TeSSI improves with the number of tasks available. Nunes and Gini claim that larger batch sizes of tasks allow TeSSI to better capture synergies among tasks [64]. In the last experiment, the spatial and temporal data in Solomon ' s dataset [90] is used. The results are shown in Figure 3.3c. In the experiment, more tasks are allocated using TeSSI. Also, it is evident that TeSSI can capture the spatio-temporal synergy of the clustered tasks and produce solutions with higher qualities.

TeSSI is an efficient algorithm that allocates tasks systematically. In the original paper of TeSSI, the objective MINIMAX and a combination of MINIMAX and MINISUM (that is, the sum of the travelled distances) are used. In principle, the algorithm can work with any other objective and bidding rule. The main downside of the algorithm is that it does not guarantee optimal and complete solutions. In TeSSI, task allocation is done in a hill-climbing fashion like the way that is performed in SSI. Further, due to the introduction of temporal constraints, it is possible that in some rounds of the auction, single or multiple tasks cannot be allocated as no agent can include the tasks in its schedule without violating constraints. In such cases, a possible solution is to introduce more agents to the assignment problem.

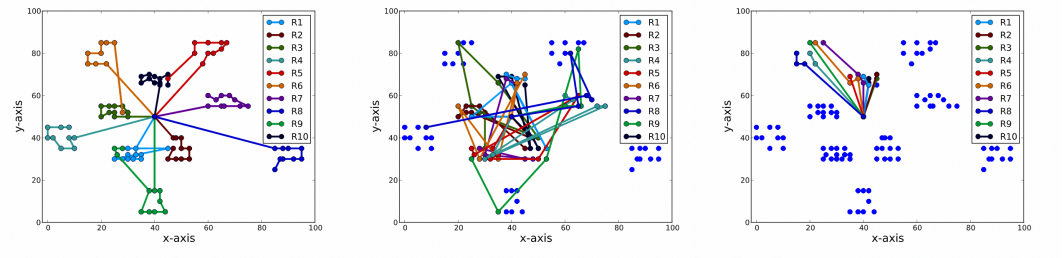
Having considered the time window of tasks, TeSSI is suitable to be applied to the task allocation problems of automated aircraft ground handling, as certain ground handling operations have to be done in certain time windows. Also, according to the experiments performed by Nunes and Gini [64], tasks that need to be done



(a) Results for simulations allocating 20-100 tasks to 10 agents (left), and 100 tasks to 5-50 agents.



(b) Results of simulation allocating different numbers of tasks to 10 agents.



(c) Results of allocations done by TeSSI (left), CBBA (mid) and Greedy (right) on Solomon ' s instance C101.

Figure 3.3: Results of experiments comparing the performance of TeSSI, CBBA, and Greedy. Retrieved from [64].

at adjacent aircraft stands are likely to be captured by TeSSI, making the implementation of the algorithm promising. However, in TeSSI, tasks that cannot be assigned are simply discarded, which is an unacceptable operation in aircraft ground handling. This disadvantage should be fixed if the algorithm is being implemented in our study. Moreover, as mentioned in the original paper [64], one possible extension is to include task execution uncertainties, including uncertain task duration. An extension of TeSSI, pTeSSI, which takes task duration uncertainties into account is discussed in the next section.

pTeSSI

To deal with the uncertain task duration, which is realistic in real-world dynamic settings, Rizzo proposed probabilistic Temporal Sequential Single Item (pTeSSI) auction [71]. pTeSSI is a variant of TeSSI, in which the schedules of agents are represented as a simple temporal network with uncertainty (STNU). By allowing agents to bid on non-deterministic tasks, pTeSSI enables allocating tasks with uncertain temporal constraints.

The bidding procedure of pTeSSI works the same as TeSSI, except for the bidding rules of the agents. In Rizzo's study [71], two objectives are considered, namely MINIMAX and MINISUM. To capture the probabilistic of agents' plan, the risk of unsuccessful dispatches are included in the bidding rules. Let $P = \{P_1, P_2, ..., P_m\}$ be the final allocation of tasks where P_i represents the set of final allocated tasks to agent a_i. The minimum time it takes agent a_i to complete the tasks in P_i is denoted as M_i, and D_i represents the minimum distance that must be traveled by agent a_i to complete its schedule. Besides, let $R_i(P_i)$ represent the probability that the schedule P_i will not be successfully executed, and therefore the time M_i and distance D_i cannot be achieved. The objectives that pTeSSI seeks to minimize are presented in the following:

- MINIMAX: $min_P max_i [M_i(P_i) + \rho \cdot R_i(P_i)]$
- MINISUM: $min_P \sum_i [D_i(P_i) + \rho \cdot R_i(P_i)]$

where ρ is a parameter indicating the penalty factor of the risk. Also, M_i and D_i are normalized to a $[0,1]$ range, being in the same range as the scheduling risk R_i. The normalization makes the factor ρ scale-agnostic and easier to interpret. Let $S = \{S_1, S_2, ..., S_i\}$ be the current schedules of agents at a given round, and t is the next task to be allocated. For the two objectives, agent a_i's bid for task t can be represented by the following:

- MINIMAX: $M_i(S_i \cup t) + \rho \cdot R_i(S_i \cup t)$
- MINISUM: $D_i(S_i \cup t) + \rho \cdot R_i(S_i \cup t) - (D_i(S_i) + \rho \cdot R_i(S_i))$

Note that for the MINIMAX objective, the maximum traveled distance in the previous round is a system-level factor that can be dropped.

The determination of D_i and M_i can be efficiently done using STNU. The schedule of agents is represented as STNUs, in which tasks are sequences of nodes and edges. Figure 3.4 shows an example of STNU containing two pickup and delivery tasks. Each pickup and delivery task is divided into four different actions. Every action consists of a start node s and a finish node f, and the two nodes are connected via a *contingent* constraint f_a-s_a=X_a represented by a curve arrow. The task duration X_a is a uniformly distributed random variable that lies in the interval $[X_a^l, X_a^u]$, in which X_a^l is the lower bound and X_a^u is the upper bound of X_a's distribution. As noted by Rizzo [71], the distribution of task duration X_a are first assumed to be uniformly distributed but they can be extended to any distribution. There are also *requirement* constraints in STNUs, represented by straight arrows. *Requirement* constraints enforce the sequence that tasks and actions must occur. All nodes are scheduled relative to an origin node that represents time zero.

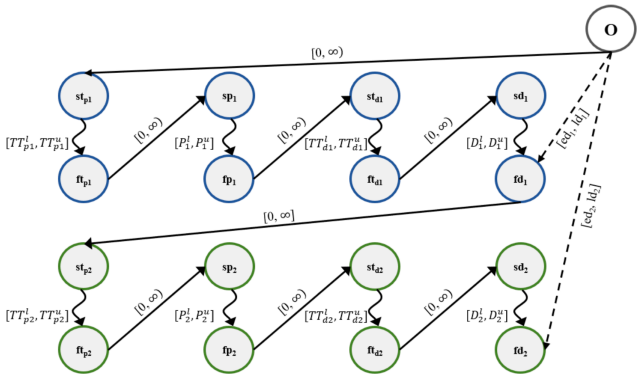


Figure 3.4: Example STNU containing two pickup and delivery tasks. Retrieved from [71].

Using its STNU, every agent can determine its bid for an auctioned task t. The agent adds a task t to all valid insertion positions in its STNU and generates the associated constraints. An insertion position is valid if the earliest delivery time of the new task is no larger than the latest delivery time that would follow it. Let r denote the risk associated with the new schedule of an agent, representing the probability that the contingent constraints in the agent's STNU cannot be satisfied. As long as r is smaller than a certain threshold of the agent, it may be profitable for agents to bid on a task even if successful dispatches of the resulting schedule are not guaranteed. After exploring all the valid insertion points in its STNU, the agent saves the lowest bid and the corresponding insertion point and submits the bid to the auctioneer. Like the TeSSI auctions, pTeSSI auctions are done in multiple rounds. At any given round, tasks that cannot be assigned to any agent are discarded. The auction continues until all the tasks are allocated.

The overall complexity of pTeSSI is the same as regular TeSSI even though pTeSSI is solving more complicated problems while generating bids [71]. Similar to TeSSI, pTeSSI does not guarantee optimality and completeness. pTeSSI is suboptimal as it does not capture all possible synergies between tasks due to its hill-climbing-fashioned allocation. However, following from the general results on single sequential item auctions, bounded-suboptimality can be proved [53]. As for completeness, pTeSSI discards tasks that cannot be assigned to any agent. Thus, the solution of pTeSSI can be incomplete. However, this situation should happen less than that in TeSSI as risks under a certain threshold are allowed in the bidding phases.

In his experiments, Rizzo evaluates TeSSI and pTeSSI in several simulations of online unmanned aerial vehicle food delivery operation in the Amsterdam area [71]. Orders are generated randomly with different demand levels, namely 60, 120, and 180 orders in total. As the results shown in Table 3.2, the successful total deliveries with pTeSSI are higher than that of TeSSI. Taking reasonable risks, the allocations of pTeSSI can be more flexible. Moreover, the auction durations of pTeSSI are much lower than that of TeSSI, and the auction durations of pTeSSI are all in a matter of seconds. The experiments performed by Rizzo [71] show that pTeSSI

is suitable for online food delivery systems. The efficient calculation time, together with the ability to capture task duration uncertainties makes pTeSSI a suitable candidate for solving online task allocation problems in automated aircraft ground handling. However, even though the bounded-suboptimality of pTeSSI is proven, its limited ability to capture synergies among tasks is still of great concern. In the next section, sequential auctions that allocate tasks in bundles will be discussed.

Table 3.2: Comparison between the total deliveries and auction duration of TeSSI and pTeSSI. Adapted from [71].

		Total deliveries		Auction duration	
Demand level	Objective	TeSSI	pTeSSI	TeSSI	pTeSSI
60	MAX-T	55.3	59.8	1.61	0.59
	SUM-DIST	50.1	59.7	2.92	0.94
120	MAX-T	77.8	109.9	11.4	2.08
	SUM-DIST	80.25	111.8	17.2	3.82
180	MAX-T	79.3	127.7	32.5	3.5
	SUM-DIST	89.7	131.0	44.2	5.4

3.3.2. Sequential single cluster (SSC) auctions

Although in SSI auctions, some synergies are taken into account when agents generate bids for their successive tasks, the considered combination of tasks is still quite limited. Within the pool of tasks, some collaborative tasks largely reduce the team cost if being performed together. In SSI, those collaborative tasks may be allocated to different agents due to the sequential allocation nature of the auction, resulting in higher team costs. Fully combinatorial auctions can be used to find the lowest team cost of the allocation problem, but the required amount of computation is considerable. To better capture synergies among tasks and restraint the required amount of computation, sequential single cluster (SSC) auctions were introduced as middle-ground options between SSI and fully combinational auctions. In this section, various SSC auctions are introduced.

SBB

Sequential bundle bid (SBB) auction is proposed by Koenig et al. [50] as a generalization of SSI. To reduce the team cost of SSI auctions, in SBB auctions, multiple tasks are assigned during each round. In every round of the SBB auction, each agent bids on all non-empty bundles of at most k out of m tasks to be auctioned. This allows agents to capture synergies among the k tasks it bids on. The bundle size k represents the trade-off between SSI auctions ($k = 1$) and combinatorial auctions ($k = m$). After receiving bids from the agents, the auctioneer then allocates exactly k tasks to one or multiple agents. Koenig et al. claimed that the determination of the winning bids can be done with a runtime that is linear in the number of bids. Also, SBB auctions have been shown to successfully reduce the required computational and communication effort by a large amount. However, during the bidding phases, agents need to examine the cost of a large combination of tasks to determine the set of tasks to bid on. Still, As long as the generation of agents' bids can be performed within a reasonable time, SBB can be a candidate task allocation mechanism for the applications of automated aircraft ground handling.

SSC

Although SBB auctions help to lower the total team cost, the involved calculation of bid generation and winner determination can take quite some time in cases with larger sizes. To better pursue the idea of bidding on subsets of tasks, sequential single cluster (SSC) auctions are proposed by Heap and Pagnucco [38]. In SSC, fixed clusters of tasks are built before auctions. In every round of the auction, agents in an SSC auction will either win all items in a cluster or none. The authors utilized the k-mean algorithm to form clusters. In principle, any existing clustering algorithm that takes locations in space as its input can be implemented.

In the original study of Heap and Pagnucco [38], clusters are formed based on single locations of the tasks. Moreover, in another study of Heap and Pagnucco [39], they studied a variant of SSC auction that is designed specifically for online pick-up and delivery problems. Clustering the location pairs of pick-up and delivery points is essentially more complicated than clustering single locations. They proposed a two-level clustering approach that first forms clusters of tasks based on their pick-up locations. Then, within each cluster, tasks

with delivery locations in the vicinity are again clustered. Figure 3.5 illustrates the idea of this two-level clustering approach. In addition, the agents may have capacity constraints representing the maximum number of tasks they can execute at any given time throughout the process. The capacity constraints can be taken into account while generating clusters. This variant of SSC worked well in the experiments conducted by Heap and Pagnucco.

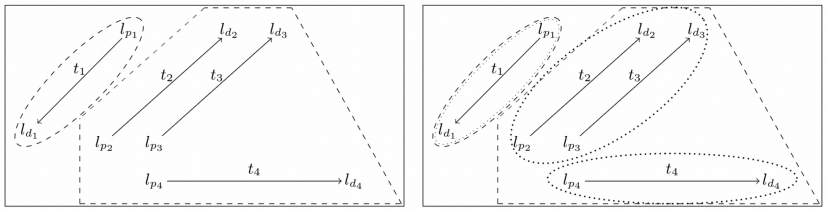


Figure 3.5: Illustration of the two-level clustering approach. First, tasks are clustered based on their pick-up locations (left). Then, they are clustered again based on their delivery locations (right). Retrieved from [39].

The authors performed some experiments comparing SSC and SBB over the percentage of improvement against typical SSI. It was shown that SSC outperforms SBB for both the MINIMAX and MINISUM objectives [39]. In addition, Heap and Pagnucco proposed an extension of SSC to deal with the dynamically emerging tasks. As the aircraft ground handling tasks are most known beforehand, the online extension of SSC will not be discussed in this literature study.

To some degree, pick-up and delivery problems resemble the task allocation problems in automated aircraft ground handling. The GSEs have to depart from the depot or other parking locations to the aircraft stands to perform the ground handling operations. Also, the GSE vehicles have some capacity limits. For example, the water service trucks can only carry the water for 3 to 4 flights in their tank. These similarities make task allocation algorithms for pick-up and delivery problems possible to adapt to solvers for automated aircraft ground handling. Nevertheless, to implement SSC into task allocation of automated aircraft ground handling, some modifications are needed. Most importantly, the temporal constraints of the tasks need to be accounted for. A possible solution is to add a temporal layer in the clustering phases, grouping the tasks with close pick-up or delivery times together.

pTeSSB

To remedy the limited amount of inter-task synergies that are able to be considered, Rizzo also proposed an extension of pTeSSI in his master thesis, called probabilistic Temporal Sequential Single Bundle (pTeSSB) auctions. In pTeSSB, it is possible for agents to de-commit to subsets of tasks in their schedule and sends them to the auctioneer to be re-auctioned. Note that although the re-auction process increases the captured synergies, it also prolongs the duration of the allocation process as more tasks are needed to be allocated at the same time. To remedy this effect, Rizzo also proposed a mechanism that bundles tasks and auctions them together.

For agents to decide which tasks to be re-auctioned, bids that are proportional to the potential reduction of the team objective are generated. Agents de-commit from the tasks in their schedule if the bids exceed a threshold. The re-auctioning procedure increases the number of tasks to be allocated at once, and thus increases the required round of auction. To mitigate this effect, bundles of tasks are generated. Other than only considering the spatial characteristics of events as previous literature does, Rizzo proposed a mechanism that bundles tasks that can be accomplished sequentially with acceptable risks. He defines a parameter *distance* that characterizes both the proximity of sequential pickup and delivery locations and the coordination of the time windows (Note that this *distance* does not represent the actual spatial distance). Tasks are grouped as a bundle if the largest distance across all tasks in the bundle is below a threshold *distance*.

In the bidding phase, the generation of agents' bids is pretty much the same as the bids generation in pTeSSI. The main advantage is that bundled tasks are already ordered, so the number of insertion points to check can be greatly decreased. Rizzo also demonstrated experimentally that pTeSSB has shorter auction du-

rations than pTeSSI. Experiments similar to the comparison between TeSSI and pTeSSI are performed. Rizzo evaluated pTeSSI and pTeSSB with demand levels of 120, 180, and 240 orders in total. As the results shown in Table 3.3, the successful total deliveries with pTeSSB are higher than that of pTeSSI, while the auction durations of pTeSSB are lower than that of pTeSSI.

Table 3.3: Comparison between the total deliveries and auction duration of pTeSSI and pTeSSB. Adapted from [71].

	Total deliveries		Auction duration	
Demand level	pTeSSI	pTeSSB	pTeSSI	pTeSSB
120	109.9	111.9	3.97	3.48
180	127.7	137.2	11.60	8.60
240	127.9	148.7	23.25	16.67

Although pTeSSB performs well in online pickup and delivery problems, whether it is suitable for applications of aircraft ground handling should be closely examined. Like other sequential auctions, pTeSSB does not guarantee the completeness of the solution. Completeness of the solutions can also be told from the total deliveries out of the total demand shown in Table 3.3. Having ground handling tasks that cannot be assigned is tricky in aircraft turnaround, and it can cause unbearable flight delays. Moreover, as the aircraft ground handling tasks are generally known beforehand, the advantage that can be taken from the re-auction processes of pTeSSB may be limited. Overall, pTeSSB may not be the best task allocation solver for applications in automated aircraft ground handling.

Up to this point, different generations of bundles of tasks are discussed. In SBB, agents bid on all possible bundles of at most k tasks to be auctioned. The k tasks are chosen by agents and may not necessarily be the same for all agents. The winner determination of SBB bids is more complicated than the bundles generated by the auctioneer. In contrast, in the extension of SSC, tasks are clustered by the auctioneer according to the pickup and delivery locations. The downside of this bundle generation technique is that the time windows of the tasks are not considered. Likewise, bundles are also centrally generated in pTeSSB. It utilized a special parameter *distance* that includes the spatial and temporal characteristics of tasks. However, non of the above discussed bundle generation technique fits the application in automated aircraft ground handling perfectly. In SBB auctions, agents have to evaluate a large combination of tasks to decide the subset to bid on. Also, the winner determination is rather complicated. The bundle generation in SCC does not involve the temporal constraints of the tasks, which might lead to schedules that are not able to be executed. Moreover, although the bundles in pTeSSB already captured the locations and time windows of the tasks, there are tighter restrictions to be considered in automated aircraft ground handling operations. For example, a baggage vehicle that just emptied its cart by loading baggage to an outbound flight is not able to load another outbound flight without picking up the baggage from the baggage center, but with its empty carts, it can unload the baggage from an inbound flight. As the above mentioned mechanisms do not fit the need in automated aircraft ground handling, we therefore study another bundle generation method. The bundle generation problems (BuGP) defined by Gansterer and Hartl [33] will be introduced in the following section.

3.3.3. Combinatorial auctions

Combinatorial auctions (CA) allocate tasks in a single round. Every agent bids on all possible bundles of tasks. The possible bundles are all subsets of tasks. By this means, all synergies between tasks are taken into account. After receiving the bids from agents, the auctioneer has to solve the winner determination problem (WDP) and announce the allocation. As CA captures all the synergies, optimal solutions can be guaranteed. However, CA is not applicable in large online problems due to its required amount of computation. In the following, a bundle generation approach that generates limited but attractive bundles is presented. The original CA is also discussed later in this section.

Centralized bundle generation problems (BuGP)

To efficiently accelerate the auctioning process by generating attractive bundles, Gansterer and Hartl defined the bundle generation problem (BuGP) [33]. BuGP aims to provide limited numbers of bundles that maximize the total coalition profit for the agents. The original study of Gansterer and Hartl was designed for solving collaborative transportation problems, in which there is some sensitive information that agents are not willing to share. Hence, the objective function of the problem can only be approximated by a proxy. On the other hand, in our study, agents have a common goal of pursuing efficient automated ground handling operations that makes the flight on time. There is no sensitive information of agents to be concealed. If BuGP is being applied to our research, an explicit objective function can be used while generating bundles.

To limit the number of bundles agents have to bid on, as well as to restraint the complexity of the WDP, the auctioneer offer bundles that can be combined to feasible solutions in WDP. During the bundle generation phase, the auctioneer constructs candidate solutions, which are combinations of non-overlapping bundles covering all tasks to be assigned. Figure 3.6 illustrates an example of building candidate solutions.

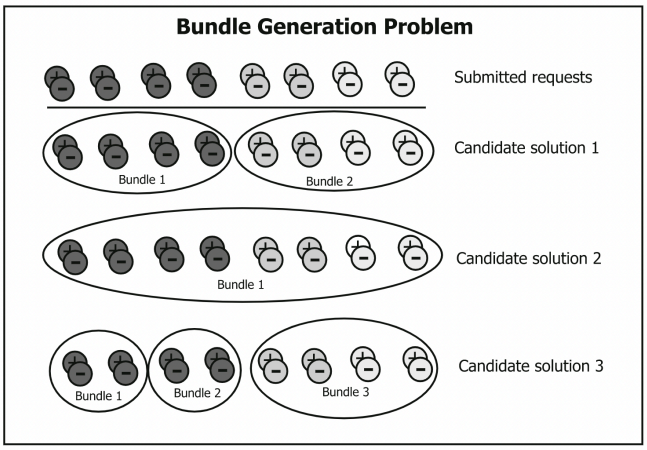


Figure 3.6: Illustration of building candidate solutions. Adapted from [33].

A genetic algorithm-based framework is used to produce the final offered bundles. A fitness function specific to the allocation problem is used to evaluate the value of a solution. GA produces a population of candidate solutions, that are sets of bundles subjected to the constraints of the problem. Let R be the total number of tasks to be auctioned, the decoding of candidate solutions are strings of R numbers. Each entry b_r represents the bundle that the task is assigned to. In order to identify duplicates in the population, a normalized representation is used in candidate solutions. That is, renumbering the bundle numbers while presenting candidate solutions. An example of decoding and the normalization is shown in Figure 3.7. The crossover and mutation mechanisms used in Gansterer and Hartl's research [33] are specific to the BuGP for collaborative transportation, and they are thus not further discussed in this literature study.

Bundles from the candidate solutions are offered to agents during the bidding phase. Agents then bid on all provided bundles. However, some bundles may not be feasible for certain agents to execute. Whether there are sensitive information agents do not want to disclose, it is impractical for the auctioneer to include all the information of agents while generating bundles as the auctioneer does not know the final allocation yet. Therefore, the auctioneer has to generate bundles under incomplete information. After the bundles are generated, agents bid on all provided bundles. Finally, the auctioneer solves the WDP and announce an optimal assignment.

Gansterer and Hartl also performed a computational study of BuGP. They compared the allocation result of auctions with complete sets of bundles and with only candidate bundles generated by BuGP. It is shown that the average runtime of allocation with candidate bundles is lower than the runtime of allocating complete bundles by a magnitude. At the same time, the resulting loss of solution quality is reasonable. Utilizing the fitness score in the GA framework, it is more flexible to include the desired factors while generating bundles. Moreover, with some carefully designed mechanisms in the genetic algorithm, BuGP can be integrated

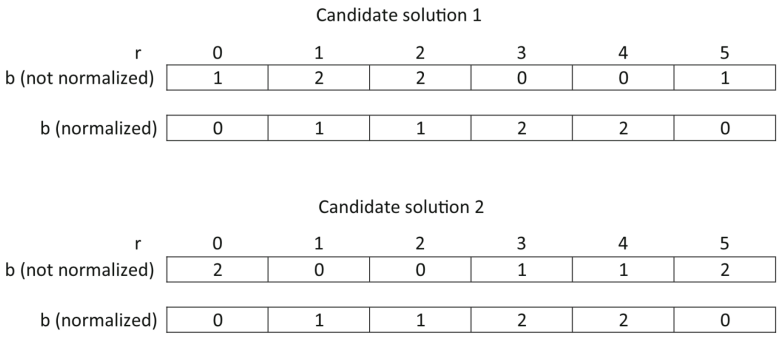


Figure 3.7: Decoding of candidate solutions before and after normalization. Retrieved from [33].

into the task allocation process of automated aircraft ground handling.

Fully combinatorial auctions

Now we introduce the original combinational auction that generates all possible bundles of tasks to be assigned (We also refer to the original combinational auction as full-CA). Considering n tasks to be allocated, the possible number of subsets is $2^n - 1$. The number of bids agents need to bid on grows exponentially with the number of tasks, which is not manageable in practical [33]. Also, solving the WDP of CA is NP-complete [74]. The bundles to be generated and the WDP makes CA inappropriate for real-world applications like task allocation of aircraft ground handling. Nevertheless, as CA gives optimal solutions, it can serve as a useful solver that can be used to evaluate the solution qualities of other auction mechanisms.

3.3.4. Summary of different auction approaches

We have discussed different auction-based task allocation algorithms, including sequential single item (SSI) auctions, sequential single cluster (SSC) auctions, and combinatorial auctions (CA). Now, the discussed algorithms are compared under some criteria, as shown in Table 3.4. The aim of this comparison is not to choose the best task allocation algorithm, but to have a comprehensive understanding of the applicability of algorithms under different criteria. The criteria are explained blow. Some analyses of the algorithms under the criteria are provided in the following paragraph.

- **Optimality, Completeness**: Whether the algorithm guarantees optimality and completeness, respectively, of the allocation results.
- **Temporal constraints, Uncertainty, and Synergy**: Whether the algorithm considers temporal constraints, uncertainties of task duration, and synergies among tasks, respectively, while allocating tasks.
- **Efficiency**: Efficiency reflects the required runtime of the approach to return feasible solutions.
- **Bundle quality**: The qualities of generated bundles. It also reflects to what extent the bundles are able to capture synergies among tasks and the preference of the agents.
- **Bundle numbers**: The number of bundles the auctioneer has to allocated and the agents have to bid on. Algorithm with lower bundle numbers receives a better mark.
- **Distributed auction**: Whether the algorithm has distributed auctioneers.

Optimality and completness

As previously discussed, the sequential allocation processes in sequential auctions (SSI and SSC) are hill-climbing approaches for optimization problems. These allocation processes do not guarantee optimal and complete solutions. On the other hand, as CA takes all synergies into account, and solves the allocation problem in a single round auction, it can provide optimal and complete solutions.

Table 3.4: Comparison of different auction-based task allocation solvers.

	SSI			**SSC**			**CA**	
Auction mechanism	SSI	TeSSI	pTeSSI	SBB	SSC	pTeSSB	BuGP	full-CA
Optimality	✗	✗	✗	✗	✗	✗	✗	✓
Completeness	✗	✗	✗	✗	✗	✗	Δ	✓
Temporal constraints	✗	✓	✓	✗	✗	✓	✓	Δ
Uncertainty	✗	✗	✓	✗	✗	✓	✓	Δ
Synergy	✗	✗	✗	Δ	Δ	Δ	Δ	✓
Efficiency	✓	✓	✓	Δ	✓	✓	✓	✗
Bundle quality	N.A.	N.A.	N.A	Δ	✗	Δ	✓	✓
Bundle numbers	N.A.	N.A.	N.A.	✗	Δ	Δ	Δ	✗
Distributed auction	✗	✗	✗	✗	✗	✗	✗	✗

Temporal constraints and uncertainty

Temporal constraints are taken into account in TeSSI, pTeSSI, and pTeSSB, while the task duration uncertainties are considered in pTeSSI and pTeSSB. The constraints and risks are already designed in the algorithms. Moreover, in BuGP, temporal constraints and risks can be taken into account in terms of the fitness score of bundles while the candidate bundles are made by the auctioneer. During the bidding phase, the agents can further evaluate the bundles and include these two criteria in their bids. In full-CA, it is still possible to reason about temporal constraints and task duration uncertainties. However, bundles that violate the temporal constraints and with unacceptable risks will still be generated in full-CA. The evaluation of the two criteria can only be expressed in agents' bids.

Synergy and efficiency

As the name of the auction approaches, SSI captures the least synergies among tasks. Only limited synergies are taken into account when agents generate bids for their successive tasks. SSC considers more synergies from the bundles formed. Lastly, full-CA captures all synergies among tasks. In general, considerations of synergies and the efficiency of allocation are somewhat on the opposite sides. The more synergies taken into account, the more computational time is required. Fortunately, for SSC, the pre-made bundles help to shorten the allocation processes and thus mitigate the increased computational time. While SSI auctions are in general the most efficient among all auction approaches, the required computational time of some SSC auctions is comparable to that of SSI auctions [38]. The only approach that requires more computation time is SBB, in which bundles are generated by the agents, and the WDP is rather complicated. Last but not least, the required computational effort of full-CA grows exponentially with the number of tasks to be allocated. The long runtime is usually not acceptable in practice.

Bundle quality and bundle numbers

For SSC and CA auctions, the quality and the numbers of the generated bundles and are also important criteria. The ways that how bundles are generated and the extent they fit the applications in aircraft ground handling are already discussed and compared in previous sections. In Table 3.4, we give the bundle quality of SSC a ✗because it does not take any time windows of tasks into account. The bundle quality of BuGP and full-CA are given ✓s because BuGP uses the fitness score of a GA framework to generate good bundles, and full-CA simply generates all possible bundles that cover the best subsets of tasks for sure. In terms of bundle numbers, agents have to generate bundles individually in SBB. The resulting number of bundles can be necessarily larger than the number of generated bundles of other SSC auctions. In full-CA, all possible bundles are generated. The amount can be significantly large as the number is exponential to the number of tasks to be assigned.

After exploring different auction mechanisms, below we make a final remark of the characteristic of auctions that are suitable to be applied in task allocation of aircraft ground handling. The procedures of sequential auctions are efficient and easy to implement in our model. However, only limited synergies are taken into account if only one task is allocated at each round of the auction. Thus, rather than SSI auctions, SSC auctions are better choices for our application. In SSC auctions, bundles can be generated in different ways. Utilizing

the SBB auction and letting the agents generate their preferred bundles is a favorable approach, given that the required computational time is acceptable in our settings. Furthermore, applying the BuGP to generate good candidate solutions is also a suitable approach. Even though a GA framework has to be constructed, all bundles will be generated before the allocation process. This approach makes the required time of bidding and winner determination shorter. The consideration of task duration uncertainties and temporal constraints should be included. A combination of pTeSSB auctions with bundles generated by BuGP can be a possible approach for allocating ground handling tasks.

4

Multi-Agent Path Finding

Multi-Agent Path Finding (MAPF) problem is an important type of multi-agent planning problem. In MAPF, paths have to be planned for multiple agents, so that agents can travel from their start locations to goal locations simultaneously without collision with other agents or obstacles in the environment. MAPF has a wide range of real-world applications, including automated warehouse [102], traffic control [25], robotics [27][103], and airport surface operations [48]. Stern et al. [92] and Felner et al. [30] have provided complete survey of MAPF algorithms and their applications.

4.1. Classical MAPF problem

In this section, the classical MAPF problem is discussed.

4.1.1. Problem Definition

The classical MAPF problem is described in the following. The classical MAPF problem contains an undirected graph, $G = (V, E)$, where V denotes vertices and E denotes edges, and a set of k agents, $a_1, ..., a_k$. Each agent a_i originates from a start position $s_i \in V$ and has a goal position $g_i \in V$. Time is assumed to be discretized. In each time step, every agent is can perform a single action, whether to $wait$ or to $move$. A $wait$ action represents that the agent stays in the current vertex for one time step, and a $move$ action means that the agent moves from its current vertex to an adjacent vertex in the graph. Let s_t^i be the vertex occupied by agent a_i at timestep t, a path $p^i = [s_0^i, ..., s_T^i, s_{T+1}^i, ...]$ is feasible for agent a_i if and only if the following conditions hold. The conditions are retrieved from [41]

1. Agent a_i originates from its start vertex, that is, $s_0^i = s^i$;
2. Agent a_i ends at its goal vertex g^i and remains there, that is, there exist a minimum T^i such that for all $t \leq T^i$, $s_t^i = g^i$;
3. Every action is either a move action along an edge or a wait action, that is, for all $t \in \{0, 1, ..., T^i - 1\}$, $(s_t^i, s_{t+1}^i) \in E$ or $s_t^i = s_{t+1}^i$.

4.1.2. Types of Conflict

The valid solution of the MAPF problem consist of a set of conflict-free paths for all agents. In classical MAPF, there are two types of conflict, *vertex conflict* and *edge conflict*, which are described below.

- A ***vertex conflict*** between agent a_j and a_k means that the two agents occupy the same vertex at the same timestep, that is, $s_t^j = s_t^k$;
- An ***edge conflict*** occurs if and only if the two agents traverse the same edge in the opposite direction at the same timestep, in other words, they swap their locations in a single timestep, where $s_t^j = s_{t+1}^k and s_t^k = s_{t+1}^j$ for some timestep t.

4.1.3. Assumptions

Some assumptions are commonly made in classical MAPF problems. Main assumptions that most MAPF algorithm assumes are [5] [15]:

- Time steps are discretized.
- Every agent has the same shape and size, and each agent only occupies a single location (vertex) at one time step.
- Each agent performs one action in exactly one time step, and the agent can only move to the 4-connected cells of its current location.
- Agent kinematics are ignored. That is, there are no limitations on velocities and accelerations of agents.
- Plan execution is perfect.

Although these assumptions do not always hold in real-world applications, the assumptions can be viewed as a basis while comparing various MAPF algorithms. MAPF variants that relax these assumptions are further discussed in section 4.3 and 4.4.

4.1.4. Objective Functions

Since MAPF is an optimization problem, the objective function has to be defined. The two most common objectives used to evaluate MAPF solutions are listed below.

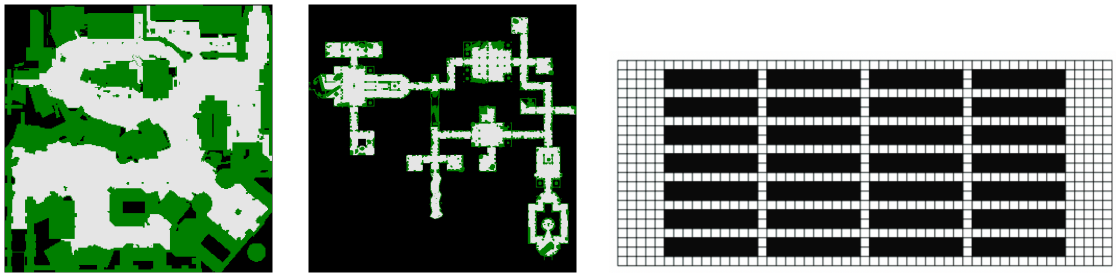
- ***makespan***: The maximum number of timesteps for agents to reach their goals.
- ***sum of costs***: The summation of the timesteps required by each agent to reach its target.

Overall, in airport ground handling operations, on-time performance is the most important objective. Thus, the objective *makespan* interprets our objective better. On the other hand, objective *sum of costs* is suitable to represent the total traveling time or distance of the vehicles. In general, with less driving distance, there are fewer risks of collisions. This parameter is thus suitable to evaluate the safety of traffic of ground handling operations. A combination of the two objectives can be used in the model of our research.

4.1.5. Benchmarks

A MAPF problem is defined by a graph and agents with a set of start nodes and goal nodes. Therefore, a benchmark for a MAPF problem includes the graph, the number of agents, and the sets of source and target vertices. In the experimental studies from literature, different MAPF solvers are compared using the same map. Stern et al. provided a thorough review on the benchmarks of MAPF problems [92]. Some types of maps commonly used in solving MAPF problems discussed in the paper are listed below.

- **Dragon Age Origins (DAO)** maps: Grids taken from the game Dragon Age Origin. The maps are publicly available in Sturtevant's repository [93]. Some most commonly used maps are den520d and brc202d, as shown in Figure 4.1a and 4.1b. The den520d map has large open spaces. In contrast, the brc202d map has fewer open spaces and has more bottlenecks. The configuration of brc202d map is similar to airside environments with connected taxiways and runways. Thus, in previous studies of ATO, the performances of MAPF algorithms on brc202d map can be viewed as an indicator in airport applications. Although the configuration of an aircraft stand is not very close to both the den520d and brc202d map, the performance of MAPF solvers on those maps are still valuable guides for applications in ground handling.
- **Open $N \times N$ grids**: $N \times N$ grids which common values of N are 8, 16, and 32. Those grids are suitable for experiments in which the ratios of agents to space are high.
- **$N \times N$ grids with random obstacles**: $N \times N$ grids, in which some grids are randomly selected to be impassible obstacles.
- **Warehouse grids**: Grids similar to real-world autonomous warehouses. The map has open spaces on the two sides and long and narrow corridors in between. Figure 4.1c illustrates an example of warehouse grids.



(a) The DAO den520d map [93]. (b) The DAO brc202d map [93]. (c) Example warehouse grids [92].

Figure 4.1: Different types of maps commonly used in MAPF.

Aircraft stands

Most aircraft ground handling operations take place at the aircraft stands. Thus, aircraft stands and the connected service road are the environments we would like to model in this study. A typical ramp layout of the aircraft Airbus 320-200 is shown in Figure 4.2. We choose A320 rather than B737 as an example here because the geometrical shapes of the two aircraft are similar, but Airbus provides more sophisticated layout maps [1]. As shown in Figure 4.2, an airport stand can be modeled as an environment with grids.

Considering the aircraft, the passenger boarding bridge, and the towing vehicle as immovable objects, there are about 20% obstacles in the environment, depending on how the grids are defined. The obstacles in the map are not distributed randomly but are clustered in the middle. Although the environment of aircraft stands does not directly fall into any above mentioned category, the DAO maps and grids with random obstacles have closer layouts to the environment we would like to model. Therefore, in the discussions of different MAPF algorithms in this literature study, we focus more on the experimental results in DAO maps (especially brc202d) and grids with obstacles. Open grid environments are also used for evaluations if the discussed algorithm did not have an experiment on grids with random obstacles.

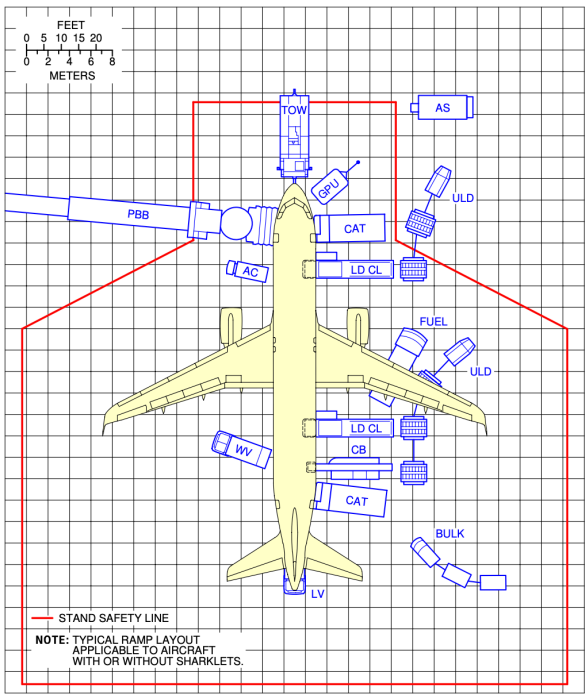


Figure 4.2: Typical ramp layout at gates of A320-200. Retrieved from [1].

4.2. Classical MAPF algorithms

In this section, existing algorithms that solve classical MAPF problems are discussed. Most of the algorithms assume the conditions described in subsection 4.1.3 hold. Some variants that relax these assumptions are introduced after presenting the original version of the algorithm.

4.2.1. A* based approaches

A* algorithm is a path finding and graph traversals algorithm. It is commonly used in a wide range of contexts. A* can generate optimal and complete path finding solutions. However, the standard version of A* can be inefficient while finding solutions for MAPF problems. The size of state-space and the branching factors grows exponentially with the number of agents. Extensions of A* are proposed in the literature to improve the efficiency of A*. In this section, several A* based approaches are discussed.

OD

Operation Decomposition (OD) [91] is proposed by Standley to reduce the exponential branching factor of the agents. OD divides each time step into n sub-steps, where n equals the number of agents. Agents are considered one at a time using a fixed random order. When a regular A* node is expanded, OD only considers the moves of the first agent and introduces an intermediate node. OD expands the intermediate node considering the possible movements of the next agent and generates further intermediate nodes. A regular A* node is generated only when the last agent is considered. The number of branching factors to be considered is reduced to the number of branching factors of a single agent. Furthermore, intermediate nodes that are not included in the solution are not expanded. Thus, the number of surplus nodes can be significantly reduced. Due to the admissibility and completeness of the A* algorithm, Standley claims that A* with OD is also admissible and complete. However, the algorithm does not guarantee optimality because of the decomposition of time steps. The decomposition could result in a situation that movements of agents ranked higher in the list block the possible movements of agents ranked lower in the list.

ID

Although OD significantly reduced the search space by avoiding exponential branching factors, the algorithm is still exponential in the number of agents. Standley developed another extension of A*, Independence Detection (ID) to mitigate this effect. The idea is to split the agents into disjoint sets, as such the optimal paths computed for each set do not interfere with paths for the other sets. In the beginning, every agent is in a set of its own. An optimal algorithm is executed in each set. After planning individual optimal paths for agents, conflicts are detected. If a conflict is found, ID tries to find other optimal paths for either agent. If the process of finding new paths fails, sets of agents involved in the conflict are merged into one set. The process is repeated until no conflicts are detected in the paths of all sets. The runtime of ID is dominated by the largest set of agents in the MAPF problem. Standley claims that ID is complete while coupling with a complete search algorithm, such as standard A* or A*+OD. Experiments are performed to compare HCA*, A*+OD, A*+ID, and A*+OD+ID. A* with both OD and ID was shown to be most efficient in terms of computational time.

EPEA*

Felner et al. developed Enhanced Partial Expansion A* (EPEA*) [29] to avoid the generation of surplus nodes in regular A* algorithm. EPEA* shares the similar idea to reduce surplus nodes like Standley's OD does, but EPEA* avoids all surplus nodes while OD does not. Only the nodes with an f value smaller or equal to that of the parent nodes are generated. EPEA* uses a mechanism called operator selection function (OSF) to identify operators that will produce the desired f-value of a given state. Felner et al. claim that EPEA* has shorter computational time than A*+OD by a factor up to full order of magnitude in their experiments.

ICTS

Increasing Cost Tree Search (ICTS) [84] is a two-level search algorithm developed by Sharon et al.. ICTS works as follows. At the high level, the algorithm searches the increasing cost tree (ICT) in a breadth-first manner. Each node in the ICT consists of a k-sized vector representing the individual cost of k agents. In the root node, the vector contains the agents' individual optimal paths without considering other agents. Child nodes in ICT are generated by increasing one of the agents' costs by 1. The low level of ICTS checks if the node is a goal node. A goal node is a node with the cost vector $C = [c_1, ..., c_k]$ such that a set of non-conflicting paths exists and the cost of the individual path for any agent a_i is exactly c_i. To check such conditions, ICTS stores all

single-agent paths of cost C_i in a special compact data structure called multi-value decision diagram. A set of non-conflicting paths for agents are generated by the cross product of the multi-value decision diagrams. The total cost of the node is the sum of the agents' costs. An example ICT with three agents is shown in Figure 4.3.

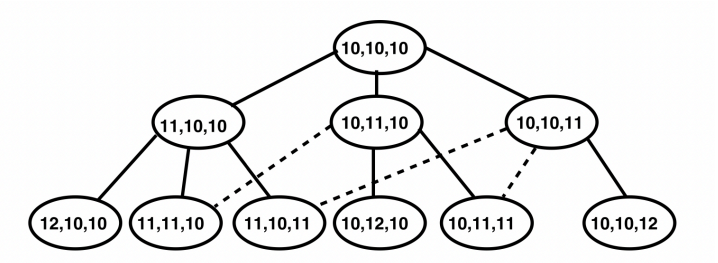


Figure 4.3: Example ICT for three agents. Retrieved from [84].

Experiments comparing ICTS and other A*-based approaches are performed by Sharon et al. [84]. On a 32×32 grid with 40 agents, ICTS+ID outperforms A*+OD+ID in terms of success rate. Also, on the DAO brc202d map, an improved version of ICTS with some pruning technique have higher success rates than A*+OD+ID. In the later work of Sharon et al. [85], EPEA* and ICTS are compared against Conflict Based Search (CBS). ICTS outperforms EPEA* on the DAO brc202d map, while the two algorithms have similar success rates on an 8×8 grid. Moreover, EPEA*, ICTS, and CBS have strength in different cases, which will be discussed in subsection 4.2.2.

4.2.2. CBS family

Conflict Based Search (CBS) [85] is a state-of-the-art MAPF solver proposed by Sharon et al. CBS can be viewed as both coupled and decoupled approaches. Numerous extensions of CBS are developed to deal with real-world scenarios. In this section, we discuss CBS and its extensions in detail.

CBS

CBS works on two levels. Initially, it computes the optimal paths for individual agents. Conflicts detected in the set of paths are resolved by applying constraints. Then, agents plan their new paths considering the applied constraints. The process repeats until an optimal and conflict-free set of paths is found. The process is described in detail below.

High-level In the high level, CBS searches a constraint tree (CT). CT is a binary tree. Each node in CT contains:

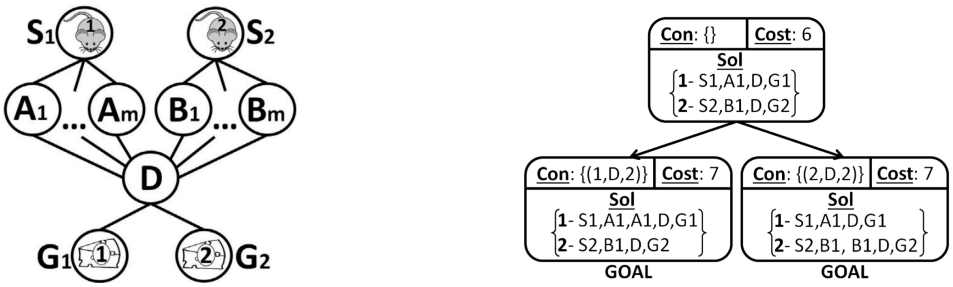
1. A set of constraints (*N.constraints*): The root node of CT contains an empty set of constraints. The child nodes inherit the constraints of the parent node and add one constraint for one agent. A constraint for agent a_i is denoted by a tuple (a_i, v, t), where agent a_i is prohibited from vertex v at time t.

2. A solution (*N.solution*): A solution is a set of paths for the agents. The path of agent a_i complies with the constraints of agent a_i. Solutions are searched in the low level. A node N in CT is a goal node when *N.solution* does not contain any conflict. A conflict is denoted by a tuple (a_i, a_j, v, t), where agent a_i and a_j occupy the same vertex v at the same time t.

3. The total cost (*N.cost*): the total cost of the current solution, which is the summation of all single-agent costs.

High-level of CBS checks if the current node is a goal node. If the current node is a goal node, CBS returns the solution. If it is not, CBS generates child nodes and introduces constraints to them. For example, if a conflict (a_i, a_j, v, t) is detected in N, constraint (a_i, v, t) is added to one of the child nodes and constraint (a_j, v, t) is added to the other child node. The high level of CBS performs the best-first search based on the cost of the nodes. Ties are broken by searching the node with fewer conflicts.

Low-level The low level search of CBS plans individual optimal paths for agents, taking into account the constraints introduced in the high level. Any single agent path finding algorithm can be implemented in the low-level search. In Sharon's work [85], A* is implemented for the low-level search algorithm.



(a) An example MAPF problem with 2 agents. The mice, 1 and 2, must reach the pieces of cheese 1 and 2, respectively.

(b) An example CBS constraint tree (CT) of the MAPF problem illustrated in Figure 4.4a.

Figure 4.4: An example MAPF problem and its CBS constraint tree. Retrieved from [85].

An example MAPF problem with 2 agents and the CBS constraint tree used to solve the problem is presented in Figure 4.4

CBS is proven to be optimal and complete. CBS was tested and compared to ICTS and EPEA* in open grids and DAO maps. In an experiment using the brc202d map, CBS solves all instances with problems up to 15 agents, while ICTS and EPEA* have success rates around 80%. The experiments with other DAO maps show that CBS is very effective while dealing with corridors and bottlenecks, but is more inefficient than ICTS or EPEA* in rather open spaces.

Sharon et al. pointed out a weakness of CBS. They found that when a group of agents is strongly coupled, that is, when there is a high rate of conflict within agents in the group, CBS needs to expand more nodes in the constraint tree in order to solve more conflicts before a solution can be found. To remedy this behavior, a variant of CBS, meta-agent conflict based search (MA-CBS) [85] was introduced.

MA-CBS

In MA-CBS, strongly-coupled agents are merged into a single meta-agent. The high level of MA-CBS is the same as that of CBS, but the meta-agent is treated as a single agent. The low-level of MA-CBS can be any MAPF solver. For example, A*, A*+OD, EPEA* are suitable for the low-level solver of MA-CBS.

A meta-agent consists of M agents. A single agent is just a meta-agent of size 1. In the constraint tree, after a conflict is found, MA-CBS has two options, to *Branch* or *Merge*. The option *Branch* is to generate two child nodes with new conflicts, just like regular CBS does. The option *Merge*, on the other hand, is to merge two conflicting agents into a single meta-agent. To determine whether to *Branch* or to *Merge* while encountering a conflict, a conflict-bound parameter B is used. If the number of conflicts found between agents is less than B, MA-CBS will do the *Branch* option; If the number of conflicts found is larger than B, agents are merged. When $B = \infty$, the constraint tree generates new nodes whenever a conflict occurs. Thus, MA-CBS(∞) is equivalent to basic CBS. In the other extreme case, if $B = 0$, the algorithm merges agents while encountering a conflict. MA-CBS(0) is identical to the independence detection (ID) mechanism introduced by Standley [91].

Sharon performed experimental studies of MA-CBS on DAO maps, using branching factor $B = 1$, 10, and 100. In general, MA-CBS outperforms algorithm A*, EPEA*, ICTS, and CBS. In the brc202d map, the map resembles an airport environment, MA-CBS(100) performs slightly better than CBS, while CBS works slightly better than MA-CBS(10) in terms of success rate. All of the three CBS algorithms outperform EPEA* and ICTS. In other DAO maps that have more open spaces, ICTS performs fairly well, while it is not the case for EPEA*. The original CBS does not work well in these kinds of open spaces. Still, in both den520d and ost003d maps, there is one MA-CBS version that outperforms all the other algorithms.

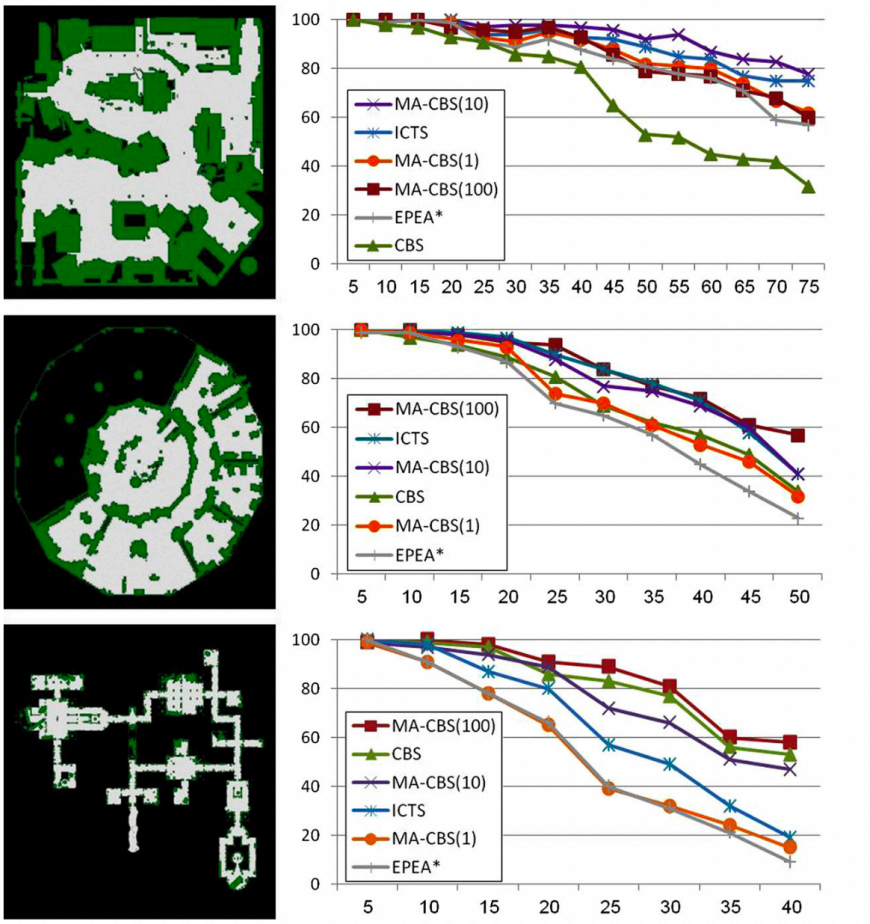


Figure 4.5: Success rate of the MA-CBS on top of ID with EPEA* as the low-level solver for different DAO maps den520d (top), ost003d (middle), brc202d (bottom). Retrieved from [85].

From the experiments of Sharon et al. [84][85], whether in open-spaces or environments with narrow corridors and bottlenecks, there is at least one variant of CBS surpassing other algorithms. Thus, CBS and its variants are considered more adaptable to different environments. Being the state-of-the-art MAPF algorithm, there are lots of developed CBS variants, and the CBS family is more discussed in literature. Therefore, CBS is considered superior to A* based approaches. CBS variants and their applications will be addressed more in the rest of this literature study.

ICBS

Besides MA-CBS [85] that Sharon et al. proposed to reduce the runtime of CBS, Boyarski et al. [10] [9] further proposed a variant of CBS by introducing three improvements. The variant is called Improved CBS (ICBS). The three improvements are:

- **Merge and restart (MR)**: Once a merge decision is made for a group of agents, the search restarts from a new root node. The agents are merged into a meta agent at the beginning of the search. MR helps to save computational time.
- **Prioritizing conflicts (PC)**: Conflicts are classified into thee types, namely *cardinal*, *semi-cardinal* and *non-cardinal*. Conflicts will be solved hierarchically based on their class.
- **Bypassing conflicts (BP)**: When a *semi-cardinal* or *non-cardinal* conflict is chosen to be resolve, the split action is not immediately performed. The path of one agent is modified if an alternative path with the same cost exists. BP reduces the size of CT and saves a considerable amount of search.

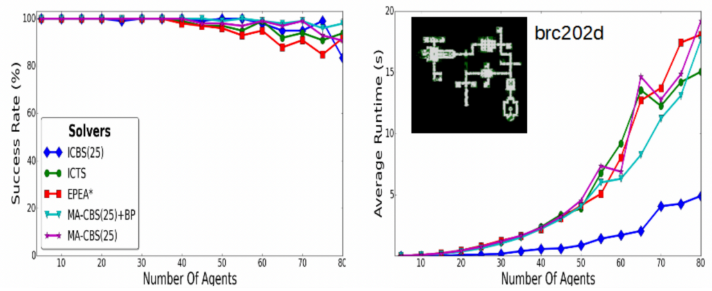


Figure 4.6: The success rate (left) and runtime performance (right) of several optimal algorithms on the DAO brc202d map. Retrieved from [10].

Boyarski et al. tested ICBS on the DAO brc202d map with several optimal algorithms. The results are shown in Figure 4.6. In terms of success rate, the performances of ICBS and other algorithms are quite close. The benefits of the newly introduced approaches are distinct in terms of runtime. ICBS outperforms other algorithms by almost a factor of 3 in the worse case with 80 agents. The quick runtime of ICBS makes it suitable for online use.

CBS is an algorithm solving MAPF problems optimally. Yu and LaValle [104] have proofed that solving MAPF optimally is NP-complete. Like all optimal MAPF solvers, CBS suffers from scalability. Barer et al. [6] presented several unbounded- and bounded-suboptimal CBS-based algorithms with relaxations on high and/or low-level searches. The idea is to trade the solution quality for better runtime.

GCBS

Barer et al. first proposed a unbounded-suboptimal variant of CBS, Greedy-CBS (GCBS)[6]. In basic CBS, both high level and low level use optimal best-first search to guarantee optimality. The high level searches for the node with the lowest cost, and the low level searches for optimal paths that satisfy the constraints. This mechanism may result in abandoning nodes with costs very close to optimal. GCBS prioritizes the CT nodes that seem closer to a goal node. Barer et al. developed a number of conflict heuristics (h_c), which is described in their study [6]. In GCBS, both high-level and low-level searches can be relaxed. The high level of GCBS favors the node with the lowest h_c. The low-level search uses an adjusted A*, applying conflict heuristics to find single agent's paths. In terms of success rate, BCBS-H(BCBS with a relaxed high-level search), BCBS-L, and BCBS-LH all outperform CBS in an experiment using the DAO brc202d map. Although GCBS provides complete solutions under certain conditions, its solutions are unbounded-suboptimal. This means that one cannot guarantee the solution cost to be within a finite factor of the optimal solution. Still, if solution qualities are not concerned, GCBS is able to provide fast and complete solutions.

BCBS

To obtain bounded-suboptimal solutions, Barer et al. proposed Bounded-CBS (BCBS) [6]. In the low level, they utilized focal search. Focal search maintains two list of nodes, an *OPEN* list and a *FOCAL* list. The *OPEN* list is similar to the *OPEN* list in A*, while *FOCAL* contains a subset nodes from *OPEN*. Consider a function f_1 that defines the nodes in *FOCAL*, and $f_{1,min}$ be the minimal f_1 value in OPEN. *FOCAL* contains all nodes n in *OPEN* such that $f_1(n) \leq \omega \cdot f_{1,min}$, where ω is a suboptimality factor. Another function f_2 is used to choose which node to expand in *FOCAL*. A focal search can be denote as $focal(f_1, f_2)$. Under the condition that f_1 is admissible, the focal search returns solution at most w times the optimal solution.

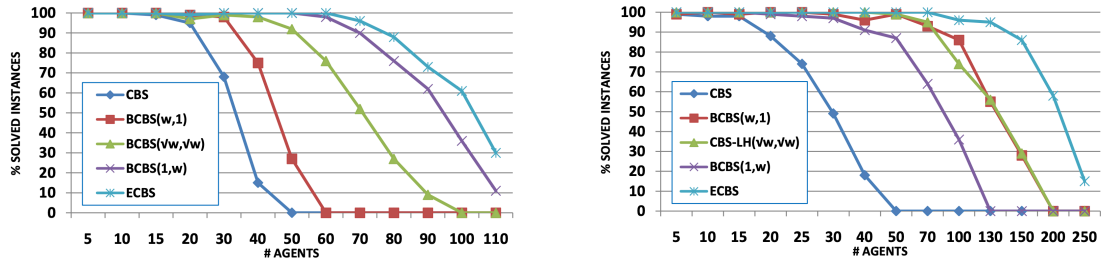
In the high level, BCBS applies $focal(g, h_c)$ to search the constraint tree, where $g(n)$ is the cost of the CT node n, and h_c is the conflict heuristics mentioned in the previous paragraph. The low level applies $focal(f, h_c)$, where f is the f-value in A*. Barer used the term BCBS(ω_H, ω_L) to describe BCSB, using ω_H as the suboptimality factor of the high level and ω_L as that of low level. In addition, GCBS is a special case, in which $\omega = \infty$ for the high and/or low-level search. The suboptimality of BCBS can be denoted as a factor $\omega = \omega_H \times \omega_L$.

ECBS

By using the suboptimality factor ω_H and ω_L, BCBS guarantee bounded suboptimal solutions. However, it is a challenge to properly determine the factor ω_H and ω_L, as stated by Barer et al. [6]. They proposed Enhanced CBS (ECBS) [6] to address the issue. Similar to BCBS, ECBS utilized focal searches in the high and low level. ECBS uses an *OPEN* and *FOCAL* list in the high level, and an $OPEN_i$ and $FOCAL_i$ list in the low level, in which i represents the agent a_i. In ECBS, for every generated CT node n, two values from the low level are returned to the high level: (1) *n.cost* and (2) *LB(n)*. *n.cost* is the cost of node n, while LB(n) is the minimal f value in $OPEN_i$, which is also the lower bound on the cost of the optimal consistent path of agent a_i. *FOCAL* in ECBS is defined as follows:

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

LB is the lower bound of the optimal solution of the entire problem ($C*$). Thus, all nodes in *FOCAL* have costs that are within ω times the optimal solution, and the cost of solution is guaranteed to be at most $\omega \cdot C*$.



(a) 32 × 32 - 20% obstacles. $\omega = 1.1$

(b) DAO brc202d map. $\omega = 1.01$

Figure 4.7: Success rate of bounded CBS versions. Retrieved from [6].

Experiments comparing CBS-based bounded suboptimal solvers were carried out by Barer et al.. The success rates of different solvers on a 32 × 32 grids map and the DAO brc202d map are shown in Figure 4.7. It is obvious that ECBS outperforms other solvers on both maps. ECBS has a success rate of 100% with up to 60 agents on a 32 × 32 grids map with 20% obstacles, and it solves problems with up to 70 agents in the DAO brc202d map. On the other hand, CBS has 100% success rates of up to 15 agents in both maps. The performances of BCBS on both maps are between CBS and ECBS. Barer et al. claim that the success of ECBS is due to its flexibility in the high-level search when the low-level return low-cost solutions.

Among the three variants of CBS proposed by Barer al el., GCBS has high success rates, but the cost of the solution is not guaranteed within a bounded factor from the cost of the optimal solution. The unbounded characteristic thus makes GCBS less favorable in real-world applications when the solution qualities are important. In such cases, BCBS and ECBS are better variants of CBS. Both of them have significantly higher success rates than that of CBS, with the cost of self-defined suboptimality. Further, ECBS performs better than BCBS in most cases. ECBS is thus a strong candidate for the application in aircraft ground handling.

ECBS+HWY

In some applications where instances are tightly coupled (i.e. there are lots of conflicts among instances), the runtime of ECBS grows severely. For example, in warehouse applications done by Kiva systems, ECBS has shown to scale poorly with many agents and narrow corridors [14]. To adapt MAPF into Kiva-like domains, Cohen et al. developed a bounded suboptimal MAPF solver, called ECBS+HWY [14][13]. The algorithm finds paths for agents that include edges from user-defined highways. In the low level search, ECBS(ω_1)+HWY(ω_2) inflates the non-highway edges by an inflation factor $\omega_2 > 1$. This encourages agents to follow the highways and avoid collisions. The algorithm is $\omega_1 \cdot \omega_2$ suboptimal. They demonstrated that ECBS+HWY can decrease the runtime and solution cost of ECBS in Kiva-like domains. In aircraft ground handling operations, Tabares et al. [3] proposed corridors in aircraft stands for the movements of GSE. The idea of encouraging agents to use the pre-defined edges is similar to the concept of highways. Therefore, implementing ECBS+HWY for path findings of GSEs could be interesting to be further explored.

4.2.3. Priority based approaches

Priority based solvers decouple MAPF problems into several individual agent path planning problems using pre-defined priorities of agents. Priority based solvers are usually really efficient. The greatest concern of priority based solvers is that it does not guarantee optimal and complete solutions. Also, solutions to the problems are quite sensitive to the way priorities are set. However, priority based approaches allow user-defined priorities of agents while solving MAPF problems. At aircraft stands, it is common that different ground handling vehicles have preferred priorities when conflicts occur. Tabares et al. provide a detailed priority list of various GSEs in their research [3]. For this reason, in real-world applications, it is worth discussing priority based approaches in more detail.

CA* and HCA*

Silver proposed some efficient algorithm that decoupled the problem into several individual path planning problems based on some pre-defined priority of agents. In Cooperative A* (CA*) [88], a priority order of agents are first generated. Single agent path plannings using A* are then performed in the defined order. After a path is made, the information of the occupied vertices (x, y) at some timestep t are stored in a three-dimensional reservation table. Agents with lower priority should avoid the path reserved while planning their paths. The single agent searches repeat until all agents find a path. Although CA* works efficiently, it is important to note that the algorithm is not complete. Agents with higher priorities can potentially block the necessary paths for agents with lower priorities. Also, the greedy behavior of prioritized agents prevents optimal solutions. To improve the heuristics used in CA*, Silver proposed another algorithm called Hierarchical Cooperative A* (HCA*) [88]. Basically, HCA* is the same as CA*, but it utilizes more sophisticated heuristics, Reverse Resumable A* search. The Reverse Resumable A* algorithm is used to calculate abstract distances to the goal node. It executes the A* algorithm in a reverse direction. The abstract distance represents the optimal distance from the current node to the goal node. Because of HCA*'s similarities with CA*, the algorithm may still lead to incomplete and suboptimal solutions.

WHCA*

In terms of optimality and completeness, the two algorithms mentioned previously is sensitive to the priority of the agents. Also, the previous algorithms have large, three-dimensional solution space, which is not suitable for real-time use. To tackle these issues, Silver developed Windowed Hierarchical Cooperative A* (WHCA*) [88]. He claimed that limiting the cooperative search to a fixed window w is a simple solution to the above mentioned issues. Every time agents plan their route, agents only consider the movements of other agents for the next w steps, while the abstract search is still executed to full depth. Then, at some regular intervals $k \leq w$ (e.g. when the agent is halfway to its current route), the window is shifted forwards, and agents plan for their new partial route. Silver suggested that in each time window w, different agent priorities can be used to improve the completeness of HCA*. WHCA* reduced the search space to a w-step time window, thus significantly reducing the computational time of HCA* and CA*. Also, WHCA* outperforms HCA* in terms of success rate. It is worth noting that WHCA* still uses a reservation table in the w-step window, thus feasible paths of agents could still be blocked by agents with higher priority. The size of the window w and the replan intervals k have a substantial effect on the performance of the algorithm.

CO-WHCA* and CO-WHCA*P

To further consider the time and locations of possible conflicts, two extensions of WHCA* were developed by Bnaya and Felner [8]. They developed Conflict Oriented WHCA* (CO-WHCA*). Unlike WHCA*, CO-WHCA* only uses a reservation table when a conflict is detected. They assume a fixed-size window w that the reservation window should be evenly positioned around the conflict. An arbitrary agent involved in the conflict is assigned to be the conflict master, which has the right to reserve a path in the reservation table. With the help of the reservation window w, agents can be at least $w/2$ step away from the conflict, thus having more time steps and possible paths to resolve the conflict. Further, Bnaya and Felner developed Conflict Oriented WHCA* with prioritization (CO-WHCA*P) by implementing a winner-determination conflict master selection method into CO-WHCA*. The prioritization is based on the likelihood of the agent to find efficient detours. Experimental results in the paper showed that both CO-WHCA* and CO-WHCA*P outperform WHCA* in terms of solution cost and success rate. Also, CO-WHCA*P achieves better solution quality than CO-WHCA*. However, it should be carefully used since it consumes a much longer solution time than CO-WHCA*.

CBSw/P

Conflict-Based Search with Priorities (CBSw/P) is an adaptation of CBS proposed by Ma et al. [57]. CBSw/P performs a best first search in the high-level search and builds a constraint tree (CT) as CBS does. In addition to the constraints, solution, and cost of a node (*N. constraints*, *N.solution*, and *N.cost*) that are recorded in every node in the CT of CBS, CBSw/P also contains an ordering of priority in every node. At the beginning of the search, the root node contains an empty set of priority orderings. While expanding a node in CT, CBS generates two child nodes. In one child node, one of the agents involved in the conflict has priority over the other agent. In another child node, the priority is set the other way around. The child nodes in CT inherit the priorities of their parent node and add a new priority order while being expanded. The agent with lower priority is prohibited to use the path at a certain time point, which is reserved by the agent with higher priority. The low-level search of CBSw/P is similar to the low-level search of CBS, which follows the constraints and priority orderings while making a new set of paths for agents. The high and low-level search continues just like CBS until a conflict-free solution is found.

PBS

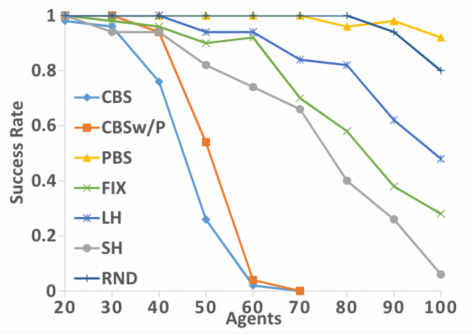
Ma et al. also developed Priority Based Search (PBS) [57]. PBS is a two-level MAPF solver using prioritized planning. Unlike CBSw/P, PBS performs a depth-first search in the high-level search. It constructs a priority tree (PT) that stores the priority order, the solution, and the cost of the solution in each node. It is important to note that PBS does not store constraints in the PT nodes, like the way CBSw/P does. PBS works as follows. In the high-level search, the root node of PT can take any initial priority order as an input (which can be empty). The low-level search is then run for all individual agents, where agents with lower priorities take the path of agents with higher priorities into account. While encountering a collision, PBS greedily chooses which agent should be given the priority. It expands the PT node with conflicts of agent i and j into two child nodes containing newly defined priorities. In one node agent i has priority over agent j, and in the other node, the priority order is the other way around. If no solution can be found in the current branch, it can efficiently backtrack the PT and explore other branches. The procedure continues until a collision-free solution is found.

Both CBSw/P and PBS are unbounded suboptimal MAPF solvers. The suboptimality of PBS, CBSw/P, CBS, and other algorithms are investigated in the experiments of Ma et al. It is shown that both CBSw/P and PBS often return near-optimal solutions. The suboptimality in two different maps are shown in Figure 4.8b and 4.8d. In a 20×20 grid with 10% obstacles, the success rate of PBS outperforms that of CBS and CBSw/P. PBS is able to solve all the instances with up to 70 agents, while CBS and CBSw/P can only solve the problem completely with up to 20 and 30 agents, respectively. In the DAO brc202d map, Both PBS and CBSw/P have a 100% success rate with up to 100 agents, while CBS can only solve the problem with about 20 agents. The success rates are shown in Figure 4.8a and 4.8c. Although PBS and CBSw/P do not guarantee optimal solutions, the two algorithms usually provide near-optimal solutions with higher success rates compared to other algorithms.

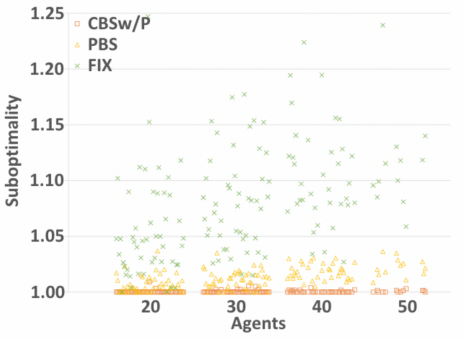
4.2.4. Other approaches

Besides the groups of MAPF algorithms discussed in previous sections, there are other kinds of MAPF solvers. One of them is the family of rule based approaches. Rule based approaches solve the MAPF problems by implementing specific rules. The rules are set based on the properties of the agents, the nature of the environment, and the interaction between the agents and the environment. Rule based solvers usually find solutions relatively fast, but the solutions are often far from optimal. Algorithms like push and swap [55], push and rotate [17], and tree-based agent swapping strategy [49] that include the "swap" operations are referred to as rule based approaches in Polydrou's literature study of his master thesis [69]. The algorithms that included "swap" operations assume that agents are capable of moving in all directions. Polydrou noted that these movements are impractical for aircraft agents as the space for turning may be limited in some places in the airport. Likewise, in our study, aircraft stands are environments that are rather dense with agents or obstacles. Some rotation movements may be impossible to be performed. To sum up, the family of rule based approaches is not suitable to be implemented in our study. A detailed discussion of rule-based approaches is thus omitted in this literature study.

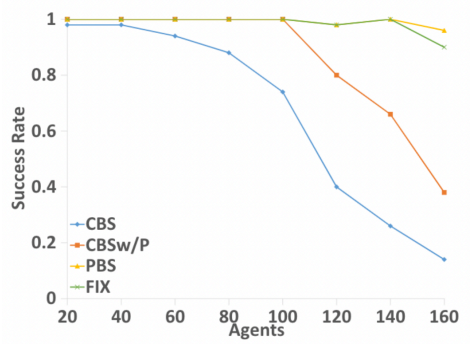
The other class of MAPF algorithms is reduction based approaches. These kinds of algorithms translate MAPF problems into standard known problems. For example, Boolean satisfiability (SAT) [94], Constraint



(a) Success rates for 20 × 20 grids - 10%obstacles.

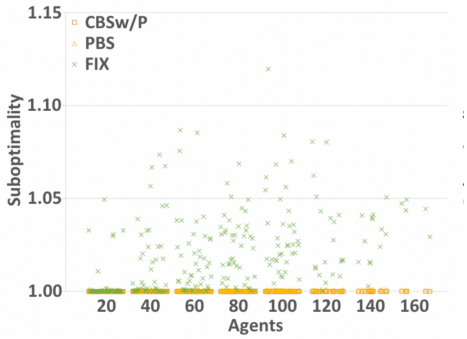


(b) Suboptimality ratios for 20 × 20 grids - 10%obstacles.



(a) Success rates for brc202d.

(c) Success rates for DAO brc202d map.



(d) Suboptimality ratios for DAO brc202d map.

Figure 4.8: Success rates and suboptimality of CBSw/P, PBS and other algorithms. Retrieved from [57].

Satisfaction Problem (CSP) [73][72], Answer Set Programming (ASP) [27], and integer linear programming (ILP) [103] can be referred to such [85][92]. Polydrou investigated the above mentioned algorithms in his literature study [69]. He mentioned that the runtimes of these algorithms are rather long and thus not suitable for online applications. Consequently, reduction based approaches will not be further considered in our study.

4.3. MAPF for real world conditions

Most of the algorithms discussed in the previous section make the assumptions described in subsection 4.1.3. However, in real-world applications, the assumptions do not always hold. Directly applying these assumptions could cause danger or inefficiency in executions. The negative effects are unacceptable in most real-world applications, including aircraft ground handling operations. Therefore, these factors should be considered during or after planning. In this section, exist MAPF variants that account for these conditions are discussed.

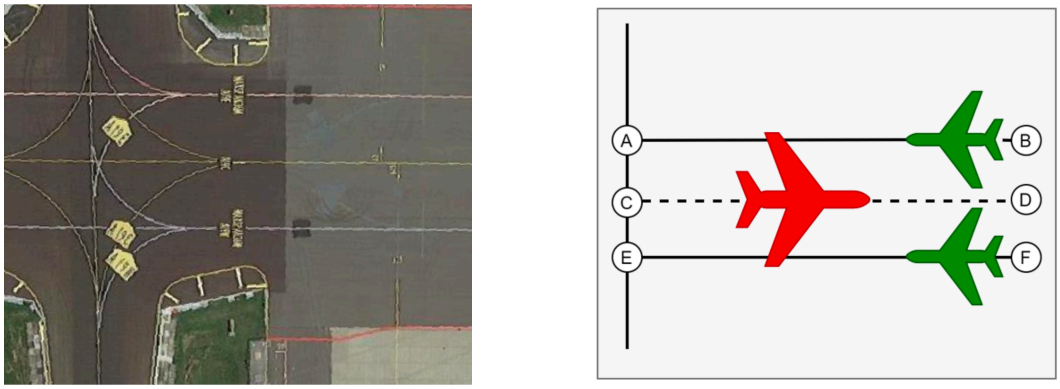
4.3.1. Continuous Time Steps

Classical MAPF approaches assume discretized time steps. This assumption is not realistic in real-world applications. Real-world agents suffer from imperfect plan execution and have uncertain action duration. Thus, setting constraints at discrete timesteps does not necessarily avoid collisions.

Safe Interval Path Planning (SIPP) [68] is a common approach to deal with MAPF problems with continuous timesteps by introducing safe intervals and conflict intervals. More details about SIPP will be discussed in subsection 4.4.1. A MAPF solver using continuous time is continuous-time conflict-based search (CCBS) proposed by Andreychuk et al. [5]. Based on CBS and an adjusted version of SIPP, CCBS is an optimal and complete MAPF solver that does not rely on the assumption of discrete timesteps. CCBS is discussed in subsection 4.4.2.

4.3.2. Geometric Shaped Agents

Classical MAPF algorithms assume that each agent has the same shape and size, and it only occupies a single location (e.g. a single cell in a grid) at a time step. This assumption is not applicable in real-world applications as real-world agents usually have geometric shapes. In this paragraph, geometric-shaped agents, which are "large" agents that can occupy multiple points at the same time, are considered in the MAPF problems.



(a) Actual situation of wingspan restricted taxiways at Schiphol.

(b) Simplified schematic situation.

Figure 4.9: Situation where classic MAPF with point agents could result in a collision between the red aircraft and the green aircraft. Retrieved from [48].

At aircraft stand areas, GSE agents can occupy multiple vertices and edges at one time point. In classical MAPF algorithms, this may happen when each agent only records a single vertex or edge in its state at a certain time step, but occupies neighboring vertices or edges at the same time point. Kamphof [48] illustrates a scenario in Schiphol Airport of this circumstance in the literature study of his master thesis. The aerial picture of the 3 taxiways at the west side of the G-apron is shown in Figure 4.9a. The 3 taxiways can be represented in the schematic in Figure 4.9b. Taxiway A-B and E-F can be used simultaneously, while taxiway C-D can only be used when neither taxiway A-B nor E-F is in use and vice versa. Classical MAPF would only prevent simultaneous agent movements on the same edge, which is not sufficient in this circumstance.

Generalized conflicts

Hönig [43] identified this situation and formalized three new types of conflicts, other than classical MAPF conflicts discussed in subsection 4.1.2. The new types of conflicts retrieved from [43], together with the examples retrieved from [48] are listed below.

- **vertex-to-vertex conflict**: Conflicts caused by two robots occupying two vertices that are close to each other at the same time. One occurs for example if the red agent in Figure 4.9b is occupying vertex C while the other agent is on vertex A.
- **edge-to-edge conflict**: Conflicts caused by two robots traversing two edges that are close to each other. One occurs for example if the red agent in Figure 4.9b is traversing edge C-D while the green agent is traversing edge A-B.
- **edge-to-vertex conflict**:Conflicts caused by one robot traversing an edge colliding with the other stationary robot. One occurs for example if the red agent in Figure 4.9b is traversing edge C-D while the green agent is staying at vertex B.

After introducing the new types of conflicts, Honig et al. solved the problem by adapting ECBS [43].

There are still several ways to account for the drawbacks caused by geometric-shaped agents. One of the possible approaches is to lower the resolution of discretization of the environment. However, Li et al. [54] pointed out that this approach can degrade the performance of the MAPF solvers and thus their practical applicability. Therefore, adapted versions of CBS to deal with large agents are proposed by Li et al.

LA-MAPF

Li et al. formalized a new problem of multi-agent path finding for large agents (LA-MAPF) [54] which takes into consideration the shape of agents. In LA-MAPF each agent has a fixed geometric shape around a reference point. Depending on its geometry shape, every agent can occupy a set of points in a graph at any time point. The agents cannot undergo transformations like rotations. Safety distances in LA-MAPF can also be included in the agents' geometry shapes. CBS can be directly adapted to solve LA-MAPF problems by considering the generalized conflicts introduced by Hönig [43], as discussed above. However, as noted by Li et al. [54], the adapted version of CBS is very inefficient for LA-MAPF, in which a great number of nodes would be generated to account for closely related conflicts.

MC-CBS

To reduce the number of nodes generated by CBS while solving LA-MAPF, Li et al. proposed a generalized version of CBS, Multi-constraint CBS (MC-CBS) [54]. MC-CBS adds multiple constraints instead of a single constraint while generating a child node in CBS. In such way, multiple related constraints can be added to the node in a single expansion, resulting in smaller CTs.

4.3.3. Non-uniform Edge Traversal Times

In classical MAPF, every agent is assumed to perform one action in exactly one timestep. Also, the agent is only able to travel to the 4-connected grids of its current location in one action. As a result, the traversal time of all edges is the same. In the study of Stern et al. [92], the authors have pointed out actions with different duration can be modeled by weighted graphs, in which the weight of each edge represents the traverse duration for an agent. By applying different weights to edges, the previously discussed algorithms can be used to solve MAPF problems with non-uniform traversal times.

Several types of weighted graphs have been used in MAPF researches. For example, MAPF in 2^k-connect grids. These kinds of maps are simple two-dimensional open-grids. The possible moves of an agent in a cell are moving to its 2^k connected-grids, where k is a parameter. An illustration of the movements to the 2^k-connected grids is shown in Figure 4.10. The traversal times (costs) are based on the Euclidean distances. When k is larger than 2, different edge traversal durations are introduced.

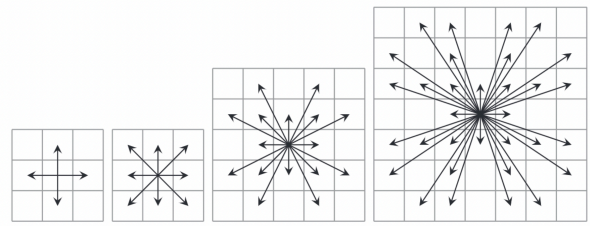


Figure 4.10: Movement to the 2^k-connected grids. Retrieved from [92].

Kamphof also discussed the uniform and non-uniform edge traversal times in the literature study of his master thesis [48]. He mentioned that while representing real-world airports, edge lengths are not uniform in the graph. Weighted graphs should be utilized, and thus agents can have different action duration. The approach can work well on graphs representing taxiways and runways in airport environments. In automated aircraft ground handling operation problems, however, the aircraft stand environments are better represented by grid-based graphs. Grid-based graphs usually have uniform edge lengths. Still, the possible motion of agents can be movements to the above mentioned 2^k-connected grids or other variants. Thus, the approaches dealing with non-uniform traversal times are still relevant to this study.

4.3.4. Kinematic Constraints and plan executions

The other assumption that classical MAPF problems often assume is that agents are not limited in their kinematic abilities. They presume that agents can accelerate, decelerate, and change directions in no time. They also assume perfect plan execution capabilities. However, these kinds of assumptions are unrealistic and cannot be neglected by real-world agents. For example, ground handling vehicles cannot accelerate and decelerate instantaneously as they have performance limitations. In airport environments, different speed

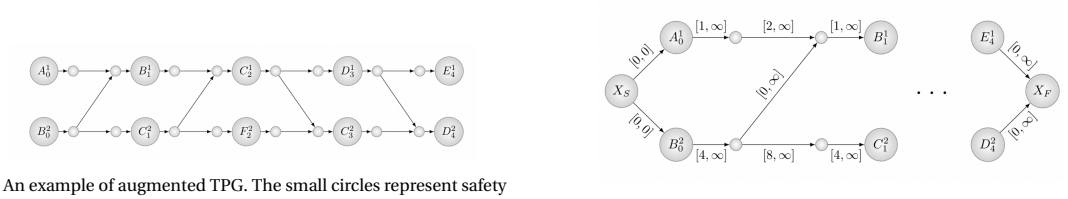
limits are imposed in different areas. Further, unforeseen disruptions, such as weather or machinery malfunctioning can also occur, affecting the execution of the plan.

Kinematic constraints therefore need to be taken into account for real-world agents to execute the MAPF plan. In Hönig et al.'s research [41], kinematic constraints are considered over agents' move actions. They proposed MAPF-POST to post-process solution from a complete MAPF solver using a simple temporal network. MAPF-POST converts a complete MAPF solution to a temporal plan graph (TPG). As indicated by the MAPF plan, TPG enforces two types of temporal precedence as follows, retrieved from [41].

- Type 1: For each agent, precedences enforce that it enters locations in the order given by its path in the MAPF plan.
- Type 2: For each pair of agents and each location that they both enter, precedences enforce the order in which the two agents enter the location in the MAPF plan.

As actual robots are not point agents and suffer from imperfect plan-execution capabilities, it is important to introduce safety distances in the plan. Thus, safety markers are added to guarantee safety distance between agents to TPG as additional vertices, resulting in augmented TPG. An example of augmented TPG is shown in Figure 4.11a.

The augmented TPG can then be transformed into a simple temporal network (STN). Vertices in STN represent events, while edges in STN represent constraints. The constraints are denoted by STN bounds [*LB*, *UB*] indicating that one event must be scheduled between *LB* and *UB* time units before another event. The STN bounds can be set based on velocity limits and non-uniform edge lengths. For Type 1 edges, the lower STN bounds (*LB*) correspond to the minimum time needed for agents to travel from one vertex to another vertex, which is related to the maximum velocities of agents. The upper STN bound (*LB*) is set to infinity unless limitations of minimum velocity are imposed. An example of STN is shown in Figure 4.11b. Following the constraints in STN, a plan execution schedule can be optimized by minimizing the *makespan* or *sum of costs*. The problem can be solved in polynomial time with both graph-based optimization and linear programming approaches. Unlike solutions from MAPF solvers, the plan execution schedule takes the maximum velocities of agents and safety buffers into account.



(a) An example of augmented TPG. The small circles represent safety markers.

(b) An example of STN.

Figure 4.11: Examples of augmented TPG and the corresponding STN [41].

Hönig et al. performed experiments of MAPF-POST using both simulations and actual robots. MAPF-POST was shown to successfully post-process the solution from ECBS+HWY and generate plan execution schedules. A major downside of MAPF-POST is that it post-process existing MAPF solutions, rather than directly taking kinematic constraints into account in the planning phase. As Ma et al. [58] noted, the resulting path might not be effective since the transformation is neglected during planning. Another limitation of MAPF-POST mentioned by Stern [92] is that it does not guarantee solution qualities. Optimal solutions from other MAPF solvers might not be optimal after being processed by MAPF-POST. Furthermore, the original MAPF-POST does not include some kinematic constraints like the acceleration limits of robots, which can be problematic while implementing MAPF-POST on heavy robots like commercial aircraft. The usage of lighter vehicles or vehicles with lower velocities, such as some ground handling vehicles, may not cause severe problems. Still, MAPF-POST is a powerful algorithm to deal with kinematic constraints and imperfect plan execution abilities of agents. The application of MAPF-POST in aircraft ground handling operations is possible under thorough assumptions.

4.4. Multi-Agent Motion Planning

In this section, MAPF approaches from the area of multi-robot path planning and motion planning will be discussed. Cohen et al. used the term *Multi-Agent Motion Planning (MAMP)* to refer to the task for finding kinodynamically feasible plans for agents [13], which is also used in this literature study. Multi-Agent Motion Planning (MAMP) is a generalization of the MAPF problem. MAMP obviates the assumptions that hold in classical MAPF problems, as the ones shown in subsection 4.1.3. MAMP aims at finding conflict-free kinematically feasible plans for agents from start to goal states within a continuous state space.

As discussed in section 4.3, some assumptions of MAPF listed in subsection 4.1.3 are not realistic for real-world problems like aircraft ground handling. The generalization of MAPF problems, MAMPs, are more appropriate means to model the movement of ground handling vehicles. In this section, we will first introduce a single-agent path finding algorithm, SIPP [68]. SIPP is the basis of different types of MAMP approaches. After that, other multi-agent solvers utilizing SIPP will be discussed.

4.4.1. Safe Interval Path Planning (SIPP)

Before introducing different MAMP solvers, a single agent planner, Safe Interval Path Planning (SIPP) will be introduced. Then, variants of SIPP, including SIPP with reservation table (SIPPwRT) and Soft Conflict SIPP (SCIPP), that can be applied to various MAMP solvers, will be presented.

SIPP

Phillips and Likhachev proposed Safe Interval Path Planning (SIPP) [68], a single agent path finding algorithm that considers other agents as dynamic obstacles. Applying SIPP to multi-agent problems, SIPP takes a predefined priority order and views the path reserved by agents with higher priorities as known obstacles. The main difference between SIPP and the priority based approaches like HCA* discussed previously is the way that the states are defined. Instead, SIPP searches in a state space consisting of pairs of agent configurations and intervals rather than configurations and time steps. A configuration is a set of non-time variables, describing an agent's physical position, heading, velocity and/or other related information.

To limit the size of the search space, safe intervals and conflict intervals are introduced. A safe interval is defined as a continuous period for a configuration, during which there is no collision. One timestep prior and one timestep after the safe interval, the configuration is in collision with dynamic obstacles. A collision interval is the opposite of a safe interval. It is a continuous period for a configuration, in which every timestep in the interval is in collision with dynamic obstacles. By using intervals, SIPP is faster than other planners that use a time dimension since the number of timesteps is usually quite large [68]. SIPP is suitable for applications with continuous time and non-uniform motion duration.

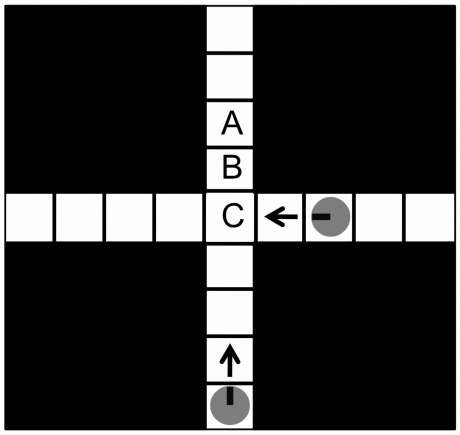
Let s denotes the states of SIPP as a pair of configuration and interval. $g(s)$ and $h(s)$ represents the g-score and the heuristic score of s, respectively. The cost of transition from state s to the new state s' is denoted as $c(s,s')$. First, SIPP is initialized. Taking the trajectories of dynamic obstacles (i.e. path reserved by agents with higher priorities) into accounts, timelines with safe and collision intervals of every spacial configuration are created. After the initialization, a modified A* search is run. The modified A* search is like usual A* except for the way it gets the successors of a state and updates the time intervals for states. A function $M(s)$ is designed to return the possible motions from state s. The resulting configurations of the motion are checked for their safe intervals. For each of the safe intervals in the new configuration, a successor with the earliest possible arrival time without collision is generated. When a feasible motion is found to a new state s', this state is added to the successors. Like in A*, successors with lowest *f-values* are considered to be expanded. Figure 4.12 shows an example of SIPP successors generation. The example is described in detail below.

Instead of using timesteps, SIPP utilizes safe intervals to represent contiguous periods. The idea significantly reduced the computation time. Thus, SIPP is suitable for online real-world applications. Besides, the way that SIPP defines successor states makes it possible to handle continuous time, large agents, agent kinematics, and non-uniform traversal time. The common assumptions of MAPF can be relaxed and the conditions concerned can be taken into account during planning. SIPP is considered a strong candidate to be implemented in the algorithms, solving paths for ground handling vehicles in this study. To better apply this potential candidate, in the following paragraphs, variants of SIPP are introduced. MAMP solvers applying

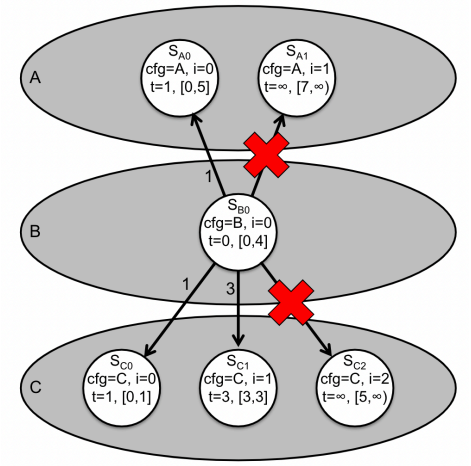
SIPP will also be discussed in the following sections.

Example SIPP problem

In this example, the situation shown in Figure 4.12a is considered. Initially, the agent is in configuration B. It is to plan its path using SIPP to avoid the two dynamic obstacles moving in the directions indicated by their arrows. The agent and the obstacles move one cell per timestep.
initialization: Three labeled configurations, (A,B,C) in Figure 4.12a will be examined. The safe intervals are determined for each configuration. The safe intervals for A are $[0,5]$ and $[7,\infty)$, as at $t = 6$ the cell is occupied by the obstacle from the bottom. Likewise, the safe intervals of configuration B is $[0,4]$ and the safe intervals of configuration C are $[0,1]$, $[3,3]$ and $[5,\infty)$.
successors generation: The start state S_{B0} is expanded. Potential successors are states with configurations of neighboring cells, which are possible configurations for the agent to move to, and their safe intervals. There are two possible successors in configuration A. The agent can move immediately to A and arrive at $t = 1$, which lies within the safety interval, and thus the state S_{A0} is feasible. If the agent would like to reach the other state that has a safe interval starting at $t = 7$, it has to wait in B until $t = 6$. However, this is not possible as the safe interval of configuration B ends already at $t = 4$. Therefore, the expansion of state S_{A1} is not feasible. Likewise, for configuration C, there are three possible successor states, but only two of them are feasible for the agent. The agent can move immediately to C and arrives at $t = 1$, or it can wait for 2 timesteps and arrives at configuration C at $t = 3$. However, it is not possible to arrive at configuration C within the time interval [5,inf) because the agent can only wait in configuration B until $t = 4$. In Figure 4.12b, arrows indicating the transition to the infeasible successors are marked by a red cross.
states expansion: The resulted valid successor states for S_{B0} are S_{A0}, S_{C0}, and S_{C1}. These states are added to the *OPEN* list. The state with the lowest *f-value* will be expanded first.



(a) An example environment with two dynamic obstacles. (A,B,C) are three configurations and the agent is located at B.



(b) An expansion of the start state S_{B0}. *cfg* denotes the configuraiton an i represents the index of safe intervals. t in the third line shows the earliest known time the agent can reach this state, together with the safe interval of this state.

Figure 4.12: An example successors generation of SIPP. Retrieved from [68].

SIPPwRT

Safe Interval Path Planning with reservation table (SIPPwRT) is a generalization of SIPP introduced by Ma et al. [58]. SIPPwRT works with continuous time and it is able to handle agents with volumes. SIPPwRT takes a pre-defined priority order. Using the priority order, agents with lower priorities have to avoid the path reserved by agents with higher priorities while planning their path. In SIPPwRT, safe intervals are defined per node in the same way as in SIPP. A reservation table is used to store the safe interval of every node, containing

all reserved intervals for that cell in increasing order of their lower bounds.

Nevertheless, agents larger than a single grid size can still collide with other agents that are about to arrive at or leave from a node. Time offsets are then introduced by Ma et al. to tighten the lower and upper bounds of safety intervals. The time offsets consider volumes, velocities, and configurations of agents. Safety margins between agents can be represented by expanded agent volumes. In SIPPwRT, each agent i is represented as a circular object with a safety radius R_i. The time offset ΔT expresses the minimum amount of time one agent needs to vacate a cell before another agent arrives at the same cell. Consider two agents, agent 1 has a safety radius R_1 and agent 2 has a safety radius R_2. The velocities of the two agents are v_1 and v_2, respectively. The minimum distance between two agents to avoid collisions can be represented as D, which should be larger than $R_1 + R_2$. The time offset ΔT that should be added to the safe intervals is distinguished in three cases as follows. Figure 4.13 illustrates two of the cases. The detailed derivation of the time offsets can be found in the original paper of Ma et al. [58].

- agent 1 and agent 2 move in the same direction:

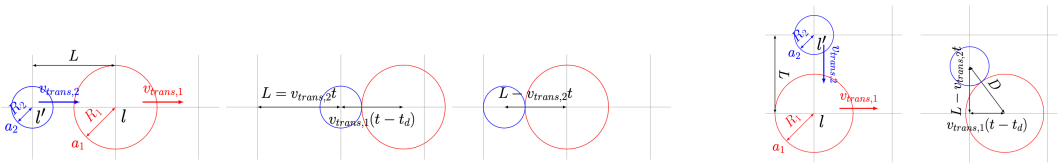
$$\Delta T = \frac{R_1 + R_2}{min(v_1, v_2)}$$

- agent 1 and agent 2 move in orthogonal direction:

$$\Delta T = \frac{\sqrt{v_1^2 + v_2^2}(R_1 + R_2)}{v_1 \times v_2}$$

- agent 1 and agent 2 move in the opposite directions. L is the distance between two cells:

$$\Delta T = \frac{L}{v_1} + \frac{L}{v_2}$$



(a) Two agents move in the same direction.

(b) Two agents move in orthogonal directions.

Figure 4.13: Illustrate different situations of movements of two agents with different sizes and velocities. Retrieved from [58].

Ma et al. proved that the paths returned by SIPPwRT are free of collisions. Furthermore, at the end of the study, Ma et al. proposed that in future works, some existing MAPF algorithms can be generalized by combining with SIPPwRT to compute continuous agent movements. Also, some kinematic constraints, such as acceleration and deceleration constraints, can be included in SIPPwRT. The two research directions help make MAPF solvers more realistic in terms of considering the kinematic properties of agents. They are both very interesting to be explored especially for applications in which agent kinematics is heavily concerned. For aircraft ground handling problems, sophisticated agent kinematics not only helps to solve the path planning problems with better solution qualities but also enhances the safety of ground handling operations.

SCIPP

Soft Conflict SIPP (SCIPP) [15] is a bounded-suboptimal single agent planner developed by Cohen et al. SCIPP is a generalized version of SIPP suitable for focal search and is applied in a MAMP solver, ECBS-CT [15].

SCIPP takes as input a start state and a goal state of the agent, a set of hard constraints for each cell c, and a reservation table specified in the high-level node. The hard constraints are stored in (c, t) pairs, in which c represents the cell and t specifies the time of the constraint. The reservation table specifies soft conflicts

in $(c, [lb, ub])$ pairs, representing the time interval that cell c is swept by agents with higher priorities. SCIPP uses a suboptimality factor ω for the focal search. In SCIPP, the node (state) with lowest *f-value* in the *FOCAL* list is expanded first. The *f-value* in the low-level search is determined by a conflict heuristic h_c. The conflict heuristic h_c represents the number of soft conflicts, which is the number of cells in which agents collide with other agents' plans when the agent moves from the current node to the next node. SCIPP returns a ω-suboptimal plan for an agent from its start state to the goal state without violating any hard constraint. It is an adapted version of SIPP suitable for the low-level search for ECBS-CT. ECBS-CT will be discussed in more detail later.

4.4.2. SIPP in multi-agent planners

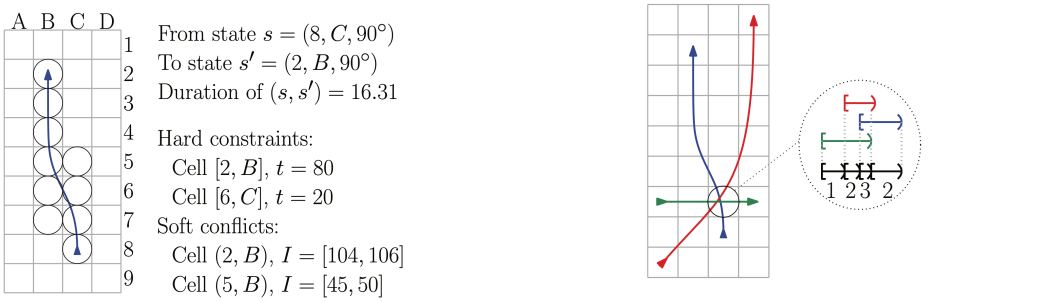
The algorithms that extend SIPP to solve MAMP problems are discussed in this section.

ECBS-CT

Enhanced conflict based search with continuous time (ECBS-CT) was developed by Cohen et al. [15]. The authors adapted ECBS (which is discussed earlier in subsection 4.2.2) to continuous time. They also developed and applied a novel bounded-suboptimal single agent path finding solver, SCIPP (discussed in subsection 4.4.1).

The authors formulated a MAMP problem and tried to solve it with their ECBS-CT. In the MAMP problem, vertices represent states and edges represent kinodynamically feasible motions between two states. A state reflects the configuration of an agent, including the coordinate of a reference point of the agent, the heading angle, velocity, acceleration, and other characteristics. The environment is represented by discretized cells. Agents with any geometric shape occupy a list of swept cells, in which each cell is associated with a time interval during which it is occupied. If two agents are planning to sweep the same cell or occupy the same cell at vertices at the same time, these two agents are in conflict. The target of the problem is to find a set of plans for agents without collisions.

To solve the above formulated MAMP problem, ECBS-CT was developed to return a solution that has a cost no more than ω times the optimal cost, where ω is a suboptimal factor. In the beginning, a high-level root is initialized, in which SCIPP is used as the low-level search for individual agents. The algorithm then performs a *FOCAL* search as ECBS does. In high-level nodes, conflicts are identified between agents in the form of $(c, [lb, ub])$, where c represents the cell and $[lb, ub]$ shows the time interval two agents are occupying the same cell c. To resolve the conflict, a constraint (c, τ) is applied, where $\tau \in [lb, ub]$ is a time point within the time interval. According to Cohen et al. [15], the reason that a time point, instead of a time interval, is utilized is to ensure the suboptimality of the algorithm. The constraints are used to expand the two successor nodes in the high-level search. In the low-level search, either constraint is taken into account while planning paths for individual agents by SCIPP. To manage the constraints, Cohen et al. used the interval map [16] as reservation tables. After performing the low-level search and updating the reservation table, *FOCAL* is maintained to include all relevant nodes in *OPEN*.



(a) Illustrates an example edge between two vertices.

(b) An example interval map showing three agents sweeping the circled cell during overlapping intervals.

Figure 4.14: Illustration of an example edge in MAMP and an example interval map. Retrieved from [15].

In the experiment of Cohen et al., they used two DAO maps (the arena and the Den420d map) and two sets of motion primitives (one set of only 5 motions and the other set of 13 motions) to evaluate the performance of ECBS-CT. A motion primitive models the kinodynamically feasible motions of the agent represented by a segment of the agent's path. The number of motion primitives influences the branching factor for the node expansion in the low-level search [48]. Overall, higher numbers of motion primitives lead to longer runtime. However, the success rate of solving MAMP problems with higher motion primitive can also be higher, due to more flexibility allowed. The results of the experiments demonstrated that ECBS-CT is a suitable approach for solving real-world MAMP problems.

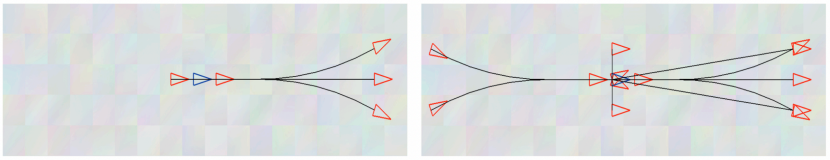


Figure 4.15: Illustrate 5 (left) and 18 (right) example motion primitives. Retrieved from [15].

The environment of aircraft stands can be modeled as open grids with some obstacles, similar to the discretized environment ECBS-CT used. Utilizing ECBS-CT as the path planning solver for GSE vehicles, the motion primitives defined in the problem can be used to represent the possible movement of GSE vehicles, instead of simple and non-realistic rectangular movements. However, in airport environments, the speed of GSE vehicles, including the turning speed, acceleration, and deceleration are restricted. These limitations have to be considered while designing the motion primitives.

Moreover, as the shape of most GSE agents is not simple circles that occupy only single grids at one time-point, defining cells that are swept by an agent is difficult. The swept cell of agents should be specified while designing motion primitives. Also, constructing interval maps like the one in Figure 4.14b required analytical representation of the agent's shape while moving according to the designed motion primitives. That is, an equation presenting the relation between elapsed time and the outline of the agent. The equations may be unavailable and computationally expensive to construct. If ECBS-CT is to be implemented to path planning of GSE vehicles in our research, the constraints may need to be detected and recorded using discretized time points.

CCBS

Continuous-time conflict based search (CCBS) was proposed by Andreychuk et al. [5]. CCBS is an adaption of CBS based on a customized version of SIPP. The problems CCBS tried to solve are quite similar to the MAMP problems formulated by Cohen et al. [15]. The main differences between CBS and CCBS are conflicts detection and resolution, constraints implementation, and the low-level search using SIPP.

CCBS can handle agents with different shape and sizes. In CCBS, conflicts can happen while agents occupying different vertices and traversing different edges. CCBS conflicts are defined as conflicts between actions. Formally, an CCBS conflict is represented as a tuple (a_i, t_i, a_j, t_j), representing the collision while agent i execute a_i at time t_i and agent j execute a_j at time t_j. In response to the conflict definitions described above, CCBS computes the unsafe interval of both actions with respect to the other action. As defined by Andreychuk et al. [5], *"the unsafe interval of action a_i with respect to agent a_j is defined as the maximal time interval starting from t_i will create a collision with performing a_j at time t_j"*. In one of the high-level successor nodes, CCBS add constraints to the node that agent i cannot perform a_i in its unsafe interval with respect to a_j, and vice versa in another node. The pair of constraints can be represented as $(i, a_i, [t_i, t_i^u))$ and $(j, a_j, [t_j, t_j^u))$.

The low-level solver of CCBS is a modified version of SIPP. This adaption of SIPP is subjected to constraints represented in the form of $(i, a_i, [t_i, t_i^u))$. There are two possible cases:

- a_i **is a move action**, in which agent i is moving from the source node v to the target node v'. If the agent arrives at v at some time point $t \in [t_i, t_i^u)$, the action moving from v to v' is removed at time t.

Additionally, a waiting action at v until time t_i^u, combining with a moving action to v' at time t_i^u are added.

- a_i **is a wait action**, in which agent i is waiting in vertex v. The waiting action within $[t_i, t_i^u)$ is forbidden by splitting the original safe intervals of v accordingly. One of the new safe intervals ends at t_i and the other starts from t_i^u.

Andreychuk et al. claimed that the modified version of SIPP returns paths with the lowest costs. The author also proofed the optimality and completeness of CCBS. CCBS requires accurate collision detection between agents and unsafe interval detection mechanisms. The collision detection needs to take into account the kinematics and geometry of agents, as the discussion of the definition of the swept cells in ECBS-CT. Moreover, as mentioned by Andreychuk et al., there is no known close-loop formula for computing safe intervals. A general approach is to discretize the detection by applying the collision detection multiple times using small time increments. The authors noted that conflict detections take a significant portion of solution time. Therefore, they implemented two heuristics to speed up the process. One is the past-conflict heuristic that checks conflicts between pairs of agents with higher numbers of past conflicts. The other heuristic is a hybrid heuristic combing cardinal conflicts and past conflicts. These conflict heuristics are very useful if CCBS is to be implemented in our model, as the conflict detection of GSE agents can be quite complicated due to their kinematics and geometry.

Experiments of CCBS were conducted on a 10×10 open-grids and the DAO den520d map. The success rate was higher than 90% with 12 agents on the open-grids and with 15 agents on the DAO map [1]. In the applications of path planning for automated ground handling, the numbers of GSE agents are between 10 and 20, but the geometrical shape is more complicated than simple circle shapes used in the experiment of Andreychuk et al.. Therefore, the applicability of using CCBS to solve the path finding problems in our research should be further investigated.

CCBS-kc

The previously introduced algorithm CCBS does not take kinematic constraints of real-world robots into account. As an extension of CCBS, CCBS with kinematic constraints (CCBS-kc) was later introduced by Andreychuk [4]. Also, the proposed algorithm allows agents to have different geometric shapes, moving speeds, and rotation speeds. The high-level search of CCBS-kc is the same as that of CCBS, while SIPP in the low-level search is further adapted.

The most important factor that affects the size of the search space is the amount of possible moving directions. A factor k is used to represent the consideration of possible movements to 2^k-connected grids. Applying the adapted SIPP as the low-level planner of CCBS-kc, it is not necessary to expand states with all possible headings. Since the agents in CCBS-kc are allowed to rotate, the heading of the agents can be connected to their motion of rotation. Moreover, Andreychuk also noticed that some states are dominant over other states and can thus be discarded. Consider two states s_1 and s_2 correspond to the same vertex on the graph but differ in headings, if $g(s_1) + rotate\_cost(s_1, s_2) \leq g(s_2)$, where $rotate\_cost$ is a function that calculate the time needed to rotate from the heading of s_1 to that of s_2, then s_2 dominants over s_1. The successors of s_2 can be generated via s_1 with lower cost essentially. Kamphof further noticed that these dominance effects can be generalized to the acceleration and deceleration motions [48]. With the domination effects between states, the search space of SIPP can be greatly reduced.

CCBS-kc was compared to CCBS on a 16×16 open-grid. Both algorithms guarantee optimal solutions, but the solution cost of CCBS-kc is higher than that of CCBS, as more traveling time is needed for agents if kinematics are considered. The experiments also showed that both CCBS and CCBS-kc have success rates lower at 80% with only 11 agents. Overall, the success rates of CCBS and CCBS-kc are questionable for implementations in path finding problems of automated aircraft ground handling. Still, the dominance feature of CCBS-kc may be useful while solving problems with kinematic considerations.

[1] with $k = 2$, where k is a parameter representing in a single agent are allowed to move to every cell located in their 2^k neighborhood in a single action.

4.5. Combined task allocation and path finding

There are different manners to solve problems in which both task allocation and path finding have to be performed. In most works, task assignment and path planning are conducted separately [42]. The goals are first assigned to agents before the non-colliding paths for agents are planned. By these kinds of approaches, the solution qualities may be compromised as some potential collisions can already be taken into account during the task allocation phases. The other manner is to model the problems as combined target-assignment and path finding (TAPF) problems [56].

In TAPF problems, agents are split into teams. A set of goals is given to each team. Ma and Koenig proposed Conflict-Based Min-Cost-Flow (CBM) that can return optimal solutions to TAPF problems in terms of *makespan*. Also, Honig et al. developed CBS with task allocation (CBS-TA), a TAPF solver based on CBS. The algorithm can be further extended to enhanced CBS with task allocation (ECBS-TA), a bounded suboptimal version of the algorithm. In the experiments of Honig et al., ECBS-TA is shown to outperform CBM in terms of average cost and average runtime. Therefore, CBM is not discussed in detail in this report. Below we describe the algorithm of CBS-TA.

In CBS-TA, only the high-level of CBS is modified. Compared to CBS, CBS-TA builds a search forest that contains multiple search trees. Two parameters are added into a regular CBS node, in which *root* represents if the current node is a root node and *assignment* shows the current task assignment which is used in the low-level search. In the beginning, CBS-TA starts a single root node with the best assignment. It expands a new root node with the next-best assignment only on demand. Honig et al. introduced a function *constrainedAssignment(I, O, C)*, where I is the set of assignments that must be included in the solution, O is the set of assignments that cannot be part of the solution, and C is the cost matrix. Optimal assignment algorithms like the Hungarian algorithm can be used to solve the assignment with cost C. The central idea of this approach is to partition the solution space by forbidding some assignments and forcefully including certain assignments.

CBS-TA is also compared to TA+CBS, in which task allocation and path finding by CBS are performed separately. The average cost and average runtime of CBS-TA are lower than that of TA+CBS. ECBS-TA is also compared with TA+ECBS. While the performance of ECBS-TA is better than TA+ECBS in cases with 5, 19, and 19 agents in 8×8 gird maps, ECBS-TA does not scale well when the number of agents increases. The required runtime of ECBS-TA in cases with 40, 70, and 100 agents in 32×32 gird maps is higher than the solution time of TA+ECBS. The large solution space of ECBS-TA hinders its applications in realistic-scaled problems. Moreover, in automated aircraft ground handling, temporal and operational constraints are applied. These constraints greatly complicated algorithms that solve task allocation and path planning simultaneously. For the above reason, TAPF solvers are considered inappropriate approaches for applications in automated aircraft ground handling. TAPF solvers are not included in the evaluation in the following section.

4.6. Evaluation of MAPF solution approaches

In this section, the above introduced path planning algorithms are compared and evaluated. First, classical MAPF algorithms are compared within their classes with more general criteria. After that, we further consider the critical criteria of applications in automated aircraft ground handling. Adpated MAPF solvers can tackle with some realistic assumptions. Also, the motion-based nature of MAMP solvers makes the algorithms more suitable to be implemented in our research. Therefore, a comparison of adapted MAPF solvers and MAMP solvers is also made. Finally, according to the comparison, we conclude the chapter with the possible path finding approaches for automated aircraft ground handling. The algorithms that were discussed in this literature study can be grouped as follows:

- **Classical MAPF solvers**:
  - A* based approaches: A*, OD, ID, EPEA*, ICTS
  - CBS based approaches: CBS, MA-CBS, ICBS, GCBS, BCBS, ECBS, ECBS+HWY, CBSw/P, PBS
  - priority based approaches: CA*, HCA*, WHCA*, CO-WHCA*, CO-WHCA*P, CBSw/P, PBS

- **Adapted MAPF solvers and MAMP solvers**: ECBS-CT, CCBS, CCBS-kc

We place CBSw/P and PBS in both CBS based and priority based classes. These two algorithms applied the branching logic of CBS but introduced priority orders during planning, so they belong to both classes.

4.6.1. Evaluation Criteria

Three categories are used to evaluate different solution approaches. Each category contains several important factors that help us to compare the algorithms. The utilized criteria are discussed in detail below.

- **Performance**: The performance of the algorithms includes the success rates, solution qualities, efficiency, and scalability. The success rate of the algorithm represents the percentage of testing problems that the algorithm can return complete solutions to in a given time. It relates to the logic of the algorithm (whether it guarantees complete solutions), but also connects to the runtime. The solution quality represents the total cost of the solution, which is equivalent to the optimality of the solution. The efficiency is based on the required computational time (runtime) of the algorithm. Lastly, the scalability measures to what extent the algorithm is able to maintain its runtime when the number of agents, obstacles, or the size of the map increase.
- **Applicability**: Applicability refers to how well the algorithm is suitable to be implemented in the area of automated aircraft ground handling. Related factors are the common assumptions of MAMP problems, including the ability to handle agents with various sizes and geometrical shapes (large agents), the adaptability to continuous time settings, the ability to deal with non-uniform edges, and the consideration of kinematics.
- **Implementation**: Implementation evaluates how easy it is to implement the algorithm in our study. Two included factors are complexity and flexibility. Complexity refers to how complex it is to implement the algorithm and to extend previous works. Algorithms with lower complexities are more preferable for implementation and thus receive better marks. Note that the complexity here does not refer to the time complexity of the algorithm. Flexibility relates to how easy it is to modify the algorithm with varying team objectives, shapes and kinematics of agents, and the configuration of the environment.

We use the categories of performance and implementation to evaluate the classical MAPF algorithms, and we use all three categories to evaluate the MAMP solvers.

4.6.2. Trade-off

Trade-offs between algorithms are presented using the criteria mentioned above.

A* based approaches

A* based approaches, including OD, ID, EPEA*, and ICTS, are generally developed earlier than CBS-based approaches and some of the priority based approaches. Those A* based algorithms have been proofed to be inferior to algorithms from other classes. EPEA* was shown to outperform OD [29], while ICTS outperforms OD+ID [84]. In terms of success rate, both EPEA* and ICTS are defeated by at least one CBS variant [85]. Moreover, as discussed in subsection 4.4.2, most MAMP solvers are based on SIPP. It is more complicated for A*-based approaches to be adapted to motion-based path planning algorithms that take geometry and kinematics of agents into account. Therefore, the trade-off between A*-based approaches will not be conducted in this literature study.

CBS based approaches

The trade-off between CBS and its extensions are presented in Table 4.1. ECBS+HWY is not included in the trade-off since it is pretty similar to ECBS but it is only suitable to be applied on maps with certain characteristics. The success rate of both CBS and MA-CBS are not very good. In different DAO maps, their success rates are already lower than 100% in problems with 20 agents [85]. The success rate of ICBS is rather high on the DAO brc202d map. Also, the success rates of GCBS, BCBS, ECBS on the same map are higher than that of CBS. Overall, the success rates only drop below 100% with more than 40 agents, which is considered acceptable in our model. Theoretically, the success rate of CBSw/P and PBS strongly rely on the applied priority order. However, if agents with higher priorities do not block the only possible paths of agents with lower priorities, CBSw/P and PBS have rather high success rates. Both algorithms can handle up to 100 agents on the DAO

Table 4.1: Comparison of CBS based algorithms.

	CBS	MA-CBS	ICBS	GCBS	BCBS	ECBS	CBSw/P	PBS
Performance								
success rate	✗	✗	✓	Δ	Δ	✓	Δ	✓
solution quality	✓	✓	✓	✗	Δ	Δ	✗	✗
efficiency	✗	Δ	Δ	✓	✓	✓	✓	✓
scalability	✗	✗	Δ	✓	✓	✓	✓	✓
Implementation								
complexity	✓	Δ	Δ	Δ	Δ	Δ	✓	✓
flexibility	Δ	Δ	Δ	Δ	Δ	Δ	✓	✓

brc202d map.

In terms of solution qualities, CBS, MA-CBS, and ICBS are able to generate optimal solutions. BCBS and ECBS can provide bounded suboptimal solutions. Finally, as the term greedy implies, bounded suboptimality is not guaranteed in GCBS. Also, due to the nature of decoupled approaches, optimalities are not guaranteed in CBSw/P and PBS. The suboptimality of both algorithms is not bounded. However, according to the experiments of Ma et al. [57], the factors of suboptimality are small. This means that CBSw/P and PBS can still generate rather good solutions. On the other hand, the efficiency and scalabilities are not that good in optimal approaches, but are rather good in algorithms that are suboptimal. The expected complexities of CBS, CBSw/P, and PBS are the lowest, because CBS is also used in other master thesis of ATO. The two priority-based algorithms have some priority order to follow when conflicts occur. The flexibilities of the two priority based approaches are better for the same reason. There is not much difference between other algorithms in terms of complexities and flexibilities.

Regarding the performance of the algorithms, ECBS and PBS are the two best algorithms. Both approaches perform well in terms of success rate, efficiency, and scalability. The solution quality of ECBS is better than PBS as it can provide bounded suboptimal solutions. Although the suboptimality of PBS is not bounded, it still returns solutions with good suboptimality, as Figure 4.8b and 4.8d shows. Considering the criteria of implementation, PBS is less complex and more flexible to be implemented. Overall, ECBS and PBS are two solution approaches that can be further adapted and applied in path finding for automated aircraft ground handling.

Priority based approaches

Table 4.2: Comparison of priority based algorithms.

	CA*	HCA*	WHCA*	CO-WHCA*	CO-WHCA*P	CBSw/P	PBS
Performance							
success rate	Δ	Δ	✓	✓	✓	✗	✓
solution quality	Δ	Δ	✗	Δ	Δ	✓	✓
efficiency	Δ	Δ	✓	✓	Δ	Δ	Δ
scalability	Δ	Δ	Δ	✓	Δ	Δ	✓
Implementation							
complexity	✓	✓	Δ	Δ	Δ	Δ	Δ
flexibility	✓	✓	✓	✓	✓	✓	✓

Another group to evaluate is the class of priority based approaches. The trade-off between different priority based algorithms is shown in Table 4.2. The comparison of CA*, HCA*, and WHCA* was provided by Silver[88]. WHCA* outperforms CA* and HCA* in terms of success rate, efficiency, and scalability using a limited search window. For the same reason, the solution quality of WHCA* is not as good as the other two algorithms. Moreover, Bnaya and Felner provided a comparison between WHCA*, CO-WHCA*, and CO-WHCA*P [8]. The

success rates and solution qualities of CO-WHCA* and CO-WHCA*P are quite close to that of WHCA*[2]. Also, CO-WHCA* and CO-WHCA*P effectively retain the required solution time without sacrificing solution qualities. Especially, the required runtime of CO-WHCA* is the shortest among the three algorithms. CO-WHCA* is also the best in scalability as its runtime is sufficiently shorter than the two others when the number of agents grows, as demonstrated by the experiments of Bnaya and Felner [8]. CBSw/P and PBS are compared to CA* in Ma et al.'s study [57]. CBSw/P was shown to have a lower success rate than CA*, while PBS has higher success rates than CA*. The runtime of both CBSw/P and PBS are higher than that of CA*. However, PBS is more scalable as its success rates have hardly dropped with increasing number of agents. In terms of solution qualities, although both CBSw/P and PBS cannot guarantee bounded-optimal solutions, their returned solutions were shown to be very close to optimal in the experiments of Ma [57].

Regarding the implementation, the complexity of algorithms that used a planning window is considered higher than the ones without. Also, algorithms of CBSw/P and PBS are considered more complex as tree structures have to be implemented. The flexibilities of all algorithms are fairly high. Priority based algorithms utilized decoupled plannings for individual agents, thus they are easier to be modified or adjusted. Overall, algorithms that perform best within the class of priority based approaches are CO-WHCA* and PBS. PBS is also one of the best algorithms in both the class of CBS and priority based approaches. This makes PBS a strong candidate of the path planning algorithm in our study. Still, some adaptions are necessary to make this classical MAPF solver applicable in the area of automated aircraft ground handling. In addition, CO-WHCA* is one of the best algorithms in its class. The idea of limited planning windows implemented in CO-WHCA* is interesting to be further explored.

Adapted MAPF solvers and MAMP solvers

Finally, the adapted MAPF solvers discussed in section 4.3 and the MAMP solvers introduced in section 4.4 are evaluated. Three categories, namely performance, applicability, and implementation will be used as the criteria of evaluation. The trade-off between algorithms is presented in Table 4.3. Unfortunately, these approaches have hardly been tested on the same maps. The marks we gave are based on limited amounts of available experimental studies, and our own reasoning, which is discussed below.

First, we compare the performance of different approaches. The success rates of LA-MAPF and MC-CBS are low. In the experiment of Li et al. [54], different variants of MC-CBS are tested. The success rates are already lower than 100% with merely 4 agents. LA-MAPF, as a predecessor of MC-CBS, is considered inferior to MC-CBS. Thus, the success rate of LA-MAPF could be even worse than that of MC-CBS. The runtime and the solution qualities of LA-MAPF and MC-CBS are also bad because of their demanding computational time. Contrarily, the solution qualities of the two algorithms are quite good as they both promise optimal solutions. Despite the that the performance of MAPF-POST is highly dependent on the applied MAPF solver, the linear programming (LP) solver solving the STN plays an important role in the performance of the algorithm. In the experiments performed by Hönig et al. [41], MAPF-POST solved the problem with 10 agents in a few seconds, while it spent 6 minutes solving the problem with 100 agents. This result implies that the efficiency of MAPF-POST with smaller problems is acceptable, but the algorithm is less scalable with larger numbers of agents. Last we evaluate the performance of MAMP solvers. The solution qualities of these three MAMP solvers are fairly good as the solutions are optimal or bounded-suboptimal. The success rate of ECBS-CT depends on the applied suboptimality factor ω and motion primitives. According to the experiments of Cohen et al.[15], the success rate is sufficient with the number of agents similar to the numbers of GSE vehicles used in common aircraft ground handling operations. However, the success rates of CCBS and CCBS-kc are questionable for real-world applications, as discussed in subsection 4.4.2. Although the runtimes of these three algorithms are acceptable when the numbers of agents are small, as the success rates imply, the scalabilities of CCBS and CCBS-kc are not as good as the scalability of ECBS-CT.

Next, we focus on the applicability of the algorithms. Four factors are included in the criteria of applicability, namely large agents, continuous time, non-uniform edges, and kinematics. Algorithms that consider these aspects receive better marks in the corresponding rows. Two adapted MAPF solvers, LA-MAPF and MC-CBS, consider the geometry of agents, but they do not take other factors into account. On the other hand,

[2]The success rate of CO-WHCA* and CO-WHCA*P are slightly higher than that of WHCA*, but are similar when the numbers of agents are smaller than 30. In our research, the number of GSE agents is often smaller than 30

MAPF-POST is able to take continuous time, non-uniform edges, and kinematics into account because of the usage of STN. STN is widely used for temporal reasoning in the field of artificial intelligence. Considerations of continuous time and non-uniform edge traversal times are the natural properties of STN. Kinematic constraints can be implemented into STN by defining different upper and lower bounds of motions. However, different sizes of agents are not included in MAPF-POST. The next three MAMP solvers utilized SIPP or SCIPP as individual agent planners. The advantage of SIPP is different characteristics of agents can be taken into account via the design of configurations and motion primitives of agents. In general, SIPP can handle agents with any geometry shape. However, we give the three algorithms Δ marks in the criteria of large agents as agents with different shapes and sizes are not included in the demonstrations of original papers [15][5][4]. Also, in the criteria of kinematics, CCBS is considered inferior to CCBS-kc as CCBS-kc is the extension of CCBS, which was designed to better deal with the kinematic of agents.

Lastly, we compare the implementation of the algorithms. Adapted MAPF solvers and MAMP solvers considered sophisticated assumptions in the path planning problems. Therefore, the implementation is essentially more complicated than classical MAPF algorithms. The complexities of MAMP solvers are yet higher than the complexities of adapted MAPF algorithms. These MAMP solvers utilize SIPP as their single agent solvers, requiring delicate designs of agent configurations and motion primitives. In terms of flexibility, LA-MAPF and MC-CBS are flexible to apply different geometry characteristics of agents. MAPF-POST utilized LP to solve the STN. Various temporal and kinematic constraints can be included while building the linear programming model. On the other hand, MAMP solvers are not as flexible as adapted MAPF algorithms. The configurations and motion primitives have to be re-designed if the characteristics of agents vary.

Table 4.3: Comparison of adapted MAPF solvers and MAMP solvers.

	LA-MAPF	**MC-CBS**	**MAPF-POST**	**ECBS-CT**	**CCBS**	**CCBS-kc**
Performance						
success rate	✗	✗	Δ	Δ	✗	✗
solution quality	✓	✓	Δ	✓	✓	✓
efficiency	✗	✗	Δ	Δ	Δ	Δ
scalability	✗	✗	✗	Δ	✗	✗
Applicability						
large agents	✓	✓	✗	Δ	Δ	Δ
continuous time	✗	✗	✓	✓	✓	✓
non-uniform edges	✗	✗	✓	✓	✓	✓
kinematics	✗	✗	✓	✓	Δ	✓
Implementation						
complexity	Δ	Δ	Δ	✗	✗	✗
flexibility	✓	✓	✓	Δ	Δ	Δ

To sum up, adapted MAPF solvers are not suitable to be directly applied to solve the path planning problems of automated aircraft ground handling, as they only consider limited motion-based assumptions of agents. Among the discussed MAMP solvers, CCBS and CCBS-kc have insufficient success rates for real-world applications. Other than CCBS and CCBS-kc, ECBS-CT has acceptable success rates, efficiency, and scalability. The returned solutions of ECBS-CT are fairly good as the suboptimality is bounded and can be decided based on different applications. In general, ECBS-CT is expected to be a suitable approach to deal with realistic problems. In the original study of ECBS-CT [15], agents have the same size and shapes. If ECBS-CT is to be implemented in our research, the geometry of agents has to be further included.

4.6.3. Possible path finding approaches for automated aircraft ground handling

In this section, we discuss some possible algorithms to be applied in our research. The best algorithms from the group of CBS based approaches are ECBS and PBS, and the best ones from the class of priority based approaches are CO-WHCA* and PBS. From the class of adapted MAPFs solver and MAMP solvers, ECBS-CT outperforms other algorithms of its class. According to the trade-off of the first two classes of approaches, PBS

is a satisfactory option for our study. However, to be applicable to solve the path planning problems for automated aircraft ground handling, some adaptions have to be made to handle the MAMP assumptions. The single agent planner SIPP can reason the assumptions of MAMP well. Therefore, a combination of PBS+SIPP is worth to be further explored. Kamphof has studied the combination of PBS+SIPP in his literature study [48]. The other algorithm that receives good marks in the trade-off is CO-WHCA*. However, as an A*-based algorithm, CO-WHCA* cannot handle the assumptions of MAMP. Combining SIPP into CO-WHCA* is not a suitable option as the states defined in SIPP are difficult to cooperate with the search windows of CO-WHCA*. Another possible approach is to combine the original priority-based algorithm, such as CA* or HCA*, with the single agent planner SIPP. This approach is equivalent to applying SIPPwRT to multiple agents with a predefined priority order, which was not included in the trade-off. Because the utilized logic behind CA* or HCA* implies the required computation is less than that of PBS, it is expected that the efficiency and scalability of SIPPwRT for multiple agents is better than that of PBS+SIPP. The success rate and the solution quality of this approach are then needed to be further investigated.

On the other hand, ECBS and ECBS-CT are both one of the best algorithms from their group. Here we favor ECBS-CT as it is an extension of ECBS that already takes the MAMP assumptions into account. As discussed previously, ECBS-CT is powerful for solving real-world path planning problems. Unlike priority based approaches that provide solutions with unbounded suboptimality, ECBS-CT guarantees bounded suboptimal solutions. The downside of ECBS-CT is that the algorithm is quite complicated to implement. Overall, priority based approaches, like PBS+SIPP and SIPPwRT for multiple agents suggested above, are simpler to implement, but there are not any guarantees about the solution qualities. ECBS-CT, however, is more complex in terms of implementation, but the solution qualities are fairly better than priority-based approaches.

In current aircraft ground handling operations, priorities of GSE vehicles are widely used, as shown in Table 2.1. To gradually introduce the automation of aircraft ground handling, the continuation of some current operations are preferable approaches. Therefore, the priority based approaches are expected to be the most appropriate algorithms to be implemented in our research. ECBS-CT will be considered if the priority based approaches (PBS+SIPP, SIPPwRT for multiple agents) are not proven to be suitable for applications of path planning for automated aircraft ground handling.

5

Research Proposal

In this final chapter, we present the research proposal for this master thesis. First, in section 5.1, the research objective and research questions are formed. In section 5.2, the research methodology, including the research scope, the modeling technique, and the solution methodologies are presented. Last, the research plans, including the model constructing plan and the time schedule are shown in section 5.3.

5.1. Research Objective and Research Questions

Based on the literature study, the following research objective is formulated:

"To design and evaluate a multi-agent control system model for automation of aircraft ground handling operations with multi-objective task allocation optimization and multi-agent path planning capabilities."

1. How can the current operation of aircraft ground handling be characterized?
   (a) What assumptions can be made regarding the current aircraft ground handling operations?
   (b) What is an appropriate scope of the ground handling operations of Amsterdam Schiphol Airport to be modeled, including the number of flights and aircraft stands to be considered, the ground handling tasks, and the required GSE to be involved?
   (c) How do the connections between task allocation and path planning for GSE be modeled?
   (d) How can the configuration of typical aircraft stands be approximated and modeled?
   (e) How can the geometrical shape and kinematic limitations of typical GSE vehicles be approximated and modeled?
2. How can the ground handling tasks of GSE vehicles be allocated?
   (a) What is the approach suitable for solving task allocation problems?
   (b) What are the objectives aimed and the constraints set while performing task allocation?
   (c) What are the proper trade-offs between pursuing short solution time and good solution qualities?
   (d) How do the solver we used performs in terms of optimality and completeness?
3. How can the path of GSE vehicles be planned?
   (a) What are the objectives aimed while performing path planning?
   (b) What assumptions of multi-agent path planning are made?
   (c) What individual agent planner can be used?
   (d) What multi-agent path planning solvers can be used?
   (e) How do we define the priorities of agents (if a decoupled approach is being used)?

(f) How do the solver we used performs in terms of optimality and completeness?

(g) How can the robustness of the solver be quantified and tested?

4. How do we implement and validate the multi-agent model?

   (a) What tool or programming language can be used to construct the model?

   (b) What data need to be acquired in order to design the model or perform simulations?

   (c) How can we acquire the required data?

   (d) How do we determine the necessary parameters in the model?

   (e) How do we perform the sensitivity analysis of the model parameters?

   (f) How can the model be validated?

   (g) How do the chosen parameters affect the result of task allocation and path planning?

5. How does the performance of the proposed multi-agent system be evaluated, in comparison to the conventional operations?

   (a) What are the suitable key performance indicators (KPI) that can be used to evaluate the performance of the proposed model?

   (b) What experiments or simulations should be performed to obtain the KPIs?

   (c) How can the proposed model provide insights on the automation of aircraft ground handling operations?

5.2. Research Methodology

5.2.1. Research scope

To narrow down the research, in this section, we present some assumptions regarding automated aircraft ground handling that are made. In this study, the configuration of pier B in Amsterdam Airport Schiphol will be used as a prototype of the model under some assumptions. We would like to focus on aircraft ground handling of European or regional flights. First, task allocation that assigns vehicles to ground handling tasks will be done. Path for ground handling vehicles will then be planned to avoid collisions while performing the assigned tasks. The general objective is to minimize the delay caused by ground handling.

Ground handling activities

Only the ground handling activities outside the aircraft cabin are considered. The considered activities are:

- Cargo unloading and loading
- Catering galleys unloading and loading
- Connecting and disconnecting Ground Power Unit
- Refueling
- Water and toilet servicing
- Connecting and disconnecting passengers Passenger Boarding Bridge

Assumptions

To properly define the scope of our research, several assumptions are made. The assumptions can be further extended if time permits. The assumptions are listed below:

- The operation of pier B in Amsterdam Airport Schiphol is used as a case study. As pier B is usually used for regional/ European flights, we will focus on the ground handling operations of the aircraft used for short-haul flights. Further, to simplified the model, the Boeing 737-800 aircraft is considered in the model.
- Regarding the infrastructure of the aircraft stands, it is assumed that all aircraft stands are equipped with PBBs, and all passengers are able to board the aircraft via PBBs. Also, all aircraft stands have an underground fuel system so that no large fuel truck will be needed to refuel aircraft.

- Only operations with powered vehicles are considered. Hence, communicating to the pilot using headphone sets, putting pylons, and aircraft marshaling are not included in the model. Moreover, seasonal operations like de-icing, and special service, for example, the use of scissors truck helping disabled passengers, are not included in the study. Also, refueling or charging of the ground handling vehicles is not included in this study.

- All Ground Service Equipment, except PBBs, are shared with the aircraft stands in the same pier. Regarding the cargo vehicles, a single combination of belt loaders, tow tractors, and open baggage carts for the cargo vehicles. GSEs of the same type are assumed to be homogeneous for simplification of the model.

- In our model, aircraft do not move. Thus, towing is not considered in the model. The path planning of PBBs are also not included in the model, as PBBs have specifically reserved paths at each aircraft stand.

- For safety reasons, refueling cannot be performed while passenger boarding and deplaning. For the same reasons, ground handling vehicles have to enter or exit the ramp through specified traffic lanes.

- The number of required GSE vehicles we assumed to serve one single flight is listed in Table 2.1. Numbers of personnel needed to support the operation of GSEs are not limited in the model.

5.2.2. Modeling automated aircraft ground handling

Most ground handling activities outside the aircraft cabin require specific GSE docking aside or connecting to the aircraft. To model the operation of aircraft ground handling activities, mechanisms with task allocation and path planning capabilities have to be implemented. In our research, task allocation and path planning of the ground handling tasks will be considered as two stages in the model. Possible agents to be included are:

- Centralized task allocation agent (single auctioneer)

- GSE vehicle agents (of all types)

- Path planning agents for aircraft stands (one agent per stand)

The environments are deterministic, static, and accessible. There are several environments in the system, a virtual base/terminal, a service road, and several aircraft stands. Multi-agent task allocation and multi-agent path finding will be done successively. In the beginning, the coordinator (auctioneer) assigns tasks to the vehicle agents. The agents travel from their parking spot at the base to the aircraft stand via the service road and enter the aircraft stand. From the predefined entrance of the aircraft stands (start), the vehicle agents travel to the working spot near the aircraft (goal). After completing the task, the vehicle agents travel back to the base via the aircraft stand and the service road or travel to the other aircraft stand to perform the next task. In particular, PBBs are gate-specific. PBB agents have a specifically reserved path in our model. Therefore, task allocation and path planning of PBBs are not considered in the model. Path planning in the environment of the service road and different aircraft stands are separated. The virtual base with a task allocation agent, and the separated path planning systems are illustrated in Figure 5.1.

Modeling the environment

The environment of the GSE depot or base is a virtual element that does not need to be modeled. Consulting the typical GSE layout on an aircraft stand, as shown in Figure 2.3, a grid-structured environment with obstacles can be built. A 64×64-grids (of grid size 1 m) environment with 15-20% obstacles will be implemented to model the aircraft stand environment. Path planning for about 11-20 vehicle agents needs to be solved. The environment of aircraft stands is also discussed in section 4.1. As the focus of multi-agent path planning is the GSE trajectories on aircraft stands, the modeling of the service road environment is first kept simple. The service road environments can be modeled by simple open grids with directional constraints, representing the traffic lanes in reality and the areas that allow vehicles to turn. If time allows, a more sophisticated model of service road can be designed.

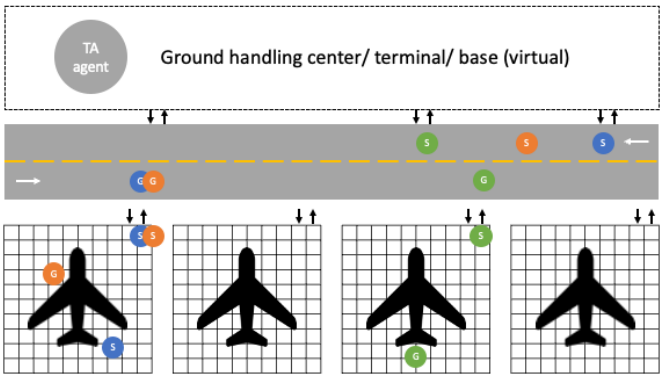
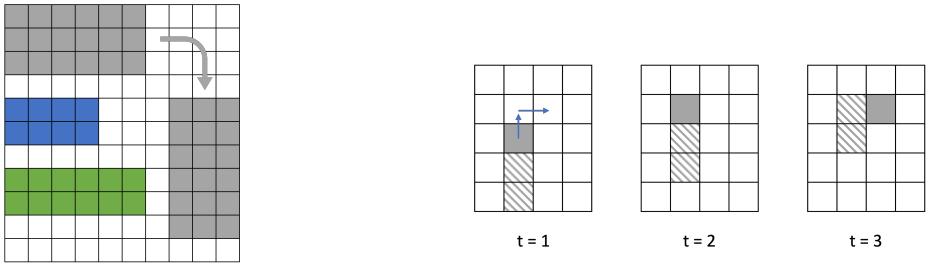


Figure 5.1: Illustrate the task allocation agent and the tasks in auction, the service road system for path planning, and several aircraft stands systems. The black arrows indicate the entrances and exits of the service road and the aircraft stands. The *s* and *g* in circles represent the start and goal locations of the GSE agents.

Modeling of agent geometry and kinematics

In aircraft stands environments, GSE vehicle agents can be represented by rectangular shapes occupying multiple grids, as shown in Figure 5.2a. The arrow in the same figure indicates the possible turning motion. The configuration of agents, that is, the grids it occupies at every time step should be further defined. Representation of the turning motion of the baggage tractors and carts can be complicated. A possible simplified representation is shown in Figure 5.2b. The gray grid represents the baggage tractor and the grids with stripes represent the following baggage carts. After the gray agent pass, cells that have been occupied will still be blocked for a certain amount of time.



(a) Representation of GSE agents.

(b) Representation of turning baggage tractors and carts.

Figure 5.2: Representation of GSE vehicles.

The kinematic of agents can be considered using motion primitives that are used in SCIPP. The motion primitives are feasible motions of agents, like the examples shown in Figure 4.15. Minimum turning radius is already considered in the design of motion primitive. The limitation of velocity, acceleration, and deceleration can also be taken into account while designing motion primitives. As the maximum speed on aircraft stands are quite low, we can use motion primitives with several constant speed without considering acceleration limits. We can also design more sophisticated motion primitives with different starting speeds and ending speeds. Currently, we keep the geometry shape and motion primitives of agents in service road environments simple. Simple point agents and straight motions with constant speed will first be utilized. If time allows, these settings can be further expanded.

5.2.3. Solution methodology

Task allocation assumptions

Auction mechanisms are considered to be implemented in the model. Several assumptions of the auction are:

- The item to be auctioned are the time slots to perform certain ground handling tasks. An example of time slots to be auctioned is shown in Figure 5.3.
- The duration of the time slots includes the time needed for the vehicles to travel to the aircraft stand, to perform the task, and to return to the base after completing the task.
- The time described above is estimated using current data.
- The duration of performing certain ground handling tasks, e.g. the duration of refueling, are assumed to be constant based on current data.
- Auctions of different ground handling vehicles are performed separately.
- The ground handling vehicle will be available for the task only from the start time of the allocated time slot. The end time is a target that the vehicle is estimated to have finished the task and have returned to the base. However, the end time can be violated. Risks related to task duration uncertainties will be considered during resource allocation.
- All tasks are assigned in advance (before planning paths for vehicle agents) within a defined time window.

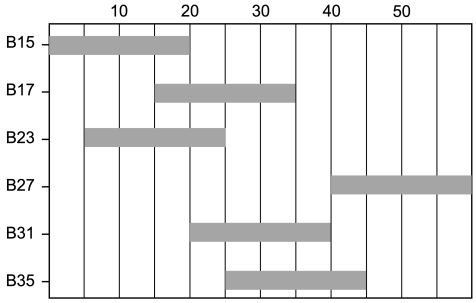


Figure 5.3: Example of a time slots of one type of GSE vehicle. The horizontal axis represents the time, while the vertical axis shows different aircraft stands.

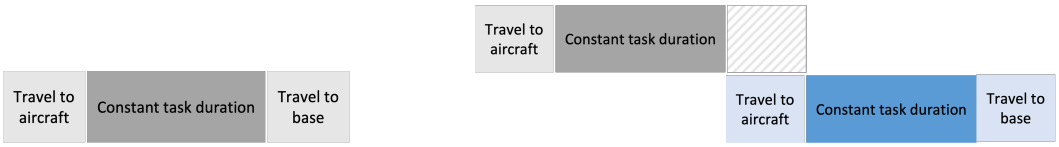
Agent bid representation

Agents bid on the time slots of ground handling tasks that the auctioneer announces, as shown in Figure 5.3. The values of the bids are agents' costs to perform the tasks. The duration of time slots that the auctioneer announces includes the time to travel from the base (the assigned entrance/exit of the service road) to the aircraft stand, a constant task duration, and the time to travel back to the base.

Figure 5.4a illustrate the components of task considered in one time slot. If an agent with a winning bid would like to bid on the other tasks that start immediately after its previous won task, it needs to consider the time constraints of the task that it would like to bid on. As shown in Figure 5.4b, the agent considers the time to travel from the previous aircraft stand to the next aircraft stand, ignoring the time to travel back to the base. Depending on the nature of ground handling vehicles, bidding on successive tasks might not be possible. Some considerations are listed below:

- Agents that can bid without limits on successive tasks: GPU vehicle and refueling carts do not have capacity constraints. Therefore, they can perform several ground handling tasks consecutively.
- Agents that bid with some limits on successive tasks: catering trucks, water service trucks, and toilet service trucks have some capacity constraints. The amount of catering galleys, water, and waste water are limited. They can only bid for limited numbers of consecutive tasks.
- Agents that bid with special limits: Cargo trucks or trolleys can only serve one flight at a time. Cargo trucks that just loaded an aircraft (i.e. unload itself) can bid on a task to unload another aircraft (i.e. load itself).

The suitable solution algorithm is already discussed in subsection 3.3.4. Adaptation of pTeSSB with centrally generated bundles by genetic algorithm will be implemented. Readers are referred to chapter 3 for detailed discussions of the task allocation solution approaches.



(a) Time slots that agents consider for their first bid.

(b) Time slots that agents consider for the successive bids.

Figure 5.4: Time slots considered in the auction.

Path planning methods

Different path planning environments, including the service road system and the aircraft stands, are shown in Figure 5.1. Paths are planned between the base and the entrance/exit of the aircraft stand, between the entrance/exit of the aircraft stand and the working spot near the aircraft, and between the exit of an aircraft stand and the entrance of the other aircraft stand.

GSE agents enter the service road system from the virtual base via single or multiple predefined entrances, depending on their parking spots. After completing some tasks, they exit the service road system (to a virtual base) via a single or multiple predefined exits. The position of the entrance and exits can be planned based on the current parking locations of pier B in Schiphol. From the service road to the aircraft stands, it is assumed that GSE agents can only enter or exit the aircraft stands via the defined entrances and exits, which are at the upper right of aircraft stands in Figure 5.1.

Multi-agent path planning algorithms will be applied to generate conflict-free paths for agents. The suitable path planning approaches are discussed in subsection 4.6.3. We will first use SIPPwRT for multiple agents to deal with the problem. For more detailed discussions of path planning approaches, readers are referred to chapter 4.

5.2.4. Simulation and model analysis

After implementing the model, test scenarios will be developed to validate the model. Sensitivity analysis will also be conducted to analyze the determination of model parameters. Further, based on the operational data of Schiphol, simulations will be performed. Lastly, statistical analysis will also be performed to evaluate the model.

5.3. Research Planning

In this section, we present the planning of the research. The model constructing plan and the planned time schedule of this thesis are included.

5.3.1. Model constructing plan

As a starting point of model development, a basic model will be built. The basic version contains simple but indispensable functions of the model. After the basic model is built, extensions of different aspects will be implemented. The model constructing plan is shown in Table 5.1. As discussed previously in this report, agents involved in automated aircraft ground handling have various temporal and operational constraints, and they can be modeled using different assumptions. In the beginning, the priority order and the number of GSE agents listed in Table 5.2 will be utilized. After that, some extensions regarding the assumption of agents will be applied.

An illustration of the basic model is shown in Figure 5.5. B_1 and B_2 in the terminal show the base of GSE vehicles, while B_c in the middle represent the fictitious base location that can be used as a heuristic in centrally generated task bundles. E_1, E_2, and E_3 show the entrances/exits of the aircraft stands. $P_{RE,1}$, $P_{RE,2}$, and $P_{RE,3}$ present the refueling locations, while the locations of other ground handling tasks can be specified in similar ways. The red lines show a possible trajectory of a GSE agent.

Table 5.1: Model constructing plan.

Item	Basic model	Extensions
Environment size	16*16 grid	64*64 grid
Agent size	1*1	(1) Large agents with the same size (2) Large agents with various sizes
Motion primitives	4 connect-grids with constant speed	(1) Smooth turning motions (2) Speed variation
MAPF algorithm	SIPPwRT for multiple agents	SIPPwRT for multiple agents / to be designed
Moving on service roads	Using the shortest distances. Collisions are neglected at first.	SIPPwRT for multiple agents / to be designed
Number of GSE base	2	Data from pier B
Number of aircraft stands	3	Data from pier B
Algorithm of auction	TeSSI	(1) pTeSSI (2) pTeSSB with centrally created bundles
Agents' bid	(1) Shortest additional distance (2) Shortest total distance	(1) Linear combination of the two (2) Risk parameters included
Visualization	Based on AE4422 warm-up exercise	To be designed

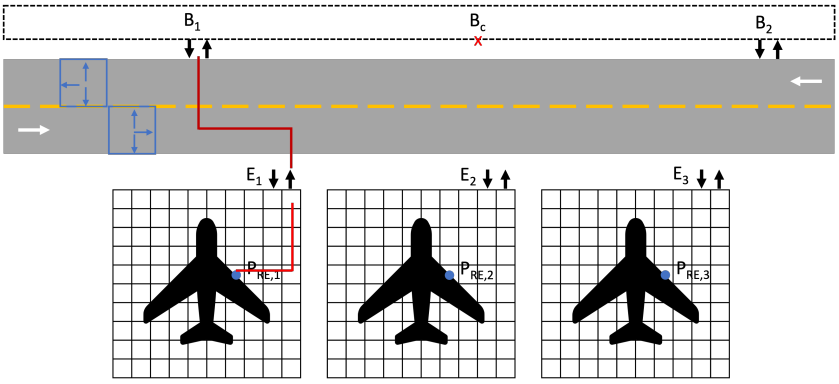


Figure 5.5: Illustration of the basic model.

Table 5.2: Agent specifications in the basic model and their extensions.

GSE	Abbreviation	Priority	Number	Extension
Refueling	RE	1	1	Refueling cannot be done with passenger boarding
Catering	CA	2	1	2 trucks for 3 aircraft (2 trucks working for 1 aircraft)
Baggage	BA	3	2	Belt loader and tractor + carts arrive separately. Start baggage handling when all vehicles arrive.
Water	WA	4	1	1 truck for 3 aircraft
Toilet	WC	5	1	1 truck for 3 aircraft
GPU	GP	6	1	

5.3.2. Time Schedule

The research planning with divisions of tasks and expected working duration is shown in Table 5.3. A basic model will be constructed before we move on to the full version of the model. Also, before implementing both models, thorough conceptualization will be made to clarify the idea and logic of the multi-agent system. After implementing the system, we will test the model with different test scenarios. Validation and verification will be performed. Moreover, simulations will be performed using real-world data from Amsterdam Airport Schiphol, the case study of our research. Finally, we conclude the reearch and prepare the final paper and presentaiton.

Table 5.3: Research planning.

High level tasks	Low level tasks	Duration
Start-up	Familiarize with existing ABM tools and original algorithms	2
Basic model conceptualisation	Design conceptual system model	1
	Design sequential auction mechanism	2
	Develop SIPPwRT for multiple agents	2
Basic model implementation	Implement the basic model	1
	Perform visualization of multi-agent system	1
Full model conceptualisation	Define agent configuration and potion primitives	1
	Design genetic algorithm for bundle generation	1
Full model implementation	Implement the full model	2
	Modify the visualisation of the model	1
Testing	Develop test scenarios	1
	Test model with the scenarios	2
	Validate and verify model	2
	Sensitivity analysis of model parameters	1
Evaluation and analysis	Develop simulation scenarios	1
	Perform simulation	2
	Perform sensitivity and statistical analysis	2
Concluding thesis work	Draw conclusion and discussion of the research	1
	Write the final paper and prepare for the presentation	4

III

Supporting work

A

Bundle Generation Heuristics

In the research paper, we briefly introduced the bundle generation of the ground handling tasks. Also, whether a set of tasks is available to be bundled and their *bundle value* used in the model are explained. Considering the availabilities and the*bundle values*, tasks bundles can be generated using the defined bundle generation heuristics in the model. In this appendix, the bundle generation heuristics used in the model is introduced in detail.

The procedures of the bundle generation is shown in 1. The inputs of the function are the set of tasks T to be allocated, a parameter *sizes* representing the sizes of bundles, which is the number of tasks included in a bundle, and a parameter r specifying the percentage of tasks that are bundled. The aim of the function is to generate a set of sets of task ids as its output. The task ids in the same set specified the tasks in the same bundle. In the beginning, the set of sets of bundled ids ID_{bid} are set to be an empty set. All tasks to be allocated are sorted by the start time of their time windows, and the number of tasks in T is recorded. Tasks in T are taken out of the set once they are assigned to a bundle. The following procedures repeat until a sufficient percentage of tasks are assigned to bundles, as specified by the bundling rate r.

First, a set of values of bundles *values* and a set of sets of bundled and ordered task ids *id orders* are set to be empty sets. For all desirable sizes of bundles, we ran a loop in all sets of neighboring tasks of size s in the task set T. Then, a function *isAvialable* is used to evaluate if it is feasible to bundle those tasks. The function takes into account the time windows and the duration of the tasks, and the traveling distance between those tasks. Detailed elaboration of the function *isAvialable* is presented in section 3.2.2 in the research paper. If the bundle is feasible, a bundle value and the optimized order of the tasks in the bundle are generated by the function *makeBundleValue*, which is also explained in in section 3.2.2 in the research paper. The bundle value is added to the set *values*, and the set of ordered and bundled task ids are added to the set *id orders*. Applying all possible bundle sizes, s, after all sets of neighboring tasks are explored, the id sets in *id orders* are sorted by their corresponding bundle values in the set $values$, making the candidate bundles.

Next, the first r percents of tasks in ID_{bid} are examined and possibly combined into bundles, where r is the bundling rate r. For the ids of the tasks in a set in ID_{bid}, we first check if all tasks are still in the set of tasks to be allocated T. This has to be checked because several sets of neighboring tasks are explored, which include overlapping tasks. If all tasks in a set are still in T, the tasks with indicated task ids are officially combined as a bundle. The set of task ids are added to the set ID_{bid}, and the tasks are removed from T. The procedure repeat for several set of task ids in *id orders*. As the heuristic does not guarantee optimized bundles, we would like to only generate bundles with rather good qualities. Therefore, we only explore sets of task ids that contain approximately the number of tasks we would like to bundle. After some bundles are formed and some tasks are removed from T, the whole bundling procedure is executed again to generate good candidate bundles.

Algorithm 1 Bundle generation heuristic

Input:
  T: Set of tasks to be allocated
  size: size of bundles; the number of tasks includes in a bundle
  r: bundling rate; the percentage of tasks that are bundled
Output
  ID_{bid}: Set of sets of bundled task ids to be allocated
$ID_{bid} = \emptyset$
Sort T by the start times of their time windows.
number of tasks = $len(T)$
while $len(T) > (1-r) *$ number of tasks **do**
  values $= \emptyset$
  id orders $= \emptyset$
  for $s \in$ *size* **do**
    for i in *range*(0, number of tasks $- s + 1$) **do**
      neighboring tasks = set of tasks from the i^{th} task of T to the $(i+s-1)^{th}$ task of T
      if bundling neighboring tasks *isAvailable* **then**
        bundle value, bundle order = *makeBundleValue*(neighboring tasks)
        values = *values* $\cup$ bundle value
        id orders = *id orders* $\cup$ bundle order
      end if
    end for
  end for
  Sort *id orders* by corresponding *values*.
  for i in *range*($len($*id orders*$) * r /$ *mean*(s)) **do**
    if all tasks with ids in the i^{th}set in *id orders* is in T **then**
      $ID_{bid} = ID_{bid} \cup$ the i^{th}set in *id orders*
      $T = T -$ task with ids in the i^{th}set in *id orders*
    end if
  end for
end while

B

Functions of the SIPP algorithm

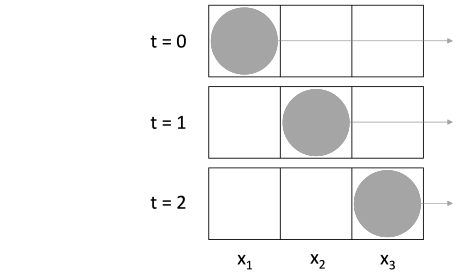
In the research paper of this study, we briefly discussed how the SIPP algorithm works. To better fit our need for path planning for GSE vehicle agents, we modified some functions in the original SIPP in the model. Here we discuss functions *possibleMoves* and *getSuccessor*, which differentiate our proposed method from the original SIPP algorithm. Also, function *updateSafeInterval* of prioritized SIPP used in the model is introduced.

possibleMoves While generating successors for a state, possible moves from the current state are used. Therefore, a set of feasible moves for agents are defined before the path planning processes start. Simple moves like moving to the four connected grids are common moves. These are also the moves considered in our model. Moreover, depending on the current grid the agent stands, directional constraints may apply. In the service road map, agents always travel on the right side. Thus, in some grids, moving in a certain direction is banned.

Other motions can also be implemented in the model. For example, moving in a single direction for more than a grid or performing a turning motion is possible in SIPP. These sophisticated motions can possibly help model the realistic vehicle movement better. Note that the more possible motions the SIPP algorithm uses, the more successor states that the algorithm should explore during path planning. This can possibly prolong the path planning process. However, longer motions can also result in fewer intermediate states between the start and goal of an agent, resulting in shorter path planning durations.

getSuccessor This function generates successors of the current state of an agent. Successors are possible states that an agent can move to from the current state. To generate successors, the agent first checks its possible movement from the current state. Then, the configuration of the agent after performing the movement is considered. If the configuration overlaps with any static obstacles or goes outside the map, the potential successor is pruned. Most importantly, the safe intervals of the occupied grids in the new configuration are checked. The earliest time that the agent can move to a grid is a buffer time later the start of the grid's safe interval. The buffer time is needed because before the time that the reference point of an agent is located at the center of a grid, part of the agent may already enter the grid. The buffer time equals the required time for the agent to move a unit-grid length. Similarly, the latest time that an agent can stay at a grid is a buffer time before the end of the safe interval. The idea is similar to the time offsets implemented by Ma et al. in the SIPPwRT algorithm [58]. For the algorithm of generating successors, readers are referred to the *getSuccessors* function of the original SIPP algorithm [68].

updateSafeIntervals After paths for agents with higher priorities are planned, the safe intervals of grids on the map are updated to avoid collision while planning paths for agents with lower priorities. In MAPF problems, there are two kinds of conflicts, vertex conflicts, and edge conflicts. Safe intervals are updated in a way that both types of conflicts are avoided. For example, in the scenario in Figure B.1, where unit-sized agent is moving to the right from grid x_1. It starts at $t = 0$ and moves one grid per time step. The position of the agent at different time steps is shown in Figure B.1. A grid is considered to be occupied if any part of the agent is inside the grid. The updated safe intervals are equal to the initial safe intervals subtracted by the occupied intervals, as shown in Figure B.1.



	x_1	x_2	x_3
Initial safe interval	[0, ∞]	[0, ∞]	[0, ∞]
Occupied interval	[0, 1]	[0, 2]	[1, 3]
Resulting safe interval	[1, ∞]	[2, ∞]	[0, 1], [3, ∞]

Figure B.1: Path for an agent on a grid map and its resulting safe intervals.

C

Additional Information of Experimental Analysis

C.1. Related data of Experiment A: Traffic demand scenarios

The results of task allocation of different types of GSE vehicles of Experiment A are shown in Table C.1. As discussed in the research paper, the overall makespan in the busy scenario is lower than that of the normal scenario. We conjecture that results are related to the lower allocation rates in the busy scenario. The allocation rates for refueling, catering, water servicing, and toilet servicing tasks have similar results. Among all ground handling tasks, the baggage handling task has the highest allocation rates in all traffic demand scenarios. Because there are one baggage unloading task and one baggage loading task for every flight, the number of tasks to be allocated is the largest among all ground handling tasks. At the same time, the number of available vehicles for baggage handling is double the number of vehicles for other ground handling tasks. Larger numbers of available vehicles can make the allocation more flexible, resulting in higher allocation rates. The other possible reason for the high allocation rates is the relatively shorter computational time for allocating baggage handling tasks. Finally, regarding the computational time, one can observe that for all types of ground handling tasks, the computational time of allocations do not scale very well. Compared to the offpeak scenarios, the allocation computational time double in the normal scenarios, and they double again in the busy scenarios.

C.2. Related data of Experiment B: Bundling

In this section, we present and discuss the detailed task allocation computational time and the bundle generation time of the model from Experiment B. Considering only the task allocation time, the allocation times in the model with bundles are generally higher than that of in the model without bundles. The only exception that the task allocation computational time is reduced is the allocation of baggage handling tasks. This may be associated with the unit capacity of the baggage vehicles. In the SPDP optimization, the constraint related to unit capacity generates a strong cut on the solution space since the alternative links and nodes are highly reduced compared to non-unit capacity constraints. Therefore, the optimization algorithm only needs to explore a limited solution space, resulting in lower computational time. However, considering the bundle generation time, the total required time for assigning baggage handling tasks is still longer than the required time of the model without bundles.

Table C.1: Comparison of the simulation results of different traffic demand scenarios.

task	makespan (min)	task allocation rates	computational time (s)
offpeak RE	142.53	0.9231	18.38
normal RE	202.62	0.8750	20.29
busy RE	166.13	0.7500	43.09
offpeak-normal p	**3.88E-18**	**9.05E-21**	**2.49E-09**
normal-busy p	**3.85E-18**	**3.95E-23**	**3.90E-18**
offpeak-normal A	**0.0000**	**1.0000**	**0.2889**
normal-busy A	**1.0000**	**1.0000**	**0.0000**
offpeak CA	138.65	1.0000	2.78
normal CA	220.47	1.0000	15.08
busy CA	209.07	0.9500	43.16
offpeak-normal p	**3.88E-18**	**1.97E-09**	**3.90E-18**
normal-busy p	6.04E-01	**3.82E-08**	**3.90E-18**
offpeak-normal A	**0.0000**	0.6800	**0.0000**
normal-busy A	0.6400	0.6484	**0.0000**
offpeak BA	203.48	1.0000	0.90
normal BA	221.05	1.0000	1.32
busy BA	218.71	1.0000	2.08
offpeak-normal p	**3.88E-18**	-	**3.90E-18**
normal-busy p	2.65E-01	**1.31E-05**	**3.90E-18**
offpeak-normal A	**0.0000**	0.5000	**0.0000**
normal-busy A	**0.7440**	0.5950	**0.0000**
offpeak WA	140.12	1.0000	7.73
normal WA	199.97	0.8750	32.20
busy WA	218.15	0.7500	63.52
offpeak-normal p	**3.88E-18**	**3.55E-20**	**3.90E-18**
normal-busy p	**5.65E-07**	**1.26E-19**	**3.90E-18**
offpeak-normal A	**0.0000**	**0.9950**	**0.0000**
normal-busy A	**0.1670**	**0.9923**	**0.0000**
offpeak WC	144.25	1.0000	9.10
normal WC	222.88	0.9375	16.70
busy WC	191.88	0.8000	40.53
offpeak-normal p	**3.85E-18**	**3.15E-22**	**3.90E-18**
normal-busy p	**3.89E-18**	**1.71E-19**	**3.90E-18**
offpeak-normal A	**0.0000**	**1.0000**	**0.0000**
normal-busy A	**1.0000**	**1.0000**	**0.0000**
offpeak overall	153.81	0.9846	38.89
normal overall	213.40	0.9375	85.59
busy overall	200.79	0.8500	192.38
offpeak-normal p	**3.90E-18**	**1.41E-18**	**3.90E-18**
normal-busy p	**5.27E-18**	**3.25E-18**	**3.90E-18**
offpeak-normal A	**0.0000**	**1.0000**	**0.0000**
normal-busy A	**0.9852**	**0.7143**	**0.0000**

Table C.2: Data of model computational time with and without task bundles

	no bundles	bundles		
task	task allocation	task allocation	bundle generation	total
RE	20.29	51.65	1.13	52.78
CA	15.08	21.07	0.95	22.02
BA	1.32	1.15	2.40	3.55
WA	32.20	35.64	0.61	36.25
WC	16.70	38.92	1.18	40.10
overall	85.59	148.42	-	154.69

D

Variability of Simulation Results

In the experimental analysis, most data obtained are not normally distributed. Hence, we perform abundant runs of simulations until the coefficient of variation stabilize. In this section, we show the evolution of the coefficients of variation for all experiments discussed in Section 5 of the research paper.

D.1. Settings of the Task Allocation Experiments

In this section, the simulation settings of the task allocation experiments are listed in Table D.1 to **??**. The simulation sets used are also presented. The evolution of the coefficients of variation of the simulation sets are shown in section D.3.

D.1.1. Experiment A: Traffic demand scenarios

Table D.1: Simulation settings of Experiment A

simulation set	traffic demand scenario	objective	task bundles	delay	replan	
II	offpeak	makespan	no	no	no	basic model
I	normal	makespan	no	no	no	basic model
III	busy	makespan	no	no	no	basic model

D.1.2. Experiment B: Bundling

Table D.2: Simulation settings of Experiment B

simulation set	traffic demand scenario	objective	task bundles	delay	replan	
I	normal	makespan	no	no	no	basic model
IV	normal	makespan	yes	no	no	basic model

D.1.3. Experiment C: Replanning

Table D.3: Simulation settings of Experiment C

simulation set	traffic demand scenario	objective	task bundles	delay	replan	
V	normal	makespan	no	yes	no	basic model
VI	normal	makespan	no	yes	yes	basic model

D.1.4. Experiment exploring the task allocation objectives

Table D.4: Simulation settings of the experiment exploring the task allocation objectives

simulation set	traffic demand scenario	objective	task bundles	delay	replan	basic model
I	normal	makespan	no	no	no	basic model
VII	normal	sum-T	no	no	no	basic model

D.2. Settings of the Path Planning Experiments

The simulation settings of the path planning experiment is listed in Table D.5. The simulation sets used are also presented. The evolution of the coefficients of variation of the simulation sets are shown in section D.3.

Table D.5: Simulation settings of the path planning experiment

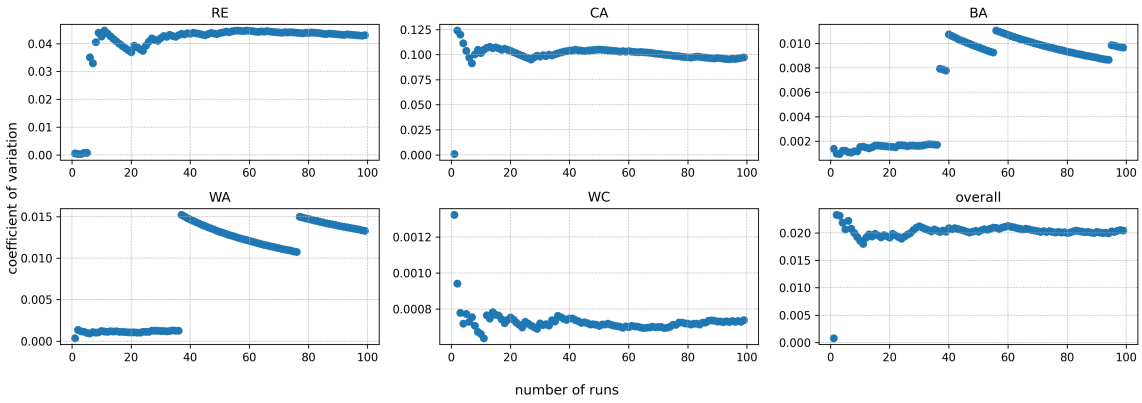
simulation set	traffic demand scenario	objective	task bundles	delay	replan	environment
I	normal	makespan	no	no	no	basic model
VIII	normal	makespan	no	no	no	extension model

D.3. Evolution of Coefficients of Variation

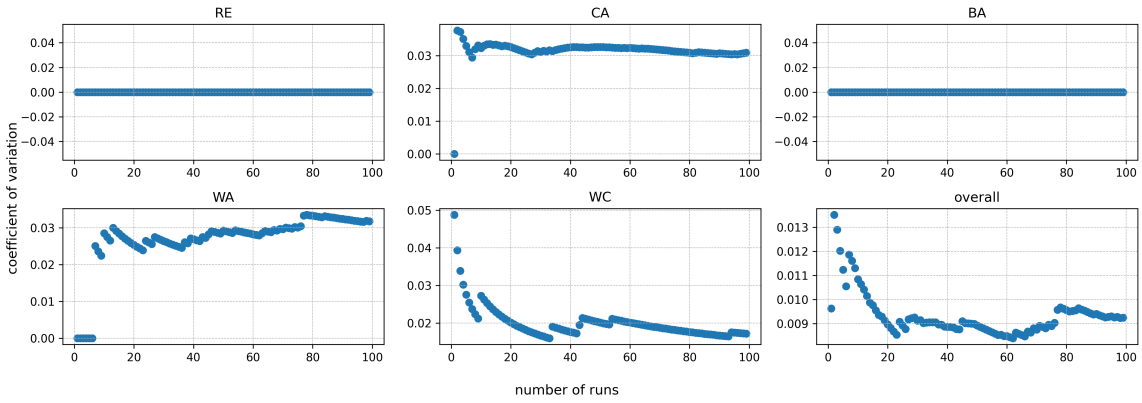
In this section, the evolution of the coefficients of variation of the simulation sets are presented.

D.3.1. Simulation I

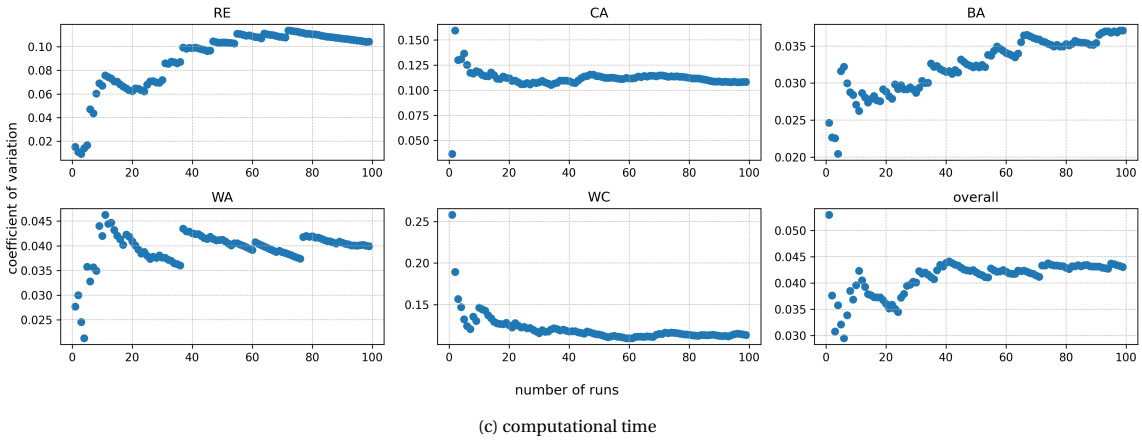
- traffic demand scenario: normal
- objective: *makespan*
- task bundles: no
- delayed: no
- replan: no
- environment: basic model



(a) makespan



(b) allocation rate

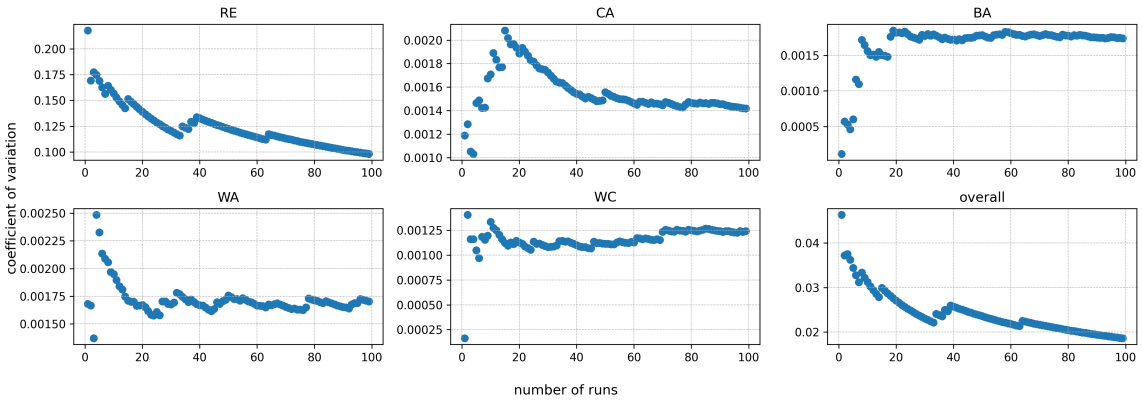


(c) computational time

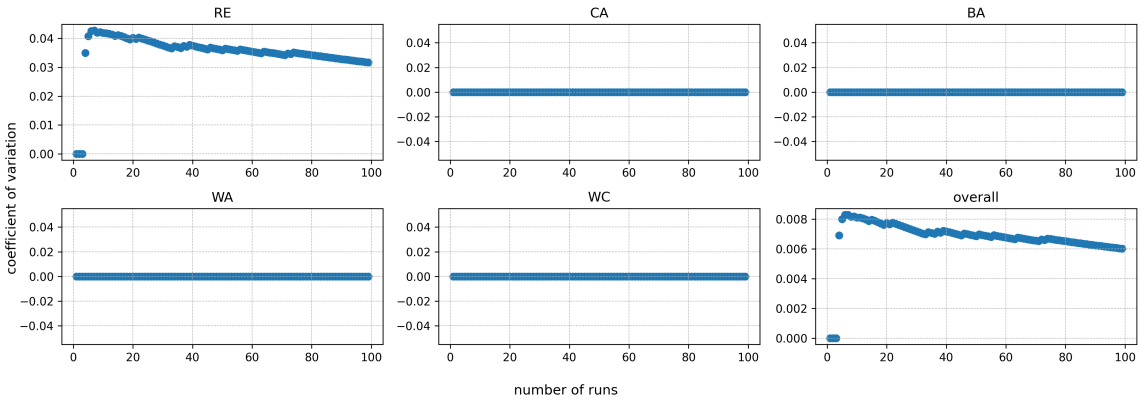
Figure D.1: Evolution of coefficients of variation for performance indicators in simulation I.

D.3.2. Simulation II

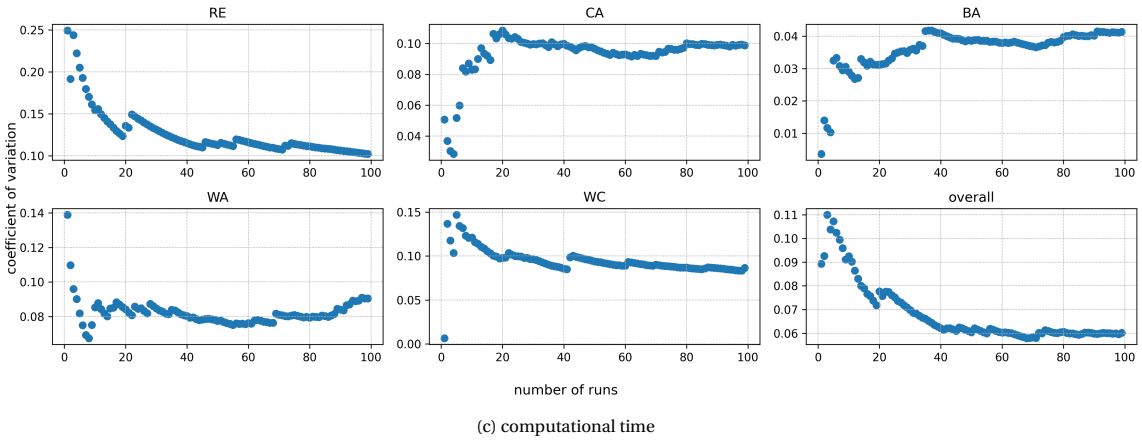
- traffic demand scenario: offpeak
- objective: *makespan*
- task bundles: no
- delayed: no
- replan: no
- environment: basic model



(a) makespan



(b) allocation rate

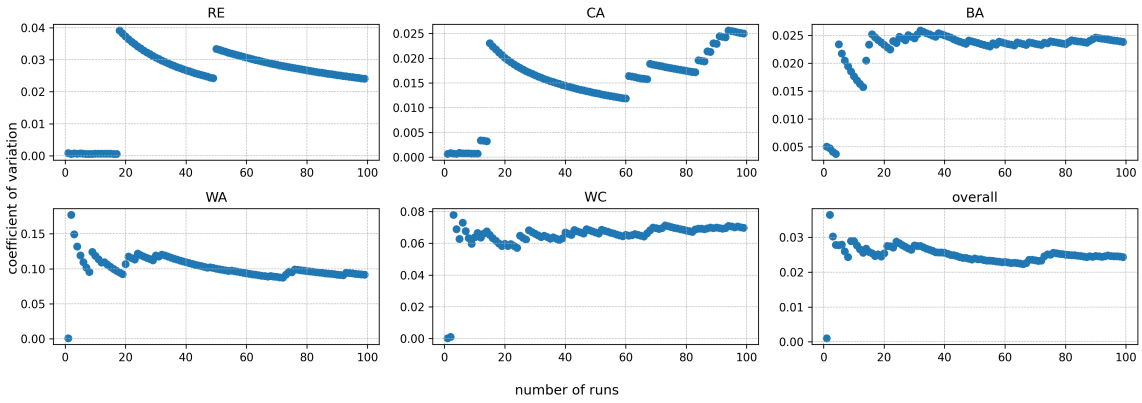


(c) computational time

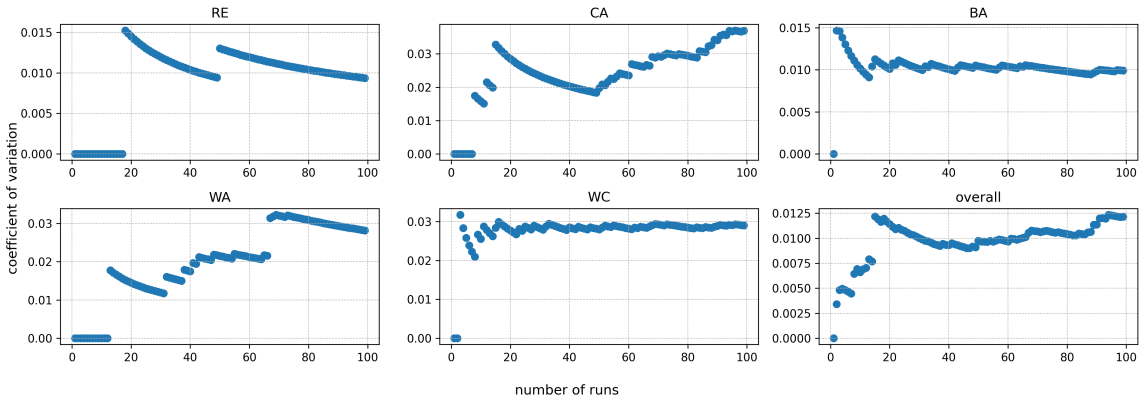
Figure D.2: Evolution of coefficients of variation for performance indicators in simulation II.

D.3.3. Simulation III

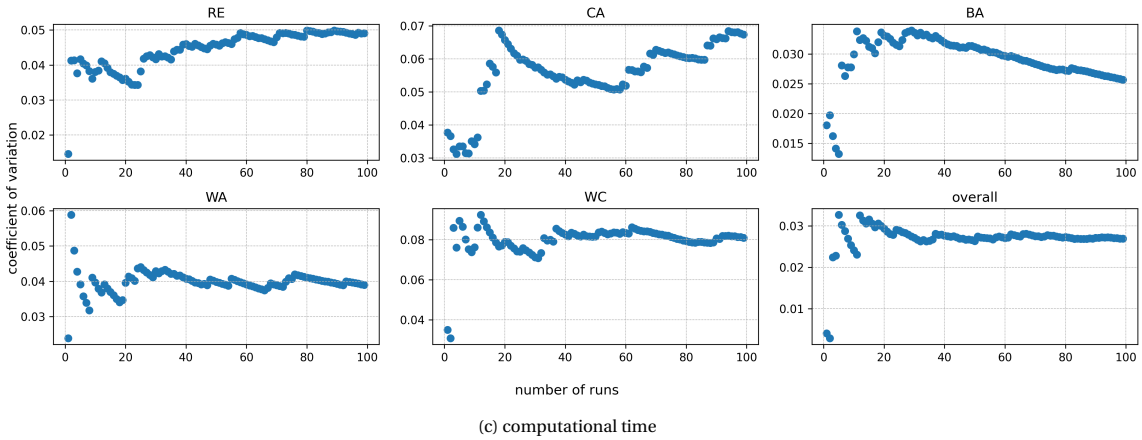
- traffic demand scenario: busy
- objective: *makespan*
- task bundles: no
- delayed: no
- replan: no
- environment: basic model



(a) makespan



(b) allocation rate

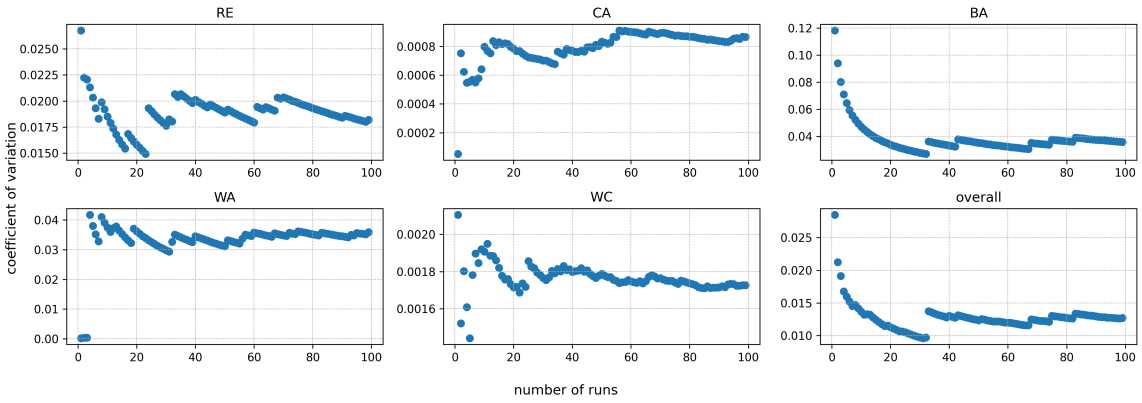


(c) computational time

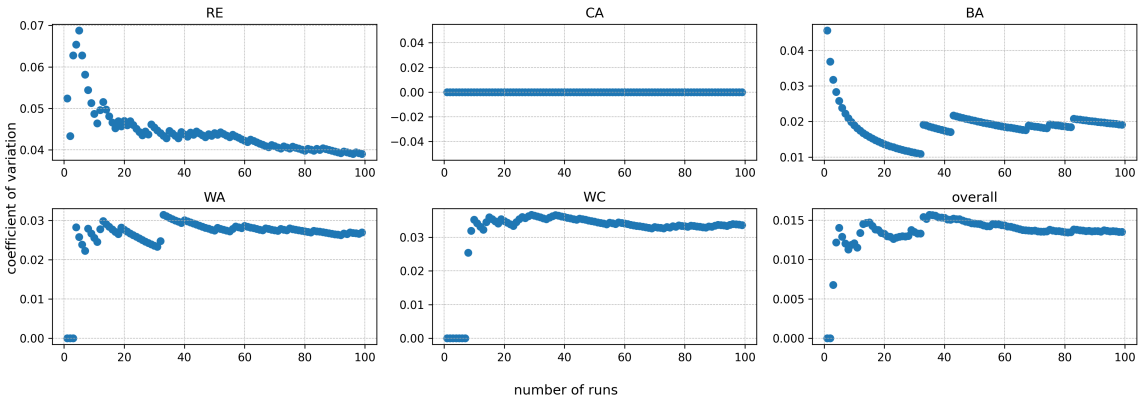
Figure D.3: Evolution of coefficients of variation for performance indicators in simulation III.

D.3.4. Simulation IV

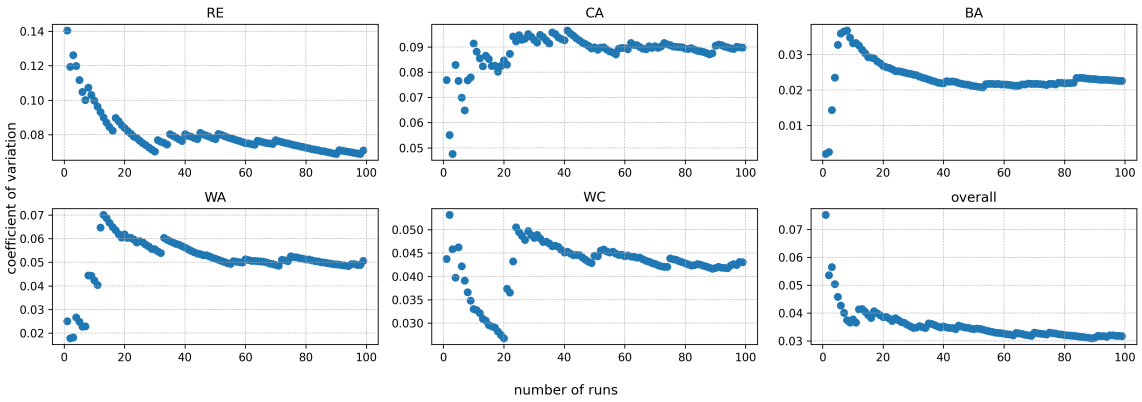
- traffic demand scenario: normal
- objective: *makespan*
- task bundles: yes
- delayed: no
- replan: no
- environment: basic model



(a) makespan



(b) allocation rate

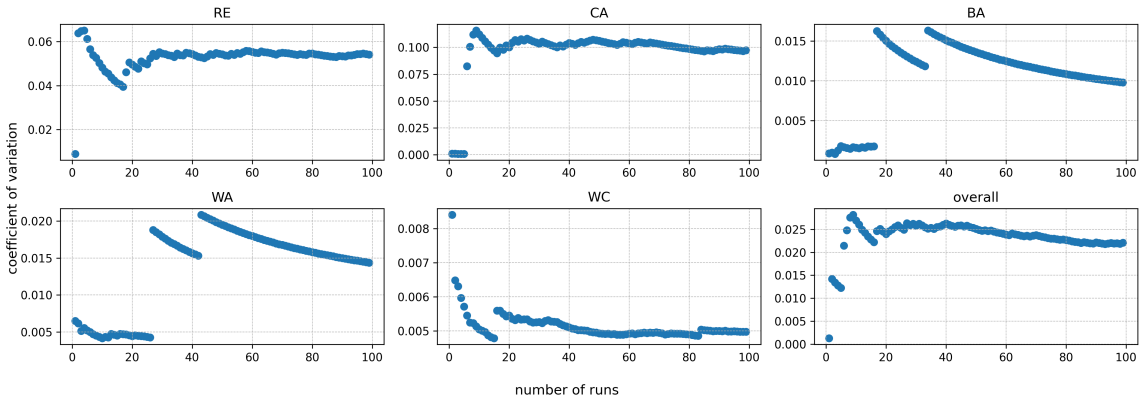


(c) computational time

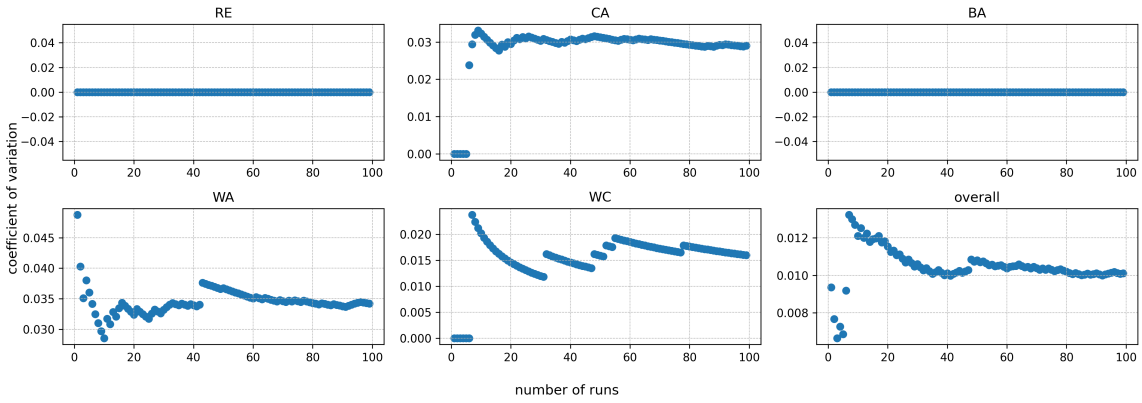
Figure D.4: Evolution of coefficients of variation for performance indicators in simulation IV.

D.3.5. Simulation V

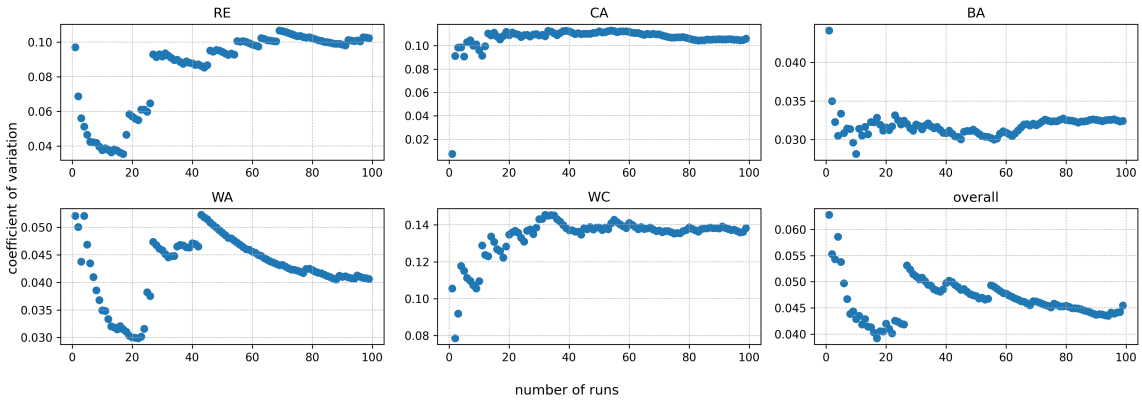
- traffic demand scenario: normal
- objective: *makespan*
- task bundles: no
- delayed: yes
- replan: no
- environment: basic model



(a) makespan



(b) allocation rate

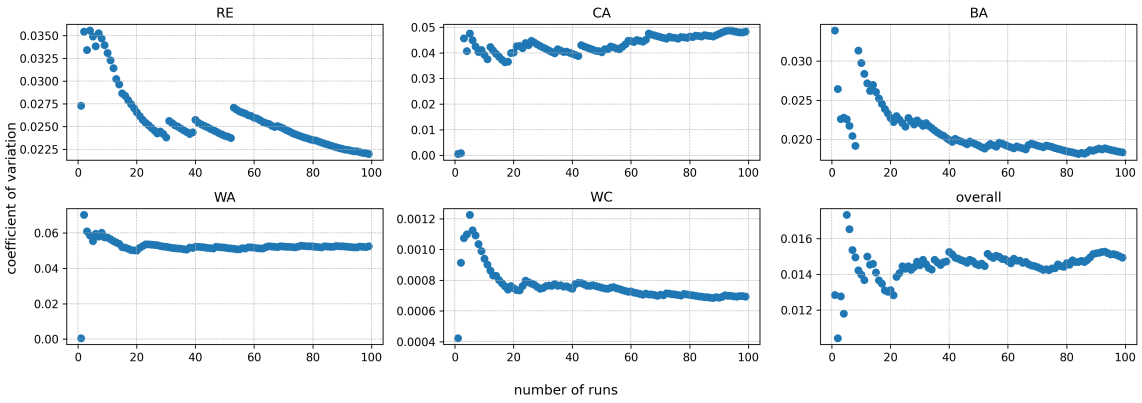


(c) computational time

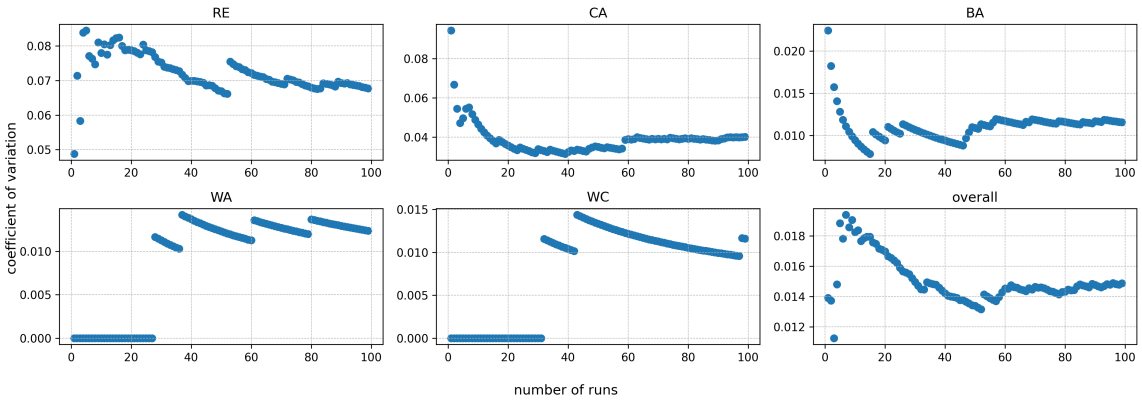
Figure D.5: Evolution of coefficients of variation for performance indicators in simulation V.

D.3.6. Simulation VI

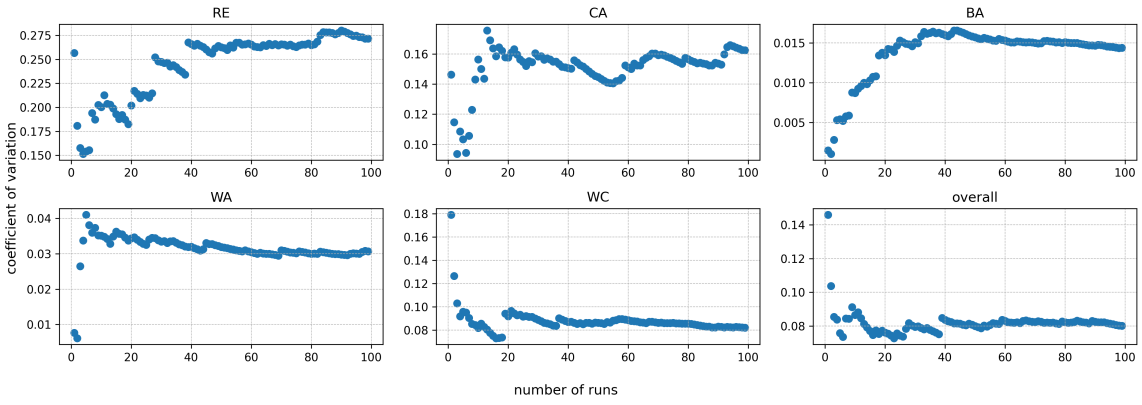
- traffic demand scenario: normal
- objective: *makespan*
- task bundles: no
- delayed: yes
- replan: yes
- environment: basic model



(a) makespan



(b) allocation rate

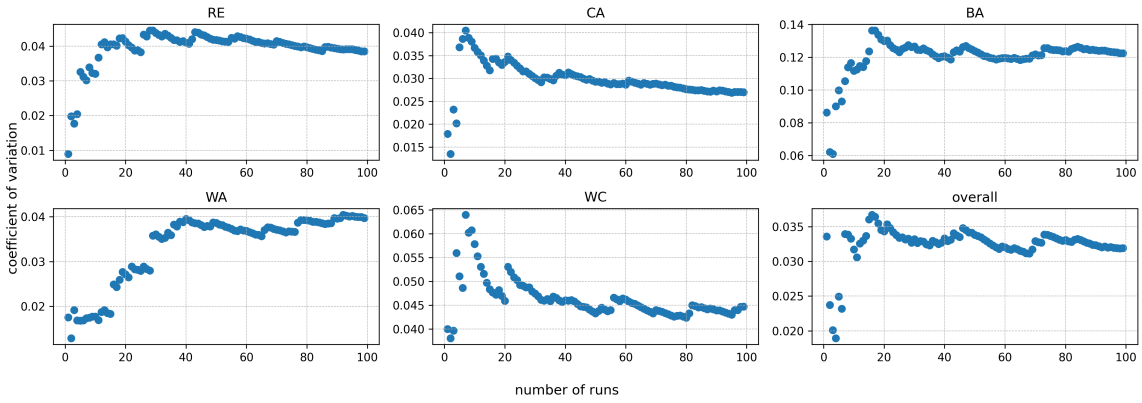


(c) computational time

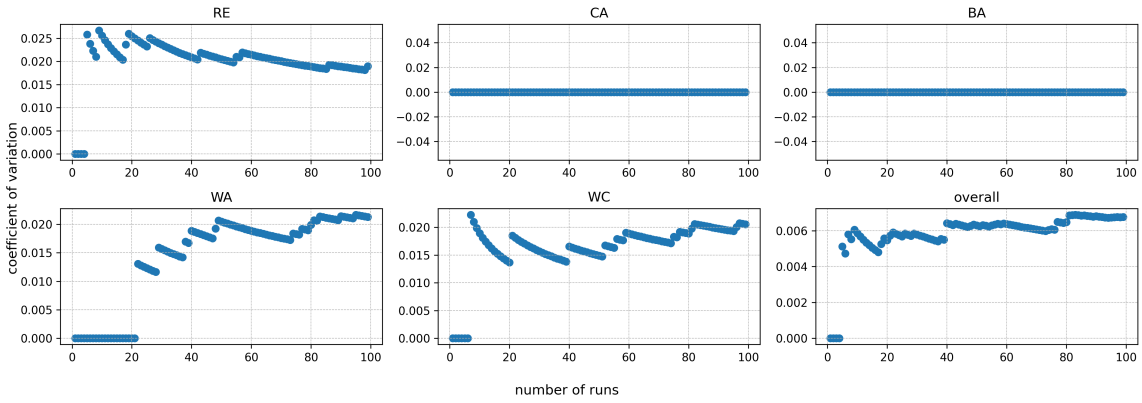
Figure D.6: Evolution of coefficients of variation for performance indicators in simulation VI.

D.3.7. Simulation VII

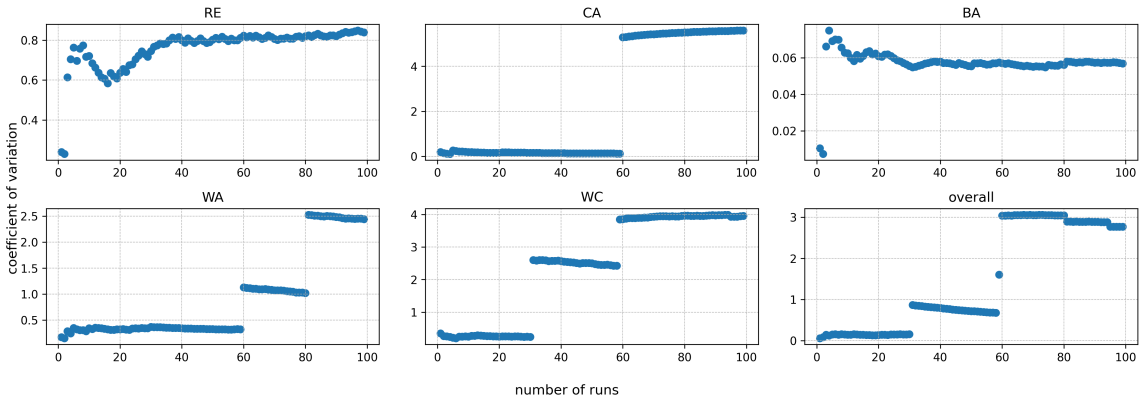
- traffic demand scenario: normal
- objective: *sum-T*
- task bundles: no
- delayed: no
- replan: no
- environment: basic model



(a) makespan



(b) allocation rate

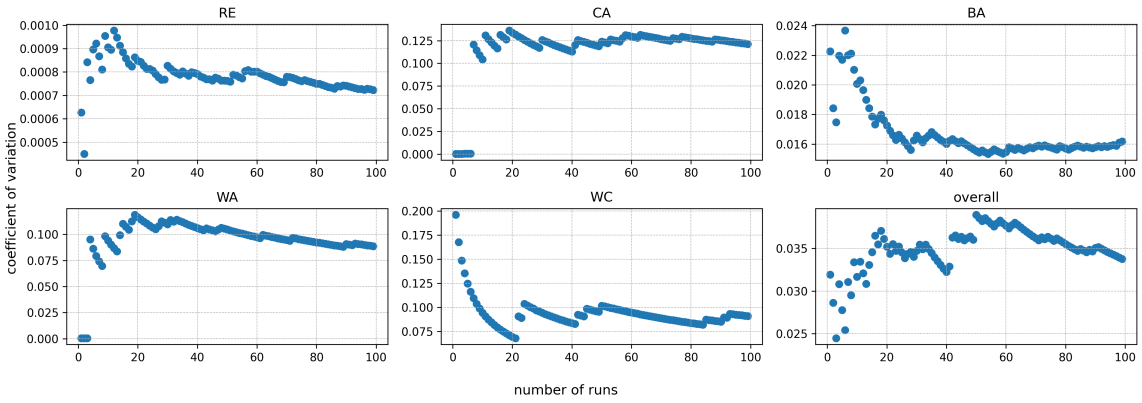


(c) computational time

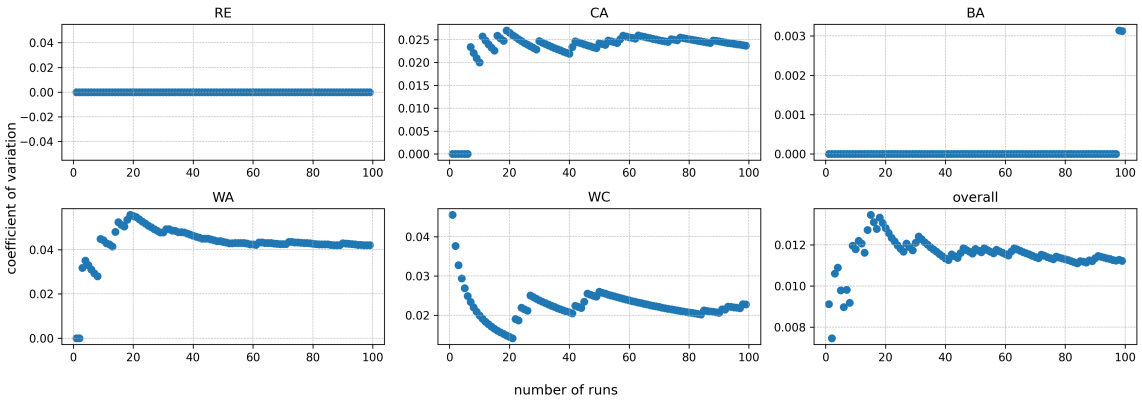
Figure D.7: Evolution of coefficients of variation for performance indicators in simulation VII

D.3.8. Simulation VIII

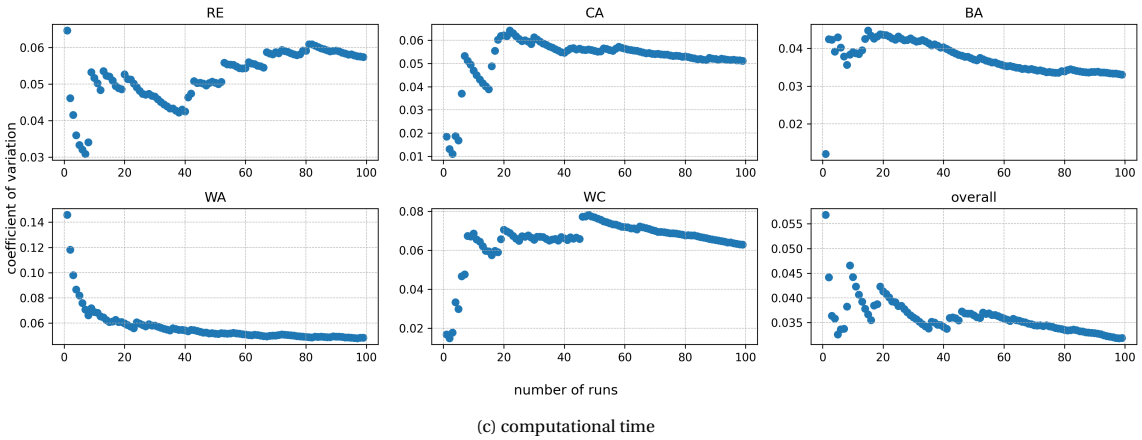
- traffic demand scenario: normal
- objective: *makespan*
- task bundles: no
- delayed: no
- replan: no
- environment: extension model



(a) makespan



(b) allocation rate



(c) computational time

Figure D.8: Evolution of coefficients of variation for performance indicators in simulation VIII.

E

Additional Sensitivity Analysis

In the sensitivity analysis on the processing duration of refueling tasks in the research paper (section 5.4.2), we investigate the makespan of the model by varying the numbers of available vehicles and the task duration. At the same time, we also investigate the task allocation rates and computational time of the model by varying the same parameters. Contour plots of the allocation rates and computational of the model are shown in Figure E.1 and Figure E.2, respectively.

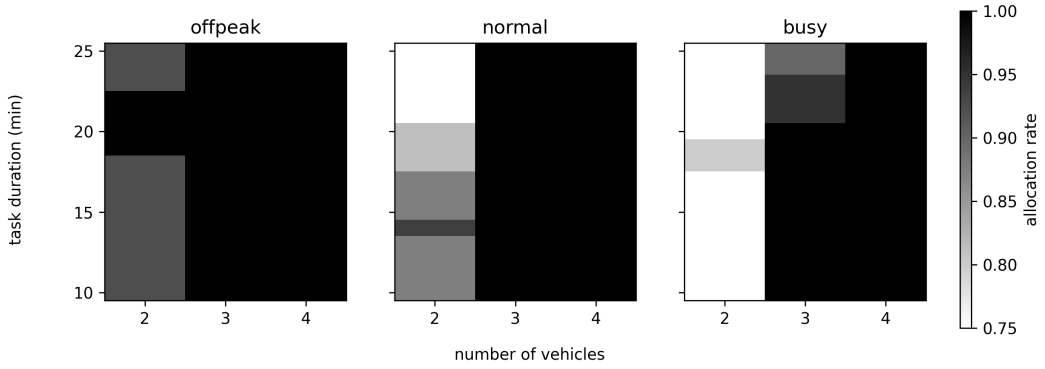


Figure E.1: Contour plot of allocate rates with the *makespan* bidding rule for three traffic demand scenarios, with varying numbers of available vehicles and task duration.

The allocation rates of the ground handling tasks are highly related to the numbers of available vehicles. In Figure E.1, with three or more available GSE vehicles, the allocation rates for different task processing duration are all one in the offpeak and normal traffic demand scenarios. In the busy scenario, the allocation rates of models with three available vehicles are slightly less then 1 if the task processing duration are longer than 20 minutes. For models with only two available ground handling vehicles, the allocation rates are lower in the busier scenarios.

From Figure E.2, it is obvious that in all traffic demand scenarios, the computational time of models with lower task duration and with only two available vehicles are longer. As there are more agents involved in the auction in models with three or four GSE vehicles, there are more bids to be generated. Thus, we presumed that the computational time in such scenarios would be longer. However, in models with lower numbers of available vehicles, more tasks are allocated to every agent. The higher numbers of allocated tasks cause longer SPDP optimization solution time. According to the results of this sensitivity analysis, the affect for optimizing larger numbers of tasks are greater then the affect of having larger numbers of agents involved in the auction.

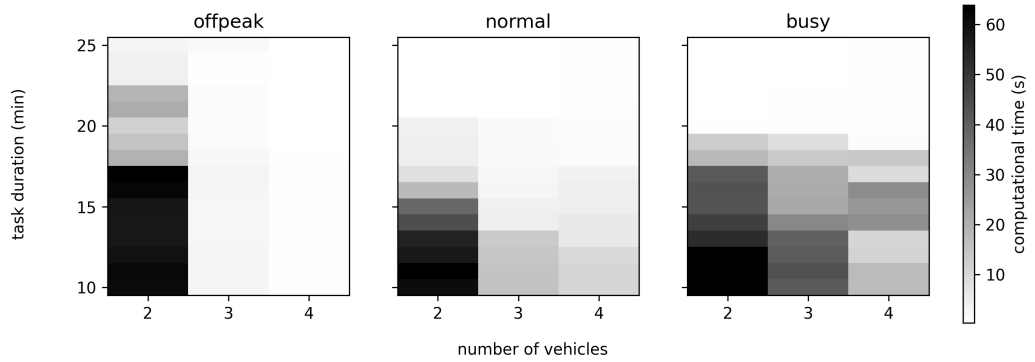


Figure E.2: Contour plot of computational time with the *makespan* bidding rule for three traffic demand scenarios, with varying numbers of available vehicles and task duration.

Bibliography

[1] SAS Airbus. A320 aircraft characteristics airport and maintenance planning. *Issue Sep*, 1:397, 2005.

[2] Boeing Commercial Airplanes. 737: Airplane characteristics for airport planning, sept. 2013. Technical report, D6-58325-6, 2013.

[3] 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.

[4] Anton Andreychuk. Multi-agent path finding with kinematic constraints via conflict based search. In *Russian Conference on Artificial Intelligence*, pages 29–45. Springer, 2020.

[5] Anton Andreychuk, Konstantin Yakovlev, Dor Atzmon, and Roni Stern. Multi-agent pathfinding with continuous time. *arXiv preprint arXiv:1901.05506*, 2019.

[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 *Seventh Annual Symposium on Combinatorial Search*, 2014.

[7] Bram Benda. Agent-based modelling and analysis of non-autonomous airport ground surface operations. 2020.

[8] Zahy Bnaya and Ariel Felner. Conflict-oriented windowed hierarchical cooperative a. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3743–3748. IEEE, 2014.

[9] Eli Boyarski, Ariel Felner, Guni Sharon, and Roni Stern. Don't split, try to work it out: Bypassing conflicts in multi-agent pathfinding. In *ICAPS*, pages 47–51, 2015.

[10] 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 *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[11] Luc Brunet, Han-Lim Choi, and Jonathan How. Consensus-based auction approaches for decentralized task assignment. In *AIAA guidance, navigation and control conference and exhibit*, page 6839, 2008.

[12] Paul C Chu and John E Beasley. A genetic algorithm for the generalised assignment problem. *Computers & Operations Research*, 24(1):17–23, 1997.

[13] Liron Cohen and Sven Koenig. Bounded suboptimal multi-agent path finding using highways. In *IJCAI*, pages 3978–3979, 2016.

[14] Liron Cohen, Tansel Uras, and Sven Koenig. Feasibility study: Using highways for bounded-suboptimal multi-agent path finding. In *Eighth Annual Symposium on Combinatorial Search*, 2015.

[15] Liron Cohen, Tansel Uras, TK Satish Kumar, and Sven Koenig. Optimal and bounded-suboptimal multi-agent motion planning. In *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[16] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

[17] Boris De Wilde, Adriaan W Ter Mors, and Cees Witteveen. Push and rotate: a complete multi-agent pathfinding algorithm. *Journal of Artificial Intelligence Research*, 51:443–492, 2014.

[18] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial intelligence*, 49(1-3): 61–95, 1991.

[19] Mauro Dell'Orco, Mario Marinelli, and Maria Giovanna Altieri. Solving the gate assignment problem through the fuzzy bee colony optimization. *Transportation Research Part C: Emerging Technologies*, 80: 424–438, 2017.

[20] Wu Deng, Junjie Xu, and Huimin Zhao. An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem. *IEEE access*, 7:20281–20292, 2019.

[21] M Bernardine Dias and Anthony Stentz. A free market architecture for distributed control of a multirobot system. 2000.

[22] M Bernardine Dias, Robert Zlot, Nidhi Kalra, and Anthony Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.

[23] Marco Dorigo and Gianni Di Caro. Ant colony optimization: a new meta-heuristic. In *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, volume 2, pages 1470–1477. IEEE, 1999.

[24] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

[25] Kurt Dresner and Peter Stone. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research*, 31:591–656, 2008.

[26] Wouter Duchateau. Multi-agent pickup-and-delivery in a distributed baggage handling system. 2021.

[27] Esra Erdem, Doga Gizem Kisa, Umut Oztok, and Peter Schüller. A general formal framework for pathfinding problems with multiple agents. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[28] EUROCONTROL. Airport collaborative decision-making. URL https://www.eurocontrol.int/concept/airport-collaborative-decision-making. Last accessed 05 October 2021.

[29] Ariel Felner, Meir Goldenberg, Guni Sharon, Roni Stern, Tal Beja, Nathan Sturtevant, Jonathan Schaeffer, and Robert Holte. Partial-expansion a* with selective node generation. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[30] 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 *Tenth Annual Symposium on Combinatorial Search*, 2017.

[31] Vanessa Frias-Martinez, Elizabeth Sklar, and Simon Parsons. Exploring auction mechanisms for role assignment in teams of autonomous robots. In *Robot Soccer World Cup*, pages 532–539. Springer, 2004.

[32] Hartmut Fricke and Michael Schultz. Improving aircraft turn around reliability. In *Third International Conference on Research in Air Transportation*, pages 335–343, 2008.

[33] Margaretha Gansterer and Richard F Hartl. Centralized bundle generation in auction-based collaborative transportation. *Or Spectrum*, 40(3):613–635, 2018.

[34] Jesús García, Antonio Berlanga, José M Molina, and José R Casar. Optimization of airport ground operations integrating genetic and dynamic flow management algorithms. *AI Communications*, 18(2): 143–164, 2005.

[35] David E Goldberg and John Henry Holland. Genetic algorithms and machine learning. 1988.

[36] J-B Gotteland and Nicolas Durand. Genetic algorithms applied to airport ground traffic optimization. In *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*, volume 1, pages 544–551. IEEE, 2003.

[37] Weian Guo, Ping Xu, Zhen Zhao, Lei Wang, Lei Zhu, and Qidi Wu. Scheduling for airport baggage transport vehicles based on diversity enhancement genetic algorithm. *Natural Computing*, 19(4):663–672, 2020.

[38] Bradford Heap and Maurice Pagnucco. Sequential single-cluster auctions for robot task allocation. In *Australasian Joint Conference on Artificial Intelligence*, pages 412–421. Springer, 2011.

[39] Bradford Heap and Maurice Pagnucco. Repeated sequential single-cluster auctions with dynamic tasks for multi-robot task allocation with pickup and delivery. In *German Conference on Multiagent System Technologies*, pages 87–100. Springer, 2013.

[40] J Holland. Genetic algorithms scientific american. 1992.

[41] Wolfgang Hönig, TK Satish Kumar, Liron Cohen, Hang Ma, Hong Xu, Nora Ayanian, and Sven Koenig. Multi-agent path finding with kinematic constraints. In *Twenty-Sixth International Conference on Automated Planning and Scheduling*, 2016.

[42] Wolfgang Hönig, Scott Kiesel, Andrew Tinka, Joseph Durham, and Nora Ayanian. Conflict-based search with optimal task assignment. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2018.

[43] Wolfgang Hönig, James A Preiss, TK Satish Kumar, Gaurav S Sukhatme, and Nora Ayanian. Trajectory planning for quadrotor swarms. *IEEE Transactions on Robotics*, 34(4):856–869, 2018.

[44] Xiao-Bing Hu and Ezequiel Di Paolo. An efficient genetic algorithm with uniform crossover for the multi-objective airport gate assignment problem. In *Multi-objective memetic algorithms*, pages 71–89. Springer, 2009.

[45] Luke Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. *AAAI/IAAI*, 2002:468–475, 2002.

[46] IATA. Airside infrastructure.

[47] Vagner Juraj, Jenčová Edina, and Szabo Stanislav. Optimization of the aircraft ground handling process. *Repüléstudományi Közlemények*, 30(2):131–138, 2018.

[48] Jorick Kamphof. Distributed multi-agent control for airport apron operations in a novel concept of autonomous engine-off taxiing (literature study), 2021.

[49] Mokhtar M Khorshid, Robert C Holte, and Nathan R Sturtevant. A polynomial-time algorithm for non-optimal multi-agent pathfinding. In *Fourth Annual Symposium on Combinatorial Search*, 2011.

[50] Sven Koenig, Craig A Tovey, Xiaoming Zheng, and Ilgaz Sungur. Sequential bundle-bid single-sale auction algorithms for decentralized control. In *IJCAI*, pages 1359–1365, 2007.

[51] Sven Koenig, Pinar Keskinocak, and Craig Tovey. Progress on agent coordination with cooperative auctions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

[52] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[53] Michail G Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton J Kleywegt, Sven Koenig, Craig A Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, volume 5, pages 343–350. Rome, Italy, 2005.

[54] Jiaoyang Li, Pavel Surynek, Ariel Felner, Hang Ma, TK Satish Kumar, and Sven Koenig. Multi-agent path finding for large agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7627–7634, 2019.

[55] Ryan J Luna and Kostas E Bekris. Push and swap: Fast cooperative path-finding with completeness guarantees. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[56] Hang Ma and Sven Koenig. Optimal target assignment and path finding for teams of agents. *arXiv preprint arXiv:1612.05693*, 2016.

[57] Hang Ma, Daniel Harabor, Peter J Stuckey, Jiaoyang Li, and Sven Koenig. Searching with consistent prioritization for multi-agent path finding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7643–7650, 2019.

[58] Hang Ma, Wolfgang Hönig, TK Satish Kumar, Nora Ayanian, and Sven Koenig. Lifelong path planning with kinematic constraints for multi-agent pickup and delivery. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7651–7658, 2019.

[59] Vijini Mallawaarachchi. Introduction to genetic algorithms —including example code, 2017. URL https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3. Last accessed 4 December 2021.

[60] Seyedali Mirjalili. Genetic algorithm. In *Evolutionary algorithms and neural networks*, pages 43–55. Springer, 2019.

[61] Asunción Mochón and Yago Sáez. *Understanding auctions*. Springer, 2015.

[62] Seyedmirsajad Mokhtarimousavi, Danial Talebi, and Hamidreza Asgari. A non-dominated sorting genetic algorithm approach for optimization of multi-objective airport gate assignment problem. *Transportation Research Record*, 2672(23):59–70, 2018.

[63] Ranjit Nair, Takayuki Ito, Melind Tambe, and Stacy Marsella. Task allocation in robocup rescue simulation domain.

[64] Ernesto Nunes and Maria Gini. Multi-robot auctions for allocation of tasks with temporal constraints. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 29, 2015.

[65] 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(04):1950020, 2019.

[66] Silvia Padrón, Daniel Guimarans, Juan José Ramos, and Salma Fitouri-Trabelsi. A bi-objective approach for scheduling ground-handling vehicles in airports. *Computers & Operations Research*, 71:34–53, 2016.

[67] Charlie Page. The aircraft turnaround: What goes on between flights, 2019. URL https://thepointsguy.com/guide/the-aircraft-turnaround-what-goes-on-between-flights/. Last accessed 08 October 2021.

[68] Mike Phillips and Maxim Likhachev. Sipp: Safe interval path planning for dynamic environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 5628–5635. IEEE, 2011.

[69] Spyros Polydorou. A learning-based approach for distributed airport surface movement operations (literature study).

[70] Cédric Pralet and Gérard Verfaillie. Time–dependent simple temporal networks: Properties and algorithms. *RAIRO-Operations Research*, 47(2):173–198, 2013.

[71] Paolo Rizzo. Auction-based task allocation for online pickup and delivery problems with time constraints and uncertainty. 2021.

[72] Malcolm Ryan. Constraint-based multi-robot path planning. In *2010 IEEE International Conference on Robotics and Automation*, pages 922–928. IEEE, 2010.

[73] Malcolm Ross Kinsella Ryan. Exploiting subgraph structure in multi-robot path planning. *Journal of Artificial Intelligence Research*, 31:497–542, 2008.

[74] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 1*, pages 69–76, 2002.

[75] Schiphol. 2 april 2020 - update: Schiphol zet volgende stap naar kern schiphol en sluit pieren. URL https://nieuws.schiphol.nl/schiphol-blijft-open-met-fors-kleinere-operatie/? Last accessed 23 November 2021.

[76] Schiphol. Safety & security pocket guide (version april 2017). Technical report, 2017.

[77] Schiphol. Schiphol airport CDM operations manual. Technical report, 2019.

[78] Schiphol. Traffic review 2020. Technical report, 2020.

[79] Schiphol. Schiphol regulations. Technical report, 2020.

[80] Schiphol. Sustaining your world vision and strategy towards the most sustainable airports. Technical report, 2020.

[81] Schiphol. An autonomous airport in 2050, 2021. URL https://www.schiphol.nl/en/innovation/blog/an-autonomous-airport-in-2050/. Last accessed 20 December 2021.

[82] Michael Schmidt. A review of aircraft turnaround operations and simulations. *Progress in Aerospace Sciences*, 92:25–38, 2017.

[83] Michael Schmidt, Philipp Nguyen, and Mirko Hornung. Novel aircraft ground operation concepts based on clustering of interfaces. Technical report, SAE Technical Paper, 2015.

[84] Guni Sharon, Roni Stern, Meir Goldenberg, and Ariel Felner. The increasing cost tree search for optimal multi-agent pathfinding. *Artificial Intelligence*, 195:470–495, 2013.

[85] Guni Sharon, Roni Stern, Ariel Felner, and Nathan R Sturtevant. Conflict-based search for optimal multi-agent pathfinding. *Artificial Intelligence*, 219:40–66, 2015.

[86] Alexei Sharpanskykh. Agent-based modelling and simulation in air transport lecture: Introduction to agents and multiagent systems. emergence., 2021.

[87] Kaveh Sheibani. Scheduling aircraft ground handling operations under uncertainty using critical path analysis and monte carlo simulation: Survey and research directions. *International Journal of Business Strategy and Automation (IJBSA)*, 1(1):37–45, 2020.

[88] David Silver. Cooperative pathfinding. *Aiide*, 1:117–122, 2005.

[89] Reid G Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on computers*, 29(12):1104–1113, 1980.

[90] Marius M Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research*, 35(2):254–265, 1987.

[91] Trevor Standley. Finding optimal solutions to cooperative pathfinding problems. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, 2010.

[92] Roni Stern, Nathan R Sturtevant, Ariel Felner, Sven Koenig, Hang Ma, Thayne T Walker, Jiaoyang Li, Dor Atzmon, Liron Cohen, TK Satish Kumar, et al. Multi-agent pathfinding: Definitions, variants, and benchmarks. In *Twelfth Annual Symposium on Combinatorial Search*, 2019.

[93] Nathan R. Sturtevant. Moving ai lab. URL https://movingai.com/index.html. Last accessed 05 November 2021.

[94] Pavel Surynek. Towards optimal cooperative path planning in hard setups through satisfiability solving. In *Pacific Rim International Conference on Artificial Intelligence*, pages 564–576. Springer, 2012.

[95] EUROCONTROL Airport CDM Team. Airport CDM implementation – the manual. Technical report, 2017.

[96] Dusan Teodorovic, Panta Lucic, Goran Markovic, and Mauro Dell'Orco. Bee colony optimization: principles and applications. In *2006 8th Seminar on Neural Network Applications in Electrical Engineering*, pages 151–156. IEEE, 2006.

[97] Future travel experience. Royal schiphol group building a roadmap towards a "fully autonomous airside operation" in 2050, 2021. URL https://www.futuretravelexperience.com/2021/03/royal-schiphol-group-building-a-roadmap-towards-a-fully-autonomous-airside-operation-in-2050/?fbclid=IwAR0IUOGOjTKdMBoQpEtQ0yxEc68tHqnhTnN5IGVdmIqWt31vhx4WmKFO_qc. Last accessed 20 December 2021.

[98] Gwo-Hshiung Tzeng and Yuh-Wen Chen. The optimal location of airport fire stations: a fuzzy multi-objective programming and revised genetic algorithm approach. *Transportation Planning and Technology*, 23(1):37–55, 1999.

[99] Pim van Leeuwen and Cees Witteveen. Temporal decoupling and determining resource needs of autonomous agents in the airport turnaround process. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 185–192. IEEE, 2009.

[100] Cheng-Lung Wu. Monitoring aircraft turnaround operations–framework development, application and implications for airline operations. *Transportation Planning and Technology*, 31(2):215–228, 2008.

[101] Cheng-Lung Wu and Robert E Caves. Modelling and simulation of aircraft turnaround operations at airports. *Transportation Planning and Technology*, 27(1):25–46, 2004.

[102] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. *AI magazine*, 29(1):9–9, 2008.

[103] Jingjin Yu and Steven M LaValle. Planning optimal paths for multiple robots on graphs. In *2013 IEEE International Conference on Robotics and Automation*, pages 3612–3617. IEEE, 2013.

[104] Jingjin Yu and Steven M LaValle. Structure and intractability of optimal multi-robot path planning on graphs. In *Twenty-Seventh AAAI Conference on Artificial Intelligence*, 2013.

[105] Hui Zhao and Liqin Cheng. Ant colony algorithm and simulation for robust airport gate assignment. *Mathematical Problems in Engineering*, 2014, 2014.