

Forecasting Aircraft Stand Snow Removal Capacity

A Case Study at Amsterdam Airport Schiphol

MSc Thesis Transport, Infrastructure and Logistics
Lara de Geus

Delft University of Technology



Forecasting Aircraft Stand Snow Removal Capacity

A Case Study at Amsterdam Airport Schiphol

by

Lara de Geus

to obtain the degree of Master of Science
at the Delft University of Technology

Student number:	4965868	
Project Duration:	November, 11, 2024 - May 16, 2025	
Thesis committee:	Dr. J. M. (Jaap) Vleugel	TU Delft, Chairmain
	Dr. J. A. (Jan Anne) Annema	TU Delft, Supervisor
	Ir. M. B. (Mark) Duinkerken	TU Delft, Supervisor
	J. (Jop) Bolsius, Msc.	Schiphol, Supervisor
Faculty:	Faculty of Civil Engineering	



Preface

This document marks the completion of my Master's program in Transport, Infrastructure and Logistics at the Delft University of Technology. It also signifies the end of my time as a student in Delft, closing a chapter that I have thoroughly enjoyed.

My fascination with the aviation sector began during an elective course I took as part of my master's program. This interest led me to connect with Royal Schiphol Group, where I had the opportunity to undertake my thesis project. When I first received the topic, *forecasting snow removal at aircraft stands*, I couldn't help but wonder, "How big can this really be?" Little did I know that the scale and complexity of the winter operations at Schiphol would leave me truly amazed, and it is precisely what made the past few months so enjoyable and rewarding. The enthusiasm and support from everyone involved in the winter organization, combined with their genuine interest in my research, constantly motivated me and provided a lasting sense of fulfillment throughout the project.

First and foremost, I would like to express my deepest gratitude to Jop, my mentor at Schiphol, for making this project an unforgettable experience. Your enthusiasm and dedication truly made a difference and your extensive knowledge and insights significantly shaped my understanding of the project. From exploring airside operations together while driving between aircraft to visiting the Tower to observe their role in winter operations, every moment was invaluable. The highlight, without a doubt, was participating in the real snow operation on January 5, where I had the unique opportunity to experience my research topic firsthand.

I am also sincerely grateful to all my APM colleagues at Schiphol for creating such a welcoming and supportive environment, making my time there truly enjoyable.

A special thanks goes to my university supervisors Jaap Vleugel and Jan Anne Annema, for their invaluable guidance and support throughout this project. Jaap, your creativity and fresh perspective encouraged me to think beyond conventional solutions. Jan Anne, your thoughtful feedback greatly enhanced the academic quality of my research. I also truly appreciate your flexibility and prompt responses, which greatly facilitated the progress of my work.

Lastly, I would like to thank my family and close friends for their support and encouragement throughout this journey. Your interest in my project and the enjoyable distractions made the process all the more pleasant.

Enjoy reading it!

*Lara de Geus
Delft, May 2025*

Summary

With the growing demand for air travel, minimizing operational downtime at airports has become increasingly critical for maintaining efficiency. Winter weather is a significant contributor to such disruptions, often resulting in flight delays, cancellations, and substantial economic losses. This underscores the importance of effective winter maintenance strategies, including aircraft de-icing and snow removal from runways, taxiways, and aircraft stands. While aircraft de-icing and runway and taxiway clearance have been widely studied, snow removal at aircraft stands has received limited attention, despite being critical nodes for boarding, disembarking, and baggage handling. Insufficient cleaning can delay inbound and outbound flights, reducing overall airport capacity. Gaining a deeper understanding of aircraft stand snow removal performance is therefore essential to improve winter resilience at airports.

This study aims to quantify aircraft stand snow removal capacity and develop a data-driven forecasting model to predict cleaning performance under varying operational and environmental conditions, using Amsterdam Airport Schiphol as a case study. In doing so, it addresses the following research question:

"How can a data-driven forecasting model be designed to improve the estimation of aircraft stand snow removal capacity and enhance airport capacity planning during snowfall?"

The research followed the Define-Measure-Analyze-Design-Verify (DMADV) methodology, a structured approach derived from Lean Six Sigma that is particularly suited for developing new processes or products. The Define phase introduced the research problem and identified key gaps in the literature, notably the limited focus on aircraft stand snow removal, particularly in relation to airport capacity during winter weather. Existing models also lack standardized snowfall classifications, limiting the comparability of results. Furthermore, this study introduces a novel application of ground radar data by analyzing the movements of specialized ground vehicles rather than aircraft. In addition to identifying these knowledge gaps, the literature review outlined six main categories of factors influencing winter maintenance performance: preparation, resources, airport-specific factors, environmental influences, operational efficiency, and human factors. These factors and their interdependencies form the basis for assessing current snow removal performance at aircraft stands.

Following the problem definition, the Measure phase analyzed current aircraft stand cleaning operations at Schiphol Airport. As shown in Figure 4.11, the process was broken down into its chronological phases, highlighting key preparatory and operational activities.

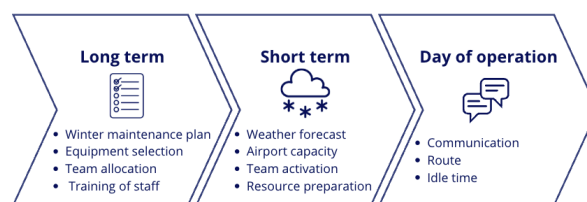


Figure 1: Chronological phases of snow removal process

The analysis provided insight into how airport capacity is determined prior to a snow event to ensure safe and manageable airport operations. Hourly airport capacity is set by the most limiting factor of three factors: runway availability, de-icing capacity, or aircraft stand availability. Airlines are advised to cancel flights accordingly. While Schiphol's runway and de-icing capacities are relatively well understood and supported by forecasting tools, stand cleaning capacity remains poorly quantified and is often overlooked in strategic discussions. This can lead to overestimated inbound capacity, resulting in queues, delays, and unplanned cancellations, highlighting the need for data-driven investigation of aircraft stand snow removal capacity.

To support this, insights from both the literature review and current state analysis were combined to develop a swimlane and causal diagram, identifying capacity-defining events and key influencing variables. These diagrams formed the conceptual foundation for a classification algorithm that categorizes historical radar tracking data from each cleaning team into one of three operational phases, cleaning, traveling, or idle.

The classified phases enabled cleaning capacity estimation and performance analysis, focusing on team availability, task durations, routing patterns, and overall capacity. Key findings included:

- On 50% of snow operation days, one fewer team than planned was available.
- Idle time varied considerably; a 35-minute break per 8-hour shift was assumed.
- Wide-body stands required longer cleaning times than narrow-body stands.
- Consecutive cleaning tasks within the same bay significantly reduced travel time compared to *cross-bay* operations.
- Although efforts were made to limit *cross-bay* movements, the ratio of *within-bay* to *cross-bay* has remained stable at 55%–45% since 2017.
- Cleaning time was the dominant component of the total cleaning cycle.
- Capacity (cleaning cycles/per hour) showed a right-skewed distribution, peaking at 2-3 cycles per hour.

In the Analyze phase, the relationship between key variables and aircraft stand cleaning capacity was systematically examined. Minute-level KNMI data were used to incorporate weather variables alongside operational factors identified in the Measure phase and clear classifications were applied to weather conditions. The analysis revealed that only snow depth (low or high) significantly affected cleaning capacity, with low accumulation (< 50 mm) associated with increased capacity. Other weather variables showed no significant impact, likely due to limited variability across the observed events. Among operational factors, stand type and route type significantly influenced cleaning capacity.

These results formed the basis for defining scenario-specific capacities. However, not all scenarios with combined variables proved statistically significant, for example, no clear capacity difference was observed between Nabo and Wibo stands for *cross-bay* routes. Based on these insights, four distinct capacity scenarios were defined to reflect key environmental and operational conditions. In addition, three weather forecast uncertainty levels, Low, Middle, and High, were established. The Middle scenario serves as the baseline, while the Low and High scenarios represent more favorable and more severe weather conditions, respectively. The final capacity scenarios are shown in Table 1.

Stand	Route	Snow	Capacity		
			Low	Middle	High
Nabo	within-bay	Low	7.2	4.9	3.2
Nabo	within-bay	High	5.7	3.7	1.9
Nabo	cross-bay	Low	3.3	2.2	1.8
Nabo	cross-bay	High	3.3	2.2	1.8
Wibo	within-bay	Low	5.5	3.4	2.0
Wibo	within-bay	High	5.5	3.4	2.0
Wibo	cross-bay	Low	3.3	2.2	1.8
Wibo	cross-bay	High	3.3	2.2	1.8

Table 1: Categorized capacities

The forecasting model was designed as a simulation-based tool that replicates aircraft stand snow removal operations under varying environmental and operational conditions. The design, visualized using a flow diagram (Figure 9.1), illustrates how time-dependent constraints, such as capacity limits, team availability, and stand suitability, interact to determine cleaning assignments and queue formation.

The model uses two types of adjustable input variables:

- **Situation-specific inputs**, that vary per snow event (exemplified in Table 2a)

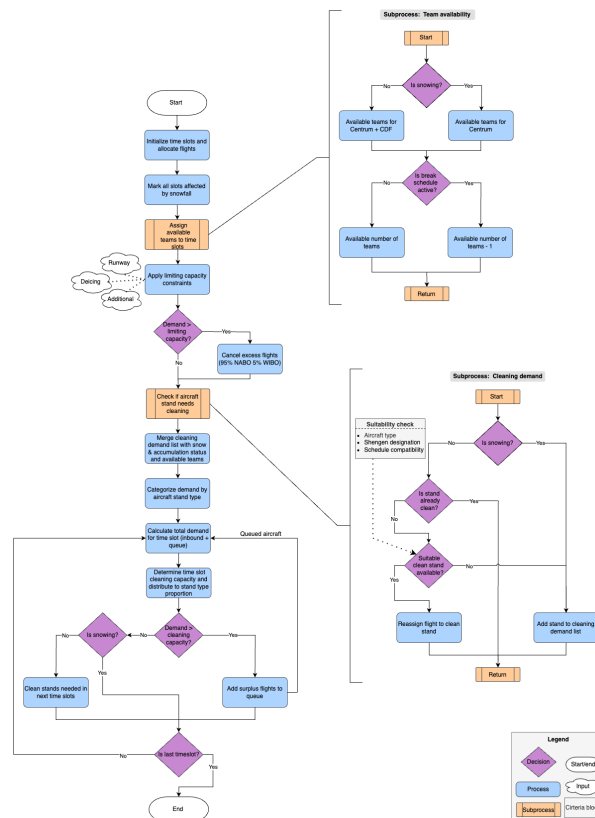


Figure 2: Design

- **Operational inputs**, reflecting structural process assumptions (Table 2b).

The model processes scheduled flight data and dynamically adjusts capacity per time slot based the Nabo/Wibo distribution, route type ratio, and snow depth level. Cleaning demand is compared against available capacity, flagging time slots where demand exceeds capacity and queues emerge, highlighting potential delays and guiding decision-making.

Variable	Values
Scenario	Middle
Time window of forecasted snow	03:00 - 11:00
Time window of forecasted accumulation levels	03:00 - 05:00; low 05:00 - 10:00; high 10:00 - 11:00; low
Number of teams active at centrum	4
Number of teams active at CDF	2
Runway capacity restriction	03:00 - 06:00; C 06:00 - 07:00; D 07:00 - 08:00; E 08:00 - 09:00; D 09:00 - 10:00; E 10:00 - 11:00; D
Deicing capacity restriction	07:00 - 10:00; 12 10:00 - 11:00; 24 11:00 - 15:00; 26
Additional capacity restriction	-

(a) Initial user-defined inputs virtual situation

Variable	Values
Time slot minutes	30 minutes
Maximal Inbound capacity per runway availability scenario	E: 10 flights/hour, D: 17 flights/hour, C: 35 flights/hour, B: 68 flights/hour
Runway clean time	40 minutes
Handling time type	Nabo: 50 minutes, Wibo: 75 minutes
Break duration	35 minutes
Start time of first break	after 3 hours
Shift duration	8 hours
Route type weights	within-bay: 55%, Cross-bay: 45%
Situation-specific capacity	See Table 6.7

(b) Inputs reflecting operational assumptions

Table 2: Model inputs

The forecasting tool is demonstrated using the virtual snow scenario discussed during the sector briefing training in November 2024. While the session concluded that runway and deicing capacity would be the primary constraints between 06:00 and 15:00, and that stand capacity would not present a limitation, the model results suggest otherwise (Figure 3). Queues begin forming around 06:30, with congestion intensifying after 10:00, posing a significant risk to operational continuity, including delays and potential cancellations. This supports the conclusion from the Measure phase that limited data-driven insight into stand cleaning capacity leads to its underestimation in strategic planning, risking overestimated inbound capacity and resulting queues.

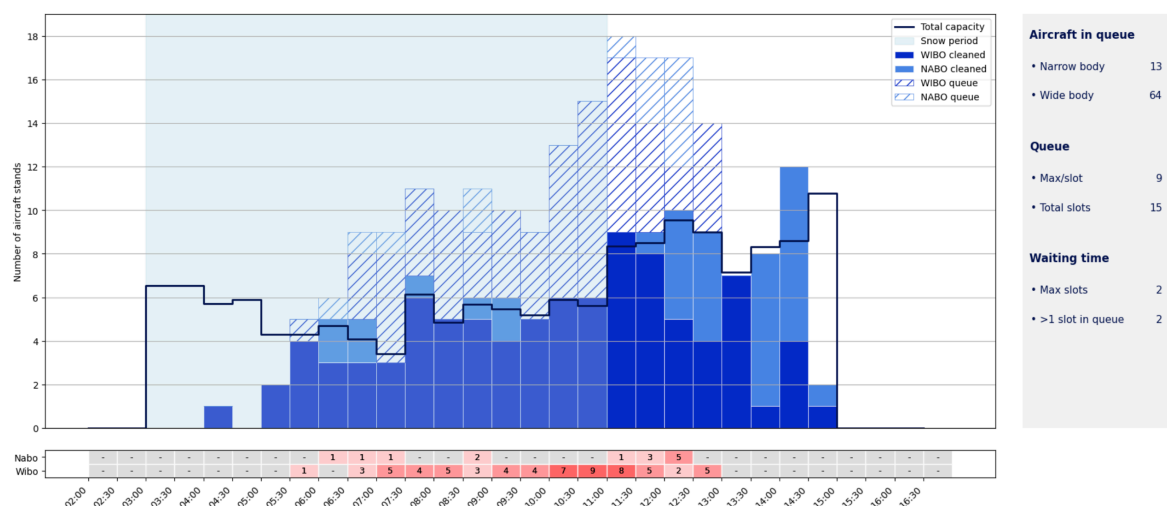


Figure 3: Model output base scenario

The emergence of queues when stand cleaning capacity is disregarded underscores the strategic importance of the developed model as a decision-support tool. The model enables users to proactively explore and evaluate interventions aimed at reducing delays and mitigating congestion through the input variable *Additional Capacity Restriction*. For example, applying a restriction of **12 flights per hour between 10:00 and 12:00** results in congestion dissipating rapidly after 10:00, reducing both the total number of queued aircraft and the maximum queue length (Figure 4). This illustrates the tool's value for proactive scenario testing and congestion mitigation.

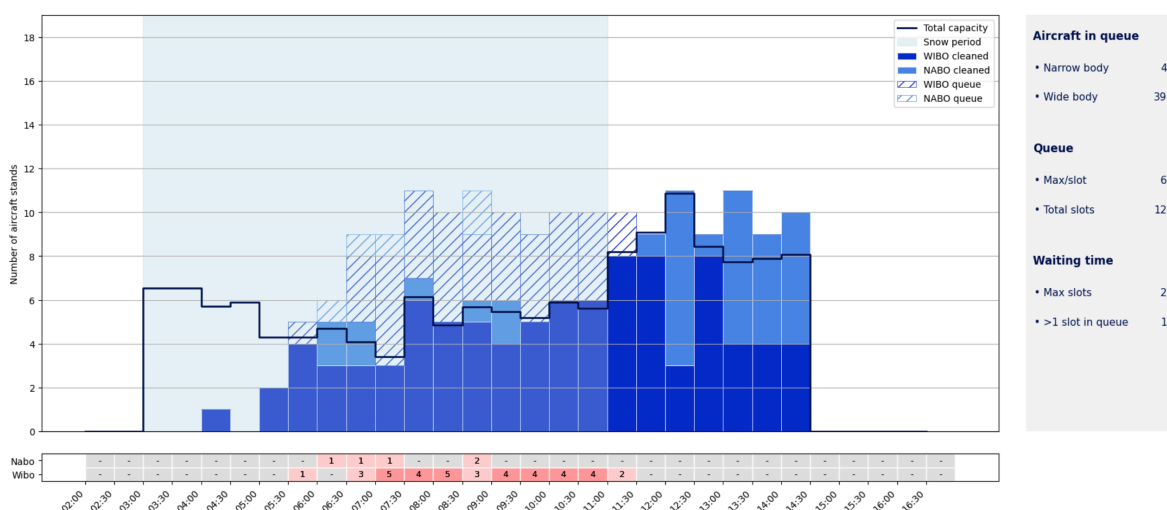


Figure 4: Model output with additional restrictions

Beyond its operational value during snow events, the forecasting model offers valuable insights for strategic optimization by simulating structural changes, such as shifting the *within-bay* versus *cross-bay* ratio or optimizing team deployment by identifying periods of over- and underutilization.

The model was validated using a historical snow event (January 5, 2025), a sensitivity analysis, and stakeholder feedback from Schiphol. Both the real-life scenario and sensitivity analysis confirmed realistic behavior. Stakeholders confirmed the model's realism and value in capturing the impact of stand cleaning capacity and supporting capacity planning during sector briefings.

Although designed for Schiphol, the model is adaptable to other airports that employ dedicated stand cleaning teams during winter operations. Schiphol-specific inputs, such as the distinction between *centrum* and CDF teams or predefined runway scenarios, can be easily adapted to suit other airport environments.

Despite offering valuable insights and a practical forecasting tool, some limitations remain. The limited weather variability in the dataset restricted the assessment of severe conditions, leading to non-significant effects for most weather-related variables, despite literature and current state analysis highlighting the importance of snow conditions. Additionally, key factors such as stand occupancy, airside traffic, and human elements (e.g. staff experience and coordination) were beyond the study's scope but likely influence performance.

Moreover, to keep the simulation operationally useful, simplifying assumptions were made. For example, route types were assigned based on a fixed Nabo/Wibo ratio rather than dynamic team positions, which may not fully capture real-world complexity.

Future research should aim to improve model realism by dynamically modeling route types using cleaning team locations and by incorporating more diverse snow event data. Integrating operational factors like stand occupancy, airside traffic, and human behavior, via advanced simulation or qualitative analysis, would further enhance model accuracy.

In conclusion, the developed forecasting model allows users to simulate stand cleaning operations by choosing the applicable scenario-based inputs. With the model, capacity bottlenecks can be anticipated, and mitigation strategies can be evaluated. It also serves as a strategic tool for scenario testing, helping to assess the impact of operational variables such as routing strategies or team availability, ultimately contributing to more robust and efficient winter operations.

Contents

Preface	i
Summary	ii
Nomenclature	xiii
1 Introduction	2
1.1 Problem Statement	2
1.2 Objective and research questions	3
1.3 Schiphol Airport	3
1.4 Research scope	4
1.5 Societal and academic relevance	5
1.6 Structure	5
2 Methodology	7
2.1 Desk research	7
2.2 Literature research	8
2.2.1 Purpose and scope of literature review	8
2.2.2 Search techniques and restriction criteria	8
2.3 Interviews	9
2.4 Process visualization	9
2.4.1 Swimlane diagram	9
2.4.2 Causal diagram	10
2.5 Data analysis	10
2.5.1 Data sources	10
2.5.2 Data analysis of key factors	11
2.6 Forecasting model	11
2.7 Verification and validation	12
3 Literature review	13
3.1 Impact of snowfall on airports	13
3.2 Factors influencing the snow removal process	14
3.2.1 Preparation	14
3.2.2 Resources	14
3.2.3 Airport specific factors	15
3.2.4 Environmental conditions	15
3.2.5 Operational efficiency	17
3.2.6 Human factors	17
3.3 Analysis of surveillance system data	18
3.4 Conclusion	18
3.5 Discussion	20
4 Current state analysis	22
4.1 Long term preparation	22
4.1.1 Winter maintenance plan	22
4.1.2 ASCT composition	22
4.1.3 ASCT cleaning area	24
4.1.4 Training of staff	24
4.2 Short-term preparation	25
4.2.1 External winter weather forecasting	25
4.2.2 Weather scenarios at Schiphol	26

4.2.3	Determining airport capacity	27
4.2.4	ASCTs activation	30
4.2.5	Preparation of equipment and resources	30
4.3	During the snow removal operation	31
4.3.1	Communication	31
4.3.2	Routes and destinations	31
4.3.3	ASCT's idle time	32
4.4	Conclusion	32
4.4.1	Swimlane diagram	33
4.4.2	Causal diagram	33
5	Current cleaning performance	36
5.1	Algorithm for capacity determination	36
5.1.1	Missing information	37
5.1.2	Detecting outliers	38
5.1.3	Algorithm verification and validation	38
5.2	Exploratory data insights on snow removal operation	39
5.2.1	Team availability	39
5.2.2	Team's Idle time	39
5.2.3	Clean time	40
5.2.4	Travel time	40
5.2.5	Cleaning cycles	43
5.2.6	Capacity	44
5.3	Conclusion	44
6	Determinants of Cleaning Capacity	48
6.1	Impact of weather parameters	48
6.1.1	Data preprocessing	48
6.2	Impact of weather variables	49
6.3	Impact of operational parameters	51
6.4	Capacity categorization	51
6.5	Conclusion	53
7	Forecasting model design	55
7.1	Simulation design framework	55
7.1.1	Design requirements	55
7.1.2	Input variables	56
7.1.3	Assumptions	56
7.1.4	Design	58
7.1.5	Model output	58
7.1.6	Model verification	60
7.2	Forecasting tool demonstration	60
7.2.1	Current situation	60
7.2.2	Strategic value of the model	61
7.3	Beyond forecasting	61
7.3.1	conclusion	63
8	Operational value assessment	65
8.1	Real life scenario analysis	65
8.2	Sensitivity analysis	66
8.3	Peer reviews	67
8.4	conclusion	68
9	Conclusion and Discussion	69
9.1	Conclusion	69
9.1.1	Answer to the main research question	69
9.1.2	Practical contribution	70
9.1.3	Scientific contribution	71
9.2	Discussion	72

9.2.1	Limitations	72
9.2.2	Recommendations	72
References		74
A Scientific paper		76
B Statistics		88
B.1	Algorithm parameter limits	88
B.2	Distribution of cleaning data	89
B.3	Supplementary results exploratory analysis	90
B.3.1	Idle time	90
B.3.2	Clean time	91
B.3.3	Travel time	91
B.3.4	Cycle duration	91
B.4	Impact of weather variables on cleaning capacity	92
C Model		94
C.1	Deicing forecasting tool	94
C.2	Model logic	94
D Code		96
D.1	Classification algorithm	96
D.2	Simulation model	107

List of Figures

1	Chronological phases of snow removal process	ii
2	Design	iv
3	Model output base scenario	v
4	Model output with additional restrictions	v
1.1	Layout Schiphol	6
2.1	Schematic overview of applied methodologies	8
2.2	Geofence polygons	12
2.3	Sample of radar data	12
4.1	Long term preparations of the snow removal process	22
4.2	Composition ASCTs (Bolsius & Scholten, 2024)	23
4.3	Special vehicles of ASCT	23
4.4	Cleaning area of ASCTs (Schiphol Operations, n.d.)	24
4.5	Short term preparations of the snow removal process	25
4.6	Snow scenarios (Schiphol Group, 2024a)	27
4.7	Airport capacity determinants	27
4.8	Winter Runway Availability Scenario (Service Owner Sneeuw en Gladheid, 2023)	29
4.9	De-icing tool	29
4.10	Sector briefing output	30
4.11	Day of snow removal operations	31
4.12	Color classifications aircraft stand	32
4.13	Swimlane diagram	34
4.14	Factors influencing the snow removal process	35
5.1	Visualization of V5 between cleaning E19	37
5.2	Number of active teams during each snow operation	39
5.3	Number of breaks for each vehicle on each snow day	40
5.4	Distribution of break durations	40
5.5	Variation of clean times per aircraft stand type	41
5.6	Visualisation of each route type	41
5.7	Variation in ASCT travel time across route type	42
5.8	Comparison of ASCT route type distributions	43
5.9	Proportional distribution of routes within <i>centrum</i> over the years	43
5.10	Distribution of cycle times across stand and route types	44
5.11	Impact of clean time versus travel time on cycle time	45
5.12	Distribution of hourly cleaning capacity	45
6.1	Distribution of snow depth across snow depth categories.	51
6.2	Distribution of snow removal capacity across aircraft stand and route types	52
7.1	User interface of situation-specific input variables	58
7.2	Simulation model logic	59
7.3	Model output virtual snow event without additional capacity restrictions	61
7.4	Model output virtual snow event with additional capacity restrictions	62
8.1	Model output of real life situation on January 5, 2025	66
9.1	Design	70

B.1	Distributions of cleaning, travel, and cycle times	90
B.2	Distribution of cleaned stands per type	91
B.3	<i>Cross bay</i> travel time across the amount of bay shifts	91
B.4	Frequency distributions of travel time share in cycle duration	92

List of Tables

1	Categorized capacities	iii
2	Model inputs	iv
2.1	Literature search engines and their yields	9
2.2	Data sources	11
3.1	Snow-related variables	17
3.2	Factors influencing snow clearing capacity of aircraft stands	19
4.1	ASCT callsigns (Bolsius & Scholten, 2024)	23
4.2	Classified weather variables	25
5.1	Descriptives of travel time across number of bay shifts	43
6.1	Weather variables	49
6.2	Aggregated variables and the corresponding methods	49
6.3	Categorized weather variables	50
6.4	Snow depth categories and their statistical significance	50
6.5	Statistical test results aircraft stand and route type	51
6.6	Statistical test results aircraft stand, route type and snow depth level combinations	53
6.7	Categorized capacities	53
7.1	User-defined situation-specific inputs	57
7.2	Inputs reflecting operational assumptions	57
7.3	Initial user-defined inputs virtual snow event	60
8.1	User-defined inputs on January 5, 2025	65
8.2	Actual performance on January 5, 2025 (Daily performance - Power BI, 2025)	66
8.3	Results per validation-scenario	67
B.1	Limits and time windows used in the algorithm	89
B.2	Boxplot statistics for parameter limit selection, in seconds	89
B.3	Descriptive statistics of ASCT's idle time	90
B.4	Descriptive statistics of clean times across stand type	91
B.5	Descriptive statistics of ASCT travel times across route type	91
B.6	Descriptives of cycle time duration	92
B.7	Correlation of weather variables with capacity and corresponding p-values	93

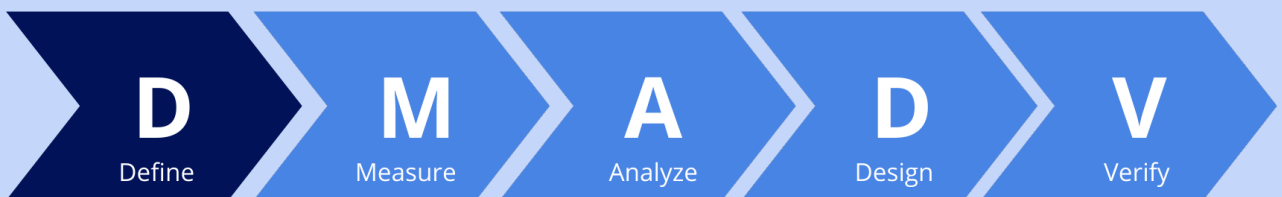
Nomenclature

Abbreviations

Abbreviation	Definition
APC	Apron Control & Planning
APOC	Airport Operations Center
ASCT	Aircraft Stand Cleaning Team
CDF	Central Deicing Facility
FMA	Flow Manager Airside
KNMI	Koninklijk Nederlands Meteorologisch Instituut
LVNL	Luchtverkeersleiding Nederland
Nabo	narrow-body
Wibo	wide-body



Define



Introduction

1.1. Problem Statement

With the growing demand for air travel, minimizing downtime at airports is becoming increasingly critical to maintain operational efficiency (Fernández et al., 2016; Pačaiová et al., 2021; Šváb et al., 2019). At Amsterdam Airport Schiphol, for example, a single day of airport closure can lead to costs of approximately €2.3 million for the airport and between €40 and €80 million for airlines (Schiphol, 2018). Winter weather is a significant contributor to operational disruptions, posing serious challenges to safety, punctuality, and cost management. These disruptions often result in flight delays, cancellations, and substantial economic losses for the aviation sector (Janic, 2009; Koščák et al., 2012; Merkert & Mangia, 2012; Pačaiová et al., 2021; Šváb et al., 2019). Beyond disruption-related costs, winter maintenance activities themselves impose significant expenses for airports and ground handlers (Koščák et al., 2020). This highlights the need for an efficient winter maintenance plan to effectively manage and mitigate the operational and financial impacts of adverse winter conditions (Pačaiová et al., 2021; Šváb et al., 2019).

To ensure operational continuity and the safety of passengers, airport personnel, and flights, various winter maintenance activities are carried out. These primarily include anti-icing and de-icing. Anti-icing involves the prevention of ice accumulation on surfaces through the application of specialized fluids that create a temporary barrier, slowing down the formation of ice. In contrast, de-icing focuses on removing ice or snow that has already accumulated. In aviation, anti-icing and de-icing are applied to both aircraft and airport pavements. Aircraft anti-icing and de-icing (hereafter referred to as aircraft de-icing) are essential for safe operations, as snow or ice on an aircraft can significantly impair its aerodynamics. However, they also result in longer gate handling times or longer taxi-out times, depending on where the aircraft is deiced, contributing to prolonged stand occupancy and departure delays (Alsalous & Hotle, 2024; Janic, 2009). Pavement anti-icing and de-icing are equally crucial, ensuring that all essential infrastructure, including runways, taxiways, taxilanes, and aircraft stands, remains fully operational for the safe movement of aircraft, vehicles, personnel, and passengers (Janic, 2009; Koščák et al., 2020; Zhang et al., 2022). Runways must maintain adequate friction levels to enable safe takeoffs and landings, taxiways and taxilanes require sufficient rigidity to support turning maneuvers, and aircraft stands must be cleared to facilitate passenger boarding and deboarding, baggage handling, and other ground operations (Koščák et al., 2020; Pačaiová et al., 2021; Šváb et al., 2019). To ensure smooth and efficient winter operations without interference between snow removal teams and airside traffic, particularly aircraft, carefully planned and coordinated winter maintenance strategies are essential.

Numerous studies have explored winter maintenance activities at airports under adverse weather conditions. As outlined in Chapter 3, these studies predominantly focus on aircraft de-icing and snow removal from runways and taxiways, while the clearing of aircraft stands has been largely overlooked. However, aircraft stand availability is a critical determinant of overall airport capacity, making efficient stand operations essential during winter weather events. This study addresses this gap by focusing specifically on snow removal from aircraft stands, using Schiphol Airport as a case study.

1.2. Objective and research questions

The primary objective of this research is to accurately quantify aircraft stand cleaning capacity under varying conditions and to design a forecasting model that predicts this capacity based on anticipated weather and operational factors. By establishing a data-driven approach to capacity estimation, this research aims to enhance airport-wide capacity planning during winter operations and support the optimization of snow removal strategies for aircraft stands.

To achieve this objective, the following main research question has been formulated:

"How can a data-driven forecasting model be designed to improve the estimation of aircraft stand snow removal capacity and enhance airport capacity planning during snowfall?"

The primary research question is addressed through the following sub-questions:

1. What is the current process for snow removal at aircraft stands?
2. What key factors are theoretically expected to influence the capacity of snow removal operations at aircraft stands?
3. What insights do historical data offer about the performance of current snow removal operations?
4. Which factors are shown by historical data to significantly affect aircraft stand snow removal capacity?
5. How can these insights be used to design a forecasting model that supports winter weather capacity planning under varying situations?
6. To what extent does the forecasting model provide accurate and usable capacity forecasts during simulated snow events?

The main research question and its sub-questions are explored using Schiphol Airport as a case study. Further details regarding the case context are provided in the following subsection. The generalisability of the findings beyond Schiphol is discussed in Section 9.1.

1.3. Schiphol Airport

This research investigates the winter maintenance strategies used at Schiphol Airport, located near Amsterdam. As the largest airport in the Netherlands and one of Europe's busiest hubs, Schiphol processed nearly 474.000 aircraft movements and welcomed over 66.5 million passengers in 2024 (Schiphol, 2024). It ranks second in Europe for direct connectivity and fourth globally as a connected hub (ACI EUROPE, 2024). Given its critical role in international air travel, ensuring smooth and efficient operations is essential to avoid delays and maintain high performance.

Schiphol Airport operates six runways, of which five serve primarily commercial traffic. During daytime operations, up to three runways are used simultaneously, based on factors such as weather, safety, and runway maintenance. At night, operations are limited to two runways. Schiphol's layout features eight aprons in the *centrum* area (A to H), as well as several additional aprons located outside *centrum*, including J, K, M, P, R, S, U, and Y. An extensive network of taxiways and taxilanes connects the aircraft stands to the runways. Figure 1.1 illustrates Schiphol's main infrastructure.

Upon landing on one of Schiphol's six runways, a commercial aircraft receives instructions from the corresponding ground controller, an LVNL (Luchtverkeersleiding Nederland) air traffic controller stationed in the tower who manages airside movements of aircraft and vehicles. The ground controller directs the aircraft to its designated parking position along taxiways and taxilanes, a process known as "taxiing". Following the instructed route, the aircraft reaches its designated parking position, referred to as an "aircraft stand". Once the aircraft is securely parked, a jet bridge (or mobile stairs for remote stands) is positioned to facilitate passenger disembarkation. Simultaneously, ground crews begin unloading baggage for transport to the terminal.

During winter maintenance operations, close coordination between Apron Control & Planning (APC, a Schiphol department in the tower responsible for towing), and snow removal teams is crucial to minimize disruptions with aircraft, as both the teams and aircraft operate within the same maneuvering area. At

Schiphol, snow removal activities on critical infrastructure involve three snow fleets (Snow Fleet 1 to 3) dedicated to clearing snow from runways, runway exits, taxiways, and taxilanes, as well as six Aircraft Stand Clearing Teams (ASCTs, numbered 4-9) responsible for removing snow from aircraft stands. These efforts aim to maintain airport capacity during snowfall as close as possible to normal levels. Inadequate coordination, however, can significantly reduce operational capacity (Bolsius & Scholten, 2024). The snow removal process from aircraft stands is further detailed in Chapter 4.

During snowfall, Schiphol's operational capacity must be proactively adjusted to reflect reduced infrastructure availability and to ensure safe, manageable flight operations. This capacity is determined during sector briefings, where the most constraining factor, either runway availability, de-icing capacity, or aircraft stand availability, defines the overall hourly airport capacity. While Schiphol's runway and de-icing capacities are relatively well understood and supported by forecasting tools, the capacity for aircraft stand snow removal remains poorly quantified and lacks a solid empirical foundation. This gap hinders accurate capacity forecasting during snow events and increases the risk of operational mismatches. In particular, underestimated stand availability can result in queues of aircraft awaiting cleared stands, causing delays and, in some cases, unforeseen flight cancellations. As such, aircraft stand snow removal capacity warrants further investigation and data-driven modeling.

1.4. Research scope

The winter season spans from November 15 to April 15, during which employees are on standby for potential snow operation.

As previously noted, Schiphol's layout comprises eight aprons in the *centrum* area (A to H), along with several additional aprons located outside *centrum*, including J, K, M, P, R, S, U, and Y. For this study, the focus is on ASCTs 4 to 9, which are responsible for cleaning the aircraft stands at aprons A, half of B, C, D, E, F, G, and the Central de-icing Facility (CDF). The other aprons either have a dedicated snow removal team or are out of service. For instance, aprons R, S, and U have dedicated cleaning teams that operate independently, without direct coordination with APC. The same applies to apron H and half of B, although ASCTs 4 to 9 may be called to assist with cleaning at these aprons if necessary. Aprons P and J (also called the CDF) are designated for de-icing operations. During snowfall, ASCT 4 is assigned to continuously clean this area. Once snowfall has stopped and aprons J and P are fully cleared and made non-slippery, ASCT 4 will split into two teams, ASCT 4 and ASCT 9, to provide additional support to the other ASCTs working at *centrum*. Aprons K and M fall under the responsibility of ASCT *Schiphol East*. Finally, apron Y is typically used for lining up, refueling and relieving snow fleets 1 to 3. However, if apron Y is needed to accommodate aircraft, ASCTs 4 to 9 will clean it accordingly.

Aircraft stands can be categorized based on the size of the aircraft they accommodate, either wide body or narrow body, and by the origin or destination of the flight, classified as Schengen or non-Schengen. For the purpose of determining snow removal capacity based on historical data, only the aircraft stand size will be taken into account, as this directly influences the time required for cleaning. While Schengen status may influence the order in which stands are cleared, since certain arriving flights must be assigned to specific stands, which may affect the route taken by a cleaning team, this operational effect is not included in the capacity analysis. However, the forecasting tool does incorporate the distinction between Schengen and non-Schengen stands. This is necessary because the model may reassign arriving aircraft to available clean stands, in which case compliance with Schengen regulations must be ensured.

The capacity determination process relies on high data granularity, as snow operations take place at most a few days per year, and may not occur at all in some winters. These events are highly irregular and difficult to predict in advance. As a result, each snow event provides valuable but scarce data, making it crucial to capture detailed, high-resolution information to maximize operational insights. Furthermore, the forecasting model developed in this research is designed to be continuously updated as new data becomes available, allowing for improved accuracy and adaptability over time.

The forecasting model is used one day before snowfall (D-1) or on the day of snowfall (D-0).

Finally, only influencing factors that can be directly retrieved from the data are included in the scope of this research. Factors that cannot be captured from the data are considered out of scope.

1.5. Societal and academic relevance

This research contributes to both academic knowledge and practical airport operations by addressing a largely unexplored but operationally critical aspect of winter maintenance: the snow removal capacity of individual aircraft stands. While most existing literature has focused on runway and taxiway clearance or aircraft de-icing, stand cleaning has received limited attention. Yet, stand availability directly influences inbound capacity, and insufficient cleaning performance can lead to operational bottlenecks and delays.

From an academic perspective, this thesis provides a novel, data-driven methodology to quantify aircraft stand cleaning capacity under varying conditions. It leverages high-resolution radar data to analyze cleaning team behavior and identifies key determinants through statistical analysis. In doing so, it addresses two notable gaps in existing literature: the limited understanding of stand-level winter maintenance performance and the utilization of ground radar data from airside vehicles for operational analysis, both of which are discussed in Chapter 3.

The societal relevance lies in its direct applicability to real-time decision-making at Schiphol Airport during snow events. Schiphol's capacity during such conditions depends on the combined effectiveness of runway clearing, de-icing operations, and stand cleaning. While the capacities of the first two are well-established, the precise capacity of aircraft stand cleaning remains uncertain. By accurately forecasting this capacity, the model developed in this research supports more informed capacity planning, enabling better coordination of operations, reducing unnecessary bottlenecks and delays, and improving the overall passenger and airline experience during winter disruptions.

1.6. Structure

This research follows the five phases of the DMADV methodology: Define, Measure, Analyze, Design, and Verify. Each chapter aligns with one of these phases and collectively contributes to the development of a data-driven forecasting model for aircraft stand snow removal capacity.

The current chapter introduces the research topic, outlining its context, objectives, and relevance, contribution to defining the problem. Chapter 2 describes the overall research approach, elaborating on the DMADV methodology and the tools and techniques applied throughout the project. Chapter 3 presents the literature review, identifying gaps in current knowledge on winter airport operations and highlighting relevant factors affecting snow removal performance, meteorological snow classifications, and applicable data analysis techniques. The former contribution to the definition of the problem, and the latter contributing to the current state analysis. Chapters 4 and 5 together constitute the Measure phase. They describe Schiphol Airport's current snow removal operations and quantify the current performance of the operations using historical data. In Chapter 6, the analysis of operational and environmental factors is conducted to identify key drivers of cleaning capacity, forming the basis for model input. Chapter 7 introduces the design of the forecasting model, while Chapter 8 assesses the model's performance through scenario analysis, sensitivity testing, and peer reviews.

Finally, Section 9 addresses the main research question, highlighting both the practical and academic contributions of this study. Furthermore, the section discusses the limitations and offers recommendations for future work. Additional details and supplementary analyses can be found in the appendices.

Throughout this report, two styles of highlighted boxes are incorporated to emphasize important information:

Statements relevant to the development of the model (input data and assumptions).

Key conclusions from chapters, sections, or subsections.



Figure 1.1: Layout Schiphol

2

Methodology

To achieve the primary objective of this study, quantifying aircraft stand cleaning capacity and developing a forecasting model to predict this capacity under varying scenarios, thereby improving winter weather airport capacity planning, the DMADV methodology was selected as the guiding framework. DMADV, which stands for Define, Measure, Analyze, Design, and Verify, is derived from the Lean Six Sigma methodology and builds on the DMAIC framework (Define, Measure, Analyze, Improve, Control). While DMAIC is typically applied to the optimization and continuous improvement of existing processes, the focus of this research lies in the design of a new, data-driven model to support capacity planning during snow events. As such, DMADV provides a more appropriate structure. It is particularly well-suited for projects involving the development of new processes, products, or models that must meet specific performance requirements, aligning closely with the goals of this research (Majumdar & Selvi, 2014).

In the **Define** phase, the introduction outlines the problem, defining the goals and scope, supported by conclusions drawn from the literature review. This phase provided the overall direction and focus for the project. The **Measure** phase involved identifying and collecting relevant data to gain insight into current snow removal operations at Schiphol Airport. The **Analyze** phase then explored the key factors on cleaning capacity in depth. In the **Design** phase, a forecasting model was developed to estimate aircraft stand cleaning capacity under various conditions, incorporating insights from the Measure and Analyze phases. Finally, the **Verify** phase involved validating the model using a real-life snow event and conducting a sensitivity analysis to assess robustness. Stakeholder feedback was incorporated to make refinements, ensuring the model's reliability for capacity planning during winter conditions. This phase ensured that the forecasting tool met the predefined requirements and could reliably support capacity planning during winter weather conditions.

Figure 2.1 provides an overview of the research phases, alongside the methodologies applied at each stage. The following sections provide an explanation of each methodology and its application within the research process.

2.1. Desk research

To address the first two sub-questions, a thorough review of documents related to winter operations at airports was conducted. Key internal resources, such as *Werkboek Sneeuw & Gladheid* and *Winteroperatie: Sneeuw & Gladheid*, were analyzed (Bolsius and Scholten, 2024; Bolsius et al., 2023; Service Owner Sneeuw en Gladheid, 2023). These findings were complemented with insights from relevant literature and interviews with KNMI and DTN. Together, these sources contribute to a thorough understanding of the current snow removal process and the key factors influencing aircraft stand cleaning operations.

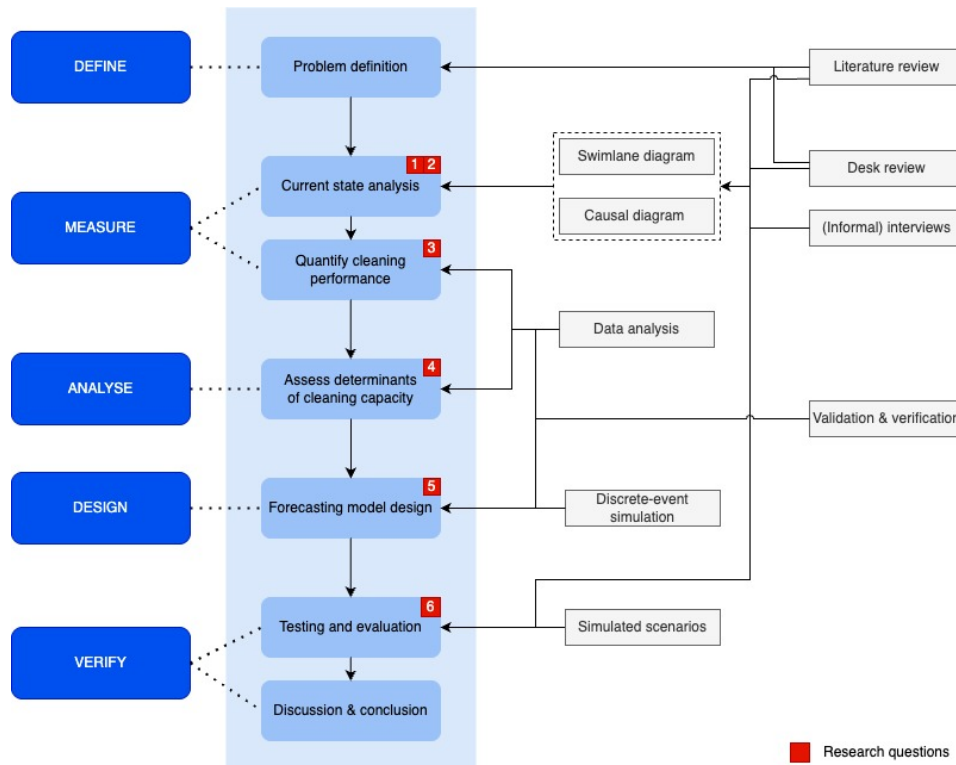


Figure 2.1: Schematic overview of applied methodologies

2.2. Literature research

A systematic literature review is conducted to synthesize research findings in a transparent, systematic and replicable manner. This process involves four key steps, as outlined by Bryman (2016). First, the purpose and scope of the literature review are defined. Secondly, search techniques are employed to identify studies relevant to the defined purpose and scope. Subsequently, the studies found are appraised and narrowed down according to predefined criteria. Finally, the remaining studies are analyzed and synthesized.

2.2.1. Purpose and scope of literature review

The literature review aimed to explore existing research on winter maintenance operations at airports, with a primary focus on the factors influencing these operations and the specific snow characteristics that impact airport functionality and maintenance strategies. Additionally, studies utilizing ground radar data from airports are reviewed to offer a foundation for the data analysis conducted in this research.

While the primary emphasis is on airport operations, relevant studies on road winter maintenance are also included where applicable to provide additional insights.

2.2.2. Search techniques and restriction criteria

A comprehensive search was undertaken to obtain studies that align with the defined purpose and scope of this research. Most of the research findings were collected through Scopus, using the search terms outlined in Table 2.1. These specified keywords enhance the replicability of the literature review. The search was limited to studies published within the last fifteen years, starting from 2010. However, a study from 2009, identified through snowballing, was considered highly relevant and has therefore been included in the analysis. Table 2.1 also presents the number of papers that the initial search yielded. However, after screening the titles and abstracts, only the articles listed in the third row of the table were deemed relevant to the research purpose. Furthermore, backward and forward snowballing techniques were applied, resulting in the final selection of articles, as detailed in Table 2.1. In Chapter 3, the final step of the literature review is carried out by analyzing and synthesizing the remaining studies.

Search engine	Search term	Papers yielded
Scopus	airport AND ("winter operations" OR "winter maintenance" OR "snow removal") [67]	Zhang et al., 2022 Pačaiová et al., 2021 Koščák et al., 2020 Šváb et al., 2019 Fernández et al., 2016 Merkert and Mangia, 2012 Myers et al., 2012 Mohammadi et al., 2024
	"winter maintenance" AND "weather conditions" AND snow AND (airport OR road) [10] "ASDE-X Data" [14]	Alsalous and Hotle, 2024 Mirmohammadsadeghi et al., 2019 Srivastava, 2011 Pan et al., 2022 Friso et al., 2018
	"A-SMGCS Data" [7]	
Snowballing	Alsalous and Hotle, 2024 Mohammadi et al., 2024 Zhang et al., 2022 Koščák et al., 2020	Adikariwattage et al., 2012 Shu-Ling et al., 2011 Janic, 2009 Klein-Paste, 2018 Koščák et al., 2012
	Šváb et al., 2019 Fernández et al., 2016	Keis, 2014 Preis and Fricke, 2020

Table 2.1: Literature search engines and their yields

2.3. Interviews

Since Schiphol relies on weather forecasts from KNMI and DTN, gaining a detailed understanding of the content and structure of the forecasts they provide is essential. Therefore, two unstructured interviews were conducted with both meteorological organizations, offering deeper insight into the type and granularity of weather data they deliver. The information obtained from these interviews formed the basis for the weather classifications presented in Table 4.2 in Subsection 4.2.1.

In addition, a sector briefing training session was attended to gain insight into the decision-making process when adverse weather is forecasted and capacity adjustments become necessary. This session involved key stakeholders including Schiphol (APOC, FMA), LVNL, KLM, and KNMI, and provided valuable perspective on how these parties coordinate their responses to winter weather disruptions and what considerations influence capacity-related decisions.

Finally, informal interviews held during winter operations meetings, along with ongoing discussions with the Schiphol Service Owner Winter Operations, contributed to mapping the current snow removal process, identifying key factors influencing cleaning capacity and developing the forecasting model.

2.4. Process visualization

Two conceptual models were developed to provide an overview of the current snow removal process and its key influencing factors. These consist of a swimlane diagram and a causal diagram, both constructed using insights from the literature review and further supported by findings from desk research and informal interviews.

2.4.1. Swimlane diagram

The swimlane diagram (also known as a cross-functional flowchart) is developed to illustrate the current aircraft stand snow removal process at Schiphol. Unlike a standard flowchart, the swimlane diagram explicitly assigns responsibilities to different departments involved in the process. Each swimlane represents a key stakeholder, ensuring that roles, interactions, and task sequences are clearly delineated. By mapping out the workflow, decision points, and communication flows, the swimlane diagram provides a comprehensive process overview. This structured visualization not only clarifies the roles of

different stakeholders in ensuring efficient and timely aircraft stand clearance but also identifies gaps in information that hinder accurate capacity calculations.

2.4.2. Causal diagram

The causal diagram illustrates the key factors influencing the snow removal process and their interdependencies. It highlights cause-and-effect relationships, helping to identify dependencies and critical variables that impact snow removal efficiency. The diagram consists of external, independent, intermediate, and dependent factors, along with arrows indicating their relationships. External factors directly impact the process but cannot be controlled or influenced by the system itself. Independent factors exist within the system, influencing other variables without being affected themselves. Intermediate factors are influenced by other factors while also impacting additional variables. Finally, the dependent factor is influenced by other factors and represents the unknown variable in this research.

Relationships between factors can be positive, where an increase in one factor leads to an increase in the next; negative, where an increase in one factor results in a decrease in the subsequent factor; or neutral, for factors that cannot exhibit an increasing or decreasing effect.

This visualization provides a structured overview of how different environmental, operational, and logistical factors interact, forming the foundation for the data analysis.

2.5. Data analysis

Based on the swimlane and causal diagrams presented in Figures 4.13 and 4.14 in Subsection 4.4, three operational states were identified as critical for calculating snow removal capacity: cleaning, traveling and idle time. Additionally, factors likely to influence these activities, and which can be captured through data analysis, were also considered. To extract these states from the data and assess the impact of influencing factors, a comprehensive data analysis was conducted, addressing the third and fourth sub-questions of this study.

2.5.1. Data sources

Table 2.2 provides an overview of the data sources used in the data analysis, including descriptions and their intended purposes within this research. To calculate cleaning capacity, the Casper dataset serves as the primary data source. While theoretically, the Casper, CISS, and VGRS datasets could each provide the necessary information to identify the relevant operational activities for capacity calculation, with VGRS and CISS offering more accessible formats, practical limitations affect their usability.

The VGRS dataset is highly incomplete, with inconsistent timestamps and a significant amount of missing cleaning tasks. The CISS data, although more consistent, is only available from 2021 onwards and occasionally fails to differentiate consecutive cleaning tasks within the same bay, as these tasks are allowed to occur without communicating with APC. Due to these limitations, Casper data is chosen as the primary data source, while CISS and VGRS data are used to verify and validate the algorithm developed to extract operational states. Specifically, when the algorithm detects a cleaning period, CISS and VGRS data help determine whether adjacent aircraft stands were cleaned simultaneously, an event that is challenging to detect through the algorithm alone.

The LHSP and CISS datasets both provide start and end times of snow removal activities; the earliest start and latest end times from these are used to define the total cleaning window. Finally, the CISS dataset supports the estimation of the cleaning operation's starting location.

Both the Casper and KNMI datasets coincidentally lack data for 2021. Although snow events occurred early that year, they coincided with the COVID-19 lockdown, which significantly disrupted normal airport operations. Consequently, the 2021 data would not be directly comparable to other years, and its absence is not considered a major limitation. In contrast, data from the years 2017 to 2025 are deemed suitable for comparison, as no substantial changes in snow removal procedures or team structures occurred during this period.

Casper data records all vehicle movements at one-second intervals, capturing each vehicle's longitude and latitude (see Figure 2.3). To make this data usable for analysis, it is combined with Schiphol's Geofence polygons (see Figure 2.2). By mapping the vehicle coordinates to the corresponding polygons,

Data source	Resolution and timeframe	Description	Purpose
Ground Radar data (Casper data) (<i>not publicly available</i>)	One-second interval, 2017-2025 (excl. 2021)	A dataset capturing aircraft and vehicle movements. For this research, the data is filtered to include vehicles V4-9 and their timestamps, latitude and longitude.	To reconstruct the movements of the ASCT coordinators and classify them as cleaning, traveling or idle time.
Geofence polygons (<i>not publicly available</i>)	-, 2025	A GeoJSON file containing segments of runways, taxiways, taxilanes, and aircraft stands, with names and positional information.	To ensure precise location identification by matching each vehicle's GPS coordinates (from Casper data) to a specific segment, such as a runway, taxiway or aircraft stand.
Central Information System Schiphol (CISS) (<i>not publicly available</i>)	Event-based, 2021-2025	An Aerodrome Operational Database, including aircraft stand assignments, departure and arrival times, and other operational data. For this research, the dataset is filtered for vehicles V4-9, including timestamps and positional information.	To support the detection of cleaning periods and to verify algorithm outcomes. Also used to determine the start and end times of the snow removal operation and the initial position of ASCTs.
VOP Gladheid Registratie Systeem (VGRS) (<i>not publicly available</i>)	Event based, 2017-2025	Snow removal and spraying tasks fulfilled by the ASCTs and sprayers, including the recorded times of execution.	To support the detection of cleaning periods and to verify algorithm outcomes.
Luchthaven status panel (LHSP) (<i>not publicly available</i>)	Event-based, 2017-2025	Start times of airport statuses, including snow status.	To identify periods of active snow removal and determine operation start and end times.
KNMI weather data (<i>not publicly available</i>)	12-second-interval, 2017-2025 (excl. 2021)	12-second measurements of 114 weather variables from 26 weather stations across Schiphol.	To assess the influence of winter weather conditions on cleaning capacity.

Table 2.2: Data sources

a new dataset is generated that provides referenceable locations for all ASCTs, specifying whether the vehicle is on a runway, taxiway, taxilane, or aircraft stand at any given second. This enriched dataset enables both the visualization of vehicle movements and the extraction of operational activities.

2.5.2. Data analysis of key factors

An algorithm was developed to classify the states of the ASCT coordinator vehicles into three categories: cleaning, traveling, or taking a break. The algorithm also accounts for and handles missing data to ensure continuity in the classification process. A detailed explanation of the algorithm, including its logic and implementation, is provided in Section 5.1. Additionally, Section 6.1.1 describes the preprocessing of weather data to enable its integration with the classified vehicle dataset for further analysis.

2.6. Forecasting model

The forecasting model was implemented as a discrete-time simulation, designed to estimate the relation between cleaning capacity, tailored to a given scenario, and the inbound flight schedule, in order to identify potential queues and operational bottlenecks. The simulation progresses in fixed time steps (e.g., 30-minute intervals), during which system states are updated to reflect changes in flight arrivals, cleaning capacity and queue formation. This approach is particularly suitable given that the sector briefing output is presented in hourly time steps, making continuous tracking of aircraft stand snow removal unnecessary. Therefore, breaking down the simulation into discrete intervals aligns well with the temporal resolution of available data and operational decision-making processes.

The simulation model operates on scenario-specific input, which users can configure to reflect upcoming

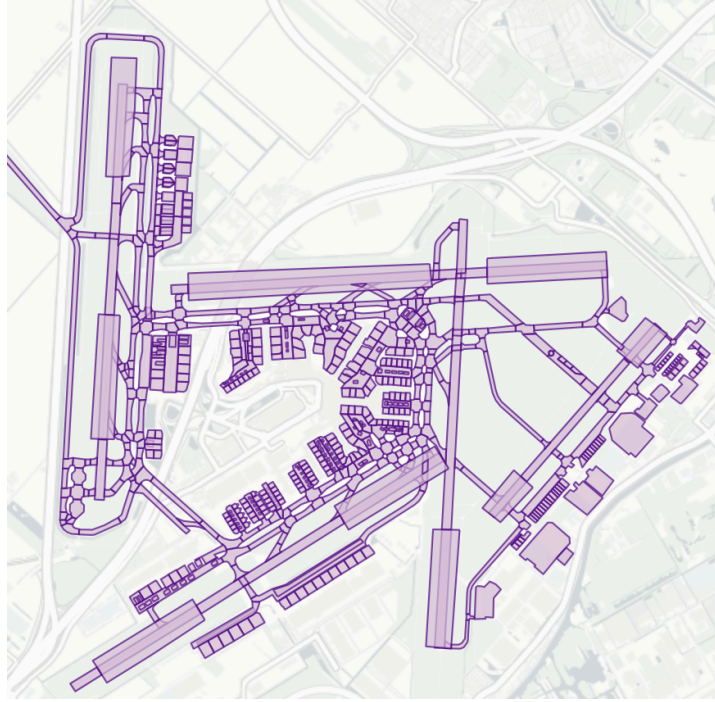


Figure 2.2: Geofence polygons

	casper_id	vehicle_registration	vehicle_callsign	timestamp	longitude	latitude
0	8096571	V5	V5	2024-11-12 00:48:46	4.741491	52.294224
1	8096571	V5	V5	2024-11-12 00:48:47	4.741491	52.294224
2	8096554	V9	V9	2024-11-12 00:48:45	4.754573	52.320591
3	8096554	V9	V9	2024-11-12 00:48:46	4.754514	52.320545
4	8096529	V6	V6	2024-11-12 00:48:45	4.741858	52.294052

Figure 2.3: Sample of radar data

ing conditions, such as expected snowfall and team availability. This flexible input structure ensures that the model output remains directly aligned with the forecasted operational and environmental context, making it suitable for real-time decision support.

A detailed explanation of the simulation framework, including design requirements, input structure, assumptions, and model logic, is provided in Section 7.1.

2.7. Verification and validation

To assess the reliability, robustness, and practical applicability of the forecasting model, a structured validation process was conducted. This process consisted of three complementary components:

- Real-life scenario validation, in which the model was applied to a historical snow event at Schiphol Airport (January 5, 2025). The model output was compared with operational expectations and evaluated in consultation with the Service Owner Winter Operations to assess realism and predictive accuracy.
- Sensitivity analysis, involving systematic variation of key input parameters to evaluate the model's responsiveness to different operational configurations.
- Peer review, in which three key stakeholder groups reviewed the model's structure, output, and usability. These sessions assessed the tool's alignment with real-world workflows and its value as a decision-support instrument during winter operations.

3

Literature review

Table 2.1 gives an overview of the reviewed literature. Using the defined search techniques and restriction criteria, 22 studies were selected for analysis. This literature review identifies knowledge gaps and establishes a foundation for addressing the sub-question *What key factors are theoretically expected to influence the capacity of snow removal operations at aircraft stands?*. To easily compare the articles, the analysis is structured into three categories: the impact of snowfall on airports, factors influencing the snow removal process, and the analysis of surveillance system data. The following sections discuss each category in detail.

3.1. Impact of snowfall on airports

Severe winter conditions at airports can cause delays, resulting in significant costs for society, airports, and air carriers (Merkert & Mangia, 2012). During such events, it is crucial for airports to maintain smooth, reliable and efficient operations. This underscores the importance of optimizing winter maintenance services to minimize disruptions and associated expenses.

Preis and Fricke (2020) emphasize that winter operations are optimized at both strategic and operational level. At strategic level, key decisions include determining the size and composition of the winter fleet and developing procedures for staff training and activation (Preis & Fricke, 2020). For example, Adikariwattage et al. (2012) focus on determining the optimal number of de-icing trucks required for gate de-icing, while Preis and Fricke (2020) estimate the ideal fleet size for standard cleaning vehicles used on aprons and specialized equipment for runways and taxiways.

At operational level, detailed plans for vehicle and personnel allocation must be formulated, encompassing the assignment of vehicles to specific areas and the design of optimal routes (Preis & Fricke, 2020). Snow and ice removal priorities are typically divided into four distinct zones: runways, taxiways, connecting ways, and aprons (Koščák et al., 2012; Zhang et al., 2022). Several studies have focused on optimizing winter maintenance strategies for these areas. For instance, Preis and Fricke (2020) designed snow removal routes, while Zhang et al. (2022) and Fernández et al. (2016) optimized the sequence of snow clearance. Additionally, Koščák et al. (2020) proposed a dynamic allocation model for snow removal vehicles. While these studies enhance winter maintenance efficiency, they overlook the inclusion of aircraft stand clearance in their models. Fernández et al. (2016) is the only study that explicitly mentions aircraft stands clearance. However, since the cleaning priority of aircraft stands is determined by the flight arrival schedule, their research focuses solely on optimizing the cleaning sequence for runways, taxiways, and aprons, without addressing the complexities of aircraft stand clearance.

Beyond optimizing winter operations, it is crucial to assess the impact of snow removal efficiency on overall airport capacity. One study by Myers et al. (2012) addresses this topic through the development of a Winter Weather Airport Capacity Model, which incorporates snow-clearing speed and weather conditions to predict departure rates during winter weather. However, this analysis is limited to runway and taxiway snow removal, and does not consider the capacity implications of snow removal on aircraft

stands, leaving a significant aspect of airside operations unexamined.

3.2. Factors influencing the snow removal process

While no study specifically examines the snow removal process for aircraft stands, the broader factors influencing winter maintenance operations are likely applicable. Therefore, this section explores the key elements affecting the efficiency and effectiveness of snow removal at airports. Table 3.2 provides an overview of these factors, categorized into six main groups: preparation, resources, airport-specific factors, environmental influences, operational efficiency, and human factors.

3.2.1. Preparation

Careful preparation for winter weather events can be done well in advance. Pačaiová et al. (2021) and Janic (2009) emphasize the importance of establishing a "snow committee" at any airport that encounters snow and ice events, regardless of their frequency. This committee plays a crucial role in ensuring smooth operations by facilitating effective communication with snow removal teams. It is also responsible for organizing preseasonal meetings for planning and preparation, as well as regular meetings throughout the winter season to address ongoing challenges. These gatherings are essential for developing a comprehensive winter maintenance document and operating procedures tailored to the unique conditions and scenarios that may arise during the season.

A robust and well-structured winter maintenance plan is essential for airports exposed to winter weather (Fernández et al., 2016; Koščák et al., 2020). According to Šváb et al. (2019), these plans must be customized to reflect the specific environmental and operational conditions of each airport. An effective plan should detail essential aspects, including the weather conditions that trigger snow removal, required staff, equipment, and other resources, areas to be cleared, designated driving routes, operational procedures, and a structured communication and information-sharing plan among all stakeholders (Janic, 2009; Merkert & Mangia, 2012; Shu-Ling et al., 2011).

Proper training is crucial for the successful execution of the winter maintenance plan. This involves not only basic training but, more importantly, regular refresher sessions to maintain readiness (Fernández et al., 2016; Merkert & Mangia, 2012; Preis & Fricke, 2020). Šváb et al. (2019) stress that even the most advanced technology is ineffective without properly trained personnel to operate and manage it efficiently. Ensuring that equipment is in optimal condition is equally important, which requires regular inspections by technicians (Pačaiová et al., 2021).

Short-term preparation measures include preventive spraying and staff readiness. Shu-Ling et al. (2011), in their study on snow removal from roadways, highlight the importance of pre-treating roads with snow-melting agents to maximizing the efficiency and effectiveness of mechanical snow removal. This proactive approach not only simplifies snow removal during storms but also prevents ice formation, extending the impact of the mechanical clearing process and maximizing overall snow removal capacity. Additionally, Šváb et al. (2019) emphasize that accurate and timely weather forecasts play a vital role in ensuring the readiness of snow removal teams.

By combining long-term planning with proactive short-term measures, airports can ensure efficient and effective responses to even the most severe winter weather events.

3.2.2. Resources

Several studies highlight the impact of the type and quantity of equipment on snow removal capacity, with four studies providing a more in-depth analysis. The algorithm developed by Zhang et al. (2022) optimizes performance by exploring various combinations of equipment types and quantities, using the snow removal capacity of different equipment teams as input. Koščák et al. (2012) focuses on optimizing the deployment of equipment sets, specifically snow blowers and sweepers with varying performance levels, by incorporating factors such as the size of the area to be cleared, various characteristics of snow, and the density of air traffic. Similarly, Koščák et al. (2020) optimizes equipment configurations but employs a different, more flexible model that integrates additional variables, including environmental impact and dynamic meteorological conditions. Lastly, Preis and Fricke (2020) estimates the minimum fleet size for winter maintenance by analyzing the area to be cleared and the performance capacity of different vehicle types, which is determined by the operating speed, the technical configuration of the

vehicle, and efficiency factors such as time and spatial utilization.

Once the optimal fleet size is determined, it is crucial to ensure that this capacity can be fully mobilized when needed. Pačaiová et al. (2021) emphasize the importance of having an adequate reserve fleet and conducting periodic inspections to ensure that all vehicles are in satisfactory technical condition at all times. Despite these preventive measures, Pačaiová et al. (2021) account for the exceptional scenario in which a repair is required during snow operations. Similarly, Šváb et al. (2019) highlight the need for on-the-spot vehicle inspection during snow events to ensure the equipment is always ready for immediate deployment.

Pačaiová et al. (2021) emphasize that, despite efforts to plan for an optimal number of staff, shortages can still occur in certain situations, potentially prolonging the overall duration of the process. Although the influence of staff availability is frequently mentioned, no research provides an in-depth analysis of this factor. Most studies focus on individual vehicles or entire fleets, making the number of staff inherently tied to the required equipment. However, the number of teams also significantly affects snow removal capacity and is not strictly tied to the equipment used. Janic (2009) states that overall system capacity depends on the number of servers and the service rates of each individual server, referring to the de-icing process in his research. This principle is also applicable to snow removal at aircraft stands, where multiple teams (servers) may operate simultaneously, each with varying service rates, influencing the overall snow removal capacity.

3.2.3. Airport specific factors

Fernández et al. (2016) highlight the impact of the flight arrival schedule on the cleaning order of aircraft stands and the routes between consecutive tasks. Janic (2009) also underscores the effect of the arrival schedule, noting that high arrival demand in the morning results in more traffic delays compared to lower arrival demand in the late afternoon, thereby illustrating the relationship between the arrival schedule, traffic volume, and its operational impact. Šváb et al. (2019) further emphasize the differences in the winter maintenance process depending on whether air traffic is present, highlighting the impact of traffic volume. Additionally, Shu-Ling et al. (2011) highlights the critical impact of traffic volume on operating speed and overall snow removal capacity, where high traffic volume negatively impacts the operating speed.

3.2.4. Environmental conditions

Weather forecasts play a crucial role in the planning and organization of winter airport maintenance. Pačaiová et al. (2021) and Šváb et al. (2019) emphasize that airports heavily rely on weather forecasts to anticipate and prepare for upcoming winter conditions. Both studies highlight the importance of well-defined winter maintenance procedures tailored to specific types of weather events, enabling airports to select the most appropriate response based on forecasted conditions. In addition, Myers et al. (2012) note that airports may respond differently to identical weather conditions due to variations in processes, operational efficiency, and infrastructure. As a result, a universal relationship between winter weather and airport capacity might not be feasible, necessitating customized protocols at each airport.

Effective use of weather forecasts allows airports to optimize the allocation and readiness of technology, equipment, and personnel, whether in anticipation of heavy snowfall or milder winter events. Forecasts must be both highly accurate and available well in advance to support timely decision-making. Moreover, airports must remain agile, continuously updating their plans in real time as weather conditions evolve, to maintain an efficient and responsive operational approach (Pačaiová et al., 2021; Šváb et al., 2019). Myers et al. (2012) further underscore the significance of weather forecasts by incorporating forecast inaccuracies into their winter weather airport capacity model, thereby improving the model's ability to predict operational impacts.

Given the need for airports to manage a variety of winter weather conditions, and with a particular focus on snowfall in the context of this research, the impact of different snow conditions is discussed in the following paragraphs. Table 4.2 summarizes the snow-related variables mentioned across the reviewed studies. It can be concluded that most studies use the same terms to distinguish snow conditions, mainly focusing on snow intensity, type, density and depth.

Types of snow

Many studies distinguish between slush, wet snow, and dry snow. Wet snow and slush typically form around 0 °C, but can also occur at lower temperatures due to the application of anti-icing and de-icing chemicals on runways (Klein-Paste, 2018). Klein-Paste (2018) further report that, when examining air-plane braking performance on snow-contaminated runways, wet snow is perceived as more slippery than an equivalent amount of slush, while dry snow provides superior braking performance compared to both. Furthermore, Myers et al., 2012 highlight that light, dry snow is easier to remove from runways than dense, wet snow of equal depth, as the latter contains a higher water content. Both Janic (2009) and Keis (2014) differentiate between dry snow and wet snow/slush when defining thresholds for initiating winter maintenance operations, noting that the threshold for wet snow and slush is significantly lower than for dry snow. Furthermore, Fernández et al. (2016) and Preis and Fricke (2020) highlight that snow type affects the operating speed of snow removal vehicles.

Snow accumulation

Snow accumulation is typically measured by snow depth (height) and volume, and is influenced by snowfall intensity, duration, and temperature. Several studies reference snow intensity using qualitative classifications such as "light," "moderate," and "heavy." However, only Janic (2009) defines specific thresholds for these categories, but limited to "light" and "heavy" snowfall. The absence of standardized criteria for snow intensity levels across studies limits the ability to directly compare findings on the impact of snowfall on airport operations. For instance, Myers et al. (2012) incorporate snow intensity levels and precipitation rate into their winter weather airport capacity model but do not provide exact boundaries for these classifications. Their correlation analysis revealed that snow intensity had a stronger association with capacity reductions than precipitation rate, underscoring snow intensity as a critical factor. Similarly, Adikariwattage et al. (2012) define five winter weather scenarios based on snow intensity, snow type, and other winter conditions, but again without specifying quantitative criteria.

Both Janic (2009) and Koščák et al. (2020) use snow accumulation, measured through snow depth or intensity, to establish thresholds for initiating snow removal activities. In addition, Koščák et al. (2020) emphasize the role of snow depth and total snowfall amount in determining the optimal deployment of winter maintenance equipment.

Finally, Shu-Ling et al. (2011) developed a snow accumulation prediction model alongside a snow removal capacity model to assess the feasibility of accomplishing the snow removal tasks within the designated time period.

Snow characteristics

Snow characteristics such as weight, density, and water content, are critical parameters for accurately forecasting winter weather and implementing effective winter maintenance strategies. Snow weight significantly influences the selection and quantity of equipment needed and plays a key role in determining the efficiency and performance of snow removal operations (Koščák et al., 2020). It is also an important factor in estimating the time needed to complete snow clearance tasks (Janic, 2009).

Myers et al. (2012) argue that the water equivalent of snow, a measure of the water content within snow, is a more relevant metric than snow depth when assessing the impact of snowfall on airport operations. For example, as discussed earlier, dry snow is easier to clear from runways than wet, dense snow of the same depth, as the latter contains a higher water content. In addition, they emphasize that snow density directly affects the rate at which snow melts.

Finally, Janic (2009) note that newly fallen snow typically exhibits much lower density than older, compacted snow, making it easier to remove but more prone to displacement by wind.

Atmospheric conditions

Temperature and wind are key factors influencing the effects of snowfall. Myers et al. (2012) incorporate temperature into their snow depth model to estimate melting rates, noting that higher temperatures accelerate snowmelt. Similarly, Shu-Ling et al. (2011) report that snow coverage tends to be thicker at lower air temperatures and thinner at higher wind speeds due to increased snow displacement. Keis (2014) further emphasize the role of air temperature in determining precipitation type, affecting whether snowfall occurs as dry snow, wet snow, or rain.

In addition, Mohammadi et al. (2024) account for both newly fallen snow and drifting snow as explanatory variables in their model, recognizing the contribution of wind-driven snow accumulation to overall surface conditions.

Factor	Reference
<i>Accumulation</i>	
depth/height	Janic, 2009, Klein-Paste, 2018, Koščák et al., 2020, Mohammadi et al., 2024, Myers et al., 2012, Shu-Ling et al., 2011
duration	Mohammadi et al., 2024
volume	Koščák et al., 2012
Intensity	Adikariwattage et al., 2012, Janic, 2009, Keis, 2014, Koščák et al., 2020, Mohammadi et al., 2024, Myers et al., 2012
<i>Characteristics</i>	
density	Janic, 2009, Koščák et al., 2012, Koščák et al., 2020, Shu-Ling et al., 2011
weight	Janic, 2009, Koščák et al., 2012, Koščák et al., 2020
water content	Myers et al., 2012, Shu-Ling et al., 2011
<i>Atmospheric conditions</i>	
Temperature	Janic, 2009, Keis, 2014, Klein-Paste, 2018, Myers et al., 2012, Shu-Ling et al., 2011
Wind	Mohammadi et al., 2024, Preis and Fricke, 2020, Shu-Ling et al., 2011
<i>Type</i>	
dry, wet, slush	Adikariwattage et al., 2012, Janic, 2009, Keis, 2014, Klein-Paste, 2018, Koščák et al., 2012, Myers et al., 2012

Table 3.1: Snow-related variables

3.2.5. Operational efficiency

Several studies mention that snow removal from runways, taxiways and aprons has priority over other airport areas. Zhang et al. (2022) develop an algorithm that dynamically adjusts cleaning priorities based on real-time conditions, including snow accumulation, proximity to critical zones, and initial area prioritization. Similarly, Fernández et al. (2016) optimize the snow removal sequence for these areas, where vehicles follow predetermined routes. Furthermore, Preis and Fricke (2020) focus on the optimal routing of snow removal vehicles, with runways prioritized over taxiways and aprons.

In contrast, the cleaning order of aircraft stands is determined by the arrival schedule of incoming flights. After one aircraft stand is cleared, the team is promptly reassigned to the next (Fernández et al., 2016). In line with this, Shu-Ling et al. (2011) emphasize the importance of prioritizing specific road segments based on predicted travel demand.

As earlier emphasized by Preis and Fricke (2020), the operating speed of snow removal machines is a crucial factor influencing snow removal capacity. The speed depends on multiple factors, including the technological capabilities of the equipment (Fernández et al., 2016), the snow conditions (Fernández et al., 2016; Preis & Fricke, 2020), and traffic flow (Shu-Ling et al., 2011).

3.2.6. Human factors

According to research by Pačaiová et al. (2021), the variability in the duration of snow clearing operations is primarily driven by human factors. The effectiveness of these operations is significantly influenced by the varying levels of experience among snow coordinators. While the development of a practical winter maintenance document serves as the foundation for the effective implementation of the airport's snow management strategy, each coordinator's practical knowledge further enhances the overall efficiency of the operations. Beyond experience, professionalism plays a key role in determining performance capacity. Mechanical equipment cannot reach optimal capacity if staff members lack

the necessary skills and competency (Merkert & Mangia, 2012; Shu-Ling et al., 2011). Moreover, time utilization factors, such as breaks and start-up times, further impact snow removal capacity (Preis & Fricke, 2020).

In addition to individual expertise and proficiency, effective collaboration among airport departments is another critical human factor affecting the overall duration of the snow clearing process (Janic, 2009; Merkert & Mangia, 2012; Pačaiová et al., 2021; Šváb et al., 2019). Proper collaboration specifically aims to prevent chaotic operations, inconsistencies in communicated information, and misaligned leadership, all of which can significantly hinder performance (Šváb et al., 2019). Moreover, effective communication between snow removal teams and the airport dispatcher is crucial to ensure that the vehicles used by the removal teams do not interfere with airside traffic (Janic, 2009). Furthermore, Merkert and Mangia (2012) highlight that a winter management plan should include adequate planning for proper communication and information exchange among all parties involved.

3.3. Analysis of surveillance system data

Several studies have used data from surveillance systems that monitor the position and identification of aircraft and vehicles within the airport movement area, using radar, multilateration, and satellite-based technologies. For instance, Srivastava (2011) enhance departure taxi time predictions by transforming and enriching raw ASDE-X data, including digitizing airport surface layouts and cleansing and validating the data. Mirmohammadsadeghi et al. (2019) use surveillance data to obtain actual values for aircraft approach speed and runway occupancy time in their runway capacity simulation model, improving its accuracy. Similarly, Pan et al. (2022) use surveillance data to simulate aircraft ground movements to detect taxiway and runway conflicts and identify capacity limitations, and Friso et al. (2018) combine radar, surveillance, and weather data (wind, visibility, temperature) to predict abnormal runway occupancy times.

While most studies focus on evaluating aircraft performance, Alsalous and Hotle (2024) use surveillance data to determine de-icing times by analyzing aircraft movements within a centralized de-icing facility. They combine surveillance and airport layout data to visualize surface movements and extract de-icing event details. A specialized algorithm analyzes individual speed profiles to distinguish between traveling, de-icing, and waiting segments.

The type of data used in these studies closely aligns with the data available for this research. However, nearly all reviewed studies focus on aircraft taxi times and runway occupancy. Notably, Alsalous and Hotle (2024) is one of the few exceptions, although it still relies exclusively on aircraft data. Their approach to extracting operational segments through a dedicated algorithm closely resembles the methodology applied in this research. Nonetheless, this study distinguishes itself by focusing on specialized ground vehicles rather than aircraft movements, offering a novel perspective on the utilization of surveillance system data.

3.4. Conclusion

The literature review underscores the critical impact of winter conditions on airport operations and the importance of optimizing snow-clearing strategies to maintain airport capacity and operational safety. The findings reveal significant gaps in current literature and provide valuable insights into the key factors and weather conditions influencing the winter maintenance operations.

Research on snow removal from runways and taxiways is relatively well-developed, with multiple studies focusing on optimizing equipment deployment, routing strategies, and vehicle allocation. In contrast, aircraft stand cleaning remains largely overlooked, with only a single study (Fernández et al., 2016) briefly mentioning it. Moreover, although snow removal efficiency plays a critical role in determining airport capacity, only one study (Myers et al., 2012) has examined this relationship, and its scope is limited to runway operations, leaving the capacity impact of stand cleaning entirely unexplored.

In addition, while most reviewed studies incorporate winter weather conditions into their models, there is no standardized approach for defining and categorizing snow conditions. Many studies refer to qualitative classifications such as "light," "moderate," or "heavy" snowfall, yet few establish clear quantitative thresholds. This lack of standardization limits the comparability and generalizability of findings. While

Factor	Scientific literature
<i>Preparation</i>	
Equipment inspection and maintenance	Pačaiová et al., 2021
Preventive spraying	Shu-Ling et al., 2011
Training of staff	Fernández et al., 2016, Merkert and Mangia, 2012, Preis and Fricke, 2020, Šváb et al., 2019
Winter maintenance plan	Fernández et al., 2016, Janic, 2009, Koščák et al., 2020, Merkert and Mangia, 2012, Pačaiová et al., 2021, Preis and Fricke, 2020, Shu-Ling et al., 2011, Šváb et al., 2019
<i>Resources</i>	
Amount of equipment	Koščák et al., 2012, Koščák et al., 2020, Preis and Fricke, 2020
Amount of staff	Janic, 2009, Merkert and Mangia, 2012, Pačaiová et al., 2021, Preis and Fricke, 2020, Shu-Ling et al., 2011
Equipment type	Janic, 2009, Koščák et al., 2012, Koščák et al., 2020, Preis and Fricke, 2020, Zhang et al., 2022
Number of teams	Janic, 2009
Repair time of equipment failure	Pačaiová et al., 2021, Šváb et al., 2019
<i>Airport specific factors</i>	
Area to be cleaned	Fernández et al., 2016, Janic, 2009, Koščák et al., 2020, Preis and Fricke, 2020, Zhang et al., 2022
Flight arrival schedule	Fernández et al., 2016, Janic, 2009
Traffic volume	Koščák et al., 2012, Shu-Ling et al., 2011
<i>Environmental influences</i>	
Weather conditions	Fernández et al., 2016, Janic, 2009, Koščák et al., 2012, Koščák et al., 2020, Myers et al., 2012, Pačaiová et al., 2021, Preis and Fricke, 2020, Shu-Ling et al., 2011, Šváb et al., 2019, Zhang et al., 2022
Weather forecast accuracy	Myers et al., 2012, Pačaiová et al., 2021, Šváb et al., 2019
<i>Operational efficiency</i>	
Cleaning priority	Fernández et al., 2016, Preis and Fricke, 2020, Shu-Ling et al., 2011, Zhang et al., 2022
Driving speed	Fernández et al., 2016, Koščák et al., 2012, Koščák et al., 2020, Shu-Ling et al., 2011, Preis and Fricke, 2020
Route between consecutive tasks	Fernández et al., 2016, Preis and Fricke, 2020
<i>Human factors</i>	
Cooperation	Janic, 2009, Merkert and Mangia, 2012, Pačaiová et al., 2021, Šváb et al., 2019
Efficient time utilization	Preis and Fricke, 2020
Experience of staff	Pačaiová et al., 2021, Shu-Ling et al., 2011
Professionalism of staff	Merkert and Mangia, 2012, Shu-Ling et al., 2011

Table 3.2: Factors influencing snow clearing capacity of aircraft stands

insights from the existing literature will inform the analysis of how weather conditions influence snow removal capacity, precise classifications will need to be developed within the scope of this research.

Another notable gap is the limited use of airport surveillance data for analyzing winter maintenance processes. While this data has been widely applied to study aircraft taxi times and operational efficiency,

its use in optimizing winter operations remains minimal. The only exception is the study by Alsalous and Hotle (2024), which leveraged surveillance data to analyze de-icing times but still focused exclusively on aircraft movements. In contrast, the present study will extend the application of surveillance data to specialized ground vehicles operating on the maneuvering area, offering a novel perspective on the application of surveillance data.

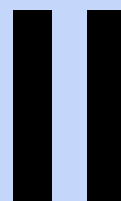
Beyond identifying these knowledge gaps, the review highlights key factors that influence winter maintenance operations, which are grouped into six primary categories: preparation, resources, airport-specific factors, environmental influences, operational efficiency, and human factors. The identified factors and their interdependencies serve as the foundation for the causal diagram presented in Figure 4.14. However, given the absence of studies specifically modeling snow removal at aircraft stands, these factors and their relationships must be adapted to accurately reflect the stand cleaning process. This adaptation is essential to address the second sub-question: ***"What key factors are theoretically expected to influence the capacity of snow removal operations at aircraft stands?"***.

3.5. Discussion

The knowledge gaps identified in this review are particularly interesting given the critical role aircraft stands play in overall airport operations. Inefficient snow removal at stands can cause significant delays, reduce airport capacity, and disrupt flight schedules. A deeper understanding of stand-specific snow removal processes enables airports to make more informed operational decisions, optimize vehicle routing and resource allocation, and ultimately minimize disruption. Therefore, developing a comprehensive understanding of snow removal capacity at aircraft stands during snowfall events is essential for maintaining operational resilience.

Since snow removal efficiency is likely affected by prevailing weather conditions, it is important to quantify their influence on the cleaning process. Establishing clear, quantitative definitions for key weather parameters will improve the predictability of snow removal performance and facilitate the comparison and generalization of findings across different airports.

To address the identified gaps, this study will adopt a data-driven approach that integrates historical ground vehicle radar data with weather observations, thereby accounting for both operational and environmental variability. The data processing methodology developed by Alsalous and Hotle (2024) will serve as a foundational reference, but will be adapted to focus specifically on specialized ground vehicles operating at aircraft stands, offering a novel extension of this approach to an underexplored aspect of winter airport operations.



MEASURE



4

Current state analysis

This chapter provides a chronological overview of the snow removal process for aircraft stands at Schiphol Airport, addressing subquestions 1 and 2: *What is the current process for snow removal from aircraft stands?* and *What key factors influence the capacity of snow removal operations at aircraft stands?*. It begins with long-term preparations, covering strategic decisions made years in advance (Section 4.1). Short-term preparations commence once snowfall is forecasted, typically around three days in advance (Section 4.2), and the day-of-operation processes focus on the real-time execution of snow removal activities (Section 4.3).

4.1. Long term preparation

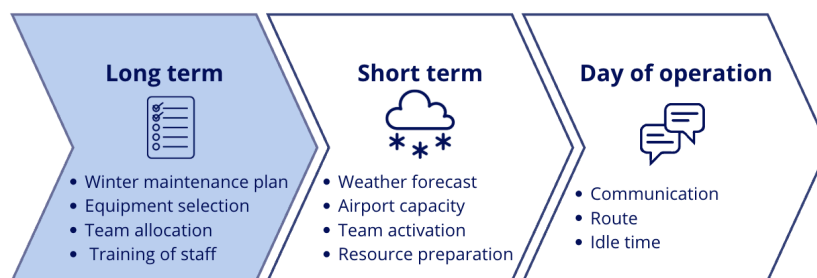


Figure 4.1: Long term preparations of the snow removal process

4.1.1. Winter maintenance plan

At Schiphol, the winter maintenance plan (Bolsius & Scholten, 2024; Service Owner Sneeuw en Gladheid, 2023) is updated annually. This plan outlines all processes and regulations related to snow removal at the Airside area and serves as a guideline for theoretical training sessions (Informal interview Service Owner Winter operations, November 2024).

4.1.2. ASCT composition

Typically, an ASCT consists of an ASCT coordinator, known as a Snow Star, who is a Schiphol employee driving a dedicated vehicle. The coordinator leads a team comprising a small and a large tractor, both equipped with front-mounted brooms, and a loader with a snowplough, as illustrated in Figures 4.2 and 4.3. Unlike the ASCT coordinator, the other vehicles are operated by contractors (Bolsius & Scholten, 2024).

The small tractor with front-mounted broom is responsible for clearing the area around the jet bridge, represented by the red-shaded area in Figure 4.2, while the other two vehicles handle the remainder of the aircraft stand. The type of equipment likely influences the cleaning speed due to variations in snow removal effectiveness. For example, the large tractor with front-mounted broom is much more efficient

for wet snow compared to the loader with snowplough. To ensure all equipment is fully operational during snowfall events, periodic technical inspections are conducted, verifying the functionality and readiness of each vehicle. Moreover, backup vehicles are available for the coordinators in case one becomes non-operational (Informal interview Service Owner Winter Operations, January 2025).

The ASCT-coordinator oversees snow clearing operations, ensuring both the effectiveness and safety of its team. The coordinator is also responsible for communicating with APC regarding task assignments and vehicle movements. The coordinator's vehicle is equipped with a transponder, providing APC with real-time operational information, including the team's callsign, current position, origin and destination (Bolsius & Scholten, 2024). Table 4.1 lists the callsigns of all teams, which are used in the algorithm described in Subsection 5.1.

In addition to the above-mentioned vehicles, a sprayer may operate either as part of the team or independently within its designated area, depending on the availability of sprayers relative to the number of ASCTs. In practice, however, the sprayer almost always operates independently, maintaining direct communication with APC (Bolsius & Scholten, 2024).

Table 4.1: ASCT callsigns (Bolsius & Scholten, 2024)

Team	Callsign
ASCT 4	V4
ASCT 5	V5
ASCT 6	V6
ASCT 7	V7
ASCT 8	V8
ASCT 9	V9

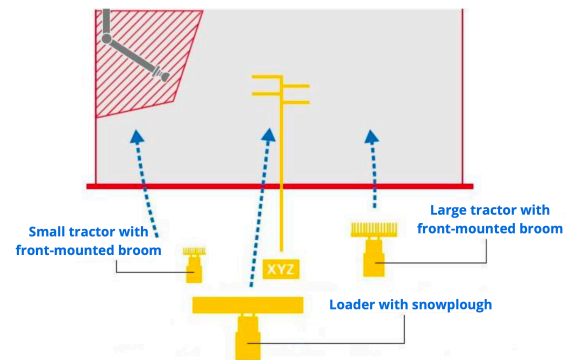


Figure 4.2: Composition ASCTs (Bolsius & Scholten, 2024)



(a) Loader with snowplough



(b) Large tractor with front-mounted brooms



(c) Small tractor with front-mounted brooms

Figure 4.3: Special vehicles of ASCT

An exception to the standard ASCT composition occurs with *ASCT 4* and *ASCT 9*, which are responsible for clearing the Central de-icing Facility (CDF). Until the end of the winter season 2023/2024, *ASCT 4* comprised one coordinator and four loaders with snowploughs. However, this configuration made it impossible for *ASCT 4* to assist *ASCTs 5-8* once snowfall had ceased and the CDF was cleared. To address this limitation, starting in the winter of 2024/2025, *ASCT 4* is reconfigured to consist of two coordinators, two loaders with snowploughs and two large tractors with front-mounted brooms. After snowfall ends and the CDF is cleared, *ASCT 4* will split into two separate teams: *ASCT 4* and *ASCT 9*. Each team will include one coordinator, one loader with snowplough and one large tractor with front-mounted broom. Although neither team will have a small tractor with front-mounted broom, the red-shaded area will still be cleared as thoroughly as possible using the available snow vehicles, ensuring the bridge remains deployable. This reconfiguration will increase the ASCT's cleaning capacity after snowfall, as *ASCT 4* and *ASCT 9* will be able to assist the other teams. However, the absence of the small tractor with front-mounted broom will likely result in longer cleaning times (Bolsius & Scholten, 2024; Bolsius et al., 2023).

4.1.3. ASCT cleaning area

As outlined in the research scope, only ASCTs 4 through 9 are included in this research. These teams are specifically assigned to snow clearance operations at *centrum* (aprons A, B (odd side), C, D, E, F, and G), and the CDF, which encompasses aprons J and P. These areas are highlighted in green in Figure 4.4. Aprons excluded from the research either have their own dedicated ASCT or are at that time out of service. However, if necessary, ASCTs 4 through 9 may assist with snow clearing at pier B (even side) or H, which are the two rightmost blue areas shown in Figure 4.4 (Aircraft Operations Schiphol, 2024; Bolsius & Scholten, 2024).

ASCT 4 is primarily designated for the continuous cleaning of the CDF. Once snowfall has ended and the CDF is completely cleared and made non-slippery, ASCT 4 will divide into two teams: ASCT 4 and ASCT 9 (Bolsius & Scholten, 2024).

After snowfall, ASCT 4 and 9 provide assistance to the other teams at *centrum*.
During light snowfall, ASCT 4 and 9 can assist the other teams intermittently.

During snowfall, apron Y is primarily used for lining up, refueling and relieving Snow Fleets 1 through 3. However, if apron Y is needed for aircraft accommodation, ASCTs 4 through 9 will ensure it is cleaned accordingly (Bolsius & Scholten, 2024). Apron Y is therefore also shown in blue in Figure 4.4, specifically as the leftmost blue area.



Figure 4.4: Cleaning area of ASCTs (Schiphol Operations, n.d.)

Aircraft stands used for passenger handling are categorized in two main types: narrow body (Nabo) and wide body (Wibo). Nabo stands have a significantly smaller surface area than Wibo stands, which likely affects the required cleaning time. In addition to size, stands also differ in their operational designation: they may be used for Schengen flights, non-Schengen flights, or be equipped to handle both.

4.1.4. Training of staff

As earlier mentioned, the winter maintenance document serves as a key reference for theoretical training sessions. Before joining the winter crew, new Snow Stars must pass both theoretical and practical exams. In addition, each year in October and November, all Snow Stars and contractors operating ASCT equipment must participate in a night training session. If snowfall is minimal or absent between November and January, an additional training session is scheduled in January or February for all Snow Stars and contractors. These training sessions provide the winter crew with valuable experience, im-

proving their familiarity with routes, vehicle coordination, team management and communication with APC (Informal interview Service Owner Winter Operations, November 2024).

4.2. Short-term preparation

The accuracy of the weather forecast determines the timeline for short-term preparations, which typically begin up to three days in advance. However, if snowfall is predicted at short notice, the reduced preparation time can disrupt the efficiency of operations.

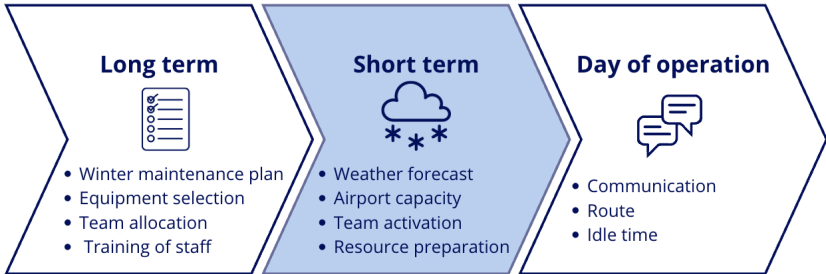


Figure 4.5: Short term preparations of the snow removal process

4.2.1. External winter weather forecasting

Schiphol primarily relies on the Koninklijk Nederlands Meteorologisch Instituut (KNMI) for weather forecasts, which provides regular updates on expected weather conditions. Additionally, Schiphol can request forecasts from DTN or consult Roadmaster, a forecasting application developed by DTN. Both KNMI and DTN base their predictions on a range of meteorological variables relevant to winter operations, including the expected start and end times of snowfall (duration), snow intensity, snow depth, temperature, wind, and visibility (Interview DTN, December 2024; Interview KNMI, January 2025). In addition to these variables, specific weather classifications are used to characterize weather conditions. The classifications relevant to this research are based on interviews with KNMI and DTN, as well as operational documents provided by Schiphol. These classifications are shown in Table 4.2 and discussed in the following subsections.

Variable	Category name	Category value
Snow Intensity (mm/h)	Light	<i>Intensity</i> < 5
	Moderate	$5 \leq \textit{Intensity} \leq 50$
	Heavy	<i>Intensity</i> > 50
Snow type	Wet	<i>Nattebol</i> $T \geq 0$
	Dry	<i>Nattebol</i> $T < 0$
Wind speed (Knots)	Light (< 3 Bf)	<i>Knots</i> < 7
	Moderate (3-6 Bf)	$7 \leq \textit{Knots} \leq 22$
	Heavy (>6 Bf)	<i>Knots</i> > 22
Visibility	BZ0 scenario 0	<i>Visibility</i> ≥ 1500
	BZ0 scenario A	$550 \leq \textit{Visibility} < 1500$ OR $200 \leq \textit{Ceiling} \leq 300$
	BZ0 scenario B	$350 \leq \textit{Visibility} < 550$ OR <i>Ceiling</i> ≤ 200
	BZ0 scenario C	<i>Visibility</i> < 350

Table 4.2: Classified weather variables

Snow intensity and accumulation

Snowfall intensity, combined with its duration, determines the total accumulation. Another key metric of accumulation is snow depth, measured by sensors that calculate the distance to the ground. However, strong winds can cause drifting snow, leading to inaccuracies in these measurements as snow dunes may form (Interview KNMI, January 2025).

Temperature and snow type

Both KNMI and DTN classify snow as either dry or wet, with temperature playing a crucial role in determining the type. In general, dry snow typically forms when temperatures are below 0°C, while wet snow occurs at or just above freezing. However, when air temperatures hover around 0°C, accurately predicting the form of snowfall becomes challenging (Interview DTN, December 2024; Interview KNMI, January 2025). Wet snow tends to accumulate only during high precipitation rates. As it melts, it transitions into slush—a mixture of snow and water. Slush can also form from the melting remnants of older dry snow.

In cases where air temperatures are below 0°C at higher altitudes but slightly above freezing near the ground, winter rain may occur, sometimes accompanied by hail. Hail accumulation can also affect ground conditions, potentially necessitating winter maintenance measures (Interview KNMI, January 2025).

The ground temperature further influences whether snow accumulates or melts upon contact. As snow builds up, it gradually cools the surface, slowing the melting process (Interview DTN, December 2024).

Wind

Wind plays a crucial role in winter weather conditions once it reaches a certain strength. Wind speeds between 3 and 4 on the Beaufort scale can already influence snowfall patterns. Dry snow, in particular, is highly susceptible to wind drift, meaning that even after it has settled, it can be blown across surfaces, forming snow dunes. In contrast, wet snow remains in place once it has accumulated, as wind only affects it while it is falling (Interview DTN, December 2024; Interview KNMI, January 2025). According to KNMI, wind speeds of 6 Beaufort or higher can lead to "snow yachting," a phenomenon where blowing snow severely reduces visibility, creating hazardous conditions (Interview KNMI, January 2025).

Wind direction is not expected to have a direct impact on cleaning speed. While it can influence the distribution of snow accumulation, particularly through the formation of snow dunes under strong wind conditions, it does not affect the overall snow removal process. This is because cleaning activities are carried out at aircraft stands situated in all wind directions across the airport. Consequently, localized accumulation patterns caused by wind are likely to be averaged out across the operation. For this reason, wind direction is not further considered in this research (Informal interview, Service Owner Winter Operations, 2025).

Visibility

During snowfall, significantly reduced visibility often indicates heavy snowfall; however, strong winds causing drifting snow can also impair visibility. Poor visibility increases travel times for snow removal teams, reducing overall operational efficiency.

Visibility categories are determined based on both horizontal visibility and cloud ceiling height, following the classifications used by Schiphol's 'Reduced View Operation' protocol (Beperkt Zicht Operatie, BZO) (Informal interview, Service Owner Winter Operations, 2025).

4.2.2. Weather scenarios at Schiphol

Until the winter of 2023/2024, a single snow scenario was implemented, in which all teams, including Snow Fleets 1 through 3 and ASCTs 4 through 8, were mobilized whenever snowfall was predicted with sufficiently high probability and accumulation. Since the winter of 2024/2025, this approach has been replaced by two distinct scenarios: 'Snow Big' and 'Snow Small'. *Snow Big* retains the procedure of the previous general scenario, mobilizing all teams. In contrast, *Snow Small* is activated for lighter snowfall or less accumulation forecasts, deploying only Snow Fleet 1 and 2 ASCTs. The snow conditions associated with each scenario are detailed in Figure 4.6. Prior to the introduction of the two scenarios, the snow conditions now associated with *Snow Big* served as the threshold for mobilizing snow fleets and ASCTs. When conditions would have fallen under the current *Snow Small* category, it was typically managed through proactive spraying or by quickly mobilizing snow teams as needed (Bolsius & Scholten, 2024; Bolsius et al., 2023; Schiphol, 2018).

Preventive spraying helps maintain the operating speed of ASCTs and enhances the efficiency of aircraft stand cleaning by preventing ice formation on the roads. It is almost always performed, except in cases when temperatures are extremely low, pavement temperatures are very low, and very dry

snow is forecasted. However, there is no specific threshold or regulation defining when spraying can be omitted (Informal interview Service Owner Winter Operations, February 2025).

	Snow Small		Snow Big	Snow Big with snow disposal
Probability [%]	30-100	10-40	41-100	60-100
Accummulation per 3 hours [cm]	< 1	≥ 1	≥ 1	≥ 5
Type	Light snowfall		Moderate snowfall	Heavy snowfall

Figure 4.6: Snow scenarios (Schiphol Group, 2024a)

4.2.3. Determining airport capacity

Schiphol's operational capacity is determined daily in sector briefings involving multiple stakeholders. In the event of forecasted snowfall, standard airport capacity is proactively adjusted based on anticipated conditions to ensure operational predictability, maintain manageable traffic flow, and avoid aircraft congestion in the airside area. These briefings take place starting one day before snowfall (D-1) and continue throughout the operation. They are led by APOC and include representatives from FMA, KLM, KNMI, and LVNL, who collectively assess the most critical bottlenecks in Schiphol's hourly inbound and outbound capacity. Based on these assessments, airlines may be advised to cancel flights in advance to align scheduled operations with the reduced capacity. The primary constraints affecting capacity during snowfall are: *runway clearing capacity, de-icing capacity or aircraft stand clearing capacity* (Figure 4.7) (Airport Operation Center Schiphol, 2024; Bolsius & Scholten, 2024).

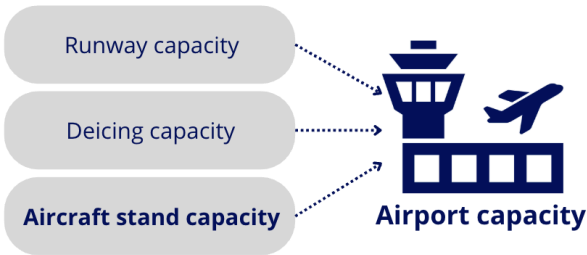


Figure 4.7: Airport capacity determinants

Runway clearing capacity

Runway clearing capacity is predefined for different snow scenarios, as illustrated in Figure 4.8, where the maximum inbound/outbound rate is indicated in red. The capacity assessment distinguishes between periods of active snowfall, categorized as light or heavy, and post-snowfall clearing, during which remaining snow is removed from the airside area to restore full operational capacity. Based on forecasted weather conditions provided by KNMI, and the following operational guideline:

The average time required to clear a single runway is approximately 40 minutes.

FMA selects the appropriate Winter Runway Availability Scenario for each hour (Airport Operation Center Schiphol, 2024; Bolsius & Scholten, 2024).

De-icing capacity

De-icing capacity is estimated using a forecasting model developed by APOC, tailored to each handling agent. A key design requirement for the aircraft stand forecasting model developed in this research is that it must align with both the logic and visual presentation of APOC's de-icing tool, ensuring consistency and ease of use (see Subsection 7.1.1). According to APOC documentation (Iris Schiphol, n.d.),

the model operates according to several key principles. The most important are summarized below; a more detailed overview is provided in Appendix C.1.

- The model operates in 15-minute time intervals, distributing demand, based on the scheduled departures from CISS flight data, across time blocks.
- Hourly de-icing capacity is entered as an average based on an equal mix of narrow body and wide body aircraft. This capacity is then divided by 4 to align with the 15-minute time intervals.
- Three scenarios (Low, Middle, High) are used to reflect weather forecast uncertainty. The Middle scenario serves as the baseline, while the Low and High scenarios represent more favorable and more severe weather conditions, respectively.
- The visual output displays bars for de-icing demand and a line for the handler's planned maximum capacity. Stacked bars differentiate fulfilled demand by aircraft type and show queued aircraft. See Figure 4.9.
- KPIs shown alongside the graph include the number of narrow body and wide body aircraft requiring de-icing, and the maximum queue size within a time block.

Figure 4.9 presents an example of the output for KLM's de-icing operation on January 5, 2025. Between 06:30 and 11:30, projected demand significantly exceeded available capacity, resulting in a queue of aircraft awaiting de-icing, visualized by the striped red bars (Airport Operation Center Schiphol, 2024)

Aircraft stand clearing capacity

In contrast, the capacity of aircraft stand snow removal is currently estimated using a simplified calculation that assumes 4 ASCTs, a maximum clearing time of 15 minutes per stand and some additional travel time between stands. This results in a theoretical capacity of 12 stands per hour during heavy snowfall. However, this calculation is not data-driven and is only available for heavy snowfall conditions (Airport Operation Center Schiphol, 2024; Bolsius & Scholten, 2024).

Operational experience indicates that during snowfall, runway or de-icing capacity is often the primary limiting factor. However, once snowfall ceases, runway operations are typically restored quickly, and not all departing aircraft require de-icing anymore, allowing for a substantial increase in inbound traffic. Despite this recovery, aircraft stand snow removal capacity is frequently overlooked, resulting in a mismatch between available runway and de-icing capacity and the readiness of aircraft stands. Consequently, aircraft are often forced to wait on taxiways until their assigned stands are cleared, leading to substantial delays and operational inefficiencies, and even unforeseen flight cancellations (Informal interview, Service Owner Winter Operations, November 2024).

This gap in capacity assessment was reinforced during the sector briefing training session in November 2024, where aircraft stand snow removal was not discussed, despite being a known bottleneck during snow operations. These findings underscore the need for a more accurate, data-driven approach to determine stand cleaning capacity and ensure its integration into operational planning and decision-making (Sector briefing training, November 2024).

Final winter weather capacity

The briefing's outcome is a graphical representation of inbound and outbound capacity over time. Figure 4.10 shows an example of the capacity assessment on the snowday of January 5, 2025. The red line in the graph represents the runway capacity, with the corresponding Winter Runway Availability Scenario labeled above it. Between 05:00 and 12:00, the inbound and outbound capacity are reduced to 20x20 instead of 35x35, which would align with Scenario C, reflecting the de-icing capacity constraint. This reduction aligns with the forecasted de-icing capacity shown in Figure 4.9. Below the red line, the primary constraint, in this case, de-icing capacity, is indicated by a label. Other factors, such as aircraft stand capacity, can also become the primary bottleneck (Airport Operation Center Schiphol, 2024, Sector briefing training, November 2024).

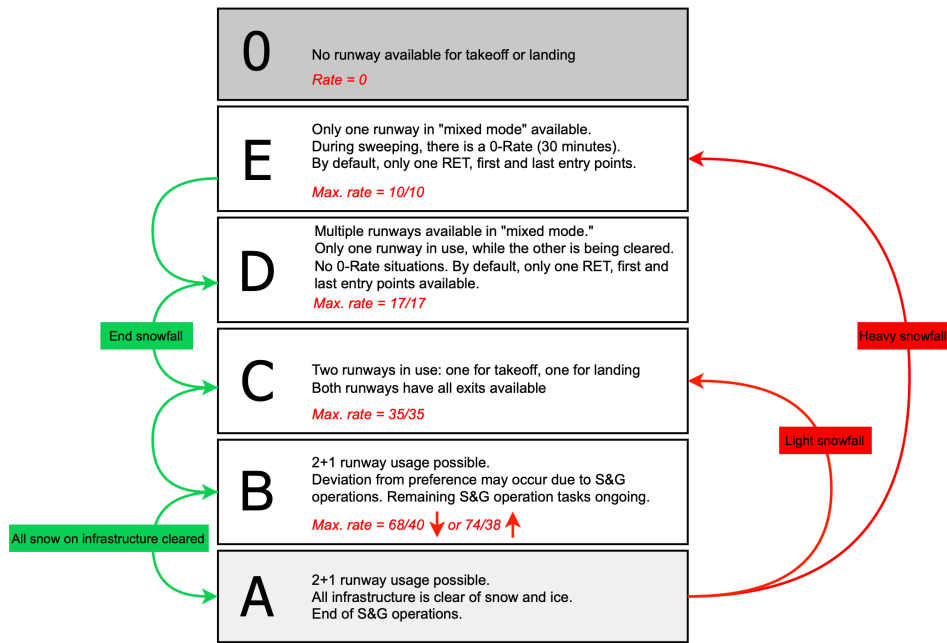


Figure 4.8: Winter Runway Availability Scenario (Service Owner Sneeuw en Gladheid, 2023)

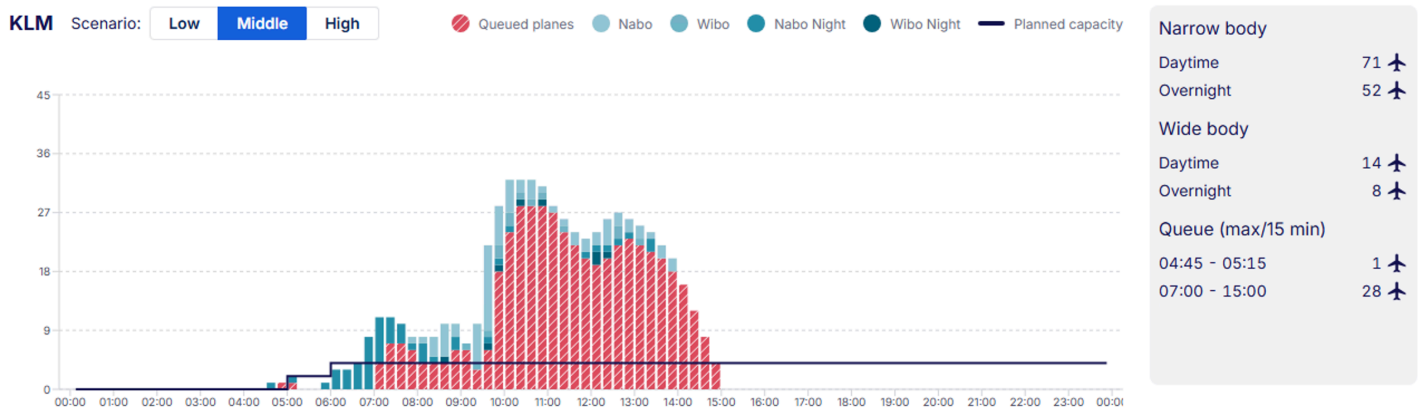


Figure 4.9: De-icing tool

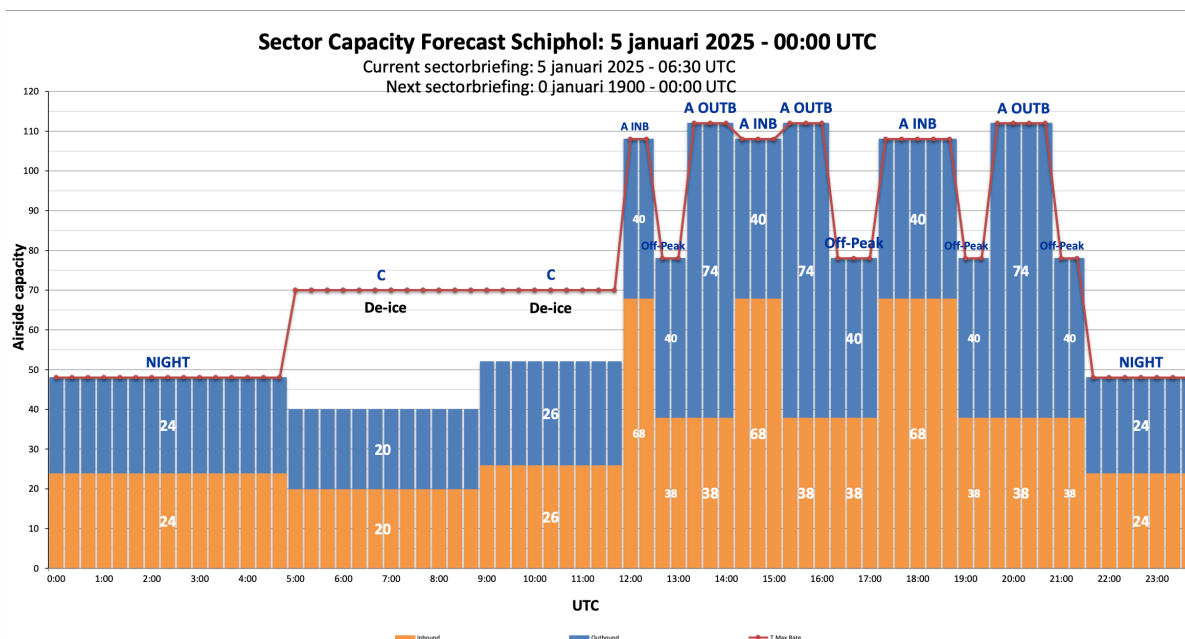


Figure 4.10: Sector briefing output

4.2.4. ASCTs activation

In anticipation of expected snow accumulation, the FMA sends a notification to all Snow Stars and contractors, alerting them to potential mobilization. Ideally, pre-warnings are issued three days in advance (D-3) to allow for early preparation. Additionally, the FMA notifies the gate planner in the control tower, enabling flexible scheduling of aircraft stands for snow storage and the staging of the snow fleets and ASCTs. At D-1 (one day prior) or D-0 (the day of the snowfall), depending on its timing, the Snow Stars and contractors receive the official call to report for duty (Schiphol Group, 2024a, 2024b).

When snowfall is predicted at very short notice, mobilization becomes significantly more challenging. This was evident during the snowfall event of November 22, 2024, when forecasts did not predict snow accumulation until just a few hours before it began. As a result, no pre-warning was issued, and all staff were called to report without warning. Since the notification was sent around 22:00, not all Snow Stars received it in time, resulting in incomplete attendance (Informal interview, Service Owner Winter Operations, November 2024).

4.2.5. Preparation of equipment and resources

To minimize the risk of equipment failures during snow operations, technical inspections are conducted on the snow removal fleet to ensure all vehicles are in optimal working condition. However, equipment failures can still occur at the time of activation. Minor issues are typically resolved on-site by technicians present during snow operations, but such failures may still temporarily reduce the number of available ASCTs (Informal interview Service Owner Winter Operations, January 2025).

Additionally, all necessary resources are strategically positioned in advance to ensure efficient deployment when snow activation is required (Informal interview Service Owner Winter Operations, January 2025).

The snow operation event of November 22, 2024 underscored the operational challenges caused by limited preparation time due to inaccurate weather forecasts. With no time for pre-snowfall inspections, critical technical issues went unnoticed. Once operations began, drivers discovered that their snow removal vehicles were still in training configuration. Resolving this issue took several hours, causing significant delays (Informal interview, Service Owner Winter Operations, January 2025).

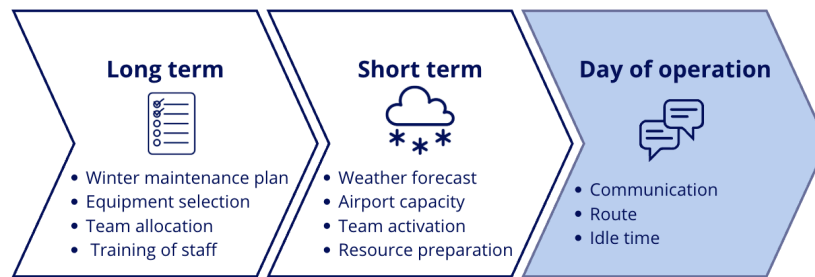


Figure 4.11: Day of snow removal operations

4.3. During the snow removal operation

4.3.1. Communication

The ASCT-coordinators communicate with APC via radio to receive and complete tasks. An exception applies to the ASCT-coordinator of ASCT 4, responsible for clearing the CDF, who communicates with the KLM ice tower instead of APC. However, if ASCT 4 (and ASCT 9) assist the other ASCTs at the *centrum*, they will resume communication with APC (Aircraft Operations Schiphol, 2024).

APC is structured into three main functions: Apron Control Inbound, Apron Control North and Apron Control South. During operations, ASCT coordinators must communicate with the three different departments: Apron Control Inbound for task assignment and handover, and Apron Control North or South for route approval, depending on the team's location (Aircraft Operations Schiphol, 2024). Each of these departments operates on a separate radio frequency, which can lead to communication challenges and coordination inefficiencies, particularly for less experienced coordinators (Winter Operation Training, November 2024).

During movement from stand to stand, ASCT-coordinators are under active control of Apron control North or South. Only when the whole team is within the aircraft stand, the coordinator can temporarily unsubscribe. In addition, there are some reporting points on taxiways, busy intersections where multiple lanes converge, where coordinators must call Apron Control North or South to ask permission to pass the point. As previously mentioned, the sprayers function independently of the ASCTs and communicate directly with APC (Aircraft Operations Schiphol, 2024).

After an aircraft stand has been cleared, the ASCT coordinator hands it over to Apron Control Inbound, assigning it to one of four status categories, Red, Orange, Yellow, or Green, as illustrated in Figure 4.12. These classifications serve as a guideline for discussions between the ASCT coordinator and Apron Control Inbound to determine whether further action is required. Once a stand is fully cleared and classified either Orange, Yellow or Green, Apron Control Inbound notifies the Gate Planner to confirm its readiness for operational use (Bolsius & Scholten, 2024; Service Owner Sneeuw en Gladheid, 2023).

4.3.2. Routes and destinations

ASCTs follow standard towing routes unless otherwise directed by Apron Control, for example, when an aircraft occupies the standard route, an occurrence more likely during periods of high traffic volume. These towing routes differ for inbound and outbound hours and are established in coordination with LVNL.

Apron Control Inbound assigns tasks to ASCTs based on the Gate Planner's prioritization and directs teams to their destinations. The Gate Planner is responsible for determining the order of snow clearance on aircraft stands while managing overall stand capacity.

During snowfall, task prioritization follows the "AS on demand" principle, meaning that aircraft stands are only cleared when an aircraft is scheduled to arrive within 15 minutes

Once snowfall has ceased, it becomes preferred to clean multiple aircraft stands simultaneously where possible. Moreover, Snow Fleet 3 may assist in these operations after snowfall.





	Aircraft stand cannot be used, too much snow/ice
	Entry line is visible Aircraft parking and jet bridge connection is possible. On the A-platform and H-platform: the passenger walkways are skid-resistant enough
	All of Orange, plus: OR: Limited handling is possible: At least luggage can be unloaded OR: Handling equipment blocks the full clearance of the aircraft stand OR: Aircraft stand is completely cleared but not yet sprayed
	The VOP is fully cleared and not slippery

Figure 4.12: Color classifications aircraft stand

In addition, to minimize travel time and enhance operational efficiency:

Efforts are made to keep ASCTs within the same bay during cleaning operations.

However, when multiple aircraft are expected to arrive in the same bay, it may be necessary to deploy ASCTs from other bays to assist. Maintaining teams within a single bay becomes especially challenging during the ‘Snow Small’ scenario, when only two ASCTs are available. In situations with low stand occupancy, it may be possible to strategically allocate incoming aircraft to adjacent stands. This enables an ASCT to clean multiple stands in succession, thereby reducing travel time and potentially shortening the overall cleaning duration (Bolsius & Scholten, 2024; Service Owner Sneeuw en Gladheid, 2023).

4.3.3. ASCT's idle time

The idle time of ASCTs includes activities such as shift handovers, refueling, breaks, and restroom visits.

ASCT coordinators, who are Schiphol employees, work in 8-hour shifts

The other vehicles are operated by contractors working 12-hour shifts. Shift handovers for ASCT coordinators typically occur at the team’s current location, whereas contractor personnel are relieved at locations designated by their own coordinator.

Refueling generally takes place at the ASCT’s current position. Vehicles such as sprayers and small tractors equipped with front-mounted brooms must monitor their levels of spraying agent to ensure timely replenishment. To maintain efficient ASCT operations, it is essential to inform APC of the expected refueling duration, which includes both fuel and spraying agent replenishment (Bolsius & Scholten, 2024).

Where possible, refueling is combined with a short break or shift handover. Breaks can be taken at the ‘Skihut’ near apron Y, while platform R is also commonly used as a designated rest location (Bolsius & Scholten, 2024).

4.4. Conclusion

To answer the first two research questions, two diagrams were developed by combining insights from the literature review and the current state analysis. Both diagrams have been validated by Schiphol’s

Service Owner Winter Operations and are presented in the following subsections.

Additionally, the current state analysis provided insight into how capacity is currently determined in advance of a snow event, an essential step in which aircraft stand cleaning capacity should play a central role. At present, capacity assessments rely on simplified assumptions rather than on data-driven estimates. As a result, aircraft stand cleaning is often overlooked entirely in strategic discussions, potentially leading to an overestimation of inbound capacity. This mismatch can cause substantial queues, delays, and broader operational disruptions.

To address this knowledge gap, the next chapter focuses on the quantification of aircraft stand snow removal performance using high-resolution historical data.

4.4.1. Swimlane diagram

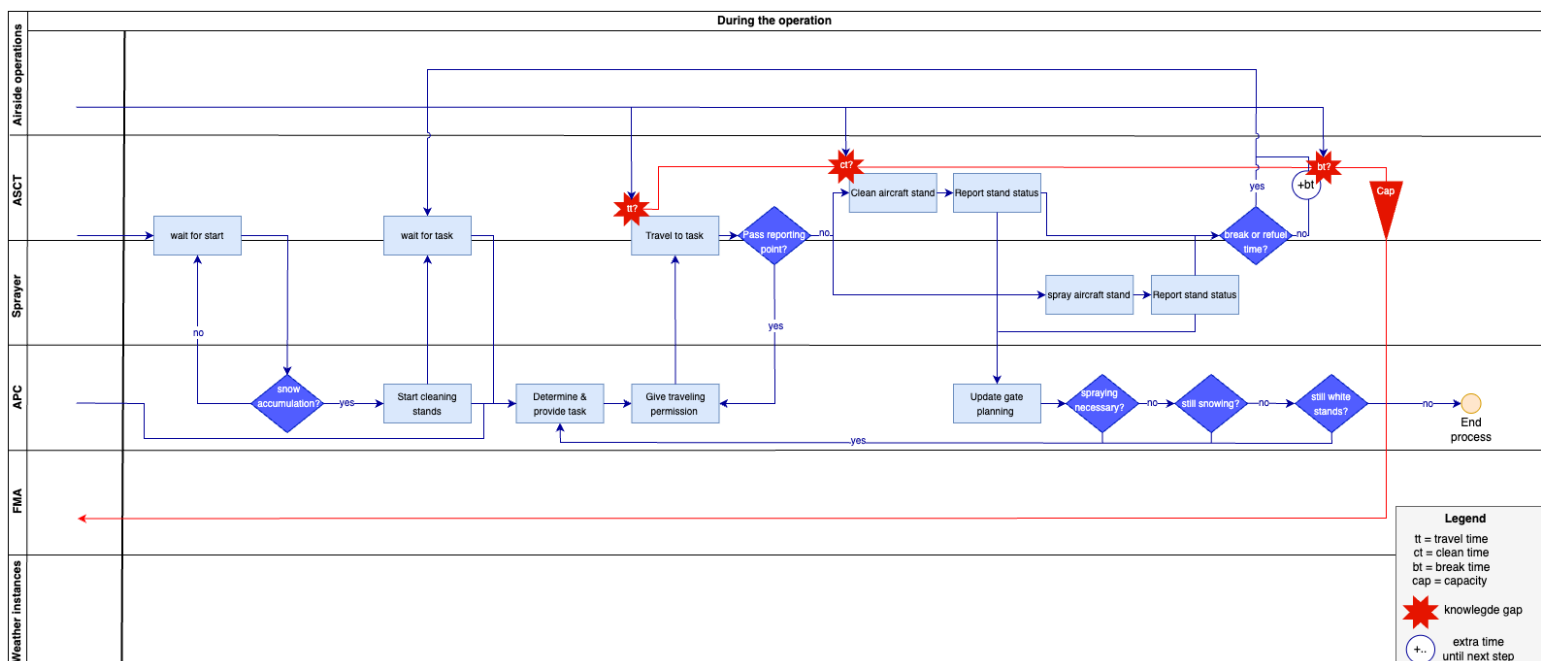
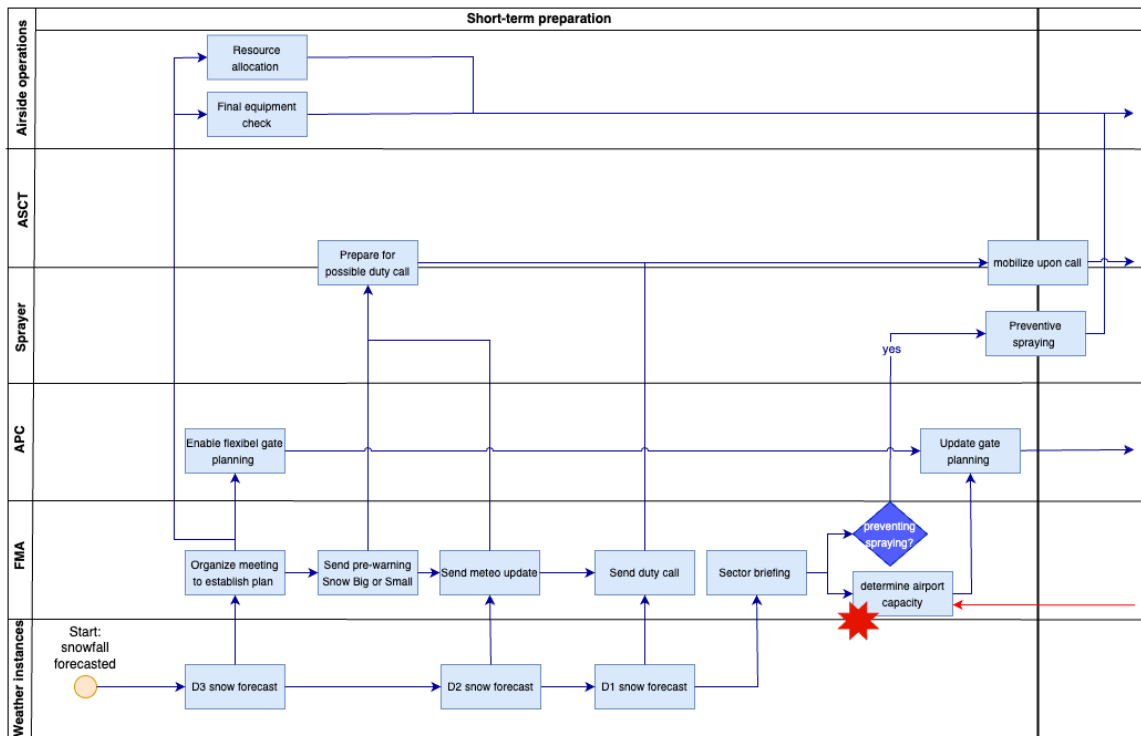
The first sub-question, ***"What is the current process for snow removal at aircraft stands?"***, is addressed through the swimlane diagram presented in Figure 4.13. This diagram maps the roles and responsibilities of the various stakeholders involved in snow removal operations and identifies key capacity-defining events. The diagram is organized into two main phases: short-term preparations and day-of-operations. Long-term preparations are excluded, as they mainly involve strategic planning rather than operational interactions between stakeholders.

As described in Section 2.4, each swimlane represents a distinct stakeholder or group of stakeholders. Apron Control Inbound, Apron Control North, and Apron Control South are grouped under a single swimlane labeled APC, due to the close alignment of their responsibilities. Red stars mark critical capacity input moments, highlighting areas where knowledge gaps exist in understanding aircraft stand snow removal capacity.

4.4.2. Causal diagram

The second sub-question ***"What key factors are theoretically expected to influence the capacity of snow removal operations at aircraft stands?"*** is answered through the causal diagram, illustrating the relationships between key factors affecting aircraft stand clearing capacity, presented in Figure 4.14. This diagram is specifically tailored to the process of cleaning aircraft stands at Schiphol, meaning certain factors identified in the literature have been omitted or reinterpreted to align with Schiphol's operations. For instance, the commonly referenced factor *Area to be cleaned* is adapted to *Type of aircraft stand*, which reflects the area size requiring clearance but is categorized only into narrow body and wide body stands. Similarly, *Cleaning priority* is interpreted differently than in most studies, as it is dedicated by the inbound flight schedule and cannot be optimized in advance. *Aircraft stand occupation* is not mentioned in the literature but is included based on the current state analysis.

The causal diagram informed the selection of input variables that could be reliably extracted from available datasets. These include aircraft stand type (Nabo, Wibo), route type (*within-bay*, *cross-bay* or *cross-zone*), number of active teams, and relevant weather conditions. Other factors, such as stand occupancy and human-related influences, while recognized as potentially significant, were excluded due to limited data availability.



(b) Process on the day of operation

Figure 4.13: Swimlane diagram



Figure 4.14: Factors influencing the snow removal process

5

Current cleaning performance

The swimlane diagram and causal diagram, presented in Figures 4.13 and 4.14, served as the conceptual foundation for the classification of operational activities, including, cleaning, traveling, and idle time, and for identifying key influencing variables.

The activity classifications were derived from historical data using a dedicated algorithm, described in Section 5.1. Based on the algorithm's output, Section 5.2 presents exploratory data analyses that provide insight into the performance of the classified activities and the influence of the selected variables.

5.1. Algorithm for capacity determination

The goal of the algorithm is to classify each radar data point into one of three operational states: cleaning, traveling, or taking a break. The primary data source used for this classification is radar data enriched with polygon information that pinpoints the precise location of each ASCT coordinator vehicle (labeled V4-9, as described in Subsection 4.1.2), within the airside area, recorded at one-second intervals. The python code is outlined in Appendix D.1.

For each snow event, the algorithm processes the data separately for each ASCT. It iterates over all timestamps (rows) during the relevant time period, examining the vehicle's location at every second. At any given moment, the vehicle is classified as either cleaning or traveling; break periods are treated as part of the cleaning status. The interpretation of location data depends on the assigned status, as outlined below:

Cleaning

When the vehicle is in the cleaning state, the algorithm accounts for the possibility that the vehicle's position may not always align precisely with the aircraft stand being cleaned. In practice, the coordinator vehicle may appear one or more polygons away due to minor movements while the team continues cleaning. To prevent prematurely ending the cleaning session, the algorithm incorporates a look-ahead window to accommodate this spatial variation. An example of this variation is shown in Figure 5.1, which illustrates the movement of vehicle V5 while cleaning stand E19.

This behavior also introduces ambiguity in distinguishing between scenarios where multiple adjacent stands are being cleaned simultaneously and situations where the ASCT coordinator is positioned on a nearby stand while the team cleans a single stand. To resolve this, the algorithm cross-references the radar-derived cleaning activity with data from the CISS and VGRS systems. These additional data sources help verify whether adjacent stands were also cleaned, addressing potential gaps in radar-based detection.

If a detected cleaning duration exceeds a predefined maximum threshold (see Table B.1 in Appendix B.1), the algorithm again consults the CISS and VGRS datasets, this time using a broader spatial scope. Instead of examining only directly adjacent stands, the algorithm considers all stands within the same bay, enabling the detection of multi-stand cleaning sessions that explain the extended duration.



Figure 5.1: Visualization of V5 between cleaning E19

When a detected cleaning period is shorter than the minimum allowed duration (see Table B.1 in Appendix B.1), the algorithm re-evaluates the sequence starting from the point where the vehicle left its previous task. The stand in question is excluded as a potential cleaning task, and the corresponding time period during which the vehicle was located at the excluded stand is reclassified as travel time.

Idle time

When the vehicle is in the cleaning state but located at platform R or Y, the algorithm assumes that the ASCT is taking a break. However, since the start and end points of operations can also occur at these platforms, the period is only classified as a break if it is detected outside the start or end of an operation.

It is important to note that breaks may also occur at other locations, such as aircraft stands or taxi-lanes. Unfortunately, these break periods are often misclassified as either cleaning or travel periods, depending on the vehicle's status, as there is no reliable way to accurately detect breaks in these cases.

Traveling

When the ASCT is not engaged in cleaning, the team is considered to be traveling to its next assignment. Travel time is measured from the moment the team leaves the previous location until it reaches the next one.

The output of the algorithm is a data file containing all identified cleaning periods, travel times, and intermediate breaks. Travel times are linked to their corresponding cleaning or break period to form complete cycles, which serve as the basis for determining cleaning capacity.

5.1.1. Missing information

The algorithm incorporates several mechanisms to address missing or inaccurate information, including absent location data (NaN values), location inaccuracies (e.g., due to GPS drift), and missing timestamps (e.g., when vehicles temporarily become untraceable due to transponder deactivation).

During cleaning periods, missing location data are handled using a look-ahead window to determine whether the vehicle is detected at the same aircraft stand shortly thereafter. If the vehicle reappears at the same stand, the NaN value is assumed to correspond to the cleaning location. If not, the cleaning period is terminated, and the NaN value is attributed to the subsequent travel time. Location inaccuracies are addressed by verifying whether the detected location remains within the same bay, in which case it is not considered an outlier. A threshold (see Table B.1 in Appendix B.1) determines how often

such deviations are tolerated as GPS drift before being interpreted as a genuine relocation. In this way, the model accounts for sporadic inaccuracies in vehicle positioning.

Missing timestamps are identified by measuring the time interval between consecutive rows. If this interval exceeds a predefined threshold (see Table B.1 in Appendix B.1), the cleaning period is terminated at the last valid timestamp, as the algorithm cannot infer the vehicle's status during the undocumented interval.

For travel periods, missing data have less impact. Missing locations are treated as part of the travel time, and location inaccuracies are less critical. However, large time gaps remain important, as the algorithm cannot determine vehicle activity during such intervals.

5.1.2. Detecting outliers

The algorithm incorporates several parameters to minimize the impact of outliers and missing data, including minimum and maximum cleaning and travel times. These parameter thresholds were derived from box plot summary statistics obtained from boundary-free algorithm results or based on practical experience. Additional details on parameter selection are provided in Appendix B.1.

To detect and exclude outliers, the algorithm applies these predefined thresholds. Cleaning cycles are excluded if their cleaning or travel durations exceed the maximum allowed values or contain data gaps that surpass the established time threshold. Additionally, snow operation days with fewer than 10 valid cleaning cycles were removed from the analysis, as these fell well below the typical range of 40 to 400 cycles observed on most days, rendering them insufficiently representative.

Further exclusions were based on location. Cleaning periods at stands located outside the research scope (those in areas HG, K, M, PD, and S) were omitted. Similarly, cleaning operations conducted at the CDF were excluded, as they fell outside the operational focus of the analysis. While these were primarily performed by V4, instances involving other vehicles were also identified and removed.

Collectively, these exclusion criteria ensured that only consistent, relevant, and representative data were retained for subsequent capacity evaluations.

5.1.3. Algorithm verification and validation

The development of the algorithm followed an iterative process during which both verification and validation were conducted to ensure its accuracy and robustness.

Verification focused on ensuring that the algorithm was correctly implemented, including accurate classification of operational states, proper functioning of threshold settings, and reliable integration with CISS and VGRS data. This is mainly done by boundary testing (how does the algorithm handle situations where data points fall exactly on threshold values) and debugging statements.

Validation involved continuously comparing the algorithm's output for selected vehicles on several snow operation days against records from the CISS and VGRS systems. Although these systems do not cover all cleaning activities, they provided valuable reference points to verify whether the algorithm correctly identified key cleaning periods.

In addition to system-based validation, manual validation was performed for selected vehicles on specific snow operation days. In these cases, cleaning periods were manually annotated through visual inspection of radar data and then cross-referenced with CISS and VGRS records. This manual annotation allowed for a direct comparison between algorithm output and human judgment.

The results of the validation methods confirmed that the algorithm successfully identified the majority of operationally relevant cleaning events. The combination of verification (technical correctness) and validation (real-world accuracy) guided successive refinements to the classification rules and parameter thresholds, improving the algorithm's robustness and accuracy.

Despite multiple development iterations, certain limitations remain. A key challenge lies in accurately distinguishing between cleaning periods and breaks. The algorithm assumes that when an ASCT is located at platform R or Y during the operation, so outside the start or end of an operation, the team is on a break. However, breaks taken at aircraft stands within the *centrum* area are not reliably detected. These can either be misclassified as cleaning events or, if the duration exceeds the maximum cleaning

threshold, excluded altogether from the analysis. Similarly, breaks occurring while the ASCT is in transit (e.g., parked on a taxi lane) may be incorrectly included in travel time or, in the case of extended durations, flagged as invalid and discarded.

5.2. Exploratory data insights on snow removal operation

After running the algorithm on all radar data, a data file was generated containing all identified travel times, cleaning periods, break periods, and cycles durations. This output serves as the basis for the exploratory data analysis, which aims to assess the performance of the current process and determine cleaning capacity.

For the data analysis, box plots and corresponding box plot statistics were chosen over the mean and standard deviations for two main reasons. First, box plots provide a non-parametric summary of the data, and do not assume an underlying distribution. This is particularly appropriate for the current dataset, where the assumption of normality does not hold, as shown in Figure 5.4 and the Figures in Appendix B.2. Second, mean values are sensitive to extreme observations, which frequently occur in this context due to irregular operational events. By using the median, the analysis better reflects typical performance, avoiding distortion from exceptionally short or long durations.

5.2.1. Team availability

Figure 5.2 shows the number of active teams during each Snow operation day. Prior to 2025, the intended number of teams active simultaneously was five. Occasions where six teams were recorded before 2025 likely reflect vehicle changes during operations due to technical issues. Since 2025, the intended number of active teams has increased to six, as vehicle V9 was added alongside V4 at the CDF. From the Figure, it can be observed that:

On 50% of the operation days, the number of active vehicles was one below the preferred maximum.

This shortfall is likely due to the unavailability of the ASCT coordinator or technical malfunctions that rendered a vehicle inoperable.

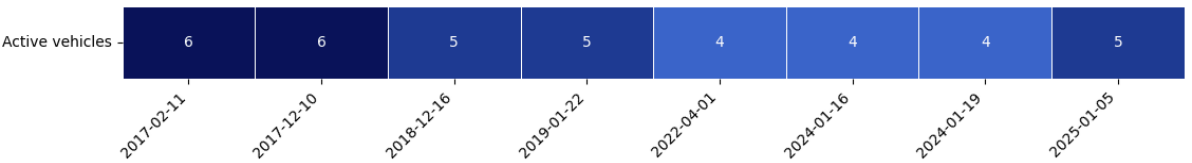


Figure 5.2: Number of active teams during each snow operation

5.2.2. Team's Idle time

Break moments were identified when a cleaning period was recorded at platform R or Y, as these locations are commonly used as designated rest areas. However, as noted in Subsection 5.1.3, there is a degree of uncertainty in detecting break periods. Breaks taken at aircraft stands, rather than at platforms R or Y, are not recognized by the algorithm and are therefore likely underreported.

The heatmap in Figure 5.3 shows the number of breaks per vehicle across all analyzed Snow operation days. It reveals that, in most cases, teams take one break during snow events lasting approximately one 8-hour shift. In contrast, longer operations, such as those in 2017 that spanned multiple days, display a significantly higher number of breaks. At the same time, the heatmap also highlights potential underreporting. For example, in 2022 and 2024, not all teams appear to have taken a break, which is unlikely given the duration of the operations. Similarly, in 2019, all teams are shown to have taken only one break, despite the snow operation extending beyond two full shifts, which is also unrealistic. To address these limitations, the heatmap findings were interpreted alongside known operational practices. Based on this combined insight, it is assumed that each team takes one break per 8-hour shift.

In addition to the number of breaks, the duration of each break is an important parameter to include in

the forecasting model to accurately reflect the availability of cleaning teams. The distribution of break durations is shown in Figure 5.4, which demonstrates a high degree of variability. While some breaks last only a few minutes and others exceed 90 minutes, most are relatively short, with a median duration of 28.3 minutes. To account for travel to designated break areas (platforms R or Y), this is supplemented by a median cross-zone travel time of 7.7 minutes, resulting in an estimated total median break time of approximately 35 minutes.

Each ASCT takes **one** break per 8-hour shift, of approximately **35** minutes.

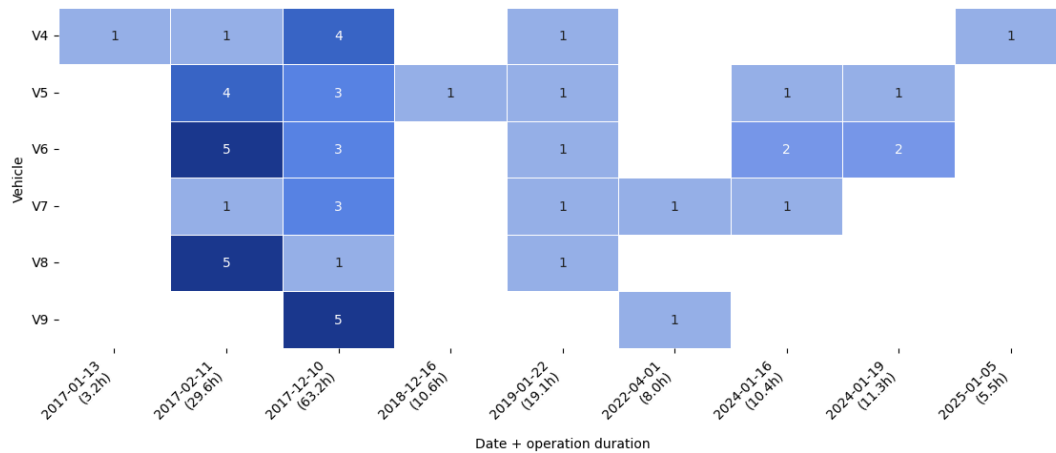


Figure 5.3: Number of breaks for each vehicle on each snow day

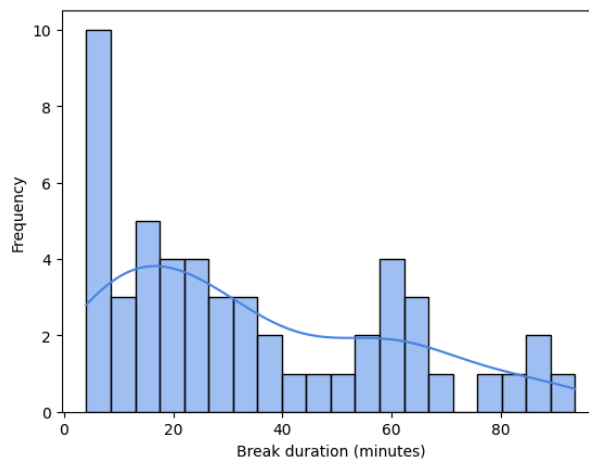


Figure 5.4: Distribution of break durations

5.2.3. Clean time

Aircraft stands are classified into two categories: narrow body (Nabo) and wide body (Wibo). Of all cleaned stands, 59% are Nabo and 41% are Wibo (see Appendix B.3.2). There is a notable difference in cleaning time between the two types, with Wibo stands generally requiring more time. This is illustrated by the box plots in Figure 5.5, and aligns with expectations, since wide body stands have a larger surface area to clean. The box plots show relatively similar interquartile ranges (IQRs) (15.2 minutes for Nabo and 16.1 minutes for Wibo, see Appendix B.3.2), indicating comparable variability in clean time.

5.2.4. Travel time

Travel time is categorized into three types: *within-bay*, *cross-bay*, and *cross-zone*. *Cross-bay* refers to movements within the *centrum* area where the team must switch to a different bay, while *Cross zone*

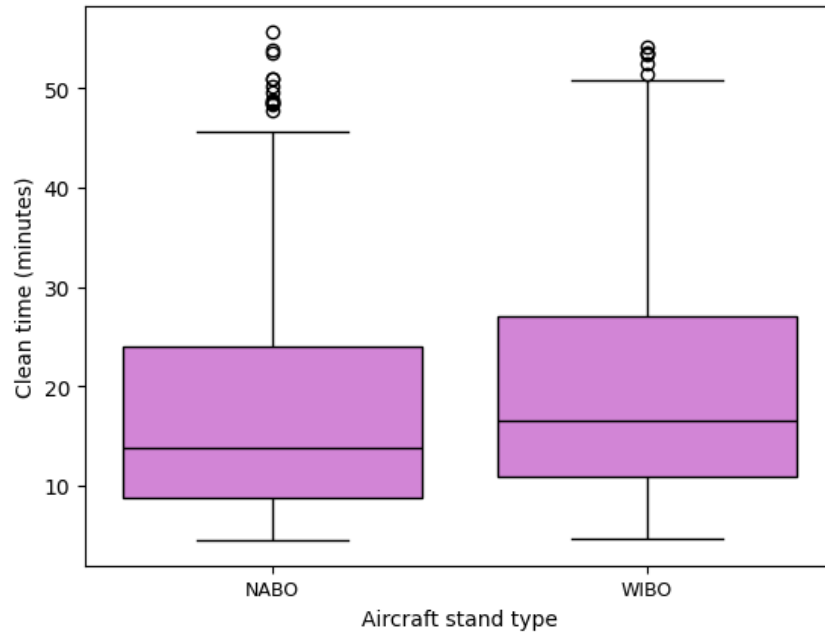


Figure 5.5: Variation of clean times per aircraft stand type

involves travel between the *centrum* and other areas of the Airside, typically occurring at the start and end of operations, during breaks, or when vehicles assigned to clean the CDF must travel to and from the *centrum*. Figure 5.6 provides a visual illustration of each route type: Figure 5.6a shows V8 traveling from B32 to B20 (*within-bay*), 5.6b shows V8 moving from C07 to B32 (*cross-bay*) and 5.6c shows V8 traveling from its starting position at R72 to E09 (*cross-zone*).



Figure 5.6: Visualisation of each route type

The box plots in Figure 5.7 illustrate substantial differences in travel time across the three route types. *within-bay* movements are by far the shortest and most consistent, with a median travel time of just 0.63 minutes. The narrow IQR (0.77 minutes) and low upper whisker (1.92 minutes) reflect limited variability, reflecting the efficiency and predictability of *within-bay* routing. In contrast, *cross-bay* and *cross-zone* movements are significantly longer and more variable, with median travel times of 6.3 and 7.7 minutes, respectively. The wide IQRs and high upper whiskers suggest greater susceptibility to delays.

These results highlight the operational advantage of assigning consecutive tasks within the same bay to minimize travel time and improve predictability.

Although efforts are made to keep ASCTs operating within the same bay, aiming to reduce travel time and enhance operational efficiency, this strategy is not consistently reflected in practice. Figures 5.8a

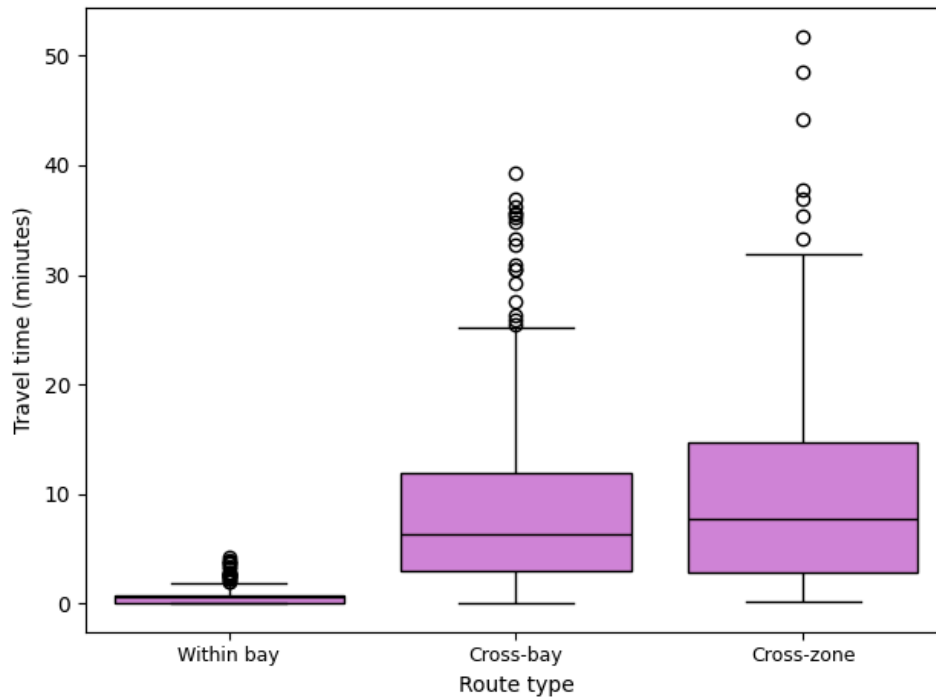


Figure 5.7: Variation in ASCT travel time across route type

and 5.8b show that *cross-bay* movements still occur frequently. Moreover, Figure 5.9 shows no observable increase over time in the proportion of *within-bay* routes across snow events.

The findings for the *centrum* area are particularly relevant, as efficiency efforts are focused here and the forecasting model is specifically designed to simulate operations within this zone. Notably:

56% of the routes are classified as *within-bay*, while **44%** are *cross-bay*.

The number of active teams is theoretically linked to route type distribution, as more teams could allow each to remain within its bay, reducing *cross-bay* movements (see the causal diagram in Figure 4.14). However, no clear trend emerges when comparing team availability (Figure 5.2) with route type proportions across snow days (Figure 5.9). For example, in 2018, the share of *within-bay* routes was high despite having the same number of teams (5) as in 2017 and 2019, when proportions were lower. Similarly, in 2022 and 2024, both with 4 active teams, route type proportions differed markedly. This suggests that team availability alone does not explain route type variation, indicating that other operational or environmental factors may be more influential.

When the strategies regarding routing are not achieved and *cross-bay* travel occurs, it becomes relevant to examine whether travel time increases with the number of bay shifts a team makes. There are eight bays in total (A, B, BC, CD, D, DE, EF, FG, G), arranged in sequential order. As such, a movement from bay A to bay G, or vice versa, represents the maximum of eight bay shifts.

Table B.6 presents the frequency of each bay shift and the corresponding median travel time for all categories with sufficient data. For the purpose of analysis, categories involving six, seven, or eight bay shifts were excluded due to their low occurrence. The table shows that most *cross-bay* routes involve only a single bay shift, with the number of occurrences decreasing as the number of shifts increases. This pattern suggests that operational planning efforts are generally successful in keeping teams at least close their current bay.

Notably, even a single bay shift results in a substantially longer median travel time compared to routes that remain within the same bay (5.8 minutes versus 0.6 minutes). The median travel times across the bay shifts categories reveal substantial variation, with one to three shifts associated with significantly

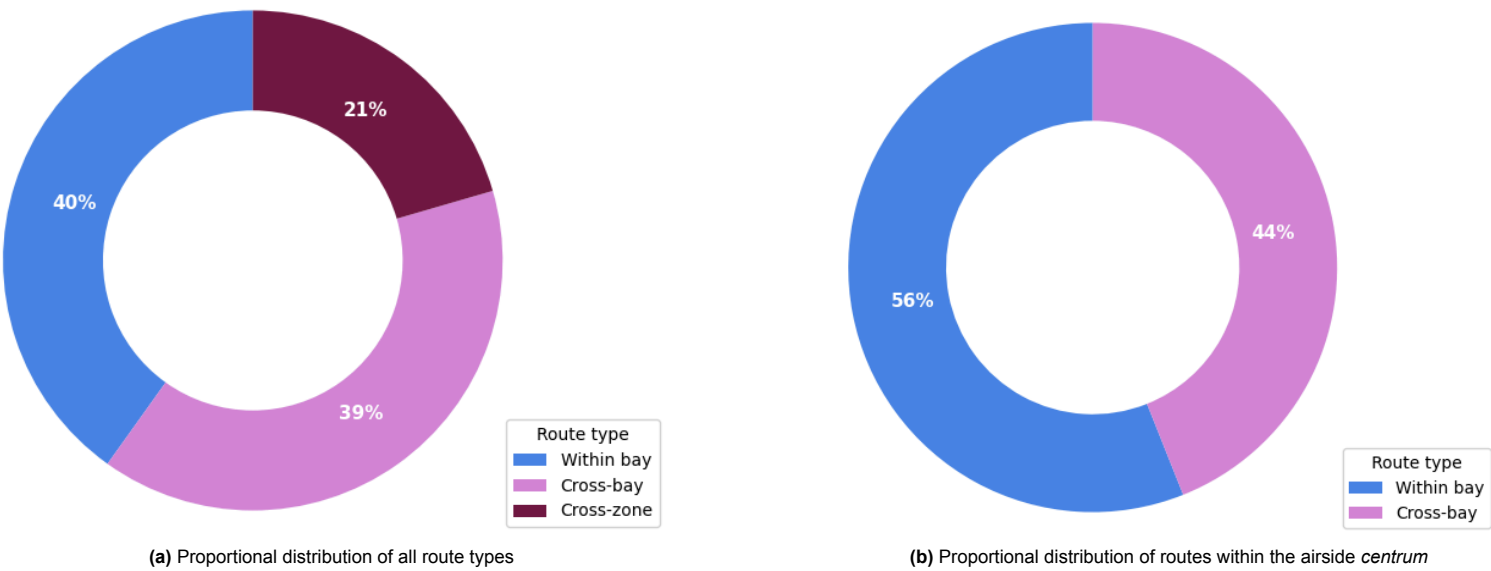


Figure 5.8: Comparison of ASCT route type distributions

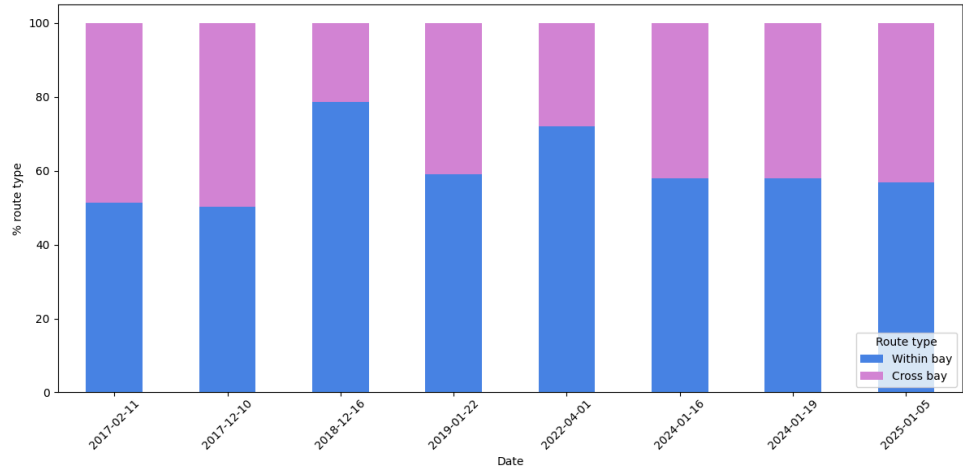


Figure 5.9: Proportional distribution of routes within *centrum* over the years

shorter travel times than the higher-shift categories.

Number of bay shifts	Count	Median travel time
1	109	5.43
2	38	6.98
3	29	7.93
4	12	11.80
5	12	11.92
6	1	-
7	2	-
8	1	-

Table 5.1: Descriptives of travel time across number of bay shifts

5.2.5. Cleaning cycles

A cleaning cycle comprises both the travel time to a task location and the task’s cleaning duration. Cycle time serves as the basis for calculating cleaning capacity in the following section. Since cleaning time varies by aircraft stand type and travel time depends on the route type, it is expected that cycle

time is influenced by both factors. This expectation is supported by the box plots in Figure 5.10, which illustrate clear variations in cycle time across different stand and route types. Notably, the difference in median cycle time between route types is more pronounced than between stand types.

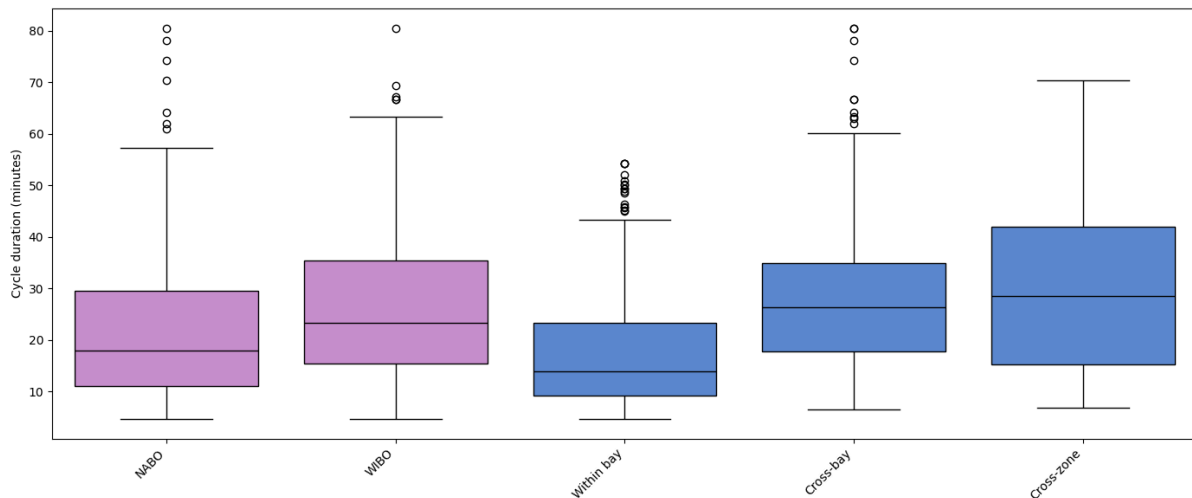


Figure 5.10: Distribution of cycle times across stand and route types

Figures 5.11a and 5.11b provide a more detailed understanding of the composition of cycle times. Figure 5.11a distinguishes between regular and long cycles, with the latter defined as those exceeding the 75th percentile (32.4 minutes). The scatterplot shows that most long cycles are driven by extended cleaning durations rather than unusually long travel times.

Figure 5.11b displays the distribution of cleaning time as a share of total cycle time, revealing that cleaning often takes up a substantial portion of the cycle, with a strong concentration near a ratio of 1.0. This suggests that, in many cases, cleaning dominates the total operational cycle. Additional graphs in Appendix B.3.4 show that this effect is primarily driven by *within-bay* operations, where travel time is minimal. In contrast, *cross-bay* and *cross zone* movements contribute more significantly to the total cycle time, with their distributions peaking around 20–30%.

Together, these figures highlight that, although route type influences the overall cycle length, cleaning duration plays the most dominant role and explains most of the variability in the upper tail of the distribution.

5.2.6. Capacity

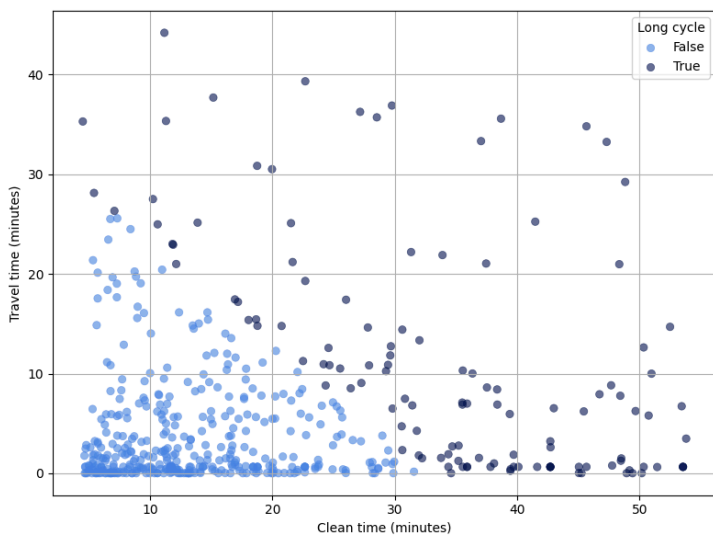
Hourly cleaning capacity is defined as the number of cleaning cycles that can be completed within one hour. For each recorded cycle, capacity is calculated by dividing 60 minutes by the total cycle duration. For instance, a 15-minute cycle corresponds to a capacity of 4 tasks per hour.

Figure 5.12 shows the distribution of cleaning capacity across all recorded cycles. The blue curve represents the estimated probability density function (PDF), highlighting a right-skewed distribution. Most capacities fall between 1 and 4 cycles per hour, with a peak around 2 to 3, corresponding to cycle durations of 30 to 20 minutes. Outliers reaching up to 12 cycles per hour reflect exceptionally short cycles, around 5 minutes, which likely result from minimal cleaning and travel times (*within-bay*). The long right tail suggests that under optimal conditions, significantly higher throughput is achievable, although such cases are rare in practice.

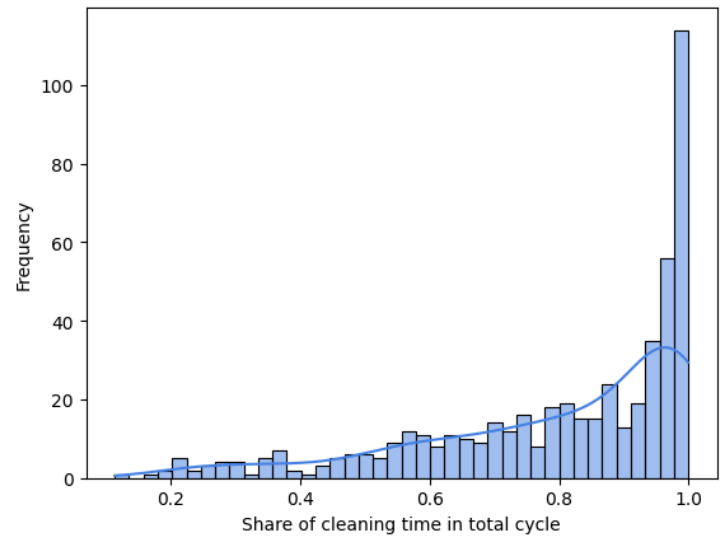
This distribution serves as the basis for the Analysis phase, in which statistical tests are conducted to evaluate the influence of various factors on cleaning capacity.

5.3. Conclusion

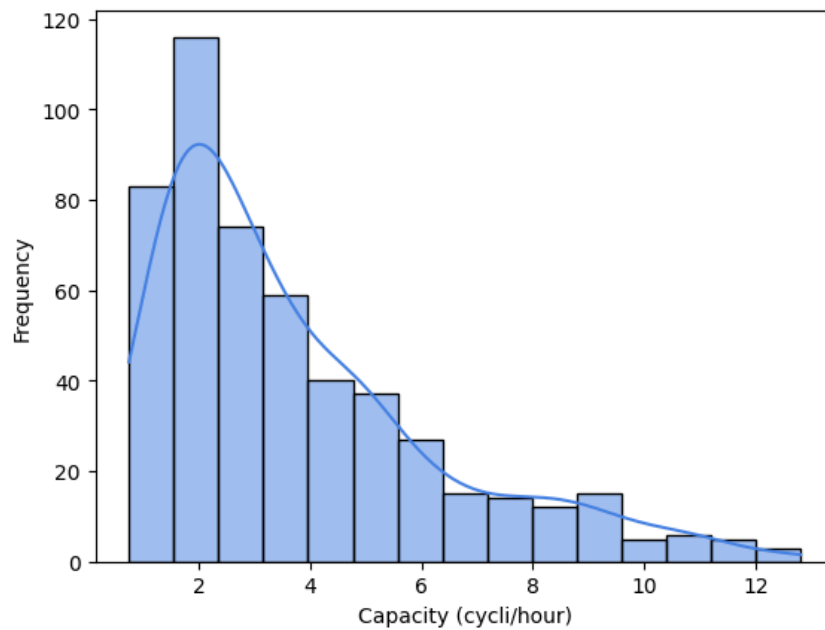
This chapter provided a comprehensive overview of current snow removal operations at aircraft stands, based on enriched radar data and a custom-built classification algorithm. The algorithm reliably distin-



(a) Composition of regular versus long cycle times



(b) Frequency distribution of clean time share in cycle duration

Figure 5.11: Impact of clean time versus travel time on cycle time**Figure 5.12:** Distribution of hourly cleaning capacity

guishes between cleaning, travel, and break periods, while accounting for location inaccuracies and missing timestamps. Key operational parameters were derived using box plot statistics, reducing the impact of outliers.

Addressing the research question ***“What insights do historical data offer about the performance of current snow removal operations?”***, this chapter demonstrated that historical data reveals clear patterns and variability in snow removal performance, driven by stand type and routing efficiency.

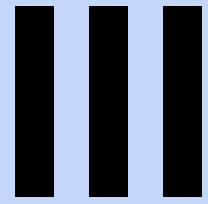
Key Insights

- 50% of the time, one fewer team was active than the preferred maximum.
- Idle time varied greatly in both frequency and duration; a 35-minute break per 8-hour shift is assumed.
- Wide body stands required longer cleaning times than narrow body stands.
- Travel time increased significantly when ASCTs moved between bays or zones.
- Even a single bay shift led to substantially higher travel time, with additional shifts compounding the effect.
- The proportion *within-bay* - *cross-bay* operations is approximately 55%–45% and has remained stable over the past years.
- Cleaning duration was the primary driver of overall cycle length.
- Capacity followed a right-skewed distribution, peaking at 2–3 cycles per hour.

For the remainder of this study, the analysis will be limited to cleaning cycles within the *centrum* area.

This decision was made because *cross zone* movements typically occur at the start and end of the operation, during intermediate breaks, or when vehicles travel to and from the CDF. These movements do not reflect the core operational dynamics that the forecasting model aims to capture. Furthermore, ASCT routing strategies are primarily focused on the *centrum* area (keep teams within their bay). By limiting the analysis to this operational context, the results obtained are directly applicable to the improvement of relevant planning and routing strategies.

The insights gained in this phase form the foundation for the next stage of the research, where statistical tests are applied to evaluate the influence of environmental and operational factors on snow removal capacity.



ANALYZE



Determinants of Cleaning Capacity

In this chapter, potential factors influencing the aircraft stand cleaning capacity are analysed. Their statistical significance and practical impact are assessed. Based on these findings, categorical capacity classes are defined, which serve as input for the final design model.

As concluded in the previous chapter (Section 5.2.6), the distribution of cleaning capacity is non-normal. Consequently, non-parametric statistical tests are used to assess the impact of relevant factors on capacity. The tests applied include the Spearman rank correlation, Mann-Whitney U test, and the Kruskal-Wallis test.

Spearman rank correlation

The Spearman rank correlation evaluates the strength and direction of the association between two variables without assuming linearity or normally distributed data. It tests the null hypothesis that no association exists between the variable and cleaning capacity. A p-value below 0.05 indicates that the null hypothesis can be rejected, providing statistically significant evidence of a relationship between the variable and cleaning capacity. Variables found to be significantly correlated with capacity are considered for inclusion in the forecasting model.

Mann-Whitney U test and Kruskal-Wallis H test

The Mann-Whitney U test is used to determine whether there is a statistically significant difference in cleaning capacity between two groups. It serves as a non-parametric alternative to the independent samples t-test when the assumption of normality is not met. For comparisons involving more than two groups, the Kruskal-Wallis H test is applied. This test extends the Mann-Whitney U test to multiple groups and serves as a non-parametric alternative to one-way ANOVA. Both tests evaluate the null hypothesis that there is no difference in cleaning capacity between the tested groups. A p-value below 0.05 leads to the rejection of the null hypothesis, indicating a statistically significant difference in cleaning capacity between the groups.

6.1. Impact of weather parameters

6.1.1. Data preprocessing

As outlined in Table 2.2 in Subsection 2.5.1, the KNMI dataset contains a lot of data. First the six locations closest to the aircraft stands at *centrum* are filtered, and the variables provided in Table 6.1 are extracted. All variables are available at both 12-second and 1-minute intervals, while the cycle times are at minute-interval. Therefore, the weather data at 1-minute intervals is used. However, there is one exception: *PWCm* is only available at 12-second intervals. For this parameter, the mode (most occurring value) of the 12-second values within each minute is calculated to align with the minute-level resolution.

To align weather conditions with cleaning cycles, and thereby assess their impact on cleaning capacity, four aggregation strategies were applied. First, weather variables were aggregated over a time window

Category	Variable (unit)	Description
Precipitation	NDm (seconds/minute)	Precipitation Duration (1 minute sum)
	NIm (mm/hour)	Precipitation intensity (1 minute average)
	PWCm (-)	Precipitation type. Relevant types for this research are: 67 (rain+snow), 70 (snow), 77 (drizzle snow), 87 (grain hail or snow), 89 (hail)
Temperature	TAm (°C)	Ambient Temperature (1 minute average)
	TBm (°C)	Wet Bulb Temperature (1 minute average)
	TDm (°C)	Dew Point Temperature (1 minute average)
	TGm (°C)	Grass Temperature (1 minute average)
Visibility	Vis (m)	Horizontal Visibility
	C1s (m)	First Cloud Layer Height
Wind speed	WSmK (knots)	Wind speed in Knots (1 minute average)
Snow accumulation	SHm (mm)	Snow Depth (1 minute average)

Table 6.1: Weather variables

equal to the minimum cycle duration (4.6 minutes), to standardize the length of all periods, enabling direct comparisons. Second, aggregation is performed over the median cycle duration (19 minutes), or the full period if the actual cycle is shorter, capturing the typical operational duration. Third, weather data is aggregated over the actual duration of each cycle, accounting for all fluctuating conditions during the operation. Lastly, a 10-minute pre-cycle window is included to capture preceding weather conditions that may affect performance. Table 6.2a provides an overview of these aggregation strategies, while Table 6.2 outlines the method used for each weather variable. Most variables are averaged across the defined windows. For *NDm*, a sum is calculated first, followed by a ratio, allowing for fair comparison across different durations. *PWCm*, being categorical, is aggregated using the mode to avoid invalid class values. For *SHm*, the maximum is taken to reflect peak snow depth, which can increase rapidly during snowfall.

Aggregation	Prefix	Aggregation method	Variables
Minimum cycle duration	first4.6_..	Average	NIm, TAm, TBm, TDm, TGm, Vis, C1s, WSmK
Median cycle duration	first19_..	Max	SHm
Full period	full_ ..	Mode	PWCm
10 minutes before period	pre10_..	Sum	NDm
		Ratio	summed NDm

(a) Aggregated weather variables

(b) Aggregation methods

Table 6.2: Aggregated variables and the corresponding methods

Besides the weather parameters, several classifications are made, as outlined in Table 6.3, based on the classifications introduced in Table 4.2 in Subsection 4.2.1. The full period aggregation variables are used for these classifications because they provide a comprehensive summary of weather conditions over the entire duration of the cleaning operation. This ensures that the classification reflects the full context in which the cleaning task was performed, rather than being influenced by short-term fluctuations that may not represent the overall operational conditions.

6.2. Impact of weather variables

To examine the impact of weather variables on cleaning capacity, a Spearman rank correlation test was performed between each aggregated weather variable and hourly cleaning capacity, as well as between each weather classification and capacity. The null hypothesis for each test stated that no relationship exists between the respective weather variable and cleaning capacity. Based on the resulting p-values,

Variable	Category name	Category value
Precipitation type	Snow	$full_PWCm \in (67, 70, 77)$
	Remaining snow	$full_SHm > 5 \text{ AND } full_PWCm \notin (67, 70, 77)$
	Rain	$full_SHm = 0 \text{ AND } full_PWCm \in (65, 55, 60, 65)$
	No precipitation	$full_SHm = 0 \text{ AND } full_PWCm = 0$
Snow Intensity	Light	$full_NIm < 5$
	Moderate	$5 \leq full_NIm \leq 50$
	Heavy	$full_NIm > 50$
Snow type	Wet	$full_TBm > 0.5 \text{ OR } full_PWCm = 67$
	Dry	$full_TBm < -0.5$
	Doubtful cases (around 0°)	$full_TDm < -2$
Wind speed	Light (< 3 Bf)	$full_WSmK < 7$
	Moderate (3–6 Bf)	$7 \leq full_WSmK \leq 22$
	Heavy (>6 Bf)	$full_WSmK > 22$
Visibility	BZO scenario C	$full_Vis < 350$
	BZO scenario B	$350 \leq full_Vis < 550 \text{ OR } full_C1s < 200$
	BZO scenario A	$550 \leq full_Vis < 1500 \text{ OR } 200 \leq full_C1s \leq 300$
	BZO scenario 0	$full_Vis \geq 1500$

Table 6.3: Categorized weather variables

presented in Appendix B.4, it was found that among the aggregated variables, only snow depth (*SHm*) showed a statistically significant correlation with cleaning capacity ($p < 0.05$). All other aggregated variables, as well as the categorized weather variables, yielded nonsignificant associations.

This lack of significance is likely due to limited variability in the observed weather conditions. For example, snow intensity was consistently classified as light, wind conditions were generally moderate, and visibility was almost always good (BZO scenario 0). Such uniform conditions reduce the potential to detect meaningful differences between categories and also explain the absence of significant effects among most aggregated weather variables.

To define categorical capacity classes suitable as input for the forecasting model, the continuous variable *Shm* was discretized. Initially, *Shm* was divided into three categories, low, medium, and high, as shown in Table 6.4, based on threshold values corresponding to snow situation classifications defined by Schiphol (see Figure 4.6).

Table 6.4b presents the results of the Kruskal-Wallis H test and the pairwise Mann-Whitney U tests. The Kurkal-Wallis H test indicated a statistically significant difference among at least two of the snow depth categories ($p = 0.003$). However, pairwise Mann-Whitney U tests did not reveal a significant difference between the low and medium categories. Consequently, these two categories were merged. For the final model, snow depth is classified into two groups: low (< 50 mm) and high (≥ 50 mm), consistent with the classification thresholds used by KNMI (KNMI, n.d.).

Figure 6.1 shows tat higher snow depths are associated with a substantially lower median capacity, suggesting that increased snow accumulation negatively impacts cleaning performance.

Category	Value	Snow depth level	P-value	Test
Low	$SHm < 10 \text{ mm}$	Low, medium, high	0.003	Kruskal-Wallis H
Medium	$Shm < 50 \text{ mm}$	Low vs medium	0.430	Mann-Whitney U
High	$Shm \geq 50 \text{ mm}$	Low vs high	0.006	Mann-Whitney U
		Medium vs high	0.004	Mann-Whitney U

(a) Snow depth classifications

(b) Statistical test results snow depth categories

Table 6.4: Snow depth categories and their statistical significancy

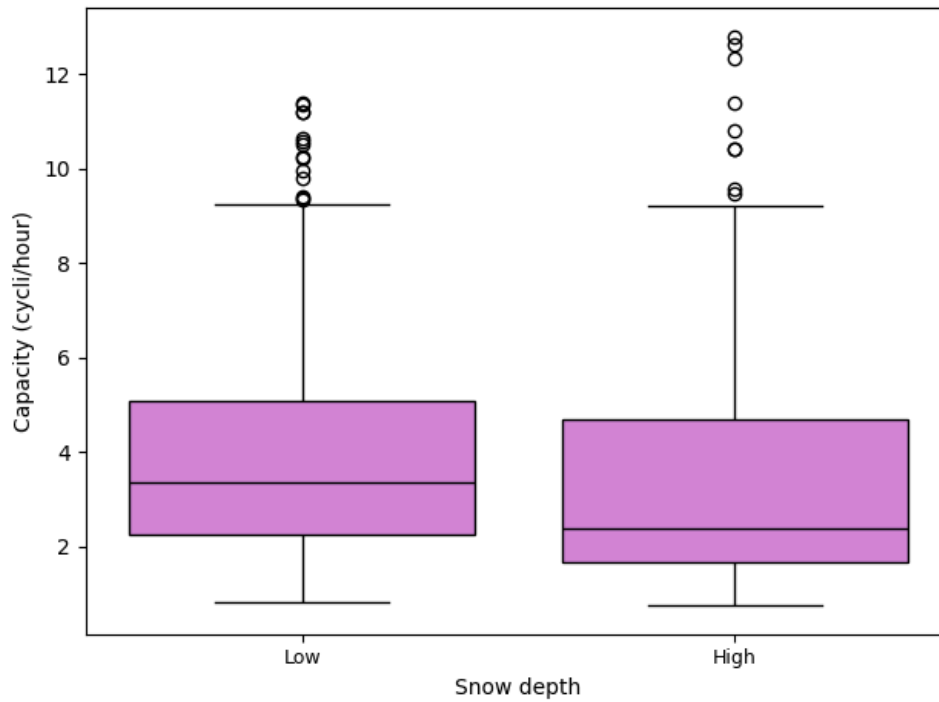


Figure 6.1: Distribution of snow depth across snow depth categories.

6.3. Impact of operational parameters

To assess whether hourly cleaning capacity significantly differs by aircraft stand type and route type, two Mann–Whitney U test were conducted. The results of these tests are presented in Table 6.5, revealing a statistically significant difference in cleaning capacity and both differentiations. The box plots in Figure 6.2 give an illustration to this difference, showing that Nabo stands are associated with higher cleaning capacities than Wibo stands, and *within-bay* routes are linked to substantially higher capacities than *cross-bay* routes.

To examine the influence of bay shifts on capacity, a Kruskal–Wallis test was applied across the five bay shift categories with sufficient data. The test did not yield a statistically significant result, indicating that the number of bay shifts does not significantly impact hourly cleaning capacity. Although bay shifts were shown to have a clear impact on travel time in Figure B.3, this does not translate into significant differences in overall cleaning capacity. A likely explanation is that while longer travel times may reduce the number of potential tasks a team could complete, this effect is offset by the variability in cleaning durations. In practice, cleaning time often dominates the total cycle duration, as concluded in Subsection 5.2.6, meaning that fluctuations in travel time have a relatively smaller influence on the overall capacity.

Variables	P-value	Test
Nabo vs Wibo	0.022	Mann-Whitney U
within-bay vs cross-bay	< 0.001	Mann-Whitney U
cross-bay shifts 1,2,3,4,5	0.491	Kruskal-Wallis H

Table 6.5: Statistical test results aircraft stand and route type

6.4. Capacity categorization

After concluding that snow depth, aircraft stand type, and route type each influence hourly cleaning capacity, the combined effects of these factors were further evaluated. A Kruskal–Wallis H test was performed across the six different combinations of these variables, with the resulting p-values outlined in Table 6.6. All combinations demonstrated a statistically significant effect, except for C4, which repre-

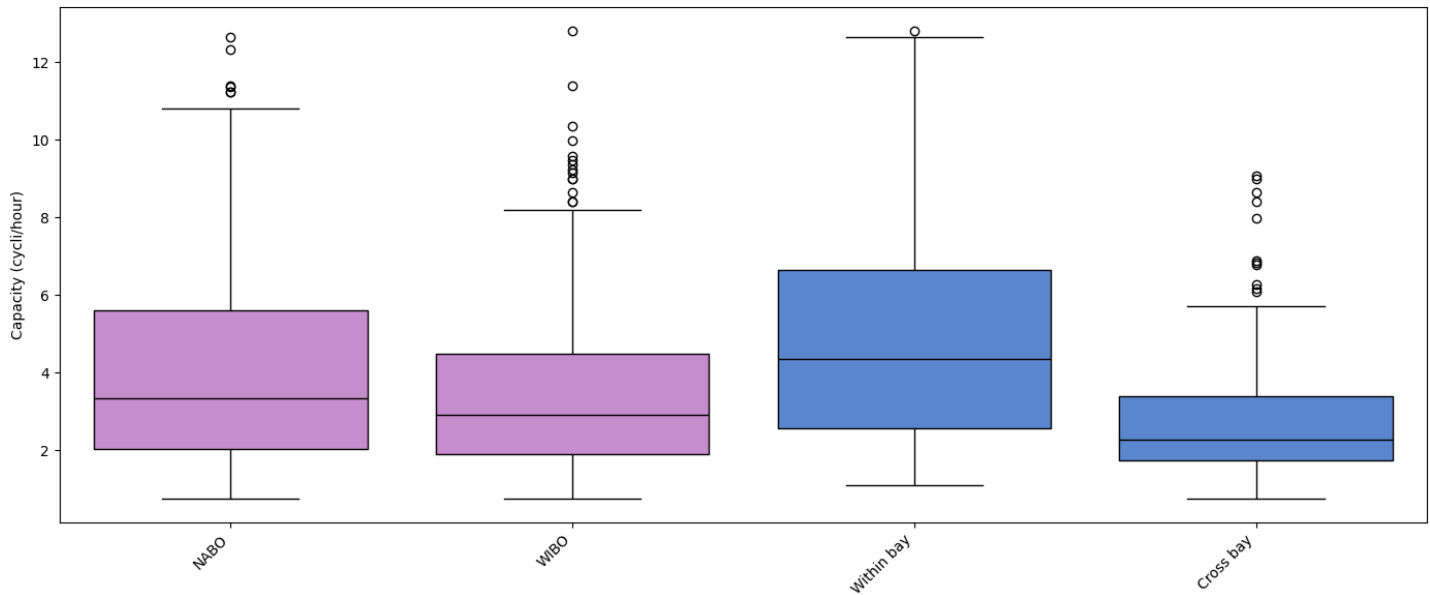


Figure 6.2: Distribution of snow removal capacity across aircraft stand and route types

sents *cross-bay* routes across stand types and snow depth levels. This indicates that when an ASCT operates on a *cross-bay* route, cleaning capacity does not significantly vary with snow level or stand type.

Subsequently, pairwise Mann-Whitney U tests were conducted for the groups that showed statistically significant results in the Kruskal–Wallis H tests, as presented in Table 6.6. Only those combinations for which statistically significant differences were found were treated as distinct capacity scenarios in Table 6.7, summarized as follows:

- For **Nabo** stands, three separate scenarios were defined: one for *cross-bay* routes, and two for *within-bay* routes, differentiated by snow level (low vs. high).
- For **Wibo** stands, two scenarios were defined, one for each route type, as snow level did not have a significant effect.
- **cross-bay routes** for Nabo and Wibo were combined into a single scenario, due to the absence of significant differences.

To define the final capacity scenarios, the median capacity was calculated for each statistically valid scenario. For merged scenarios, the average of the individual medians was used. Additionally, Low and High capacity scenarios were included to be able to account for weather forecast uncertainty in the forecasting model, consistent with the model logic of APOC's de-icing tool. The Low and High scenarios correspond to the 75th and 25th percentiles, respectively. The final capacity scenarios are presented in Table 6.7. While only statistically distinct scenarios were assigned separate capacities, all possible combinations are still included in the model with shared capacity values. This approach facilitates easy updates in the future if new data reveal additional significant effects.

Variables	Count	P-value	Test
C1. Nabo - route types - snow levels		< 0.001	Kruskal-Wallis H
C1.1. Nabo - within-bay - snow low vs. high		0.014	Mann-Whitney U
C1.2 Nabo - cross-bay - snow low vs. high		0.094	Mann-Whitney U
C2. Wibo - route types - snow levels		0.004	Kruskal-Wallis H
C2.1. Wibo - within-bay - snow low vs. high		0.546	Mann-Whitney U
C2.2. Wibo - cross-bay - snow low vs. high		0.086	Mann-Whitney U
C3. within-bay - snow levels - stand types		0.003	Kruskal-Wallis H
C3.1. within-bay - Snow Low - Nabo vs. Wibo		< 0.001	Mann-Whitney U
C3.2. within-bay - Snow High - Nabo vs. Wibo		0.845	Mann-Whitney U
C4. cross-bay - stand types - snow levels		0.072	Kruskal-Wallis H
C5. Snow depth low - stand types - route types		< 0.001	Kruskal-Wallis H
C5.1. Snow Low - Nabo - within-bay vs. cross-bay		< 0.001	Mann-Whitney U
C5.2. Snow Low - Wibo - within-bay vs. cross-bay		0.032	Mann-Whitney U
C6. Snow depth high - stand types - route types		< 0.001	Kruskal-Wallis H
C6.1. Snow High - Nabo: within-bay vs. cross-bay		0.002	Mann-Whitney U
C6.2. Snow High - Wibo: within-bay vs. cross-bay		0.006	Mann-Whitney U

Table 6.6: Statistical test results aircraft stand, route type and snow depth level combinations

Stand	Route	Snow	Capacity		
			Low	Middle	High
Nabo	within-bay	Low	7.2	4.9	3.2
Nabo	within-bay	High	5.7	3.7	1.9
Nabo	cross-bay	Low	3.3	2.2	1.8
Nabo	cross-bay	High	3.3	2.2	1.8
Wibo	within-bay	Low	5.5	3.4	2.0
Wibo	within-bay	High	5.5	3.4	2.0
Wibo	cross-bay	Low	3.3	2.2	1.8
Wibo	cross-bay	High	3.3	2.2	1.8

Table 6.7: Categorized capacities

6.5. Conclusion

This chapter identified the key operational and environmental factors influencing aircraft stand cleaning capacity during snow removal operations, thereby addressing the research question: ***"Which factors are shown by historical data to significantly affect aircraft stand snow removal capacity?"*** The main findings are summarized below:

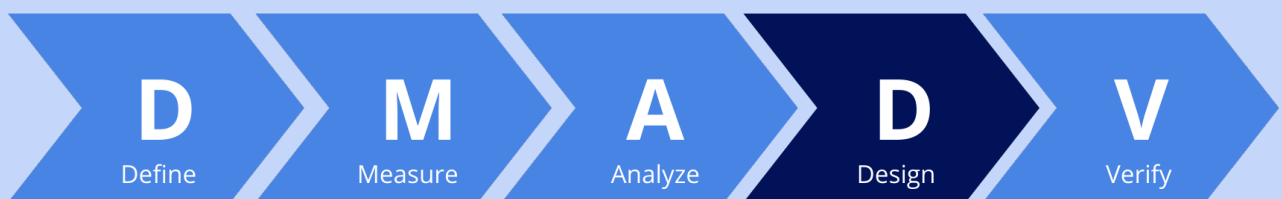
Key Insights

- Snow depth was the only weather variable with a statistically significant effect, with greater depths linked to lower capacity.
- Other weather variables showed no significant impact, likely due to limited variability in the dataset.
- Stand type (Nabo vs. Wibo) and route type (*within-bay* vs. *cross-bay*) significantly influenced capacity.
- The combined effect of snow depth, stand type, and route type revealed that only specific combinations significantly affect capacity; these were modeled separately, while non-significant variations were consolidated.

The capacity scenarios defined in Table 6.7 serve as input for the forecasting model developed in the subsequent Design phase.

IV

DESIGN



7

Forecasting model design

The objective of this research is to develop a forecasting model that predicts aircraft stand cleaning capacity based on anticipated weather conditions and operational factors. This model enables the evaluation of how available cleaning capacity aligns with the inbound flight schedule, potentially identifying queues and operational bottlenecks. It is designed to support decision-making during sector briefings, where the limiting capacity is determined and, based on this constraint, the inbound capacity is adjusted and flight cancellations are advised to airlines as necessary.

The following section first outlines the simulation framework, after which the final output is demonstrated using a virtual snow event.

7.1. Simulation design framework

The simulation is built using a discrete-time structure with 30-minute time slots, incorporating both situation-specific and operational input variables. The model logic is based on real-world constraints and historical performance data derived from the Measure & Analysis phase. The following subsections outline the design requirements, input parameters, and core assumptions, followed by a description of the final simulation design, illustrated by a flow diagram, and the model verification process.

7.1.1. Design requirements

Functional requirements

- The model must be focused on the operations within *centrum*.
- Users must be able to adjust situation-specific parameters (e.g., capacity restriction time windows, team numbers, route type ratio) without modifying the model code, making it applicable across various snowfall events and operational setups.
- The model must provide cleaning capacity estimates at sufficient temporal resolution to support slot-level planning decisions, e.g., 1-hour, 30-minute or 15-minute intervals.
- The simulation logic must realistically represent operational constraints, including the reassignment of flights to already cleaned stands where feasible, logical allocation of available capacity, utilization of surplus capacity, and incorporation of scheduled team breaks.
- The model should support rapid scenario analysis, delivering results within minutes to enable timely decision-making during operational briefings.
- The model should align with the logic of the existing de-icing model to ensure consistency and ease of use within the sector briefing, as discussed in Subsection 4.2.3.

Visualization requirements

- The model output must be intuitive and easy to interpret for operational stakeholders.
- The visual design must align with APOC's de-icing forecasting tool to ensure consistency and ease of use, as discussed in Subsection 4.2.3 and illustrated in Figure 4.9.

While maintaining logical and visual consistency with the de-icing tool is essential, enhancements that improve the model structure and output are encouraged.

KPIs

To ensure alignment with APOC's de-icing forecasting tool, the representation of KPIs is an essential design element. In addition to KPIs used in the de-icing tool, three supplementary indicators were added to provide a better understanding of the situation on airside during the snow event. The full list of KPIs is as follows:

KPIs aligned with the de-icing forecasting tool:

- Total number of queued aircraft, differentiated by Nabo and Wibo.
- Maximum queue length (aircraft per time slot)

Additional KPIs:

- Total number of time slots with queues
- Maximum waiting time experienced by an aircraft
- Number of aircraft experiencing a delay of more than 30 minutes

The KPIs adapted from the de-icing tool were modified to reflect the context of aircraft stand operations. The additional KPIs were selected based on an informal interview with the Service Owner Winter Operations, who specifically emphasized the operational relevance of the maximum waiting time. Delays exceeding 30 minutes likely trigger additional procedures, making this threshold a critical metric for supporting operational decision-making.

7.1.2. Input variables

The model incorporates two types of input variables: situation-specific inputs, which vary per snow event, outlined in Table 7.1, and operational inputs, which are only adjusted in the event of a structural change in the cleaning process, outlined in Table 7.2. Among these inputs is the cleaning capacity, which is treated as an operational input and varies based on the situation defined by the user with the situation-specific inputs.

Figure 7.1 displays the user input interface for situation specific variables. The date, end time of cleaning, model scenario and team availability can be specified using the designated widgets. The remaining input variables must be manually updated in the sheet displayed at the top of the model file. Operational inputs are located at the top of the modeling code for easy access.

7.1.3. Assumptions

The following assumptions are used in the model:

- CDF cleaning teams are assumed inactive during snowfall, resuming operations only after snowfall ends. Occasional assistance to *centrum* teams during snowfall is not modeled due to its unpredictability.
- Flight cancellations occur randomly within each time slot, following a fixed Nabo/Wibo ratio of 95%-5%, without incorporating any strategic or tactical decision-making logic. In practice, specific flight cancellations are decided by the airlines, following recommendations from APOC. For most airlines, the resulting distribution of canceled flights adheres this fixed (Informal interview APOC, March 2025).
- Flights assigned to a clean stand are excluded from the model.
- Capacity restrictions (runway, de-icing, additional) are hourly but split into two 30-minute slots. Any fractional capacity is rounded up for the first slot and down for the second slot.
- Route type is modeled as a fixed ratio of 55% *within-bay*-45% *cross-bay*, derived from historical data.
- Capacity is proportionally distributed between Nabo and Wibo stands based on the scheduled and queued flights per time slot.

Variable	Description
Inbound flight schedule	The flights scheduled during the simulation period, derived from CISS flight data
Model Scenario	Low, Middle, or High, reflecting weather uncertainty. The Middle scenario represents the baseline conditions, while the Low and High scenarios indicate more favorable or worse conditions than expected, respectively.
Time window(s) of forecasted accumulation	Predicted start and end time(s) of low and/or high accumulation levels.
End time of cleaning requirement	Predicted time when cleaning is no longer needed, for example, due to rain naturally clearing the stands.
Number of teams active at centrum	Number of teams operating at <i>centrum</i> throughout the snowfall period.
Number of teams active at CDF	Number of teams initially assigned to the CDF, who become available at <i>centrum</i> once snowfall ends.
Start time of first cleaning shift	Time when cleaning teams enter the Airside area to start the snow removal process. Based on this input, scheduled break times and potential subsequent shift start times are calculated.
Runway capacity restriction	Time window(s) during which inbound capacity is restricted, based on defined runway capacity scenarios.
De-icing capacity restriction	Time window(s) during which inbound capacity is restricted, based on predicted de-icing capacity.
Additional capacity restriction	Time window(s) with additional capacity limitations due to other operational constraints such as aircraft stand snow removal.

Table 7.1: User-defined situation-specific inputs

Variable	Values
Time slot interval	30 minutes
Maximal Inbound capacity per runway availability scenario	E: 10 flights/hour, D: 17 flights/hour, C: 35 flights/hour, B: 68 flights/hour
Runway clean time	40 minutes
Handling time per aircraft type	Nabo: 50 minutes, Wibo: 75 minutes
Break duration	35 minutes
Start time of first break	after 3 hours
Shift duration	8 hours
Route type ratio	within-bay: 55%, cross-bay: 45%
Situation-specific capacity	See Table 6.7

Table 7.2: Inputs reflecting operational assumptions

- The number of assignable and queued aircraft are defined as integers, while capacity is a fractional variable. This discrepancy often leads to positive or negative slack, which is carried over to the next time slot. Following consultation with the Service Owner Winter Operations, the integer threshold was set at 0.4 instead of 0.5.
- Excess capacity after snowfall ends (i.e., the "AS on demand" principle no longer applies) is used to clean future stands of the same type, or reallocated if only the other stand type remains.

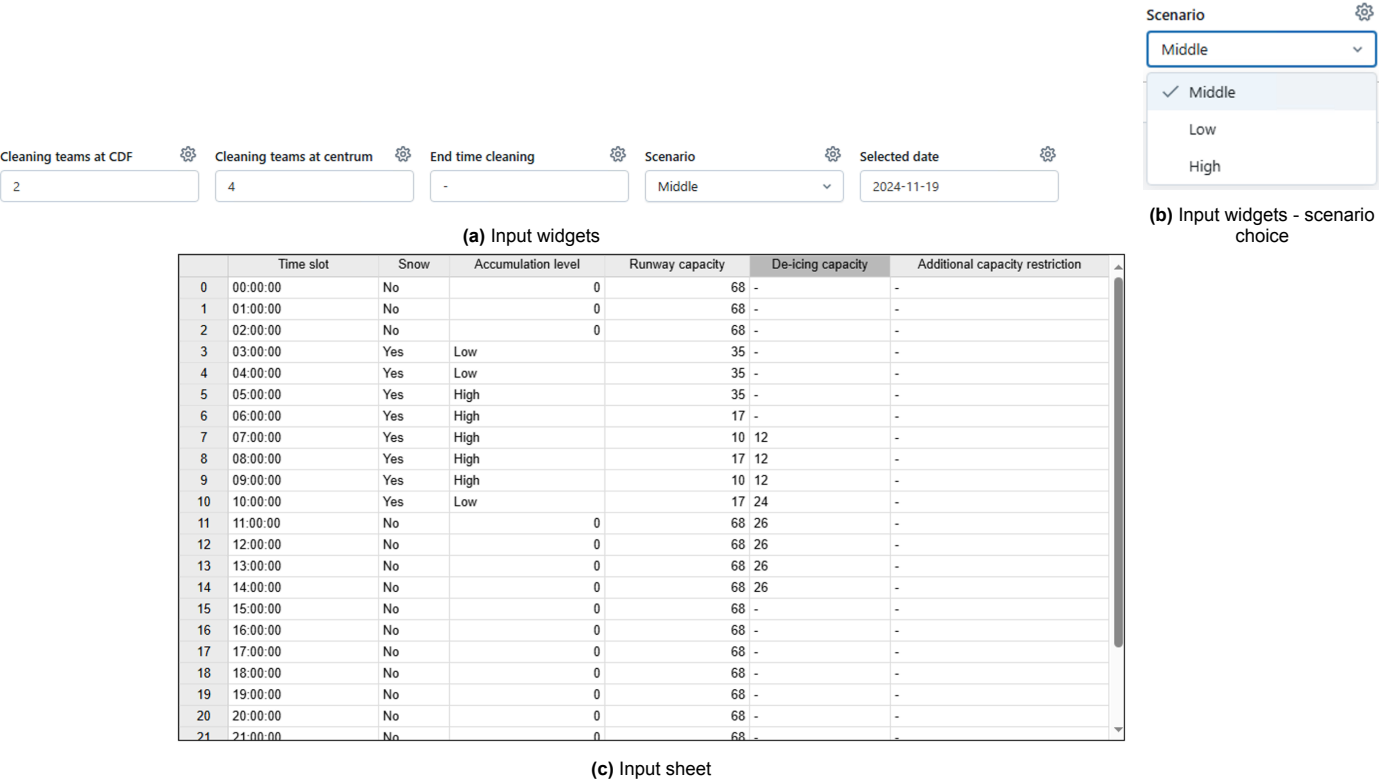


Figure 7.1: User interface of situation-specific input variables

- Aircraft arrival times are not individually modeled; instead, flights are grouped into their respective time slots. As a result, the precise waiting time per aircraft cannot be determined and is instead approximated based on slot-level timing.

7.1.4. Design

Figure 7.2 presents a comprehensive overview of the final model design. A detailed explanation of the diagram is provided in Appendix C.2 and the underlying python code is outlined in Appendix D.2. The flow diagram visualizes how time-dependent constraints such as capacity limits, team availability, and stand suitability interact to determine flight cancellations, cleaning assignments, and queue formation.

7.1.5. Model output

Figure 8.1 presents the output of the simulation model, demonstrating alignment with the design requirements outlined in Subsection 7.1.1. This visual representation illustrates the dynamic relationship between available cleaning capacity, the number of aircraft stands cleaned, and the queue of aircraft during a snow event. The x-axis represents time slots, while the y-axis indicates the number of aircraft assigned to stands that have not yet been cleaned. The graph distinguishes between Nabo and Wibo stands, displaying both assignable aircraft (solid bars) and queued aircraft (hatched bars) for each type. A bold line represents the total available cleaning capacity per time slot. Notably, the capacity does not always correspond to the solid bars of cleaned stands (assignable aircraft). This discrepancy arises from capacity slack and the utilization of Nabo capacity for Wibo stands (and vice versa), as explained in Subsection 7.1.3. The snowfall period is shaded in light blue to indicate reduced team availability during this time.

Beneath the graph, a separate bar chart shows the total number of queued aircraft per time slot, and the KPIs are displayed alongside the graph for quick reference.

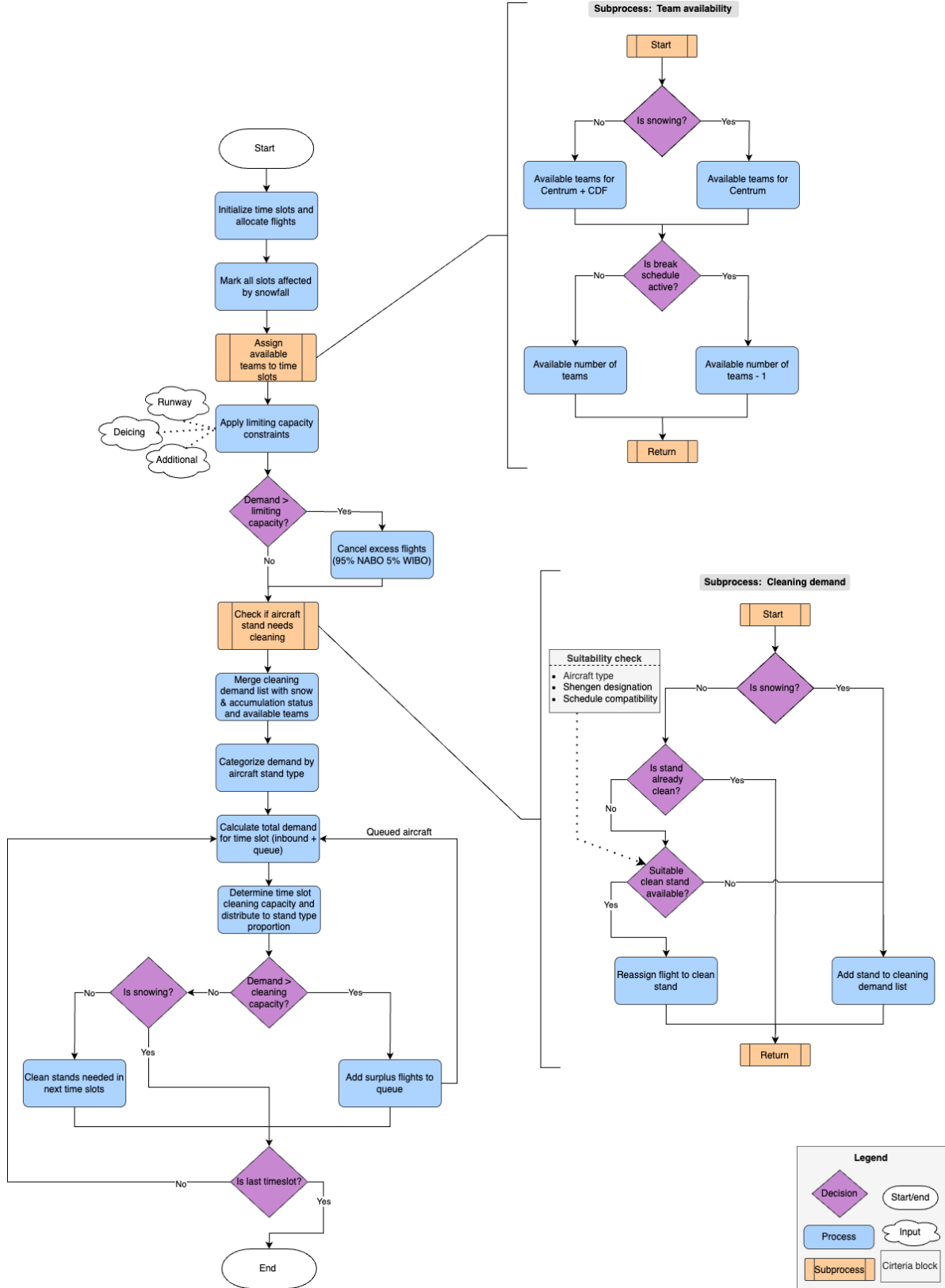


Figure 7.2: Simulation model logic

7.1.6. Model verification

To ensure that the simulation model was correctly constructed and implemented, model verification was performed. This involved assessing the internal consistency and correctness of the model's logic and computations. Several verification techniques were applied: code reviews were conducted to detect potential implementation errors, and step-by-step logging was used to trace intermediate outputs and verify alignment with expected behavior. Additionally, test runs using simplified and controlled input cases were compared against manually calculated outcomes to confirm computational accuracy. Extreme scenario testing was also employed to examine the model's response under edge conditions, such as complete unavailability of cleaning teams.

Together, these verification steps provide confidence that the model operates as intended. The validation of its practical performance is addressed in Chapter 8.

7.2. Forecasting tool demonstration

7.2.1. Current situation

The forecasting tool is demonstrated using a virtual snow situation, based on the situational context discussed during the sector briefing training in November 2024. The situation-specific input variables established during the briefing are summarized in Table 8.3. The conclusion of the sector briefing was that, between 06:00 and 11:00, the limiting factor would alternate between runway capacity and de-icing capacity. From 11:00 to 15:00, de-icing capacity was identified as the primary constraint. Aircraft stand capacity would not cause any problem.

Variable	Values
Scenario	Middle
Time window of forecasted snow	03:00 - 11:00
Time window of forecasted accumulation	03:00 - 05:00; low 05:00 - 10:00; high 10:00 - 11:00; low
End time of cleaning requirement	-
Number of teams active at centrum	4
Number of teams active at CDF	2
Runway capacity restriction	03:00 - 06:00; C 06:00 - 07:00; D 07:00 - 08:00; E 08:00 - 09:00; D 09:00 - 10:00; E 10:00 - 11:00; D
De-icing capacity restriction	07:00 - 10:00; 12 10:00 - 11:00; 24 11:00 - 15:00; 26
Additional capacity restriction	-

Table 7.3: Initial user-defined inputs virtual snow event

The output of this scenario is shown in Figure 8.1. It reveals that, during the snowfall period, there are typically 4 to 6 aircraft waiting in the field for an available stand. From 10:00 onwards, when runway and de-icing capacities increase rapidly to a restricted capacity of 24 flights per hour, with de-icing as the limiting factor, queues continue to grow, peaking at 9 aircraft. Moreover, the KPIs show that 2 aircraft have a waiting time exceeding the 30 minute threshold, with a maximum waiting time of 2 time slots. Overall, it can be concluded that this level of congestion poses a serious risk to operational continuity, potentially leading to significant delays and unplanned flight cancellations.

Contrary to the conclusions of the sector briefing, which stated that runway and de-icing capacity would be the primary constraints and stand capacity would not pose a problem, the model output suggests otherwise. It indicates that stand availability already leads to queuing from 06:30 onwards, with congestion escalating notably from 10:00. This finding reinforces the conclusion drawn in Section 4.4:

Due to limited data-driven insights into aircraft stand cleaning capacity, this factor is often overlooked in strategic planning, risking an overestimation of inbound capacity and the formation of aircraft queues on taxiways.

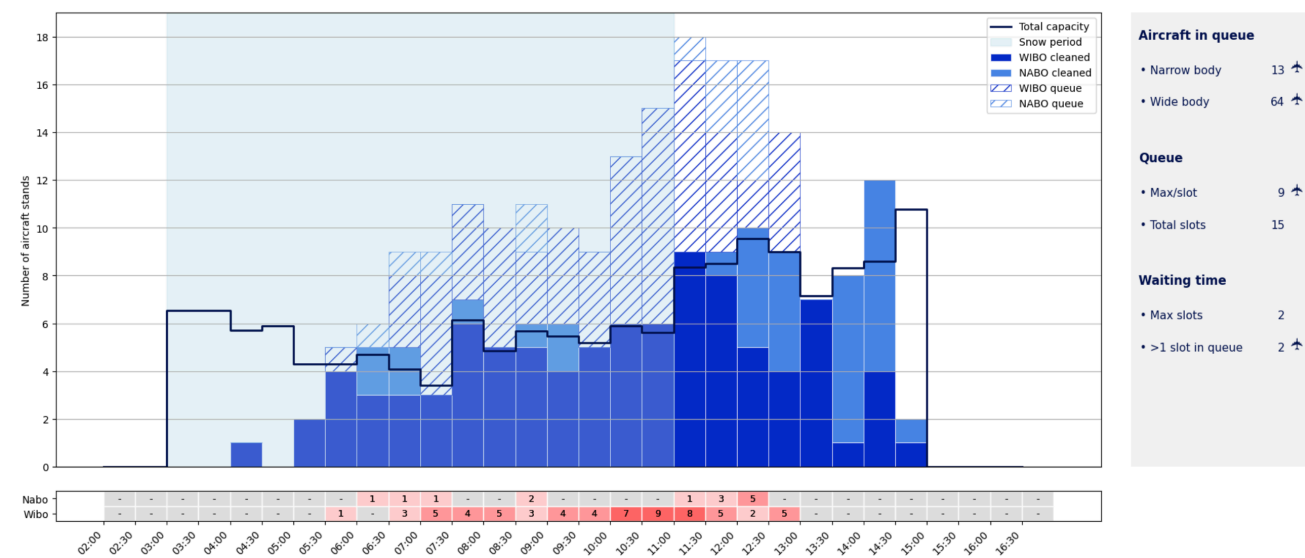


Figure 7.3: Model output virtual snow event without additional capacity restrictions

7.2.2. Strategic value of the model

The queues that emerge when aircraft stand cleaning capacity is not considered highlight the critical importance of the developed model as a strategic decision-support tool. By simulating operational scenarios under additional capacity constraints, the model empowers users to proactively explore and assess interventions aimed at reducing delays and alleviate congestion.

Through the input variable *Additional Capacity Restriction*, users can strategically test and optimize various capacity management strategies. For instance, applying a restriction of **12 flights per hour between 10:00 and 12:00** produces the output shown in Figure 7.4a. Compared to the baseline scenario (Figure 8.1), queues dissipate rapidly after 10:00, reducing both the total number of queued aircraft and the maximum queue length. However, one aircraft still experiences a waiting time of two slots.

Introducing an additional restriction of **12 flights per hour between 06:30 and 07:00**, thereby extending the existing de-icing constraint of 12 flights per hour between 07:00 and 10:00, yields the results shown in Figure 7.4b. This earlier intervention proves highly effective in mitigating early-morning queue formation, further reducing both the total number of queued aircraft and the maximum queue length. Importantly, it ensures that the maximum waiting time remains below the critical threshold of 30 minutes.

These results clearly demonstrate that even moderate, well-timed flight restrictions, informed by model simulations, can significantly reduce queuing and enhance operational efficiency. Overall, it can be concluded that:

The model enables users to assess cleaning capacity, identify bottlenecks, and evaluate operational constraints, thereby optimizing the **timing, intensity, and effectiveness** of interventions. Its output provides a robust foundation for data-driven capacity planning.

7.3. Beyond forecasting

Beyond its purpose to evaluate strategic capacity restrictions during snow events, the forecasting model offers valuable insights for a range of broader applications:

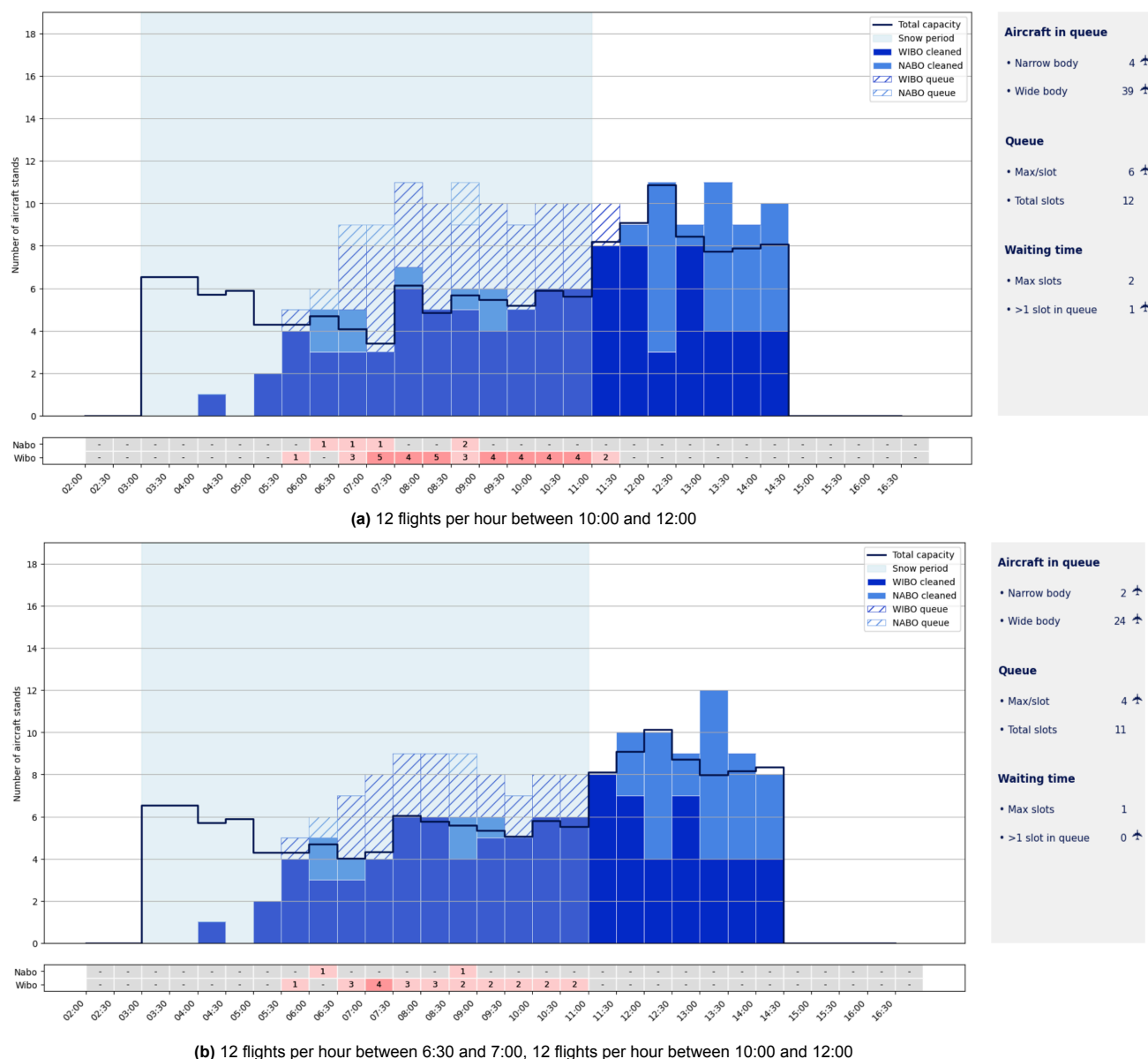


Figure 7.4: Model output virtual snow event with additional capacity restrictions

Structural operational changes:

The model enables users to simulate the effects of structural adjustments in key operational parameters. For instance, changes in the number of available cleaning teams or shifts in the *within-bay* to *cross-bay* route type ratio can be assessed. This functionality supports more informed, long-term resource planning and optimization of snow removal at aircraft stands.

Optimizing team deployment:

The model can support tactical improvements in workforce planning by revealing periods of over- and underutilization. For example, the simulation output in Figure 8.1 indicates limited cleaning activity between 03:00 and 06:00. Adjusting team schedules to start later could enhance efficiency by aligning deployment with peak operational times, thereby reducing idle periods during critical moments.

Through these additional applications, the model serves not only as an operational decision-support tool during snow events but also as a broader instrument for strategic optimization and long-term planning.

7.3.1. conclusion

In response to the sub-question ***"How can these insights (from the Measure and Analyse phases) be used to design a forecasting model that supports winter weather capacity planning under varying situations?"***, this chapter demonstrated how the key operational and environmental drivers identified through the data analysis can be integrated into a discrete-time simulation model. By incorporating situation-specific inputs and operational constraints, the model allows for flexible configuration and tailored scenario analysis, making it a robust tool for forecasting aircraft stand cleaning capacity under diverse conditions.

The model's output not only validates the importance of cleaning capacity in winter operations but also highlights its strategic value in supporting informed decision-making. The following insights summarize the model's impact:

Key Insights

Forecasting value:

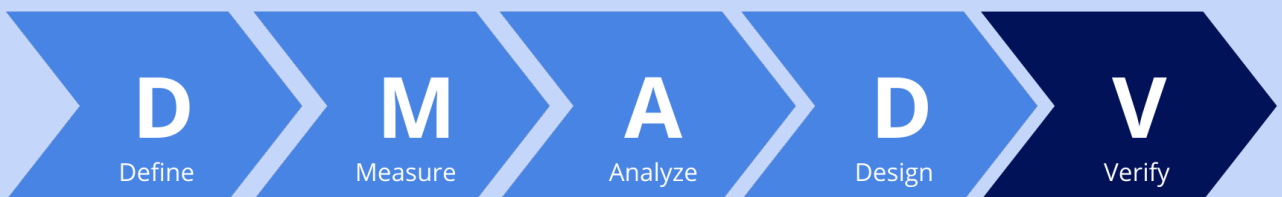
- Aircraft stand cleaning capacity is often overlooked in strategic planning, risking operational disruptions and the formation of queues on taxiways.
- The model provides a structured and transparent tool to assess cleaning capacity and identify potential bottlenecks.
- It enables users to test the timing, intensity, and effectiveness of additional capacity restrictions, supporting data-driven capacity management.
- Even minor, well-timed interventions in flight throughput can significantly reduce congestion, proving the value of proactive capacity planning.

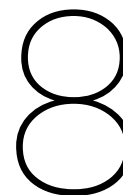
Beyond forecasting:

- The model supports long-term optimization by simulating structural changes in key variables, such as team size, workforce planning, and route type ratios.

V

VERIFY





Operational value assessment

This chapter presents the Verify phase of the DMADV framework, which focuses on validating the forecasting tool by evaluating whether it effectively addresses the problem defined in earlier stages and performs reliably under realistic operational conditions. To achieve this, three complementary validation approaches are used: a real-life scenario analysis that applies the model to an actual snow event at Schiphol Airport, a sensitivity analysis that evaluates the model's responsiveness to varying operational conditions, and peer reviews by operational stakeholders to assess usability, practical relevance, and alignment with existing decision-making processes. Together, these methods provide a comprehensive validation of the model's effectiveness and help determine its added value for winter operations at Schiphol Airport. The verification of the model is already discussed in subsection 7.1.6

8.1. Real life scenario analysis

The forecasting tool was validated using the snow event of January 5, 2025. The corresponding sector briefing output is presented in Figure 4.10 in Subsection 4.2.3. On this day, snowfall occurred between 06:00 and 08:00, followed by rain. The weather conditions were relatively mild, suggesting that the forecasting scenario would fall between the Middle and Low scenarios. As rain ensured rapid snow melting, cleaning was no longer necessary from 9:00 onwards. According to the sector briefing, the aircraft stand cleaning capacity was expected to be sufficient to meet the demand under these conditions.

Data analysis of this snow event showed that V4 started helping the teams at *centrum* from 7.45 onwards, and the *within-bay* to *cross-bay* ratio was 58%-42%. A total of five cleaning teams were available during the event, as the sixth team was excluded due to technical issues. The situation-specific input parameters used in the simulation are summarized in Table 8.1. Figure 8.1 presents the model output based on these scenario-based and operational conditions and Table 8.2 shows the actual performance.

Variable	Values
Scenario	Middle
Time window of forecasted snow	06:00 - 08:00
End time of cleaning requirement	9:00
Time window of forecasted accumulation levels	06:00 - 08:00: low
Number of teams active at centrum	4
Number of teams active at CDF	1
Runway capacity restriction	05:00 - 11:00; C
De-icing capacity restriction	05:00 - 09:00; 20 09:00 - 12:00; 26
Additional capacity restriction	-

Table 8.1: User-defined inputs on January 5, 2025

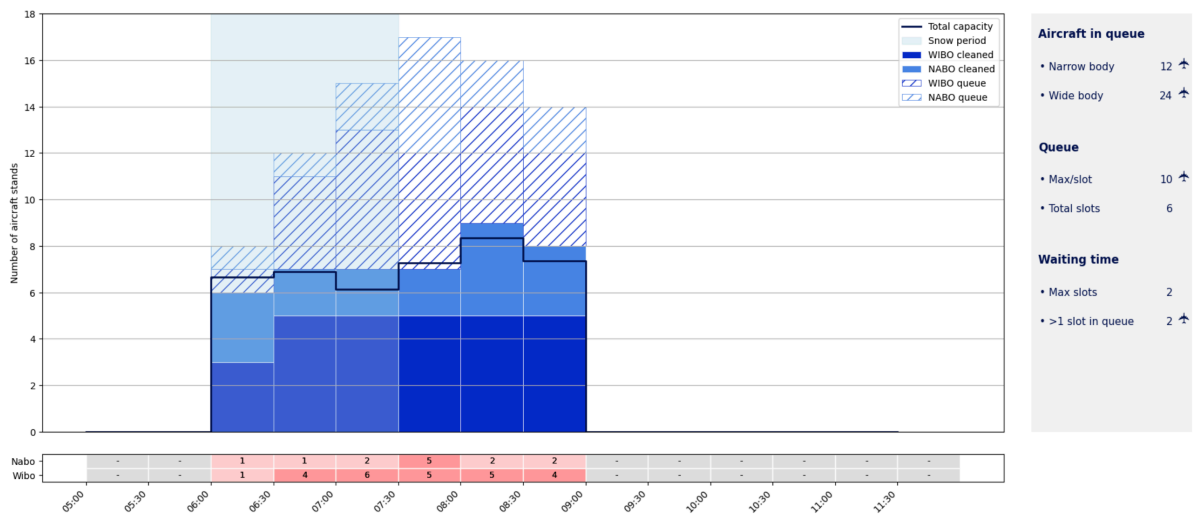


Figure 8.1: Model output of real life situation on January 5, 2025

KPI	Performance
Aircraft in queue	In total, 4 queued aircraft between 6:00 - 8:00 (2 Nabo, 2 Wibo).
Queue	A maximum of 2 aircraft per slot, 4 slots in total.
Waiting time	One aircraft experiences a waiting time of more than 30 minutes (2 slots).

Table 8.2: Actual performance on January 5, 2025 (Daily performance - Power BI, 2025)

The results indicate that the forecasting model predicted a more severe situation than what actually occurred when applying the Middle scenario. In contrast, applying the Low scenario resulted in the model predicting no queues. In reality, the actual performance was between the Middle and Low scenarios, indicating that the model was slightly conservative when using the Middle scenario.

Although this study did not find a significant impact of most weather conditions on cleaning capacity, this particular situation demonstrates that capacity exceeded the baseline scenario, likely due to favorable weather conditions. Additionally, the model is not able to capture the effect of rain. While the model predicts queues between 8:00 and 9:00, the actual performance data shows not such queues. This discrepancy can be attributed to the occurrence of rain, which significantly accelerated the cleaning process, as confirmed during the evaluation meeting of this snow event.

To validate the model output, the results were discussed with the Service Owner Winter Operations, who confirmed the plausibility of the outcomes. Based on his experience during the event, he agreed that the discrepancy between the model output and the actual situation was reasonable.

8.2. Sensitivity analysis

To assess the robustness and responsiveness of the forecasting model, a sensitivity analysis was performed by systematically varying key input parameters. Table 8.3 summarizes all scenarios along with their respective KPI values. The results show that team availability and the selected model scenario have the greatest impact on queue formation.

In scenario S1.1, where one fewer team is available (4 *centrum* + 1 CDF teams), the maximum waiting time increases to 3 slots, and the number of aircraft waiting more than one slot rises significantly, both concerning results. This scenario is highly realistic, as Section 5.2.1 showed that team availability during snow events often falls one team short of the plan. In scenario S1.2 (3 *centrum* + 1 CDF teams), the system becomes severely overloaded: the maximum queue per slot more than triples, queues occur during 24 time slots (12 hours), some aircraft have to wait up to 8 slots, and over 100 aircraft face delays exceeding 30 minutes, an alarming result. Conversely, increasing the availability to 5 *centrum* + 2 CDF almost eliminates delays, with a maximum of 3 queued aircraft per slot in only 5 slots, and no aircraft experience delays longer than 30 minutes. These findings underscore the critical role

of team availability in maintaining operational performance and minimizing delays during snow events.

The model scenarios addressing weather uncertainty also show significant effects. Under favorable conditions (S2.1 - Low), no queuing occurs, and only one aircraft experiences a brief delay. In sharp contrast, the High scenario (S2.2), representing worse-than-expected weather, leads to severe congestion, comparable to the impact of having two fewer teams available than planned.

Varying the share of *within-bay* versus *cross-bay* movements also significantly alters performance. As the proportion of *within-bay* routes increases (S3.1–S3.3), the total number of queued aircraft decreases, along with the maximum queue per slot and total queued slots. Notably, increasing the ratio of *within-bay* routes to 65% already ensures aircraft experience delays not longer than 30 minutes. On the contrary, a little decrease of *within-bay* route ratio (to 50%) directly increases the maximum waiting time to 3 slots, and lead to substantially higher total queued aircraft (90). This underscores the value of minimizing *cross-bay* travel.

Adjustments to break time (S4.1–S4.4) have only a marginal effect. Shorter breaks (30–25 min) slightly reduce queuing, while longer ones (40–45 min) modestly increase it. Notably, having slightly shorter breaks (30 minutes) already ensures that no aircraft experiences a delay longer than 30 minutes, an important KPI. The limited variation across scenarios is partly due to the model's use of 30-minute time slots, which rounds small changes in break duration to the same slot count, dampening their effect on capacity. While this simplification supports model consistency, it does not fully reflect real-world break behavior, where overlaps or partial slot usage may occur. Nevertheless, the results indicate that even a 20-minute difference in break time (25 vs. 45 minutes) has only a limited effect on system performance.

In conclusion, the sensitivity analysis confirms that the model responds logically to changes in input variables and provides valuable insight into operational decision-making. The most influential operational factors for improving system performance are team availability and routing strategy, while changes in break time are less effective as standalone measures.

Altered variables		Aircraft in Queue	Queue		Waiting time	
			Max/slot	Total slots	Max slots	> 1 slot
Base		77	9	15	2	2
S1.1	Nteams = 4 + 1	85	12	17	3	18
S1.2	Nteams = 3 + 1	116	33	24	8	105
S1.3	Nteams = 5 + 2	8	3	5	1	0
S3.1	Route type: 65%-35%	53	7	14	1	0
S3.2	Route type: 75%-25%	30	4	10	1	0
S3.3	Route type: 85%-15%	19	4	8	1	0
S3.4	Route type: 50%-50%	90	10	16	3	5
S4.1	Break time = 30	65	8	15	1	0
S4.2	Break time = 25	65	8	15	1	0
S4.3	Break time = 40	87	11	16	2	8
S4.4	Break time = 45	87	11	16	2	8
S2.1	Low	1	1	1	1	0
S2.2	High	116	37	24	9	110

Table 8.3: Results per validation-scenario

8.3. Peer reviews

Thirdly, the model was validated through peer review by two three stakeholder (groups) directly involved in snow operations and airport capacity planning.

The first review was conducted with the Service Owner Winter Operations, who initially commissioned the development of the tool. Throughout the development process, multiple iterations were made based on his feedback to ensure alignment with operational needs and constraints. The Service Owner acknowledged that the model output was both realistic and operationally credible. However, as demonstrated during validation with the real-life scenario, weather conditions can still influence outcomes, despite the statistical analysis not indicating a significant effect. Consequently, he emphasized the con-

tinued importance of considering the impact of weather conditions in decision-making. Furthermore, the Service Owner saw the model's ability to simulate different operational setups, such as team availability and routing strategies, as a key strength for improving snow removal operations. Additionally, he emphasized the tool's potential to enhance operational awareness of how stand cleaning capacity affects overall airport performance and to support more effective coordination of capacity-constrained resources during winter events.

The second peer review was conducted with APOC, the unit responsible for determining the airport capacity during winter weather operations, and one of the primary intended end-users of the forecasting tool. The review focused on the model's potential integration into APOC's decision-making processes. Representatives from APOC recognized the model's value in providing operational insight into how cleaning capacity influences overall airport performance and in supporting capacity planning during snow events, thereby improving the transparency of capacity-related decisions. They highlighted the usefulness of visualizing the expected impact of cleaning delays on inbound flow, which could inform decisions regarding total inbound capacity. Furthermore, they particularly appreciated the model's ability to simulate capacity based on user-specified operational configurations rather than relying on manual capacity calculations.

Lastly, the model was presented to the Winter Process Team during the evaluation session of winter 2024/2025. All team members recognized the tool's usability during sector briefings and emphasized its potential to support strategic optimization. This was underscored by the discussions sparked around increasing the *within-bay* ratio. Furthermore, the importance of scenario testing to validate the model was recognized.

8.4. conclusion

The verification phase addressed the following sub-question: ***“To what extent does the model provide accurate and usable capacity forecasts during simulated snow events?”***. The model evaluation revealed several important insights regarding its performance and practical applicability in an operational context. These key insights are summarized below:

Key Insights

- The real-life case study revealed that the model tended to be overly conservative under mild winter conditions, with actual performance falling between the Low and Middle scenarios.
- The model exhibited a logical response to variations in key input variables during the sensitivity analysis, indicating its robustness.
- Team availability and route type distribution were identified as critical operational factors influencing queue formation, suggesting potential levers for operational optimization.
- The choice of model scenario had a pronounced effect on outcomes, underscoring the importance of accounting for weather forecast uncertainty in capacity planning.
- Peer reviews confirmed the model's practical applicability, emphasizing its flexibility to simulate user-defined operational configurations as a key strength.

Conclusion and Discussion

This research investigated the snow removal at aircraft stands at Amsterdam Airport Schiphol, focusing on developing a data-driven model to support capacity planning during snowfall. The proposed model simulates snow removal operations under varying environmental and operational conditions, providing insights into optimal capacity management strategies. While the model was specifically designed for Schiphol, the insights and methodologies are applicable to other airports with similar operational setups. The conclusion of this research is presented in Section 9.1, followed by the discussion in Section 9.2.

9.1. Conclusion

The conclusion first addresses the main research question, followed by a discussion of the study's practical implications and scientific contributions.

9.1.1. Answer to the main research question

This research aimed to quantify aircraft stand cleaning capacity under diverse operational and environmental conditions and to develop a data-driven forecasting model to support capacity planning during winter operations. The primary research question was as follows: ***"How can a data-driven forecasting model be designed to improve the estimation of aircraft stand snow removal capacity and enhance airport capacity planning during snowfall?"***

To facilitate data analysis and model development, a swimlane diagram and causal diagram were constructed to map critical capacity-related data points and key influencing factors within the cleaning process. These diagrams formed the conceptual foundation for a classification algorithm that categorizes historical radar tracking data from each cleaning team into one of three operational phases: cleaning, traveling, or idle. These classifications enabled a systematic analysis of cleaning performance and the identification of key determinants of cleaning capacity. Results indicated that narrow body stands and travel routes within the bay substantially increase cleaning capacity. Snow depth was the only weather variable with a measurable impact, classified as low or high. Based on these three variables, scenario-specific cleaning capacities were defined to serve as input for the forecasting model. This approach significantly improves Schiphol's current theoretical calculation of aircraft stand snow removal capacity, estimated at 12 stands per hour with four active teams during heavy snowfall.

A discrete-time simulation model was subsequently developed to forecast stand cleaning capacity under anticipated operational and environmental conditions. The model design, presented in Figure 9.1, allows users to adjust scenario-specific input parameters, such as available teams and runway and de-icing capacity restrictions. By simulating how the available cleaning capacity aligns with the inbound flight schedule, the model helps identify potential queuing situations. Furthermore, the model allows users to evaluate the impact of additional capacity constraints or mitigation strategies aimed at reducing queues. This functionality supports proactive planning by optimizing the timing, intensity, and effectiveness of interventions. When combined with known runway and de-icing capacity forecasts, the tool significantly enhances data-driven capacity management, enabling more efficient and strategic

decision-making.

Feedback from operational stakeholders at Schiphol Airport confirmed the model's relevance and practical value during sector briefings. Stakeholders acknowledged its ability to improve situational awareness and inform capacity-related decision-making during snow events. Furthermore, the model was recognized for its potential to optimize the snow removal process strategically.

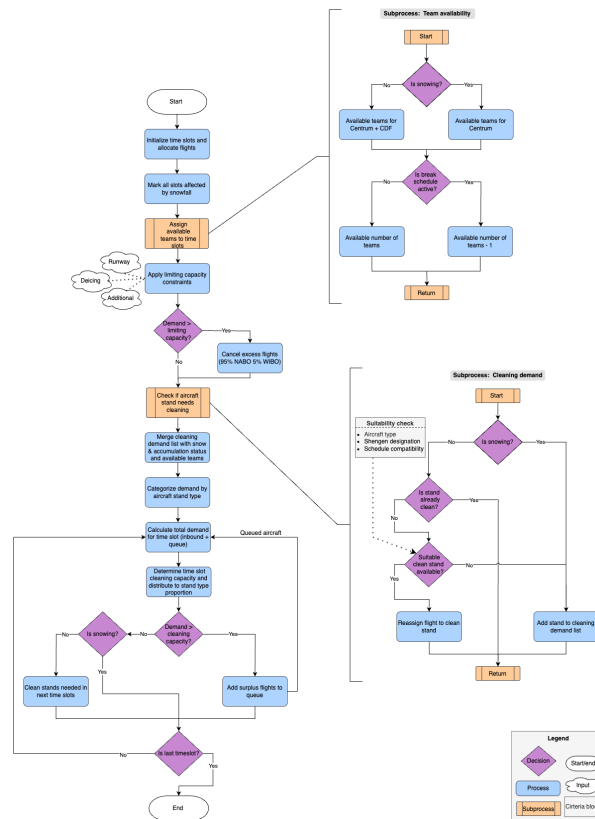


Figure 9.1: Design

9.1.2. Practical contribution

Schiphol Airport

The findings of this research have direct implications for snow operations and capacity planning at Schiphol Airport. The primary objective of the developed forecasting tool is to support decision-making during sector briefings held prior to snow events, where the airport's hourly inbound capacity is determined. This capacity is crucial for advising flight cancellations and ensuring smooth airport operations.

Currently, the hourly capacity is established by identifying the limiting factor among three key operational processes: expected runway availability (based on predefined runway scenarios), forecasted de-icing capacity (obtained from the de-icing planning tool), and estimated aircraft stand cleaning capacity. Previously, aircraft stand cleaning capacity was estimated using a simplified rule of thumb, assuming that four ASCTs could clean around 12 stands per hour. Due to the vague and non-data-driven nature of this estimate, cleaning capacity often received less attention compared to runway and de-icing limitations.

The forecasting tool developed in this study offers a more accurate, data-driven alternative. By simulating expected queues under varying environmental and operational conditions, the model can proactively identify whether stand cleaning might become the limiting factor during specific time windows. Integrating the tool with known runway and de-icing capacities allows decision-makers to foresee bottlenecks and apply targeted capacity restrictions, rather than reacting during real-time operations. This enables a more balanced and evidence-based approach to capacity determination, where all relevant components, runway, de-icing, and stand availability, are consistently considered.

Beyond forecasting, the model also serves as a strategic tool to assess the impact of various operational parameters and strategies, providing valuable insights for optimizing snow removal processes. One key finding from the sensitivity analysis is the significant impact of route type on cleaning capacity. This result reinforces the current operational strategy of keeping ASCT teams within their existing bays whenever possible, highlighting the importance of maintaining this practice during snow events. The analysis of historical route ratios, which show approximately 55% *within-bay* and 45% *cross-bay* movements, indicates that there is potential for optimization.

The model also demonstrates the substantial influence of team availability on cleaning capacity. The sensitivity analysis shows that reducing the number of active teams by just one significantly increases the number of queued aircraft and their waiting time. Given that historical data indicates one fewer team was active than planned in 50% of cases, ensuring optimal staffing levels during snow operations is crucial.

Potential for generalization

Although the forecasting tool was developed specifically for Schiphol Airport, the underlying methodology, combining historical operational data with scenario-based forecasting, is highly transferable to other airports that experience similar winter weather conditions. It is particularly relevant for airports that deploy dedicated teams for aircraft stand snow removal as part of their winter operations strategy.

To ensure accurate, airport-specific forecasting, it is recommended that each airport conducts its own analysis of historical data to determine cleaning capacities and define appropriate capacity scenarios. Since operational procedures, infrastructure, and available resources can vary considerably across airports, the influence of factors such as snow depth, stand type, or route layout may differ, and additional factors may also be relevant. This supports the findings of Myers et al. (2012), who concluded that a universal relationship between winter weather conditions and snow removal capacity is unlikely to exist, as airports respond differently to similar events.

Most user-defined scenario inputs and operational constraints are broadly applicable across different airport contexts. Those that are specific to Schiphol, such as the distinction between *centrum* and CDF teams or predefined runway scenarios, can be easily adapted to other airport environments. Moreover, route types may vary depending on the airport layout. For instance, it may be necessary to differentiate between cross-terminal and inter-terminal routes.

Finally, the added value of the forecasting tool depends on the airport's layout and runway configuration. Schiphol's multi-runway system allows for up to two runways to be used for inbound traffic during peak periods, making aircraft stand availability a more critical bottleneck during snow events. In contrast, airports with simpler layouts or lower traffic volumes, such as Eindhoven Airport, which operates a single runway in mixed-mode, are less likely to encounter stand-related capacity bottlenecks. At such airports, the relatively lower volume of inbound flights and reduced routing complexity may allow for easier coordination of snow removal, reducing the operational benefit of a detailed aircraft stand capacity forecasting tool.

In summary, although the forecasting tool was tailored to Schiphol's specific operational environment, its modular and flexible design makes it well-suited for adaptation to other airport contexts, particularly those with comparable layouts and dedicated snow removal teams. With appropriate customization of scenario inputs and operational parameters, the tool can offer meaningful support for winter operations and capacity planning across a wide range of airport settings.

9.1.3. Scientific contribution

This study makes several novel contributions to the academic literature on winter operations at airports. It is the first to focus explicitly on snow removal at aircraft stands, a critical yet underexplored aspect of winter maintenance. Prior research has largely concentrated on runways and taxiways, overlooking the operational bottlenecks caused by limited stand availability during snowfall events.

The study introduces a novel methodological framework that combines radar-based vehicle tracking data with a classification algorithm to identify cleaning activities and estimate cleaning capacity. While the capacity estimates are tailored to Schiphol Airport, the underlying methodology is transferable to other airports employing radar systems to monitor ground vehicle movements. Notably, this research

expands the scope of radar data usage from the common focus on aircraft tracking to ground support operations, offering a new direction for performance evaluation of airside operations.

In addition, this study contributes to the scarce literature on airport capacity forecasting during winter weather. Although one earlier study explored this topic, its scope was confined to runways and taxiways, neglecting the impact of stand availability. This research highlights the critical importance of incorporating stand cleaning into capacity forecasting, highlighting its relevance for maintaining smooth airport operations during snowfall.

Finally, the study addresses a methodological gap by introducing quantitative weather classification thresholds to standardize snow condition categories. This resolves inconsistencies seen in previous research, where qualitative or loosely defined definitions hindered the comparability and reproducibility of findings across different studies and operational contexts.

Overall, the study contributes both academically and practically to the field of airport winter operations. It presents a flexible, data-driven model that enhances winter weather capacity planning at Schiphol Airport while also offering potential for adaptation at other airports employing similar snow removal strategies. Additionally, the study advances the academic literature by addressing the previously overlooked aspect of stand cleaning in capacity forecasting, thereby broadening the understanding of critical factors influencing airport operations during snowfall.

9.2. Discussion

9.2.1. Limitations

While this study offers valuable insights and a practical forecasting tool, several limitations should be acknowledged.

One limitation arises when classifying operational periods, where some degree of misclassification may persist despite efforts to address noisy data, particularly when identifying breaks outside the R and Y platforms. Additionally, the weather conditions observed on the analyzed snow days exhibited limited variability, constraining the ability to assess the impact of more severe weather conditions. Consequently, this limitation led to non-significant effects for most variables, despite both the literature and the current state analysis emphasizing the influence of specific weather conditions. Moreover, the real-life case study indicated that the model tended to be overly conservative under mild winter conditions. This observation suggests that weather conditions may indeed affect operational performance, even though the statistical analysis did not reveal a significant impact.

Furthermore, several influential factors, such as stand occupancy, airside traffic, and human factors like staff experience and cooperation were beyond the scope of this study, despite their likely impact on operational performance.

To maintain operational usability, several simplifying assumptions were made during the simulation. For example, CDF-dedicated teams were assumed to be unavailable for *centrum* support during snowfall, and route type was incorporated as a fixed *within-bay* - *cross-bay* ratio rather than being dynamically derived based on team positioning. While these assumptions help maintain clarity and efficiency in the simulation, they may not fully reflect the complexity of real-world operations.

Lastly, pavement spraying operations, essential for rendering stands operational, were excluded from the scope of this study. Because sprayers are typically fewer in number than ASCTs and therefore operate independently, they were not included in the algorithm for identifying cleaning cycles. However, they can be a source of capacity limitation themselves. For example, during the snow removal operations on 5 January 2025, it was observed that spraying was a major bottleneck, with only a single sprayer available to treat all aircraft stands (Informal interview Winter performance meeting, 2025). This highlights a relevant operational factor not accounted for in the current model.

9.2.2. Recommendations

Future research

The findings and limitations of this study point to several promising directions for future research aimed at deepening the understanding of aircraft stand snow removal capacity and improving the forecasting

model.

First, continuously updating the model with data from new snow events would strengthen its predictive performance, particularly under a wider range of weather conditions. As more diverse weather patterns are captured, the model could offer deeper insights into the influence of meteorological factors on snow removal operations.

Secondly, future research could explore the impact of additional factors identified in the causal diagram (Figure 4.14). Incorporating historical flight data could enhance the model by providing insights into dynamic stand occupancy levels and traffic flow. Lower stand occupancy may increase the *within-bay* ratio by facilitating simultaneous cleaning, while high traffic flow may extend travel times between bays due to congestion. Additionally, human factors, such as staff experience and cooperation, could be investigated through observational studies, structured interviews, or comprehensive sensitivity analyses.

Third, refining specific simulation components could enhance the model's accuracy and realism. Among the simplifying assumptions outlined in Subsection 7.1.3, the static treatment of route type stands out as particularly impactful. Future work could address this by dynamically modeling route type based on task assignments, considering the position of each ASCT and the location of its next assigned stand. Integrating such routing logic would allow the simulation to more accurately represent operational behavior.

Lastly, the observed bottleneck in pavement spraying capacity on January 5, 2025, underscores the importance of integrating aircraft stand spraying into the capacity analysis. Specifically, future versions of the model should consider spraying time as part of the total cycle time. Given that historical radar data for sprayers is available, the developed classification algorithm could be used to identify spraying periods by adjusting some threshold values and logic. Ultimately, enabling the number of available sprayers to be defined as an input parameter, similar to the number of ASCTs, would provide a more realistic capacity forecast.

Schiphol airport

A key recommendation for Schiphol Airport is to carefully consider potential weather impacts when specifying model inputs, as they may still influence snow removal operations. In particular, it is important to take into account the overall weather context rather than focusing solely on individual variables, as the broader weather conditions are likely to shape operational outcomes.

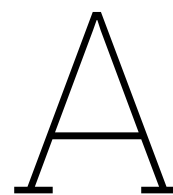
To further optimize snow removal performance, it is recommended to enhance *within-bay* movements during operations to increase efficiency and to ensure that the optimal number of teams is consistently available, as this is essential for maintaining effective operations during snow events.

Lastly, testing the model in real-life situations or through additional historical scenarios would enhance model validation and provide valuable insights into forecasting accuracy. Analyzing discrepancies between predicted and actual outcomes can help identify areas where model assumptions or parameters may require adjustment. Moreover, it can reveal specific weather conditions under which the model tends to overestimate or underestimate capacity, thereby refining the model's practical application.

References

- ACI EUROPE. (2024). Quality_of_Network.
- Adikariwattage, V., Wirasinghe, S. C., De Barros, A., & Ruwanpura, J. (2012). Fleet Optimization for Aircraft Deicing Trucks. *Proceedings, Annual Conference - Canadian Society for Civil Engineering*, 2, 1505–1514. <https://www.researchgate.net/publication/236612550>
- Aircraft Operations Schiphol. (2024). *VOP-coördinatie, verplaatsing VOP-team* (tech. rep.).
- Airport Operation Center Schiphol. (2024). *Trainingen Sectorbriefing APOC* (tech. rep.).
- Alsalous, O., & Hotle, S. (2024). Deicing Facility Capacity and Delay Estimation Using ASDE-X Data: Chicago O'Hare Simulation Case Study. *Transportation Research Record*, 2678(4), 455–467. <https://doi.org/10.1177/03611981231185147>
- Bolsius, J., & Scholten, D. (2024). Werkboek Sneeuw & Gladheid.
- Bolsius, J., van den Ham, J., & Scholten, D. (2023). Werkboek S&G, definitief versie 2023-10-3-1.
- Bryman, A. (2016). *Social Research Methods* (tech. rep.). Oxford University Press.
- Fernández, C. M., Gómez, V. F., & Arnaldo, R. M. (2016). Mathematical model for optimising the sequence for clearing snow from the manoeuvring area during winter operations. *Journal of Advanced Transportation*, 50(6), 1225–1251. <https://doi.org/10.1002/atr.1399>
- Friso, H. F., Richard, C., Visser, H. G., Vincent, T., & Bruno, D. (2018). Predicting abnormal runway occupancy times and observing related precursors. *Journal of Aerospace Information Systems*, 15(1), 10–21. <https://doi.org/10.2514/1.1010548>
- Janic, M. (2009). Modeling Airport Operations Affected by a Large-Scale Disruption. *Journal of Transportation Engineering*, 135(4), 206–216. <https://doi.org/10.1061/ASCE0733-947X2009135:4206>
- Keis, F. (2014). WHITE - Winter hazards in terminal environment: An automated nowcasting system for Munich Airport. *Meteorologische Zeitschrift*, 24(1), 61–82. <https://doi.org/10.1127/metz/2014/0651>
- Klein-Paste, A. (2018). Airplane braking friction on dry snow, wet snow or slush contaminated runways. *Cold Regions Science and Technology*, 150, 70–74. <https://doi.org/10.1016/j.coldregions.2017.02.004>
- KNMI. (n.d.). KNMI waarschuwingen.
- Koščák, P., Berežný, Š., & Ferenc, J. (2012). Optimization Operations of Winter Maintenance of Airport. *International Journal of Traffic and Transportation Engineering*, 2012(3), 40–45. <https://doi.org/10.5923/j.ijtte.20120103.02>
- Koščák, P., Berežný, Š., Vajdová, I., Koblen, I., Ojciec, M., Matisková, D., & Puškáš, T. (2020). Reducing the negative environmental impact of winter airport maintenance through its model design and simulation. *International Journal of Environmental Research and Public Health*, 17(4), 1296. <https://doi.org/10.3390/ijerph17041296>
- Majumdar, R., & Selvi, K. (2014). Six Sigma-Overview of DMAIC and DMADV. <https://www.researchgate.net/publication/346081379>
- Merkert, R., & Mangia, L. (2012). Management of airports in extreme winter conditions-some lessons from analysing the efficiency of Norwegian airports. *Research in Transportation Business and Management*, 4, 53–60. <https://doi.org/10.1016/j.rtbm.2012.06.004>
- Mirmohammadsadeghi, N., Hu, J., & Trani, A. (2019). Enhancements to the runway capacity simulation model using the asde-x data for estimating airports throughput under various wake separation systems. *AIAA Aviation 2019 Forum*, 1–16. <https://doi.org/10.2514/6.2019-3044>
- Mohammadi, N., Klein-Paste, A., & Lysbakken, K. R. (2024). Simulating winter maintenance efforts: A multi-linear regression model. *Cold Regions Science and Technology*, 227, 104307. <https://doi.org/10.1016/j.coldregions.2024.104307>
- Myers, T., Andrews, M., & Krozel, J. (2012). Winter weather airport capacity model. *12th AIAA Aviation Technology, Integration and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. <https://doi.org/10.2514/6.2012-5535>

- Pačaiová, H., Korba, P., Hovanec, M., Galanda, J., Šváb, P., & Lukáč, J. (2021). Use of simulation tools for optimization of the time duration of winter maintenance activities at airports. *Sustainability (Switzerland)*, 13(3), 1–15. <https://doi.org/10.3390/su13031095>
- Pan, Y. L., Jhang, J. C., & Gu, F. R. (2022). Using Advanced Surface Movement Guidance and Control System for Kaohsiung Xiaogang Airport Aircraft's Route Assignment Planning Based on Petri nets. *Proceedings of the 2022 8th International Conference on Applied System Innovation, ICASI 2022*, 160–165. <https://doi.org/10.1109/ICASI55125.2022.9774454>
- Preis, H., & Fricke, H. (2020). Optimizing Winter Maintenance Service at Airports, 765–771. https://doi.org/10.1007/978-3-030-48439-2_{ }93
- Schiphol. (2018). *Winteroperatie: vervanging SNLV 1 & 2 Betrouwbare capaciteit op Amsterdam Airport* (tech. rep.).
- Schiphol. (2024). *Facts and figures* (tech. rep.).
- Schiphol Group. (2024a). *BASIS.9520 Checklist Winteroperatie* (tech. rep.).
- Schiphol Group. (2024b). *BASIS.9520 Proces opstart Winteroperatie Sneeuw* (tech. rep.).
- Schiphol Operations. (n.d.). Download plattegronden.
- Service Owner Sneeuw en Gladheid. (2023). *Winteroperatie: Sneeuw & Gladheid*.
- Shu-Ling, W., Rong-Hua, Q., Chun-Fu, S., De-Xini, Z., & Hui, Z. (2011). Study on assessment model of road snow removal capacity. *Proceedings - 4th International Conference on Intelligent Computation Technology and Automation, ICICTA 2011*, 1, 1134–1138. <https://doi.org/10.1109/ICICTA.2011.285>
- Srivastava, A. (2011). Improving Departure Taxi Time Predictions Using ASDE-X Surveillance Data. *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*, 1–14. <https://doi.org/10.1109/DASC.2011.6095989>
- Šváb, P., Korba, P., Albert, M., & Kolesár, J. (2019). Information system to support the management of winter airport maintenance. *NTinAD 2019 - New Trends in Aviation Development 2019 - 14th International Scientific Conference, Proceedings*, 170–173. <https://doi.org/10.1109/NTAD.2019.8875565>
- Zhang, Y. P., Zhang, Y., Xing, X. Q., Chen, Y., Yang, N., & Mao, J. (2022). The Method of Division and Classification Quantification for Snow and Ice Removal on Airport Pavement. *Lecture Notes in Electrical Engineering*, 775, 649–664. https://doi.org/10.1007/978-981-16-5429-9_{ }50



Scientific paper

Forecasting Aircraft Stand Snow Removal Capacity

A Case Study at Amsterdam Airport Schiphol

Lara de Geus

Delft University of Technology, The Netherlands

Abstract

Winter weather can significantly disrupt airport operations, leading to flight delays, cancellations, and substantial economic losses. Proactively forecasting airport capacity during snowfall is essential to ensure timely flight cancellations and maintain operational continuity. While aircraft stand availability is a key determinant of overall airport capacity, data-driven insights into this process are lacking. This study addresses this gap by developing a forecasting model to predict aircraft stand snow removal capacity, using Amsterdam Airport Schiphol as a case study. A classification algorithm is employed to extract operational cleaning activities, cleaning, traveling, and idle periods, from historical radar data, enabling the assessment of current cleaning performance. Based on these insights, a simulation-based model was developed to forecast stand cleaning capacity under anticipated operational and environmental conditions. The model's flexible design allows users to adjust scenario-specific input parameters, such as team availability and capacity restrictions, facilitating tailored scenario analysis. By simulating the alignment between available cleaning capacity and inbound flight schedules, the model identifies potential queuing situations and facilitates the assessment of additional capacity restrictions, supporting more informed capacity planning.

Keywords: *Winter operation, Aircraft stand cleaning, Airport capacity, winter weather capacity planning, Capacity forecasting model, Ground radar data.*

I Introduction

With the growing demand for air travel, minimizing operational downtime at airports has become increasingly critical for maintaining efficiency (Fernández et al., 2016; Pačaiová et al., 2021; Šváb et al., 2019). At Amsterdam Airport Schiphol, for example, a single day of airport closure can incur approximately €2.3 million in costs for the airport itself and between €40 and €80 million for airlines (Schiphol, 2018). Winter weather is a significant contributor to such disruptions, often resulting in flight delays, cancellations, and substantial economic losses. This highlights the need for effective winter maintenance strategies, to manage and mitigate the operational and financial impacts of adverse winter conditions (Merkert & Mangia, 2012; Pačaiová et al., 2021; Šváb et al., 2019).

To ensure continuity of operations and the safety of passengers, airport personnel, and flights, airports implement vari-

ous winter maintenance activities, including de-icing of aircraft and snow removal on critical infrastructure such as runways, taxiways, and aircraft stands. Although extensive research has been conducted on snow removal from runways and taxiways and on aircraft de-icing procedures, the clearing of aircraft stands has received limited attention (see Section II). Yet aircraft stands are essential components of the airside infrastructure, serving as key nodes for boarding, disembarking, and baggage handling. Inadequate snow removal at stands can delay inbound and outbound flight operations, thereby reducing overall airport capacity. Gaining a deeper understanding of aircraft stand snow removal performance is therefore essential to improve winter resilience at airports.

Therefore, the objective of this study is to quantify aircraft stand snow removal capacity and to develop a forecasting model that predicts cleaning performance under varying operational and environmental conditions, using Amsterdam Airport Schiphol as a case study. The central research question guiding this study is:

"How can a data-driven forecasting model be designed to improve the estimation of aircraft stand snow removal capacity and enhance airport capacity planning during snowfall?"

To address this question, the study follows the Define-Measure-Analyze-Design-Verify (DMADV) methodology, a structured approach derived from Lean Six Sigma that is particularly suited for developing new processes or products (Majumdar & Selvi, 2014). Several tools are applied within each research stage. Desk research of key internal researches combined with literature review and (informal) interviews formed the basis for the problem definition and current state analysis. Through the development of a classification algorithm and subsequent data analysis, current performance is assessed and key determinants of capacity are defined. The model is build using discrete-event simulation, where demand and capacity are calculated at discrete time steps. Finally, the validation of the model is done by scenario testing and informal interviews.

The paper is structured according to the DMADV phases, systematically guiding the reader. The first two sections present the introduction of the problem (Section I), supported by conclusions drawn from the literature (Section II). Section III details the current state analysis, and Section IV the classification algorithm development and subsequent exploratory data analysis. In Section V, the determination of key determinants of cleaning capacity is employed. Section VI outlines

the model design, including the simulation framework and the model demonstration. Section VII evaluates the model's operational value through real-life scenario analysis, sensitivity testing, and stakeholder feedback. Finally, Section VIII concludes the study by summarizing key findings, discussing limitations, and suggesting avenues for future research.

II Literature review

As part of the *Define* phase, a literature review was conducted to build a deeper understanding of the current state of knowledge and identify research gaps relevant to this study.

Effective winter operations are crucial for maintaining airport reliability and minimizing disruptions during adverse weather. Research on aircraft de-icing and snow removal from runways and taxiways is relatively well developed, with a primary focus on optimizing fleet size (Preis & Fricke, 2020), routing strategies (Preis & Fricke, 2020), snow removal sequences (Fernández et al., 2016; Zhang et al., 2022), and vehicle allocation (Koščák et al., 2020). In contrast, snow removal at aircraft stands remains largely unexplored, with only a brief mention in the study by Fernández et al. (2016). As stand cleaning priorities are driven by flight arrival schedules, their work concentrated on optimizing cleaning sequences for runways, taxiways, and aprons, without addressing the specific complexities associated with aircraft stand operations.

Snowfall can significantly reduce airport capacity and cause widespread delays (Merkert & Mangia, 2012). There is one study by Myers et al. (2012) that researches the impact of snowfall on airport capacity by developing a Winter Weather Airport Capacity Model that incorporates snow-clearing speed and weather variables to predict departure rates. However, the model focuses exclusively on runway snow removal and neglects the impact of stand cleaning. Given that inefficiencies in stand snow removal can lead to congestion, parking shortages, and cascading flight delays, understanding these operations is essential for accurately predicting airport capacity during winter weather.

Existing research highlights that snow removal performance is influenced by multiple factors, which can be categorized into six key groups: preparation, resources, airport-specific factors, environmental conditions, operational efficiency, and human factors. Effective preparation involves structured winter maintenance plans, regular staff training, and preventive measures such as the application of de-icing agents (Pačaiová et al., 2021; Shu-Ling et al., 2011; Šváb et al., 2019). Resources, including the availability, reliability, and technical performance of snow removal vehicles, as well as staff availability and the number of teams active, also strongly impact operational effectiveness (Koščák et al., 2020; Pačaiová et al., 2021; Preis & Fricke, 2020). Operational efficiency is shaped by airport layout, flight schedules, traffic volume, vehicle operating speeds, and area prioritization strategies (Fernández et al., 2016; Janic, 2009; Preis & Fricke, 2020). In addition, human factors such as communication and staff experience are critical determinants of snow removal capacity, with variability in cleaning duration

often attributed to differences in personnel expertise (Merkert & Mangia, 2012; Pačaiová et al., 2021; Shu-Ling et al., 2011)

Several studies emphasize the dependence of airports on accurate weather forecasts to enable proactive and scenario-specific winter maintenance planning (Pačaiová et al., 2021; Šváb et al., 2019). Snow properties, including type (dry, wet, slush), accumulation rate, and density, directly affect clearance difficulty (Myers et al., 2012), equipment selection (Koščák et al., 2020), cleaning efficiency (Janic, 2009; Koščák et al., 2020), and vehicle operating speeds (Fernández et al., 2016; Preis & Fricke, 2020). However, many models rely on qualitative classifications (e.g., "light," "moderate," "heavy" snowfall) without clearly defined quantitative thresholds, limiting comparability and generalizability of findings across different studies and airports.

Another important research gap is the limited use of surveillance data in winter maintenance research. While surveillance systems are widely used to analyze aircraft taxi times and runway operations (Mirmohammadsadeghi et al., 2019; Pan et al., 2022; Srivastava, 2011), their application to winter operations remains limited. Only Alsalous and Hotle (2024) use surveillance data to evaluate de-icing operations, combining this data with airport layout data to visualize surface movements and extract de-icing event details. A dedicated algorithm analyzes individual speed profiles to distinguish between operational phases (traveling, de-icing, and waiting). While their approach shares methodological similarities with the present study, this research centers on specialized ground vehicles rather than aircraft movements, providing a new perspective on how surveillance data can be applied to winter maintenance operations.

Taken together, the literature reveals a clear gap in modeling and quantifying snow removal at aircraft stands and its impact on airport capacity. This study addresses this gap by applying radar surveillance data to analyze snow removal vehicle operations, offering a novel application of such data. It also introduces quantitative snow condition thresholds to overcome the lack of standardized classifications and to enhance generalizability. Lastly, the literature review identifies offers a detailed exploration of the factors influencing snow removal capacity, forming the foundation for the conceptual model used in the data analysis.

III Current state analysis

The Measure phase focused on gaining a comprehensive understanding of current snow removal operations at Schiphol Airport. To achieve this, the process was analyzed chronologically, encompassing the key preparatory and operational activities associated with each phase, as visualized in Figure 1.

During snowfall, Schiphol's operational capacity must be proactively adjusted to account for reduced infrastructure availability and to ensure safe and manageable airport operations. Capacity limits are determined during sector briefings, where the most constraining factor, either runway capacity, de-icing capacity, or aircraft stand capacity, sets the limit for the hourly

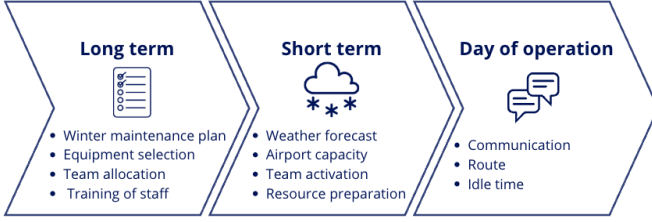


Figure 1: Chronological phases of snow removal process

airport capacity. Based on this assessment, airlines are advised to cancel flights as needed to align demand with available capacity.

While runway capacity has been predefined for various scenarios (Figure 2) and de-icing capacity is forecasted using a dedicated model (Figure 3), the capacity for aircraft stand snow removal remains poorly quantified and lacks a solid empirical foundation. This gap complicates accurate capacity forecasting during snow events and increases the risk of operational mismatches. Underestimating stand availability can lead to queues of aircraft awaiting cleared stands, causing delays and, in some cases, unforeseen flight cancellations. Therefore, aircraft stand snow removal capacity requires systematic investigation and data-driven modeling (Airport Operation Center Schiphol, 2024; Bolsius & Scholten, 2024).

This study focuses on Aircraft Stan Clearing Teams (ASCTs) 4 through 9, which are primarily responsible for snow removal at "centrum" (aprons A through H) and the Central de-icing Facility (CDF). Typically, ASCTs 4 and 9 focus on clearing the CDF during snowfall and assist the remaining teams once snowfall has ceased and the CDF has been cleared. Task assignment is coordinated by Apron Control Inbound, based on the Gate Planner's prioritization. During snowfall, the "AS on demand" principle is applied: stands are only cleared when an aircraft is scheduled to arrive within 15 minutes. To minimize travel time and enhance efficiency, ASCTs are kept within the same bay (the area within the apron) whenever possible (Bolsius & Scholten, 2024; Service Owner Sneeuw en Gladheid, 2023).

Building on insights from the literature review and the current state analysis, a causal diagram was developed (Figure 5), illustrating the key factors influencing aircraft stand snow removal and their interdependencies. The diagram is specifically tailored to Schiphol's operational context, with certain factors from the literature omitted or reinterpreted to reflect local practices. It forms the conceptual foundation for classifying operational activities, including traveling, cleaning, and idle time, and for identifying the key variables affecting cleaning performance.

IV Current cleaning performance

Building on the current state analysis, the study further examines current performance by developing a classification algorithm and conducting an exploratory data analysis.

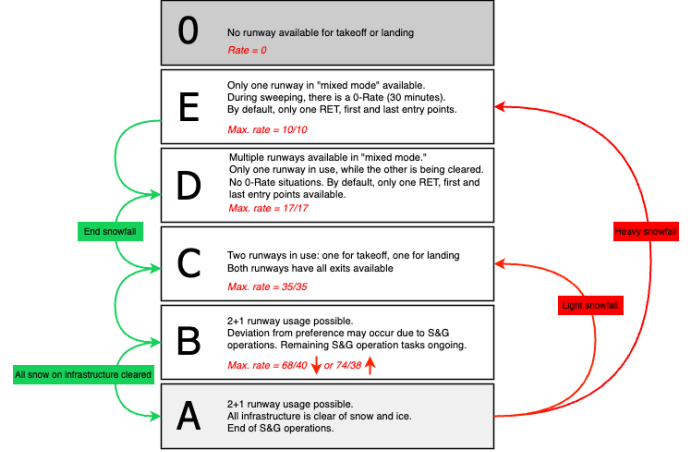


Figure 2: Winter Runway Availability Scenario (Service Owner Sneeuw en Gladheid, 2023)

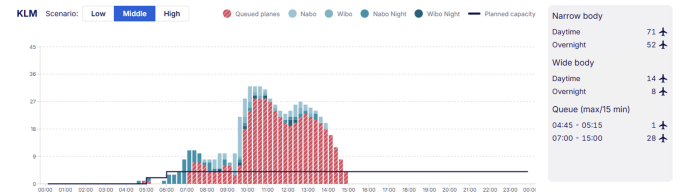


Figure 3: Output de-icing tool

Classification algorithm

To determine aircraft stand snow removal capacity, an algorithm was developed to classify each second of radar tracking data into one of three operational phases: cleaning, traveling, or idle time. Ground radar data from Schiphol Airport (2017–2025), combined with Geofence polygon information, served as the primary input. Data from 2021 were unavailable, however, this is not considered a limitation, as snow events during that year coincided with the COVID-19 lockdown, significantly disrupting normal airport operations and making these events atypical for comparison.

Cleaning activities were identified when vehicles were located at an aircraft stand, with corrections for minor GPS drift during cleaning. Cross-referencing with the CISS and VGRS information systems was used to verify whether adjacent stands had also been cleaned but were not directly captured by the algorithm. Traveling was classified when vehicles moved between stands, while idle periods were detected as cleaning periods at designated break locations.

The algorithm addresses missing or inaccurate data, including absent location records (NaN values), GPS inaccuracies, and missing timestamps, using a combination of look-ahead checks, proximity validation, and time-gap thresholds. Minimum and maximum cleaning and travel durations were defined based on box plot statistics under boundary-free conditions and operational experience, ensuring that anomalies such as excessively long or short activities were corrected or excluded.

Verification and validation were performed through manual cross-checking with CISS and VGRS records, as well as manual annotation of radar data. Although iterative refinements improved the algorithm’s robustness, some limitations remain, particularly in accurately distinguishing cleaning periods from breaks, as teams may take breaks at locations other than the designated break areas.

The classified operational phases formed the basis for calculating cleaning capacity and analyzing the impact of operational and environmental factors.

Exploratory data analysis

The following bullet points summarize key insights from historical data on aircraft stand cleaning at Schiphol Airport, focusing on team availability, task durations, routing patterns, and cleaning capacity. These findings provide a baseline understanding of operational performance and identify factors that influence the efficiency of snow removal activities.

- On 50% of the snow operation days, one team fewer than the intended maximum was available.
- Idle time varied considerably in both frequency and duration; a 35-minute break per 8-hour shift is assumed.
- Wide-body stands required longer cleaning times compared to narrow-body stands.
- Consecutive cleaning tasks within the same bay were associated with significantly shorter travel times compared to *cross-bay* operations, which involves crossing one or multiple bays. While travel time generally increased with the number of bays crossed, shifts involving two to five bays occurred far less frequently than single-bay shifts.
- Despite efforts to keep teams within their current bay, the proportion of *within-bay* to *cross-bay* operations has remained stable at approximately 55%–45% since 2017.
- Cleaning time was identified as the dominant contributor to the total cleaning cycle duration, which includes both the travel time to a task and the task’s cleaning time.
- Capacity, defined as the number of cleaning cycles that can be completed per hour, showed a right-skewed distribution, peaking between two and three cycles per hour.

V Determinants of cleaning capacity

In the Analyze phase, the relationship between key variables and aircraft stand cleaning capacity was systematically examined. Given the non-normal distribution of capacity data, non-parametric statistical tests, including Spearman rank correlations and Mann-Whitney U tests, were applied. Alongside the operational factors identified in the Measure phase, weather variables were included using minute-level KNMI data from multiple weather stations at Schiphol. These variables were

aggregated to align with individual cleaning cycles, and classifications such as snow intensity, snow type, and precipitation level were defined.

The analysis revealed that only snow depth significantly affected cleaning capacity, with greater accumulation linked to reduced cleaning capacity. To enable its use in the forecasting model, snow depth was categorized into two levels: low (< 50 mm) and high (≥ 50 mm). Other weather variables and classifications showed no significant influence, likely due to limited variability across the observed snow events. Regarding operational factors, stand type and route type were found to significantly impact capacity, while the number of bay shifts did not.

To define cleaning capacity across varying operational and environmental conditions, the combined effect of the three relevant variables was evaluated. This resulted in four distinct capacity scenarios:

- For Nabo stands, three scenarios were defined: one for *cross-bay* routes, and two for *within-bay* routes, differentiated by snow depth (low vs. high).
- For Wibo stands, two scenarios were defined based on route type, as snow level did not have a significant effect.
- *Cross-bay* routes for Nabo and Wibo were combined into a single scenario, due to the absence of significant differences.

These scenarios result in the final capacities presented in Table 1. The capacity labels Low, Middle, High correspond to the 75th, 50th, and 25th percentiles of observed capacity, respectively. The Middle values serve as the default scenario, while the Low and High values are included to meet the model’s design requirement for scenario flexibility.

Stand	Route	Snow	Capacity		
			Low	Middle	High
Nabo	within-bay	Low	7.2	4.9	3.2
Nabo	within-bay	High	5.7	3.7	1.9
Nabo	cross-bay	Low	3.3	2.2	1.8
Nabo	cross-bay	High	3.3	2.2	1.8
Wibo	within-bay	Low	5.5	3.4	2.0
Wibo	within-bay	High	5.5	3.4	2.0
Wibo	cross-bay	Low	3.3	2.2	1.8
Wibo	cross-bay	High	3.3	2.2	1.8

Table 1: Categorized capacities

VI Model design

During the Design phase, the forecasting model aimed at estimating aircraft stand cleaning capacity across a range of operational and environmental conditions was developed and verified.

VI.1 Simulation design framework

Figure 6 shows the model design, visualized using a flow diagram. It outlines how time-dependent constraints, such as capacity limits, team availability, and stand suitability, interact to determine flight cancellations, cleaning assignments, and the formation of queues. The model is developed based on the following design requirements:

Functional requirements

- The model must be focused on operations within *centrum*.
- The user must be able to adjust input parameters without modifying the model code, making it applicable across various snowfall events and operational setups
- The model must provide cleaning capacity estimates at sufficient temporal resolution to support slot-level planning decisions, e.g., 1-hour or 30-minute intervals.
- The simulation logic must realistically represent operational constraints, including the reassignment of flights to already cleaned stands where feasible, logical allocation of available capacity, utilization of surplus capacity, and incorporation of scheduled team breaks.
- The model should support rapid scenario analysis, delivering results within minutes to enable timely decision-making during operational briefings. The model should align with the logic of the existing de-icing model to ensure consistency and ease of use within the sector briefing. This includes the possibility to switch between a Low, Medium, and High scenario.

Visualization requirements

- The model output must be intuitive and easy to interpret for operational stakeholders.
- The visual design must align with APOC's de-icing forecasting tool to ensure consistency and ease of use, as illustrated in Figure 3.

While maintaining logical and visual consistency with the de-icing tool is essential, enhancements that improve the model structure and output are encouraged.

KPIs

To ensure alignment with APOC's de-icing forecasting tool, the representation of KPIs is an essential design element. The KPIs adapted from the de-icing tool were modified to reflect the specific context of aircraft stand cleaning operations. Additionally, three new KPIs were selected based insights from an informal interview with the Service Owner Winter Operations, who specifically emphasized the operational relevance of the maximum waiting time. Delays exceeding 30 minutes typically trigger additional procedures, making this threshold a

critical metric for for operational decision-making. The KPIs are presented in the model output shown in Figure 4.

Input parameters

The model incorporates two types of adjustable input variables: situation-specific inputs, which vary per snow event (exemplified in Table 2) and operational inputs, which are based on insights from the Measure and Analysis phases and are only updated in response to structural changes in the cleaning process (Table 3).

Variable	Values
Scenario	Middle
Time window of forecasted snow	03:00 - 11:00
Time window of forecasted accumulation levels	03:00 - 05:00; low 05:00 - 10:00; high 10:00 - 11:00; low
End time of cleaning requirement	-
Number of teams active at centrum	4
Number of teams active at CDF	2
Runway capacity restriction	03:00 - 06:00; C 06:00 - 07:00; D 07:00 - 08:00; E 08:00 - 09:00; D 09:00 - 10:00; E 10:00 - 11:00; D
De-icing capacity restriction	07:00 - 10:00; 12 10:00 - 11:00; 24 11:00 - 15:00; 26
Additional capacity restriction	-

Table 2: Initial user-defined inputs virtual situation

Variable	Values
Time slot minutes	30 minutes
Maximal Inbound capacity per runway availability scenario	E: 10 flights/hour, D: 17 flights/hour, C: 35 flights/hour, B: 68 flights/hour
Runway clean time	40 minutes
Handling time per aircraft type	Nabo: 50 minutes, Wibo: 75 minutes
Break duration	35 minutes
Start time of first break	after 3 hours
Shift duration	8 hours
Route type weights	Within-bay: 55%, Cross-bay: 45%
Situation-specific capacity	See Table 1

Table 3: Inputs reflecting operational assumptions

Figure 4 presents the output of the forecasting model, capturing the dynamic relationship between available cleaning capacity, the number of aircraft stands cleaned, and the queue of aircraft awaiting a cleaned stand.

Assumptions

The model is based on the following key assumptions:

- CDF cleaning teams are assumed inactive during snowfall, resuming operations only after snowfall ends. Occasional assistance to *centrum* teams during snowfall is not modeled due to its unpredictability.
- Flight cancellations occur randomly within each time slot, following a fixed Nabo/Wibo ratio of 95%-5% (Informal interview APOC, March 2025).
- Flights assigned to a clean stand are excluded from the model.
- Aircraft arrival times are not individually modeled; instead, flights are grouped into their respective time slots, approximating waiting times.
- Capacity restrictions (runway, de-icing, additional) are hourly but split into two 30-minute slots. Any fractional capacity is rounded up for the first slot.
- Route type is modeled as a fixed ratio of 55% *within-bay* and 45% *cross-bay*, based on historical data.
- Capacity is proportionally distributed between Nabo and Wibo stands based on the scheduled and queued flights per time slot, rather than using a fixed ratio.
- Integer rounding for queued aircraft uses a threshold of 0.4, as advised by the Service Owner Winter Operations.
- Excess capacity after snowfall ends is used to clean future stands of the same type, or reallocated if only the other stand type remains.

VI.2 Strategic value of the model

The forecasting tool is demonstrated using the virtual snow scenario discussed during the sector briefing training in November 2024. The situation-specific input variables established during the briefing are summarized in Table 2. During the session, it was concluded that between 06:00 and 11:00, the limiting capacity factor would alternate between runway capacity and de-icing capacity, while from 11:00 to 15:00, de-icing capacity would serve as the primary constraint. Aircraft stand capacity was not anticipated to present a limitation.

The model output of this scenario is presented in Figure 4a. In contrast to the conclusions drawn during the sector briefing, the model results reveal that stand availability already leads to queuing from 06:30 onwards, with congestion intensifying notably after 10:00. Moreover, the KPIs show that 2 aircraft have a waiting time exceeding the 30 minute threshold, with a maximum waiting time of 2 time slots. This level of congestion poses a substantial risk to operational continuity, potentially resulting in significant delays and maybe even unplanned flight cancellations. This finding reinforces the conclusion drawn in Section III that the limited data-driven insights into aircraft

stand cleaning capacity often lead to being overlooked in strategic planning, thereby risking an overestimation of inbound capacity and the subsequent formation of aircraft queues.

The emergence of queues when stand cleaning capacity is disregarded underscores the strategic importance of the developed model as a decision-support tool. By simulating operational scenarios under varying capacity constraints, the model enables users to proactively explore and evaluate interventions aimed at reducing delays and mitigating congestion. Through the input variable *Additional Capacity Restriction*, users can strategically test and optimize different capacity management strategies. For example, applying a restriction of **12 flights per hour between 10:00 and 12:00** results in the output shown in Figure 4b. Compared to the baseline scenario (Figure 4a), the queues dissipate rapidly after 10:00, reducing both the total number of queued aircraft and the maximum queue length. These results demonstrate that even moderate, well-timed flight restrictions, can significantly enhance stand availability and operational efficiency.

Overall, the model provides a robust framework for assessing cleaning capacity, identifying bottlenecks, and evaluating operational constraints. Its outputs enable users to optimize the timing, intensity, and effectiveness of interventions, offering a strong foundation for data-driven capacity planning.

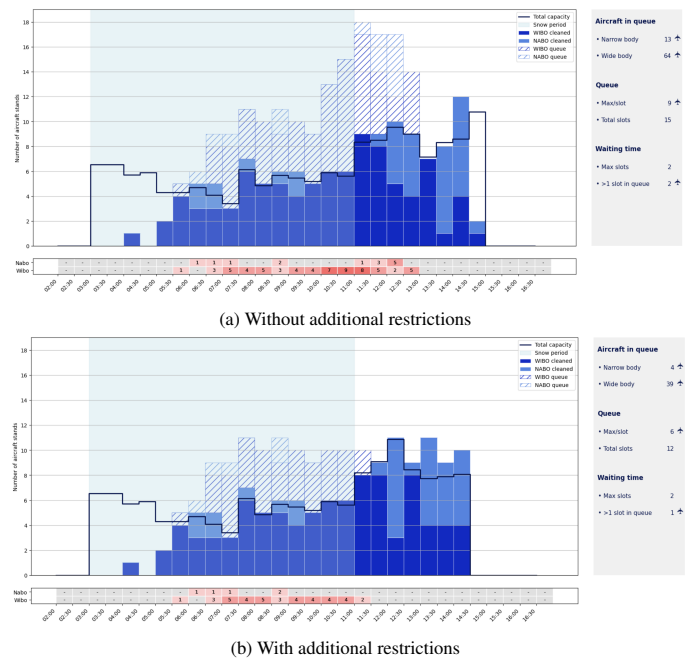


Figure 4: Model output

Beyond its primary function of evaluating strategic capacity restrictions during snow events, the forecasting model provides valuable insights for a range of broader operational applications. It enables the simulation of structural changes in key operational parameters, such as variations in the number of available cleaning teams or adjustments in the ratio between *within-bay* and *cross-bay* route types. Additionally, the model supports tactical workforce management by identifying periods

of over- and underutilization, thereby enhancing overall team deployment efficiency. By facilitating scenario analyses of operational adjustments, the model serves not only as an operational decision-support tool but also as a broader instrument for optimizing long-term aircraft stand cleaning strategies.

VII Operational value assessment

The Verify phase focuses on assessing whether the forecasting tool effectively addresses the defined operational problem and performs reliably under real-world conditions. Three complementary methods were applied: (1) a real-life scenario analysis of the January 5, 2025 snow event, (2) a sensitivity analysis of key input variables, and (3) peer reviews with operational stakeholders.

Real-life scenario

The real-life scenario involved mild snowfall between 06:00 and 08:00, immediately followed by rain. The model, using the Middle scenario, predicted a more severe situation than what actually occurred. Although this study did not find a significant impact of most weather conditions on cleaning capacity, this particular situation demonstrates that capacity was slightly higher than the baseline scenario due to the favorable conditions. Especially when the rain accelerated snow melting and improved cleaning performance, the model deviated from actual performance. Actual performance fell between the Middle and Low scenarios, with the Low scenario predicting no queuing. The Service Owner Winter Operations reviewed the discrepancy and confirmed the model's conservative tendency under these conditions.

Sensitivity analysis

Sensitivity analysis demonstrated that team availability was the most influential factor (see Table 4). Removing one team, eliminating the CDF double shift but maintaining centrum capacity, only slightly increased queuing. However, when centrum capacity was reduced (e.g. two teams were unavailable), the queue nearly tripled and delays occurred in nearly all time slots. This scenario is realistic, as no CDF double shift existed before winter 2024/2025, and Section IV showed *centrum* capacity was reduced in 50% of past snow events. The model scenarios addressing weather uncertainty also show significant effects, ranging from no queuing in the Low scenario, to extreme queuing in the High scenario.

Route type distribution also had a strong impact, with higher shares of *within-bay* movements reducing queuing, while changes in break duration had limited effect due to the model's rounding of break time into fixed slots. Overall, the model responded logically to input variations, confirming its robustness and decision-support value.

Peer reviews

Peer reviews reinforced the model's operational value, with the Service Owner of Winter Operations validating the credibil-

ity of the model outputs and highlighting its potential to increase awareness of how stand cleaning capacity affects overall airport performance. APOC representatives emphasized the model's utility for capacity planning, particularly its ability to visualize the impact of cleaning delays on inbound flow and to automatically compute capacity from user-defined configurations. They considered it as a promising addition to the sector briefing. Furthermore, members of the Winter Process Team recognized the tool's practical usability during sector briefings and emphasized its potential to support strategic optimization efforts.

In conclusion, the model has proven to be a reliable forecasting and decision-support tool. It provides actionable insights, enhances situational awareness, and supports data-driven planning, making it a valuable asset for improving winter operational resilience at Schiphol Airport.

VIII Conclusion

Key findings

The objective of this study was to quantify aircraft stand snow removal capacity and to develop a forecasting model to predict cleaning performance under varying operational and environmental conditions. This addresses the central research question: *"How can a data-driven forecasting model be designed to improve the estimation of aircraft stand snow removal capacity and enhance airport capacity planning during snowfall?"*

A classification algorithm was developed using radar data to identify cleaning activities and quantify cleaning performance. Key determinants of cleaning capacity, such as stand type, route type, and snow depth, were identified, leading to scenario-based capacity estimations, thereby improving current calculation of aircraft stand snow removal capacity. These capacities, along with parameters and constraints from the current state and performance analysis, served as inputs for the discrete-time simulation model.

The forecasting model enables users to simulate stand cleaning operations based on scenario-specific inputs, assessing how available cleaning capacity aligns with the inbound flight schedule to identify potential queuing situations. Additionally, it allows users to evaluate the impact of capacity constraints or mitigation strategies aimed at reducing queues. This functionality supports proactive capacity planning by optimizing the timing, intensity, and effectiveness of interventions, facilitating more informed decision-making.

While tailored to Schiphol, the model structure is adaptable to other airports, particularly those with comparable layouts and dedicated snow removal teams. While Schiphol-specific inputs, such as the distinction between *centrum* and CDF teams or predefined runway scenarios, were included in this version, they can be easily adapted to suit other airport environments.

Besides practical relevance for Schiphol Airport, this research makes several novel contributions to winter airport operations research, focusing on the underexplored area of aircraft stand

snow removal, particularly in capacity forecasting during winter weather. The study expands the scope of radar data usage from the common focus on aircraft tracking to ground support operations, offering a new direction for performance evaluation of airside operations. Moreover, it provides clear quantitative weather classification thresholds to standardize snow conditions categories, resolving inconsistencies in current literature.

Limitations and recommendations

Despite offering valuable insights and a practical forecasting tool, limitations remain. The weather dataset lacked variability, limiting the ability to assess the impact of more severe weather and leading to non-significant effects for most variables. As the real life scenario showed, the model might be too severe for very mild conditions. Additionally, influential factors such as stand occupancy, airside traffic, and human elements like staff experience and coordination were beyond the study's scope, though they likely affect performance.

To keep the simulation operationally useful, simplifying assumptions were made. For example, CDF teams were assumed unavailable for centrum support during snowfall, and route types were fixed based on Nabo/Wibo ratios rather than dynamic team positions, which may not fully reflect operational complexity. Finally, spraying operations, essential for rendering stands operational, were excluded from the analysis. Their limited availability can be a bottleneck, as observed on 5 January 2025, when operating with a single sprayer significantly delayed operations.

Future research should focus on enhancing model realism by dynamically modeling route types based on ASCT locations and incorporating operational factors such as stand occupancy, airside traffic, and human elements, using historical flight data and qualitative methods. Additionally, integrating spraying operations would provide a more comprehensive and realistic assessment of total cycle time and capacity.

For Schiphol Airport, continuously updating the model with data from new snow events is strongly recommended to improve predictive performance, especially under varied weather conditions. As the model captures more diverse patterns, it can offer deeper insights into the impact of meteorological factors on snow removal operations.



Figure 5: Factors influencing the snow removal process

Altered variables		Aircraft in Queue	Queue		Waiting time	
			Max/slot	Total slots	Max slots	< 1 slot
Base		77	9	15	2	2
S1.1	Nteams = 4 + 1	85	12	17	3	18
S1.2	Nteams = 3 + 1	116	33	24	8	105
S1.3	Nteams = 5 + 2	8	3	5	1	0
S3.1	Route type: 65%-35%	53	7	14	1	0
S3.2	Route type: 75%-25%	30	4	10	1	0
S3.3	Route type: 85%-15%	19	4	8	1	0
S3.4	Route type: 50%-50%	90	10	16	3	5
S4.1	Break time = 30	65	8	15	1	0
S4.2	Break time = 25	65	8	15	1	0
S4.3	Break time = 40	87	11	16	2	8
S4.4	Break time = 45	87	11	16	2	8
S2.1	Low	1	1	1	1	0
S2.2	High	116	37	24	9	110

Table 4: Results per validation-scenario

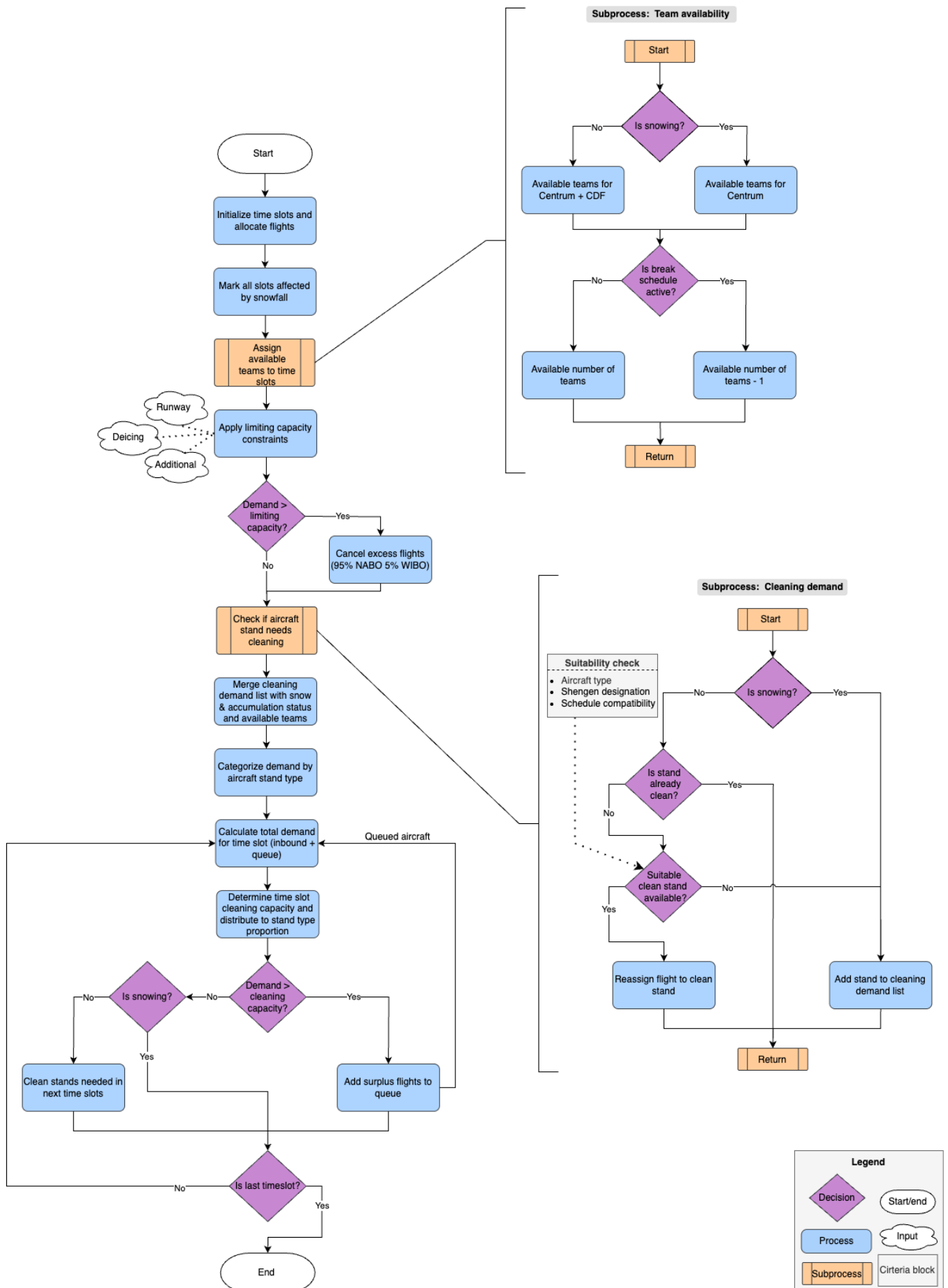


Figure 6: Simulation model logic

References

- Airport Operation Center Schiphol. (2024). *Trainingen Sector-briefing APOC* (tech. rep.).
- Alsalous, O., & Hotle, S. (2024). Deicing Facility Capacity and Delay Estimation Using ASDE-X Data: Chicago O'Hare Simulation Case Study. *Transportation Research Record*, 2678(4), 455–467. <https://doi.org/10.1177/03611981231185147>
- Bolsius, J., & Scholten, D. (2024). Werkboek Sneeuw & Gladheid.
- Fernández, C. M., Gómez, V. F., & Arnaldo, R. M. (2016). Mathematical model for optimising the sequence for clearing snow from the manoeuvring area during winter operations. *Journal of Advanced Transportation*, 50(6), 1225–1251. <https://doi.org/10.1002/atr.1399>
- Janic, M. (2009). Modeling Airport Operations Affected by a Large-Scale Disruption. *Journal of Transportation Engineering*, 135(4), 206–216. <https://doi.org/10.1061/ASCE0733-947X2009135:4206>
- Koščák, P., Berežný, Š., Vajdová, I., Koblen, I., Ojciec, M., Matisková, D., & Puškáš, T. (2020). Reducing the negative environmental impact of winter airport maintenance through its model design and simulation. *International Journal of Environmental Research and Public Health*, 17(4), 1296. <https://doi.org/10.3390/ijerph17041296>
- Majumdar, R., & Selvi, K. (2014). Six Sigma-Overview of DMAIC and DMADV. <https://www.researchgate.net/publication/346081379>
- Merkert, R., & Mangia, L. (2012). Management of airports in extreme winter conditions-some lessons from analysing the efficiency of Norwegian airports. *Research in Transportation Business and Management*, 4, 53–60. <https://doi.org/10.1016/j.rtbm.2012.06.004>
- Mirmohammadsadeghi, N., Hu, J., & Trani, A. (2019). Enhancements to the runway capacity simulation model using the asde-x data for estimating airports throughput under various wake separation systems. *AIAA Aviation 2019 Forum*, 1–16. <https://doi.org/10.2514/6.2019-3044>
- Myers, T., Andrews, M., & Krozel, J. (2012). Winter weather airport capacity model. *12th AIAA Aviation Technology, Integration and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*. <https://doi.org/10.2514/6.2012-5535>
- Pačaiová, H., Korba, P., Hovanec, M., Galanda, J., Šváb, P., & Lukáč, J. (2021). Use of simulation tools for optimization of the time duration of winter maintenance activities at airports. *Sustainability (Switzerland)*, 13(3), 1–15. <https://doi.org/10.3390/su13031095>
- Pan, Y. L., Jhang, J. C., & Gu, F. R. (2022). Using Advanced Surface Movement Guidance and Control System for Kaohsiung Xiaogang Airport Aircraft's Route Assignment Planning Based on Petri nets. *Proceedings of the 2022 8th International Conference on Applied System Innovation, ICASI 2022*, 160–165. <https://doi.org/10.1109/ICASI55125.2022.9774454>
- Preis, H., & Fricke, H. (2020). Optimizing Winter Maintenance Service at Airports, 765–771. https://doi.org/10.1007/978-3-030-48439-2_{-}93
- Schiphol. (2018). *Winteroperatie: vervanging SNLV 1 & 2 Betrouwbare capaciteit op Amsterdam Airport* (tech. rep.).
- Service Owner Sneeuw en Gladheid. (2023). Winteroperatie: Sneeuw & Gladheid.
- Shu-Ling, W., Rong-Hua, Q., Chun-Fu, S., De-Xini, Z., & Hui, Z. (2011). Study on assessment model of road snow removal capacity. *Proceedings - 4th International Conference on Intelligent Computation Technology and Automation, ICICTA 2011, 1*, 1134–1138. <https://doi.org/10.1109/ICICTA.2011.285>
- Srivastava, A. (2011). Improving Departure Taxi Time Predictions Using ASDE-X Surveillance Data. *2011 IEEE/AIAA 30th Digital Avionics Systems Conference*, 1–14. <https://doi.org/10.1109/DASC.2011.6095989>
- Šváb, P., Korba, P., Albert, M., & Kolesár, J. (2019). Information system to support the management of winter airport maintenance. *NTinAD 2019 - New Trends in Aviation Development 2019 - 14th International Scientific Conference, Proceedings*, 170–173. <https://doi.org/10.1109/NTAD.2019.8875565>
- Zhang, Y. P., Zhang, Y., Xing, X. Q., Chen, Y., Yang, N., & Mao, J. (2022). The Method of Division and Classification Quantification for Snow and Ice Removal on Airport Pavement. *Lecture Notes in Electrical Engineering*, 775, 649–664. https://doi.org/10.1007/978-981-16-5429-9_{-}50

B

Statistics

This Chapter outlines several tables and figures supporting the Measure and Analysis phase.

B.1. Algorithm parameter limits

Several parameters are incorporated into the algorithm to define boundaries that minimize the impact of outliers and missing data. Table B.1 presents an overview of these parameters and their respective values.

The parameters related to cleaning and travel times are derived from box plot summary statistics of the recorded durations under boundary-free conditions, that is, when the maximum cleaning and travel time constraints are set to a large constant M (maximum), and the median *within-bay* travel time is set to 0 (minimum). One exception is made for the minimum cleaning time, which is set at 4.5 minutes. A lower threshold of 0 minutes is considered unrealistic, as it would result in the algorithm incorrectly interpreting brief passes by an aircraft stand as completed cleaning events. According to the Service Owner Winter Operations, based on operational experience, cleaning durations only become realistic from 4.5 minutes onwards (Interview Service Owner Sneeuw & Gladheid, 2025).

To identify the thresholds for classifying outliers, the algorithm uses the lower and upper whiskers from the box plots (see Table B.2). Because travel times vary significantly across the three defined route types, route-specific travel time parameters are established. For all route types, the lower theoretical whisker of travel time distributions falls below zero, which is physically impossible. As a result, no minimum travel time is enforced. Similarly, the lower whisker for cleaning times also falls below zero; however, the practical lower limit of 4.5 minutes will still hold because of the mentioned reasoning. The maximum allowed values for both cleaning and travel times are set to their respective upper whiskers.

The parameter *Median within-bay travel time* is set to the median of the *within-bay* travel time distribution. The median is chosen over the mean due to its robustness against outliers, which are visibly present in the corresponding box plot. This choice ensures that the travel time parameter remains representative of typical behavior while reducing sensitivity to extreme values.

As previously mentioned, location inaccuracies may occur due to GPS drift or other factors. For the determination whether the location TAXILANE is within the bay, a breadth-first search is applied with a threshold of three polygons, based on the observation that a maximum of three taxi lane polygons can be present within a single bay. The threshold for the number of consecutive timestamps at which the vehicle may be detected at a location deemed too far is set to five. Beyond this point, the deviation is interpreted as an actual change in location rather than a temporary inaccuracy due to location drift.

The time gap threshold is set equal to the minimum allowed cleaning duration, as beyond this point, the algorithm can no longer reliably determine whether a stand was cleaned during the undocumented interval. The future data look-ahead window is set to the maximum allowed cleaning duration, reflecting the theoretical upper limit for the number of consecutive NaN values that may occur within a single cleaning period. Finally, the time window for cross-referencing with the CISS and VGRS datasets is

set to 10 minutes, based on the assumption that the start time of a cleaning period in these datasets, if available, should closely align with the start time identified by the algorithm.

Category	Parameter	Value
Cleaning time (min)	Min.	4.50
	Max.	56.30
Travel time (min)	Max. within-bay	4.50
	Max. cross-bay	49.92
	Max. Cross zone	65.05
	Median within-bay	0.65
	polygon threshold	3
Time & distance thresholds	distant threshold	5
	timegap threshold (min)	4.5 (min. clean time)
	future data time window (min)	56.30 (max. clean time)
Look-ahead windows	CISS/VGRS time window (min)	10

Table B.1: Limits and time windows used in the algorithm

Statistic	Cleaning time	Within-bay travel time	Cross-bay travel time	Cross-zone travel time
Median		39		
Lower whisker	4.5	0	0	0
Upper whisker	3378	270	2995	3903

Table B.2: Boxplot statistics for parameter limit selection, in seconds

B.2. Distribution of cleaning data

Figure B.4 show the distributions of the cleaning data, split by clean time, travel time and cycle time.

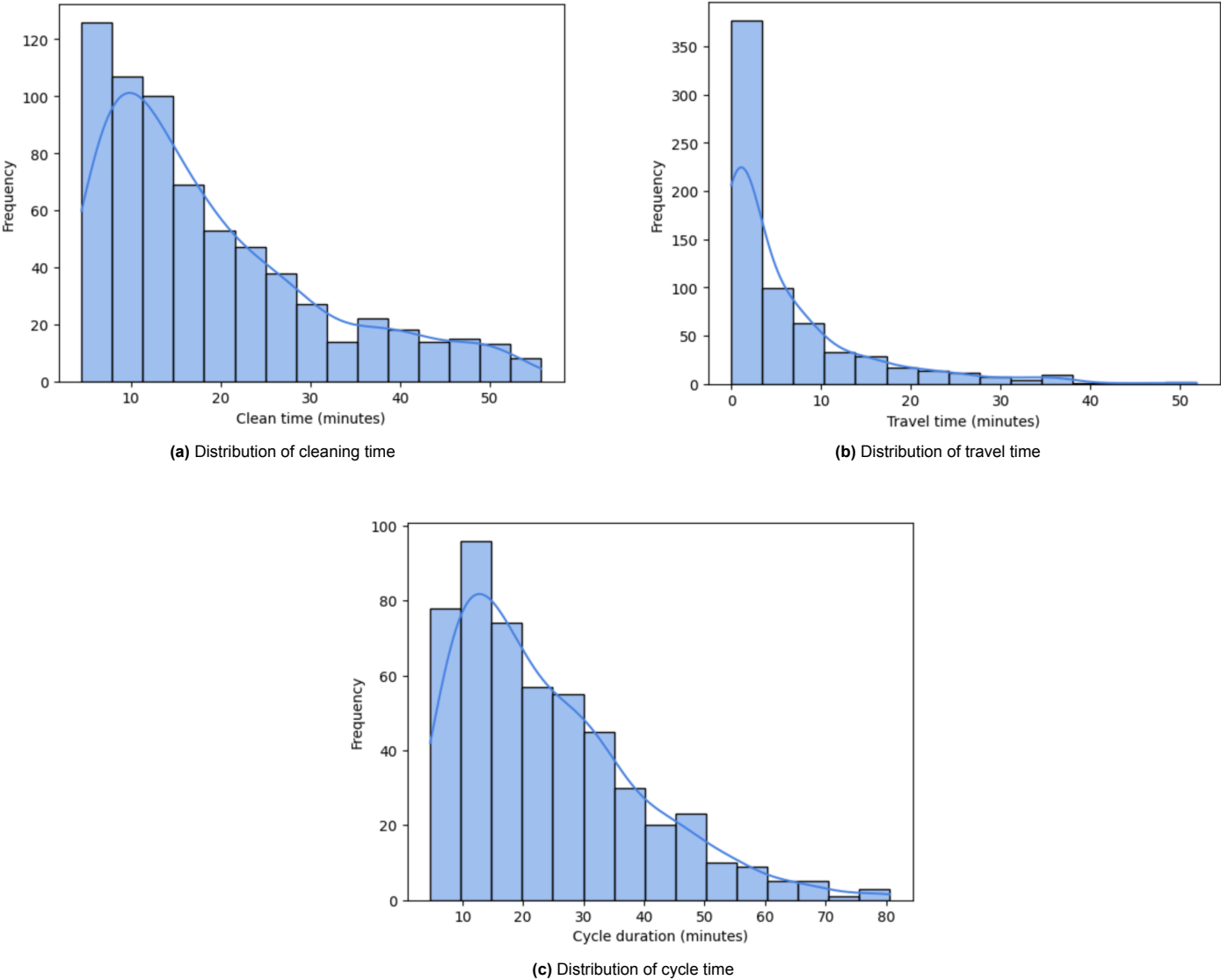


Figure B.1: Distributions of cleaning, travel, and cycle times

B.3. Supplementary results exploratory analysis

The following sub-sections support the exploratory analysis performed in Section 5.2.

B.3.1. Idle time

Statistic	Minutes
25% quartile	13.0
median	26.5
75% quartile	56.7
IQR	43.8
Lower whisker	4.1
Upper whisker	93.7

Table B.3: Descriptive statistics of ASCT's idle time

B.3.2. Clean time

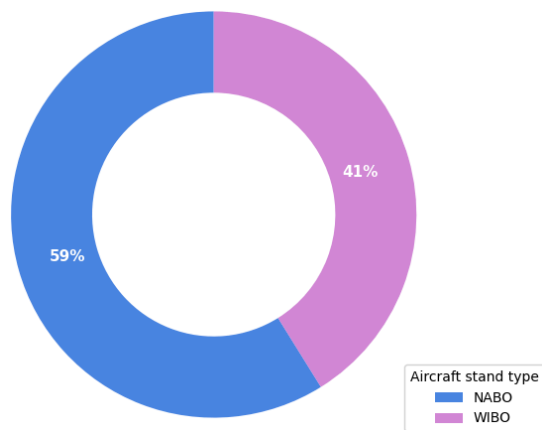


Figure B.2: Distribution of cleaned stands per type

Statistic	Nabo	Wibo
25% quartile	8.72	10.93
median	13.88	16.62
75% quartile	24.11	27.18
IQR	15.2	16.25
Lower whisker	4.5	4.6
Upper whisker	45.67	51.45

Table B.4: Descriptive statistics of clean times across stand type

B.3.3. Travel time

Statistic	Withinibay	Cross-bay	Cross-zone
25% quartile	0.03	2.97	2.78
median	0.63	6.30	7.67
75% quartile	0.80	11.92	14.70
IQR	0.77	8.95	11.92
Lower whisker	0.02	0.02	0.12
Upper whisker	1.92	25.25	31.95
Count	268	263	137

Table B.5: Descriptive statistics of ASCT travel times across route type

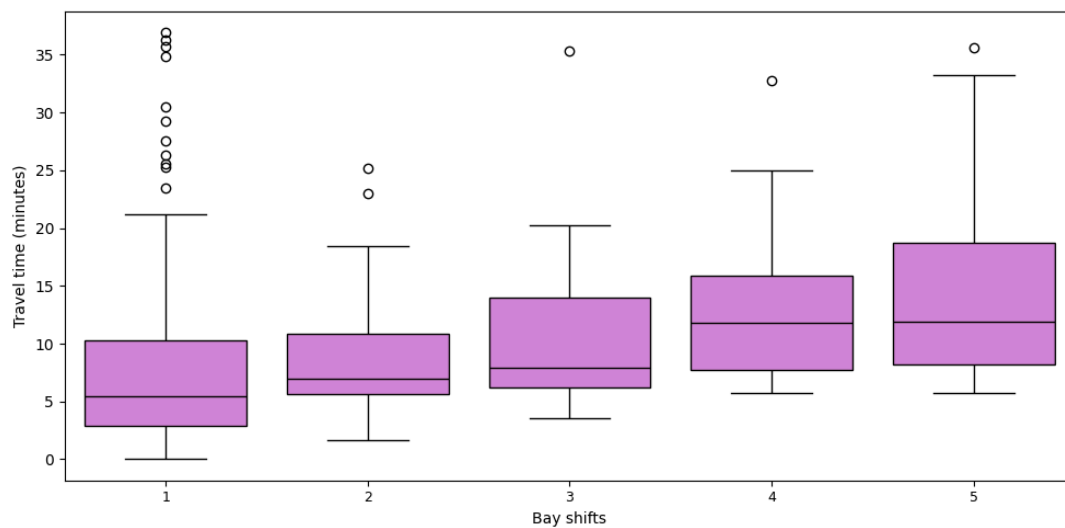
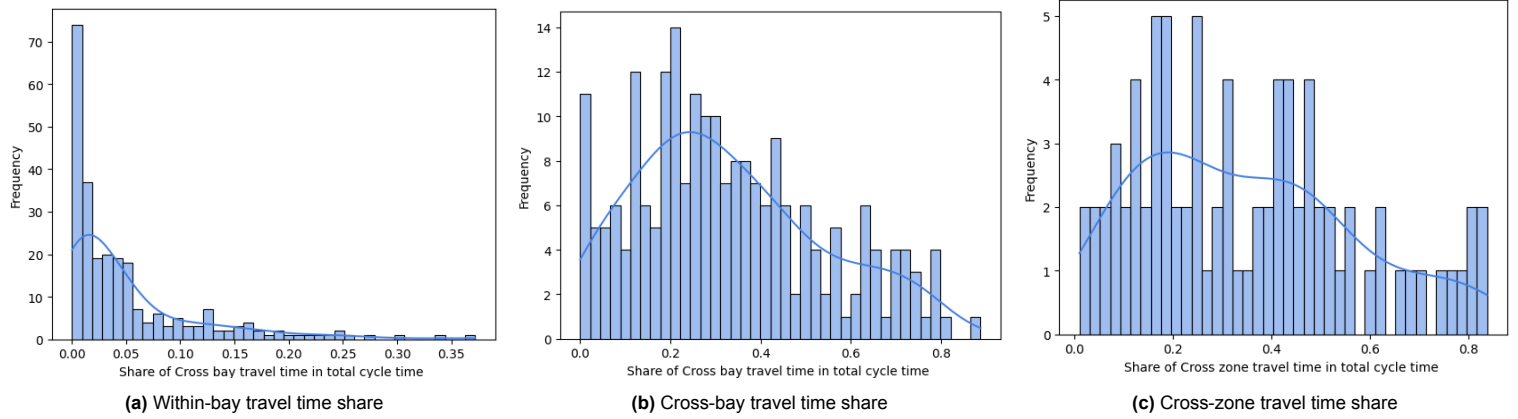


Figure B.3: Cross bay travel time across the amount of bay shifts

B.3.4. Cycle duration

Statistic	Cycle duration
25% quartile	12.35
Median	20.35
75% quartile	32.42
IQR	20.07
Lower whisker	4.68
Upper whisker	62.02

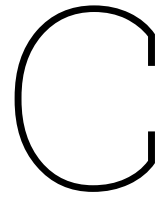
Table B.6: Descriptives of cycle time duration**Figure B.4:** Frequency distributions of travel time share in cycle duration

B.4. Impact of weather variables on cleaning capacity

Table B.7 shows the correlation coefficients and corresponding p-values of all weather variables and cleaning capacity.

Variable	Correlation coefficient	p-value
pre10_WSmK	0.030	0.534
pre10_TAm	0.010	0.830
pre10_TGm	-0.059	0.222
pre10_TBm	-0.022	0.648
pre10_TDm	-0.048	0.317
pre10_Vis	0.018	0.711
pre10_NIm	-0.062	0.200
pre10_SHm	-0.206	1.73e-05
pre10_C1s	0.032	0.508
pre10_NDm_ratio	-0.062	0.202
first4.6_WSmK	0.050	0.300
first4.6_TAm	0.006	0.897
first4.6_TGm	-0.060	0.212
first4.6_TBm	-0.022	0.643
first4.6_TDm	-0.035	0.474
first4.6_Vis	0.019	0.699
first4.6_NIm	-0.057	0.236
first4.6_SHm	-0.211	1.08e-05
first4.6_C1s	0.037	0.446
first4.6_NDm_ratio	-0.048	0.319
first19_WSmK	0.031	0.519
first19_TAm	0.003	0.955
first19_TGm	-0.056	0.251
first19_TBm	-0.025	0.607
first19_TDm	-0.039	0.416
first19_Vis	-0.002	0.971
first19_NIm	-0.071	0.144
first19_SHm	-0.229	1.62e-06
first19_C1s	0.010	0.838
first19_NDm_ratio	-0.050	0.300
full_WSmK	0.021	0.664
full_TAm	0.008	0.877
full_TGm	-0.051	0.288
full_TBm	-0.019	0.691
full_TDm	-0.035	0.465
full_Vis	-0.016	0.746
full_NIm	-0.085	0.080
full_SHm	-0.248	1.96e-07
full_C1s	0.006	0.904
full_NDm_ratio	-0.073	0.129

Table B.7: Correlation of weather variables with capacity and corresponding p-values



Model

C.1. Deicing forecasting tool

The following statements provide a detailed overview of the model logic and visual presentation of APOC's de-icing tool (Iris Schiphol, n.d.).

Model logic:

- The forecast is available from D-3 to D-0.
- The model operates in 15-minute intervals.
- De-icing demand is derived from CISS flight data, distributing the scheduled departures across the 15-minute time blocks.
- Capacity refers to the number of aircraft each handler expects to be able to de-ice per hour. These values are provided by the handlers and entered manually by APOC users.
- Hourly de-icing capacity is entered as an average, based on an equal mix of narrow body and wide body aircraft. This capacity is then divided by 4 to align with the 15-minute time intervals.
- The model incorporates three scenarios, Low, Middle, and High, to account for uncertainty in weather forecasts. The Middle scenario serves as the baseline, while the Low and High scenarios represent more favorable and more severe weather conditions, respectively.
- When demand exceeds capacity, a queue of aircraft starts to form. Conversely, when capacity exceeds demand, the queue decreases.

Model output:

- In the visual output: bars represent de-icing demand, while the line indicates the handler's planned maximum capacity. The bars are stacked to show fulfilled demand for wide body and narrow body aircraft, as well as queued aircraft. See Figure 4.9.
- KPIs displayed alongside the graph include the number of wide body and narrow body aircraft requiring de-icing, and the maximum queue size within a 15-minute block.

C.2. Model logic

In the following steps, the logic of the simulation model is explained.

1. Initialize time slots and determine snow status

- Generate time slots (e.g., every 30 minutes) and round down each flight's scheduled arrival time to the beginning of its corresponding slot (e.g., 11:29 → 11:00).
- Identify and mark all time slots affected by snowfall, based on the predicted snow start and end times.

- Assign the number of active ASCTs to each time slot. During snowfall, only teams operating in the centrum are considered available, excluding those assigned to the CDF. After snowfall ends, the team(s) dedicated to the CDF rejoin the teams at centrum, increasing the number of available teams accordingly.
- Adjust the number of active ASCTs per time slot to account for teams taking scheduled breaks.

2. Apply capacity restrictions

- Runway capacity undergoes a gradual transition towards the optimal runway scenario (Scenario B) once snowfall ceases, taking into account the average runway cleaning time.
- For each time slot, the effective capacity is determined as the minimum of the runway capacity, deicing capacity and additional restricted capacity.
- If the number of scheduled flights exceeds the available capacity for a given time slot, the surplus flights are canceled. Flight removals follow a predefined ratio of 95% Nabo and 5% Wibos.

3. Determine aircraft stands requiring cleaning

- During snowfall, all aircraft stands are assumed to require snow removal and are therefore included to the cleaning demand list.
- Starting 30 minutes before the end of snowfall (i.e., one time slot), aircraft stands that have already been cleared are assumed to remain clear and are therefore excluded from subsequent cleaning rounds.
- When *End time of cleaning requirement* is applied, all queued and newly arriving aircraft can be immediately assigned to an aircraft stand once this time is reached.
- For any flight scheduled at an aircraft stand that has not yet been cleaned after snowfall, the algorithm attempts to reassign it to an already-cleaned stand if a suitable alternative is available. Suitability is determined based on the following criteria:
 - Wide-body aircraft can only be assigned to Wibos stands, and narrow-body aircraft to Nabos stands.
 - Schengen flights must be assigned to stands that are compatible with Schengen operations or both Schengen and non-Schengen operations, and vice versa.
 - The previous flight at the alternative stand must have completed handling before the new assignment.
 - The new flight must be able to complete handling before any subsequent flight is scheduled at that stand.
- If no suitable cleaned stand is available, the flight remains assigned to its original stand, which is then marked as requiring cleaning.
- Finally, the cleaning demand list is merged with the corresponding snow status and team availability per time slot. The total demand is categorized by aircraft stand type (Nabo or Wibos).

4. Calculate capacity and cleaned stands per slot

- For each time slot, compute the available cleaning capacity per aircraft stand type, considering model scenario, route type (based on *within-bay* versus *cross-bay* ratio), accumulation level, and the number of active teams. Distribute this capacity across Wibos and Nabos stands in proportion to their total demand (including both scheduled and queued stands).
- Determine how many stands can be cleaned during each time slot and how many must be deferred to the queue. If excess capacity remains and snowfall has ended, use it to clean upcoming scheduled stands in advance.

D

Code

D.1. Classification algorithm

```
1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[ ]:
5
6
7 #Casper data
8 minimum_cleantime = 270
9 maximum_cleantime = 3378
10 max_travel_time_within_bay = 270
11 max_travel_time_within_centrum = 2995
12 max_travel_time_outside_centrum = 3903
13 constant_within_bay_travel_time = 39
14 time_range_check = 600
15 lookahead_window = maximum_cleantime
16 time_gap_threshold = minimum_cleantime
17 polygon_threshold = 3
18
19
20 # In[ ]:
21
22
23 def calculate_polygon_distance(start, end, polygon_graph):
24     """Berekent de afstand in aantal tussenliggende polygones. --> breath-first search (BFS)"""
25
26     visited = set()
27     queue = deque([(start, 0)])
28
29     while queue:
30         current, distance = queue.popleft()
31         if current == end:
32             return distance
33         if current in visited:
34             continue
35         visited.add(current)
36         for neighbor in polygon_graph.get(current, []):
37             queue.append((neighbor, distance + 1))
38     return float('inf') # Geen pad gevonden
39
40
41 # In[ ]:
42
43
44 def decide_if_end_cleaning(snow_period, current_vop, start_time, end_time, vehicle_id, break_times, cleaning_periods, travel_times, cycli, minimum_cleantime,
45     ↪ maximum_cleantime, max_travel_time_within_bay, max_travel_time_within_centrum,
46     ↪ max_travel_time_outside_centrum, constant_within_bay_travel_time, time_range_check, time_gap_detected, last_vop_end_time, previous_vop,
47     ↪ temporary_skipped_vops, skipped_vops, i, data, VGRS_data, df_ciss):
48
49     is_break = False
50     too_long_cleaning = False
51     too_long_travel = False
52     route_type = None
53
54     #als op R of Y platform tijdens de schoonmaak --> pauze. Maar alleen als het midden in het proces is, niet als het aan het begin of einde is.
55     if current_vop.startswith(('Y', 'R')) and previous_vop is not None and current_vop != data.loc[data['loctype'] == 'VOP', 'location'].iloc[-1]:
56         is_break = True
57         #als de vorige current vop ook pauze was op hetzelfde platform, verleng de pauze
58         if break_times and (vop.location_dict.get(current_vop) == 'Y-platform' and vop.location_dict.get(previous_vop) == 'Y-platform') or
59         ↪ (vop.location_dict.get(current_vop) == 'R-platform' and vop.location_dict.get(previous_vop) == 'R-platform'):
60             last_break = break_times.pop()
61             last_cyclus = cycli[-1]
62             if last_cyclus['is_break']:
63                 cycli.pop()
64
65         #Maak de break opnieuw met de nieuwe eindtijd
66         start_break_time = last_break['start_time']
67         end_break_time = end_time
68         calculate_break_time(break_times, snow_period, previous_vop, start_break_time, end_break_time, vehicle_id)
69
70         #Maak de cyclus opnieuw met de nieuwe eindtijd
71         previous_previous_vop = last_cyclus['from_vop']
72         start_time_travel = last_cyclus['travel_start']
73         route_type = last_cyclus['route_type']
74         calculate_cyclus(cycli, snow_period, vehicle_id, previous_previous_vop, previous_vop, start_time_travel, start_break_time, end_break_time, route_type,
75     ↪ is_break, too_long_cleaning, too_long_travel, time_gap_detected)
```

```

else:
    if last_vop_end_time is not None:
        route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop, vehicle_id,
        ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
        ↪ too_long_travel, time_gap_detected)
    else:
        start_index = data[(data['location'] == current_vop) & (data['timestamp'] == start_time)].index[0]
        previous_vop, last_vop_end_time = find_previous_start_position(start_index, data, df_ciss, current_vop)
        if previous_vop is not None and last_vop_end_time is not None:
            route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop,
            ↪ vehicle_id,
            ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
            ↪ too_long_travel, time_gap_detected)

        calculate_break_time(break_times, snow_period, current_vop, start_time, end_time, vehicle_id)
        calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
        ↪ too_long_cleaning, too_long_travel, time_gap_detected)

last_vop_end_time = end_time
previous_vop = current_vop
temporary_skipped_vops.clear()
is_break = False
time_gap_detected = False
too_long_travel = False

#Als V4 op CDF was, hoef je niet te kijken naar mogelijke andere locaties die die heeft schoongemaakt
elif vehicle_id == 'V4' and vop_location_dict.get(current_vop) == 'CDF':
    if last_vop_end_time is not None:
        route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop, vehicle_id,
        ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
        ↪ too_long_travel, time_gap_detected)
    else:
        start_index = data[(data['location'] == current_vop) & (data['timestamp'] == start_time)].index[0]
        previous_vop, last_vop_end_time = find_previous_start_position(start_index, data, df_ciss, current_vop)
        if previous_vop is not None and last_vop_end_time is not None:
            route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop, vehicle_id,
            ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
            ↪ too_long_travel, time_gap_detected)

        calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break, too_long_cleaning,
        ↪ too_long_travel, time_gap_detected)
        calculate_break_time(break_times, snow_period, current_vop, start_time, end_time, vehicle_id)

last_vop_end_time = end_time
previous_vop = current_vop
temporary_skipped_vops.clear()
time_gap_detected = False
too_long_travel = False

else:
    cleaning_duration = (end_time - start_time).total_seconds()
    if minimum_cleantime <= cleaning_duration <= maximum_cleantime:
        if last_vop_end_time is not None:
            route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop, vehicle_id,
            ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
            ↪ too_long_travel, time_gap_detected)
        else:
            start_index = data[(data['location'] == current_vop) & (data['timestamp'] == start_time)].index[0]
            previous_vop, last_vop_end_time = find_previous_start_position(start_index, data, df_ciss, current_vop)
            if previous_vop is not None and last_vop_end_time is not None:
                route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop,
                ↪ vehicle_id,
                ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
                ↪ too_long_travel, time_gap_detected)

        #check other VOPS based on time
        additional_cleaned_vops = {current_vop}
        if not VGRS_data.empty:
            additional_cleaned_vops = check_VGRS_adjacent_stands(i, data, VGRS_data, cleaning_periods, additional_cleaned_vops, start_time, end_time, current_vop,
            ↪ minimum_cleantime, time_range_check)

        if len(additional_cleaned_vops) == 1: #geen extra vops gevonden vanuit VGRS
            if not df_ciss.empty:
                additional_cleaned_vops = check_ciss_adjacent_stands(i, data, df_ciss, cleaning_periods, additional_cleaned_vops, current_vop, start_time,
                ↪ end_time, minimum_cleantime, time_range_check)

            if len(additional_cleaned_vops) == 1: #geen extra vops gevonden vanuit VGRS en CISS
                #standaard aanpak
                end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
                calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
                ↪ too_long_cleaning, too_long_travel, time_gap_detected)
            else:
                n = len(additional_cleaned_vops)
                cleaning_duration_per_vop = (cleaning_duration - (n - 1) * constant_within_bay_travel_time) / n
                if cleaning_duration_per_vop < minimum_cleantime:
                    #toch niet met additional vop
                    end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
                    calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
                    ↪ too_long_cleaning, too_long_travel, time_gap_detected)
                else:
                    sorted_vops = [current_vop] + list(additional_cleaned_vops - {current_vop})
                    partly_travel_time = constant_within_bay_travel_time
                    partly_start_time = start_time
                    prev_vop = previous_vop
                    prev_end_time = last_vop_end_time
                    print(f'vopje toegevoegd met CISS: {additional_cleaned_vops}')
                    for vop in sorted_vops:
                        if vop != current_vop:
                            route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, partly_end_time, partly_start_time, prev_vop, vop,
                            ↪ vehicle_id,
                            ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
                            ↪ too_long_travel, time_gap_detected)

                            partly_end_time = partly_start_time + timedelta(seconds=cleaning_duration_per_vop)
                            end_cleaning(cleaning_periods, snow_period, vop, partly_start_time, partly_end_time, cleaning_duration_per_vop, vehicle_id,
                            ↪ too_long_cleaning)
                            calculate_cyclust(cycli, snow_period, vehicle_id, prev_vop, vop, prev_end_time, partly_start_time, partly_end_time, route_type, is_break,
                            ↪ too_long_cleaning, too_long_travel, time_gap_detected)
                            partly_start_time = partly_end_time + timedelta(seconds=partly_travel_time)
                    prev_vop = vop
                    prev_end_time = partly_end_time

```



```

162
163
164     else:
165         n = len(additional_cleaned_vops)
166         cleaning_duration_per_vop = (cleaning_duration - (n - 1) * constant_within_bay_travel_time) / n
167         if cleaning_duration_per_vop < minimum_cleantime:
168             #toch niet met additional vop van VGRS --> check CISS
169             if not df_ciss.empty:
170                 additional_cleaned_vops = {current_vop}
171                 additional_cleaned_vops = check_ciss_within_bay(i, data, df_ciss, cleaning_periods, additional_cleaned_vops, current_vop, start_time, end_time,
172                 ↪ minimum_cleantime, time_range_check)
173                 if len(additional_cleaned_vops) == 1:
174                     print(f"cleaning duration van {current_vop} is te kort met VGRS en CISS werkt ook niet")
175                     end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
176                     calculate_cyclus(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
177                     ↪ too_long_cleaning, too_long_travel, time_gap_detected)
178
179             else:
180                 n = len(additional_cleaned_vops)
181                 cleaning_duration_per_vop = (cleaning_duration - (n - 1) * constant_within_bay_travel_time) / n
182                 if cleaning_duration_per_vop < minimum_cleantime:
183                     print(f"cleaning duration van {additional_cleaned_vops} is te kort ({cleaning_duration_per_vop}) met CISS data en VGRS data")
184                     end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
185                     calculate_cyclus(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type,
186                     ↪ is_break, too_long_cleaning, too_long_travel, time_gap_detected)
187
188                 else:
189                     sorted_vops = [current_vop] + list(additional_cleaned_vops - {current_vop})
190                     partly_start_time = start_time
191                     partly_travel_time = constant_within_bay_travel_time
192                     prev_vop = previous_vop
193                     prev_end_time = last_vop_end_time
194                     for vop in sorted_vops:
195                         if vop != current_vop:
196                             route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, partly_end_time, partly_start_time, prev_vop,
197                             ↪ vop, vehicle_id,
198                                     max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
199                                     ↪ too_long_travel, time_gap_detected)
200                         partly_end_time = partly_start_time + timedelta(seconds=cleaning_duration_per_vop)
201                         end_cleaning(cleaning_periods, snow_period, vop, partly_start_time, partly_end_time, cleaning_duration_per_vop, vehicle_id,
202                         ↪ too_long_cleaning)
203                         calculate_cyclus(cycli, snow_period, vehicle_id, prev_vop, vop, prev_end_time, partly_start_time, partly_end_time, route_type,
204                         ↪ is_break, too_long_cleaning, too_long_travel, time_gap_detected)
205                         partly_start_time = partly_end_time + timedelta(seconds=partly_travel_time)
206                         prev_vop = vop
207                         prev_end_time = partly_end_time
208
209             else:
210                 sorted_vops = [current_vop] + list(additional_cleaned_vops - {current_vop})
211                 partly_travel_time = constant_within_bay_travel_time
212                 partly_start_time = start_time
213                 prev_vop = previous_vop
214                 prev_end_time = last_vop_end_time
215                 print(f'vopje toegevoegd: {additional_cleaned_vops}')
216                 for vop in sorted_vops:
217                     if vop != current_vop:
218                         route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, partly_end_time, partly_start_time, prev_vop, vop,
219                         ↪ vehicle_id,
220                                     max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
221                                     ↪ too_long_travel, time_gap_detected)
222                     partly_end_time = partly_start_time + timedelta(seconds=cleaning_duration_per_vop)
223                     end_cleaning(cleaning_periods, snow_period, vop, partly_start_time, partly_end_time, cleaning_duration_per_vop, vehicle_id, too_long_cleaning)
224                     calculate_cyclus(cycli, snow_period, vehicle_id, prev_vop, vop, prev_end_time, partly_start_time, partly_end_time, route_type, is_break,
225                     ↪ too_long_cleaning, too_long_travel, time_gap_detected)
226                     partly_start_time = partly_end_time + timedelta(seconds=partly_travel_time)
227                     prev_vop = vop
228                     prev_end_time = partly_end_time
229
230 last_vop_end_time = end_time
231 previous_vop = current_vop
232 temporary_skipped_vops.clear()
233 time_gap_detected = False
234 too_long_travel = False
235
236 elif cleaning_duration >= maximum_cleantime:
237     print(f"{i}, {current_vop} te lang {cleaning_duration}, ff verder kijken")
238     if last_vop_end_time is not None:
239         route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop, vehicle_id,
240         ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
241         ↪ too_long_travel, time_gap_detected)
242
243     else:
244         start_index = data[(data['location'] == current_vop) & (data['timestamp'] == start_time)].index[0]
245         previous_vop, last_vop_end_time = find_previous_start_position(start_index, data, df_ciss, current_vop)
246         if previous_vop is not None:
247             route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, last_vop_end_time, start_time, previous_vop, current_vop,
248             ↪ vehicle_id,
249                                     max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
250                                     ↪ too_long_travel, time_gap_detected)
251
252     #kijk of er miss andere vops zijn
253     potential_cleaned_vops = {current_vop}
254     if not VGRS_data.empty:
255         potential_cleaned_vops = check_VGRS_within_bay(i, data, VGRS_data, cleaning_periods, potential_cleaned_vops, start_time, end_time, current_vop,
256         ↪ minimum_cleantime, time_range_check)
257
258     if len(potential_cleaned_vops) == 1: #geen extra vops gevonden vanuit VGRS --> check CISS
259         if not df_ciss.empty:
260             potential_cleaned_vops = check_ciss_within_bay(i, data, df_ciss, cleaning_periods, potential_cleaned_vops, current_vop, start_time, end_time,
261             ↪ minimum_cleantime, time_range_check)
262         if len(potential_cleaned_vops) == 1: #geen extra vops gevonden vanuit VGRS en CISS
263             print(f"cleaning duration van {current_vop} is te lang en geen CISS of VGRS vops gevonden")
264             too_long_cleaning = True
265             end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
266             calculate_cyclus(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
267             ↪ too_long_cleaning, too_long_travel, time_gap_detected)
268
269         else:
270             n = len(potential_cleaned_vops)
271             cleaning_duration_per_vop = (cleaning_duration - (n - 1) * constant_within_bay_travel_time) / n
272             if cleaning_duration_per_vop < minimum_cleantime:
273                 ↪ print(f"cleaning duration van {potential_cleaned_vops} is te kort ({cleaning_duration_per_vop}) met CISS data, sla alsnog de lange clean time op")
274                 too_long_cleaning = True
275                 end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
276                 calculate_cyclus(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
277                 ↪ too_long_cleaning, too_long_travel, time_gap_detected)

```

```

255
256
257     else:
258         if cleaning_duration_per_vop > maximum_cleantime:
259             too_long_cleaning = True
260             print(f"cleaning duration van {potential_cleaned_vops} is alsnog te lang ({cleaning_duration_per_vop}) met CISS erbij")
261
262             sorted_vops = [current_vop] + list(potential_cleaned_vops - {current_vop})
263             partly_start_time = start_time
264             partly_travel_time = constant_within_bay_travel_time
265             prev_vop = previous_vop
266             prev_end_time = last_vop_end_time
267             for vop in sorted_vops:
268                 if vop != current_vop:
269                     route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, partly_end_time, partly_start_time, prev_vop, vop,
270                                     ↪ vehicle_id,
271                                     ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
272                                     ↪ too_long_travel, time_gap_detected)
273                     partly_end_time = partly_start_time + timedelta(seconds=cleaning_duration_per_vop)
274                     end_cleaning(cleaning_periods, snow_period, vop, partly_start_time, partly_end_time, cleaning_duration_per_vop, vehicle_id,
275                                     ↪ too_long_cleaning)
276                     ↪ too_long_cleaning
277                     calculate_cyclust(cycli, snow_period, vehicle_id, prev_vop, vop, prev_end_time, partly_start_time, partly_end_time, route_type, is_break,
278                                     ↪ too_long_cleaning, too_long_travel, time_gap_detected)
279                     ↪ too_long_cleaning, too_long_travel, time_gap_detected
280                     partly_start_time = partly_end_time + timedelta(seconds=partly_travel_time)
281                     prev_vop = vop
282                     prev_end_time = partly_end_time
283
284             else:
285                 n = len(potential_cleaned_vops)
286                 cleaning_duration_per_vop = (cleaning_duration - (n - 1) * constant_within_bay_travel_time) / n
287
288                 if cleaning_duration_per_vop < minimum_cleantime:
289                     print(f"cleaning duration van {potential_cleaned_vops} is te kort ({cleaning_duration_per_vop}) met VGRS data, check toch maar CISS data")
290                     #check alsnog CISS
291                     if not df_ciiss.empty:
292                         potential_cleaned_vops = {current_vop}
293                         potential_cleaned_vops = check_ciiss_within_bay(i, data, df_ciiss, cleaning_periods, potential_cleaned_vops, current_vop, start_time, end_time,
294                                     ↪ minimum_cleantime, time_range_check)
295                         ↪ minimum_cleantime, time_range_check
296                         if len(potential_cleaned_vops) == 1:
297                             print(f"cleaning duration van {current_vop} is te kort met VGRS en CISS werkt ook niet")
298                             too_long_cleaning = True
299                             end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
300                             calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
301                                     ↪ too_long_cleaning, too_long_travel, time_gap_detected)
302                             ↪ too_long_cleaning, too_long_travel, time_gap_detected
303                         else:
304                             n = len(potential_cleaned_vops)
305                             cleaning_duration_per_vop = (cleaning_duration - (n - 1) * constant_within_bay_travel_time) / n
306                             if cleaning_duration_per_vop < minimum_cleantime:
307                                 ↪ print(f"cleaning duration van {potential_cleaned_vops} is te kort ({cleaning_duration_per_vop}) met CISS data en VGRS data, sla alsnog de lange clean time op")
308                                 too_long_cleaning = True
309                                 end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
310                                 calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type,
311                                     ↪ is_break, too_long_cleaning, too_long_travel, time_gap_detected)
312                                 ↪ is_break, too_long_cleaning, too_long_travel, time_gap_detected
313                         else:
314                             if cleaning_duration_per_vop > maximum_cleantime:
315                                 too_long_cleaning = True
316                                 print(f"cleaning duration van {potential_cleaned_vops} is alsnog te lang ({cleaning_duration_per_vop}) met CISS en VGRS erbij")
317
318                                 sorted_vops = [current_vop] + list(potential_cleaned_vops - {current_vop})
319                                 partly_start_time = start_time
320                                 partly_travel_time = constant_within_bay_travel_time
321                                 prev_vop = previous_vop
322                                 prev_end_time = last_vop_end_time
323                                 for vop in sorted_vops:
324                                     if vop != current_vop:
325                                         route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, partly_end_time, partly_start_time, prev_vop,
326                                             ↪ vop, vehicle_id,
327                                             ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
328                                             ↪ too_long_travel, time_gap_detected)
329                                         partly_end_time = partly_start_time + timedelta(seconds=cleaning_duration_per_vop)
330                                         end_cleaning(cleaning_periods, snow_period, vop, partly_start_time, partly_end_time, cleaning_duration_per_vop, vehicle_id,
331                                             ↪ too_long_cleaning)
332                                         ↪ too_long_cleaning
333                                         calculate_cyclust(cycli, snow_period, vehicle_id, prev_vop, vop, prev_end_time, partly_start_time, partly_end_time, route_type,
334                                             ↪ is_break, too_long_cleaning, too_long_travel, time_gap_detected)
335                                         ↪ is_break, too_long_cleaning, too_long_travel, time_gap_detected
336                                         partly_start_time = partly_end_time + timedelta(seconds=partly_travel_time)
337                                         prev_vop = vop
338                                         prev_end_time = partly_end_time
339
340                             else:
341                                 #niet mogelijk om nog te kijken naar CISS, dus sla te lange clean time op zonder de VGRS vops erbij
342                                 too_long_cleaning = True
343                                 end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning)
344                                 calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break,
345                                     ↪ too_long_cleaning, too_long_travel, time_gap_detected)
346                                 ↪ too_long_cleaning, too_long_travel, time_gap_detected
347
348                 else:
349                     if cleaning_duration_per_vop > maximum_cleantime:
350                         too_long_cleaning = True
351                         print(f"cleaning duration van {potential_cleaned_vops} is alsnog te lang ({cleaning_duration_per_vop}) met VGRS data")
352
353                         sorted_vops = [current_vop] + list(potential_cleaned_vops - {current_vop})
354                         partly_start_time = start_time
355                         partly_travel_time = constant_within_bay_travel_time
356                         prev_vop = previous_vop
357                         prev_end_time = last_vop_end_time
358                         for vop in sorted_vops:
359                             if vop != current_vop:
360                                 route_type, too_long_travel = calculate_travel_time(travel_times, snow_period, partly_end_time, partly_start_time, prev_vop, vop,
361                                     ↪ vehicle_id,
362                                     ↪ max_travel_time_within_bay, max_travel_time_within_centrum, max_travel_time_outside_centrum,
363                                     ↪ too_long_travel, time_gap_detected)
364                                 partly_end_time = partly_start_time + timedelta(seconds=cleaning_duration_per_vop)
365                                 end_cleaning(cleaning_periods, snow_period, vop, partly_start_time, partly_end_time, cleaning_duration_per_vop, vehicle_id, too_long_cleaning)
366                                 calculate_cyclust(cycli, snow_period, vehicle_id, prev_vop, vop, prev_end_time, partly_start_time, partly_end_time, route_type, is_break,
367                                     ↪ too_long_cleaning, too_long_travel, time_gap_detected)
368                                 ↪ too_long_cleaning, too_long_travel, time_gap_detected
369                                 partly_start_time = partly_end_time + timedelta(seconds=partly_travel_time)
370                                 prev_vop = vop
371                                 prev_end_time = partly_end_time
372
373                     too_long_cleaning = False
374                     too_long_travel = False
375                     last_vop_end_time = end_time

```

```

350     previous_vop = current_vop
351     temporary_skipped_vops.clear()
352     time_gap_detected = False
353
354     else: #te korte schoonmaak duur
355         if previous_vop is None:
356             #current_vop te korte schoonmaakduur. Herstart vanaf het begin.
357             skipped_vops.add(current_vop)
358             i = 0 #Ga terug naar het begin
359         else:
360             print(f"{current_vop} te kort {cleaning_duration}, einde schoonmaak")
361             #skip deze vop en ga opnieuw bekijken welke vop dan wel
362             temporary_skipped_vops.add(current_vop)
363             if vehicle_id == 'V4' and vop_location_dict.get(previous_vop) == 'CDF':
364                 for j in range(i-1, -1, -1):
365                     loc = data.iloc[j]['location']
366                     if vop_location_dict.get(loc) == 'CDF' or loc == 'A20':
367                         i = j + 1
368                         break
369             else:
370                 for j in range(i-1, -1, -1):
371                     if data.iloc[j]['loctype'] == 'VOP' and data.iloc[j]['location'] == previous_vop:
372                         i = j + 1 # Reset i naar die vorige locatie
373                         break
374
375     return last_vop_end_time, previous_vop, temporary_skipped_vops, skipped_vops, time_gap_detected, i
376
377 # In[]:
378
379
380
381 def calculate_cyclust(cycli, snow_period, vehicle_id, previous_vop, current_vop, last_vop_end_time, start_time, end_time, route_type, is_break, too_long_cleaning,
382 ↪ too_long_travel, time_gap_detected):
383     if last_vop_end_time is not None:
384         total_duration = (end_time - last_vop_end_time).total_seconds()
385     else:
386         total_duration = None
387
388     vop_type = vop_type_dict.get(current_vop, "Onbekend")
389     start_sneeuw, eind_sneeuw = snow_period
390
391     cycli.append({
392         'snow_start': start_sneeuw,
393         'snow_end': eind_sneeuw,
394         'vehicle_id': vehicle_id,
395         'from_vop': previous_vop,
396         'to_vop': current_vop,
397         'route_type': route_type,
398         'vop_type': vop_type,
399         'travel_start': last_vop_end_time,
400         'travel_end': start_time,
401         'clean_start': start_time,
402         'clean_end': end_time,
403         'total_cycle_duration': total_duration,
404         'is_break': is_break,
405         'too_long_cleaning': too_long_cleaning,
406         'too_long_travel': too_long_travel,
407         'time_gap_detected': time_gap_detected,
408     })
409
410 # In[]:
411
412
413 def find_previous_start_position(current_index, data, df_ciss, current_vop, min_duration=180):
414     #check eerst ciss data
415     if not df_ciss.empty:
416         if (df_ciss['current_position'].iloc[0] == current_vop) or (vop_location2_dict.get(df_ciss['current_position'].iloc[0]) ==
417 ↪ vop_location2_dict.get(current_vop)):
418             print(current_vop, df_ciss['current_position'].iloc[0])
419             previous_vop = df_ciss['previous_position'].iloc[0]
420             print(f'met ciss {previous_vop}')
421             for j in range(current_index - 1, -1, -1):
422                 if data.loc[j, 'location'] == previous_vop:
423                     last_vop_end_time = data['timestamp'].iloc[j]
424                     print(f'met ciss is de start index {previous_vop}')
425                     return previous_vop, last_vop_end_time
426
427     #anders deze manier
428     candidate_vop = None
429     candidate_start = None
430     candidate_end = None
431
432     for j in range(current_index - 1, -1, -1):
433         row = data.loc[j]
434
435         # Als 'loctype' NaN is, sla deze rij over
436         if pd.isna(row['loctype']):
437             continue
438         if row['location'] == current_vop:
439             continue
440         if row['loctype'] != 'VOP':
441             if candidate_vop is not None:
442                 # check of het verblijf lang genoeg was
443                 duration = (candidate_end - candidate_start).total_seconds()
444                 if duration >= min_duration:
445                     print(f'met ruwe data is de start index {candidate_vop}')
446                     return candidate_vop, candidate_end
447             else:
448                 candidate_vop = None
449             continue
450         else:
451             if candidate_vop is None:
452                 candidate_vop = row['location']
453                 candidate_end = row['timestamp']
454                 candidate_start = row['timestamp']
455             elif row['location'] == candidate_vop:
456                 candidate_start = row['timestamp']
457             else:
458                 # nieuwe VOP gevonden, check de vorige
459                 duration = (candidate_end - candidate_start).total_seconds()

```

```

459         if duration >= min_duration:
460             print(f'met ruwe data is de start index {candidate_vop}')
461             return candidate_vop, candidate_end
462
463         else:
464             # reset voor nieuwe VOP
465             candidate_vop = row['location']
466             candidate_end = row['timestamp']
467             candidate_start = row['timestamp']
468
469     # Als je de loop helemaal doorloopt, check laatste candidate nog
470     if candidate_vop is not None and (candidate_end - candidate_start).total_seconds() >= min_duration:
471         print(f'met ruwe data is de start index {candidate_vop}')
472         return candidate_vop, candidate_end
473
474     return None, None
475
476 # In[ ]:
477
478
479
480 def check_VGRS_adjacent_stands(i, data, VGRS_data, cleaning_periods, additional_cleaned_vops, start_time, end_time, current_vop, minimum_cleantime, time_range_check):
481     t_range_start = start_time - timedelta(seconds=time_range_check)
482     t_range_end = end_time + timedelta(seconds=time_range_check)
483
484     additional_cleaned_vops.update(set(VGRS_data[
485         (VGRS_data['location2'] == vop_location2_dict.get(current_vop)) &
486         (VGRS_data['Begin'].between(t_range_start, t_range_end))]['VOP']))
487     print(f'het was {current_vop} + vanuit VGRS zonder filtering: {additional_cleaned_vops}')
488
489     #filterin 1: staat de additional vop uberhaupt in de data?
490     additional_cleaned_vops.intersection_update({vop for vop in additional_cleaned_vops if not
491         data[(data['location'] == vop) & (data['timestamp'].between(start_time, end_time))].empty})
492     print(f'filtering 1: {additional_cleaned_vops}')
493
494     #filtering 2: wordt de vop al gedetecteerd door het algoritme zelf?
495     future_check = t_range_end + timedelta(seconds=time_range_check)
496     match = data.index[data['timestamp'] == future_check]
497     if not match.empty:
498         idx_future = match[0]
499         lookahead_data = data.iloc[i:i + idx_future]
500         future_locations = list(zip(lookahead_data['location'], lookahead_data['loctype'], lookahead_data['timestamp']))
501         valid_future_locations = [(loc, loctype, timestamp) for loc, loctype, timestamp in future_locations if pd.isna(loc) and pd.isna(loctype) and
502             ← pd.isna(timestamp)]
503
504         to_remove = set()
505         for vop in additional_cleaned_vops:
506             vop_timestamps = [timestamp for loc, loctype, timestamp in valid_future_locations if loc == vop]
507             if vop_timestamps:
508                 first_occurrence = min(vop_timestamps)
509                 last_occurrence = max(vop_timestamps)
510                 duration = (last_occurrence - first_occurrence).total_seconds()
511                 if duration > minimum_cleantime:
512                     to_remove.add(vop)
513             if to_remove:
514                 additional_cleaned_vops -= to_remove
515             print(f'filtering 2: {additional_cleaned_vops}')
516
517     #filtering 3 --> staat de additional vop al in de cleaning periods die pas geleden zijn gedaan
518     time_window_start = start_time - timedelta(seconds=time_range_check)
519
520     recent_cleanings = [period for period in cleaning_periods
521         if time_window_start <= period['end_time'] < start_time]
522
523     for period in recent_cleanings:
524         additional_cleaned_vops.discard(period['vop'])
525     print(f'na filtering 3: {additional_cleaned_vops}')
526
527     additional_cleaned_vops.add(current_vop)
528
529     return additional_cleaned_vops
530
531 # In[ ]:
532
533
534 def check_ciss_adjacent_stands(i, data, df_ciss, cleaning_periods, additional_cleaned_vops, current_vop, start_time, end_time, minimum_cleantime, time_range_check):
535     future_check_ciss = None
536     for a in range(len(df_ciss)):
537         row = df_ciss.iloc[a]
538
539         # 1) current_vop in current_position
540         if row['current_position'] == current_vop:
541             # Check of end_datetime van ciss dicht bij start_time van casper ligt
542             if abs(row['end_datetime'] - start_time) <= timedelta(seconds=time_range_check):
543                 # Pak de volgende rij om te zien of de previous_position overeenkomt
544                 if a + 1 < len(df_ciss):
545                     next_row = df_ciss.iloc[a+1]
546                     # Als de volgende rij's previous_position NIET gelijk is aan de huidige current_position...
547                     if next_row['previous_position'] != row['current_position']:
548                         #maar wel dezelfde location2 heeft: --> dezelfde kant van dezelfde baai
549                         if vop_location2_dict.get(next_row['previous_position']) == vop_location2_dict.get(current_vop):
550                             additional_cleaned_vops.add(next_row['previous_position'])
551                             future_check_ciss = next_row['start_datetime'] + timedelta(seconds=time_range_check)
552
553         # 2) current_vop in previous_position
554         if row['previous_position'] == current_vop:
555             # Check of start_datetime van CISS dicht bij end_time van casper ligt
556             if abs(row['start_datetime'] - end_time) <= timedelta(seconds=time_range_check):
557                 # Kijk naar de vorige rij om te zien of de current_position dezelfde is
558                 if a > 0:
559                     prev_row = df_ciss.iloc[a-1]
560                     # Als de vorige rij's current_position NIET gelijk is aan de huidige previous_position...
561                     if prev_row['current_position'] != row['previous_position']:
562                         #maar wel dezelfde location2 heeft: --> dezelfde kant van dezelfde baai
563                         if vop_location2_dict.get(prev_row['current_position']) == vop_location2_dict.get(current_vop):
564                             additional_cleaned_vops.add(prev_row['current_position'])
565                             future_check_ciss = row['start_datetime'] + timedelta(seconds=time_range_check)
566
567     #filtering 1
568     additional_cleaned_vops.intersection_update({vop for vop in additional_cleaned_vops if not
569         data[(data['location'] == vop) & (data['timestamp'].between(start_time, end_time))].empty})

```

```

569 print(f'filtering 1 ciss: {additional_cleaned_vops}')
570
571 #filtering 2
572 if future_check_ciss is not None:
573     match = (data['timestamp'] - future_check_ciss).abs().idxmin()
574     idx_future = match[0]
575     lookahead_data = data.iloc[i:idx_future]
576     future_locations = list(zip(lookahead_data['location'], lookahead_data['loctype'], lookahead_data['timestamp']))
577     valid_future_locations = [(loc, loctype, timestamp) for loc, loctype, timestamp in future_locations if pd.notna(loc) and pd.notna(loctype) and
    ↪ pd.notna(timestamp)]
578
579     to_remove = set()
580     for vop in additional_cleaned_vops:
581         vop_timestamps = [timestamp for loc, loctype, timestamp in valid_future_locations if loc == vop]
582         if vop_timestamps:
583             first_occurrence = min(vop_timestamps)
584             last_occurrence = max(vop_timestamps)
585             duration = (last_occurrence - first_occurrence).total_seconds()
586             if duration > minimum_cleantime:
587                 to_remove.add(vop)
588         if to_remove:
589             additional_cleaned_vops -= to_remove
590     print(f'filtering 2 ciss: {additional_cleaned_vops}')
591
592 #filtering 3
593 if cleaning_periods:
594     last_vop = cleaning_periods[-1]['vop']
595 else:
596     last_vop = None
597
598 if len(cleaning_periods) > 1:
599     second_last_vop = cleaning_periods[-2]['vop']
600 else:
601     second_last_vop = None
602
603 if last_vop in additional_cleaned_vops:
604     additional_cleaned_vops.discard(last_vop)
605 if second_last_vop in additional_cleaned_vops:
606     additional_cleaned_vops.discard(second_last_vop)
607 print(f'filtering 3 ciss: {additional_cleaned_vops}')
608
609 additional_cleaned_vops.add(current_vop)
610
611 return additional_cleaned_vops
612
613
614
615 # In[ ]:
616
617
618 def check_VGRS_within_bay(i, data, VGRS_data, cleaning_periods, potential_cleaned_vops, start_time, end_time, current_vop, minimum_cleantime, time_range_check):
619     t_range_start = start_time - timedelta(seconds=time_range_check)
620     t_range_end = end_time + timedelta(seconds=time_range_check)
621
622     potential_cleaned_vops.update(set(VGRS_data[
623         (VGRS_data['location'] == vop_location_dict.get(current_vop)) &
624         (VGRS_data['Begin'].between(t_range_start, t_range_end))]['VOP']))
625     print(f'het was {current_vop} + vanuit VGRS zonder filtering: {potential_cleaned_vops}')
626
627     #Filter 1: staan de additional vops uberhaupt in de data?
628     potential_cleaned_vops.intersection_update({vop for vop in potential_cleaned_vops if not
629         data[(data['location'] == vop) & (data['timestamp'].between(start_time, end_time))].empty})
630     print(f'filtering 1: {potential_cleaned_vops}')
631
632     #Filter 2: wordt de vop al gedetecteerd door het algoritme zelf?
633     future_check = t_range_end + timedelta(seconds=time_range_check)
634     match = data.index[data['timestamp'] == future_check]
635     if not match.empty:
636         idx_future = match[0]
637         lookahead_data = data.iloc[i:i + idx_future]
638         future_locations = list(zip(lookahead_data['location'], lookahead_data['loctype'], lookahead_data['timestamp']))
639         valid_future_locations = [(loc, loctype, timestamp) for loc, loctype, timestamp in future_locations if pd.notna(loc) and pd.notna(loctype) and
    ↪ pd.notna(timestamp)]
640
641         to_remove_vop = set()
642         for vop in potential_cleaned_vops:
643             potential_vop_timestamps = [timestamp for loc, loctype, timestamp in valid_future_locations if loc == vop]
644             if potential_vop_timestamps:
645                 first_occurrence = min(potential_vop_timestamps)
646                 last_occurrence = max(potential_vop_timestamps)
647                 duration = (last_occurrence - first_occurrence).total_seconds()
648                 if duration > minimum_cleantime:
649                     to_remove_vop.add(vop)
650             if to_remove_vop:
651                 potential_cleaned_vops -= to_remove_vop
652     print(f'filtering 2: {potential_cleaned_vops} zijn over')
653
654     #filtering 3 --> staat de additional vop al in de cleaning periods die pas geleden zijn gedaan
655     time_window_start = start_time - timedelta(seconds=time_range_check)
656
657     recent_cleanings = [period for period in cleaning_periods
658         if time_window_start <= period['end_time'] < start_time]
659
660     for period in recent_cleanings:
661         potential_cleaned_vops.discard(period['vop'])
662
663     print(f'na filtering 3: {potential_cleaned_vops}')
664
665     potential_cleaned_vops.add(current_vop)
666
667     return potential_cleaned_vops
668
669
670 # In[ ]:
671
672
673 def check_ciss_within_bay(i, data, df_ciss, cleaning_periods, potential_cleaned_vops, current_vop, start_time, end_time, minimum_cleantime, time_range_check):
674     future_check_ciss = None
675     for a in range(len(df_ciss)):
676         row = df_ciss.iloc[a]
677

```

```

678     # 1) current_vop in current_position
679     if row['current_position'] == current_vop:
680         # Check of end_datetime van ciss dicht bij start_time van casper ligt
681         if abs(row['end_datetime'] - start_time) <= timedelta(seconds=time_range_check):
682             # Pak de volgende rij om te zien of de previous_position overeenkomt
683             if a + 1 < len(df_ciss):
684                 next_row = df_ciss.iloc[a+1]
685                 # Als de volgende rij's previous_position NIET gelijk is aan de huidige current_position...
686                 if next_row['previous_position'] != row['current_position']:
687                     # en dezelfde location heeft: --> binnen dezelfde baai of platform
688                     if vop_location_dict.get(next_row['previous_position']) == vop_location_dict.get(current_vop):
689                         potential_cleaned_vops.add(next_row['previous_position'])
690                         future_check_ciss = next_row['start_datetime'] + timedelta(seconds=time_range_check)
691
692     # 2) current_vop in previous_position
693     if row['previous_position'] == current_vop:
694         # Check of start_datetime dicht bij end_time ligt
695         if abs(row['start_datetime'] - end_time) <= timedelta(seconds=time_range_check):
696             # Kijk naar de vorige rij om te zien of de current_position dezelfde is
697             if a > 0:
698                 prev_row = df_ciss.iloc[a-1]
699                 if prev_row['current_position'] != row['previous_position']:
700                     # Voeg de current_position van de huidige rij toe
701                     if vop_location_dict.get(prev_row['current_position']) == vop_location_dict.get(current_vop):
702                         potential_cleaned_vops.add(prev_row['current_position'])
703                         future_check_ciss = row['start_datetime'] + timedelta(seconds=time_range_check)
704
705     #filtering 1
706     potential_cleaned_vops.intersection_update({vop for vop in potential_cleaned_vops if not
707         data[(data['location'] == vop) & (data['timestamp'].between(start_time, end_time))].empty})
708     print(f'filtering 1 ciss: {potential_cleaned_vops}')
709
710     #filtering 2
711     if future_check_ciss is not None:
712         match = data.index[data['timestamp'] == future_check_ciss]
713         idx_future = match[0]
714         lookahead_data = data.iloc[:idx_future]
715
716         future_locations = list(zip(lookahead_data['location'], lookahead_data['loctype'], lookahead_data['timestamp']))
717         valid_future_locations = [(loc, loctype, timestamp) for loc, loctype, timestamp in future_locations if pd.isna(loc) and pd.isna(loctype) and
718             pd.isna(timestamp)]
719
720         to_remove = set()
721         for vop in potential_cleaned_vops:
722             vop_timestamps = [timestamp for loc, loctype, timestamp in valid_future_locations if loc == vop]
723             if vop_timestamps:
724                 first_occurrence = min(vop_timestamps)
725                 last_occurrence = max(vop_timestamps)
726                 duration = (last_occurrence - first_occurrence).total_seconds()
727                 if duration > minimum_cleantime:
728                     to_remove.add(vop)
729             if to_remove:
730                 potential_cleaned_vops -= to_remove
731             print(f'filtering 2 ciss: {potential_cleaned_vops}')
732
733     #filtering 3
734     if cleaning_periods:
735         last_vop = cleaning_periods[-1]['vop']
736     else:
737         last_vop = None
738
739     if len(cleaning_periods) > 1:
740         second_last_vop = cleaning_periods[-2]['vop']
741     else:
742         second_last_vop = None
743
744     if last_vop in potential_cleaned_vops:
745         potential_cleaned_vops.discard(last_vop)
746     if second_last_vop in potential_cleaned_vops:
747         potential_cleaned_vops.discard(second_last_vop)
748     print(f'filtering 3: {potential_cleaned_vops}')
749
750     potential_cleaned_vops.add(current_vop)
751
752     return potential_cleaned_vops
753
754 # In[ ]:
755
756 def end_cleaning(cleaning_periods, snow_period, current_vop, start_time, end_time, cleaning_duration, vehicle_id, too_long_cleaning):
757     """
758     Handelt het einde van een schoonmaakperiode af door het toe te voegen aan de lijst.
759     """
760     vop_type = vop_type_dict.get(current_vop, "Onbekend")
761
762     duration_minutes = cleaning_duration/60
763     start_sneeuw, eind_sneeuw = snow_period
764
765     cleaning_periods.append({
766         'snow_start': start_sneeuw,
767         'snow_end': eind_sneeuw,
768         'vehicle_id': vehicle_id,
769         'vop': current_vop,
770         'vop_type': vop_type,
771         'start_time': start_time,
772         'end_time': end_time,
773         'duration_seconds': cleaning_duration,
774         'duration_minutes': duration_minutes,
775         'too_long_cleaning': too_long_cleaning,
776     })
777
778 # In[ ]:
779
780 def calculate_travel_time(travel_times, snow_period, start_time, end_time, previous_vop, current_vop, vehicle_id, max_travel_time_within_bay,
781     max_travel_time_within_centrum, max_travel_time_outside_centrum, too_long_travel, time_gap_detected):
782     tt = (end_time-start_time).total_seconds()

```

```

787     tt_minutes = tt/60
788     route_type = None
789
790     if vop_location_dict.get(current_vop) == vop_location_dict.get(previous_vop): #nu aangenomen dat heel platform A dan binnen de baai is.
791         route_type = 'Within bay/platform'
792         if tt > max_travel_time_within_bay:
793             too_long_travel = True
794     elif vop_locationtype_dict.get(current_vop) == vop_locationtype_dict.get(previous_vop):
795         route_type = 'Within zone'
796         if tt > max_travel_time_within_centrum:
797             too_long_travel = True
798     else:
799         route_type = 'Cross-zone'
800         if tt > max_travel_time_outside_centrum:
801             too_long_travel = True
802
803     start_sneeuw, eind_sneeuw = snow_period
804
805     travel_times.append({
806         'snow_start': start_sneeuw,
807         'snow_end': eind_sneeuw,
808         'vehicle_id': vehicle_id,
809         'from_vop': previous_vop,
810         'to_vop': current_vop,
811         'route_type': route_type,
812         'start_time': start_time,
813         'end_time': end_time,
814         'travel_time_minutes': tt_minutes,
815         'travel_time_seconds': tt,
816         'time_gap_detected': time_gap_detected,
817         'too_long_travel': too_long_travel,
818     })
819
820     return route_type, too_long_travel
821
822 # In[ ]:
823
824
825 def calculate_break_time(break_times, snow_period, current_vop, start_time, end_time, vehicle_id):
826
827     break_duration_seconds = (end_time - start_time).total_seconds()
828     break_duration_minutes = break_duration_seconds/60
829
830     start_sneeuw, eind_sneeuw = snow_period
831
832     break_times.append({
833         'snow_start': start_sneeuw,
834         'snow_end': eind_sneeuw,
835         'vehicle_id': vehicle_id,
836         'vop': current_vop,
837         'start_time': start_time,
838         'end_time': end_time,
839         'duration_seconds': break_duration_seconds,
840         'duration_minutes': break_duration_minutes,
841     })
842
843     return True
844
845 # In[ ]:
846
847
848 def determine_cleaning_capacity(snow_period, data, df_ciss, VGRS_data, vehicle_id, lookahead_window, minimum_cleantime, maximum_cleantime, max_travel_time_within_bay,
849 ↪ max_travel_time_within_centrum,
850 ↪ max_travel_time_outside_centrum, constant_within_bay_travel_time, time_range_check, time_gap_threshold, polygon_threshold,
851 ↪ polygon_graph):
852
853     cleaning_periods = []
854     travel_times = []
855     break_times = []
856     cycli = []
857     all_vops = set(data.loc[data['loctype'] == 'VOP', 'location'])
858     skipped_vops = {vop for vop in all_vops if vop.startswith(('K', 'R', 'Y', 'S', 'U'))}
859     temporary_skipped_vops = set()
860     in_cleaning = False
861     start_time, current_vop = None, None
862     last_vop_end_time = None
863     previous_vop = None
864     time_gap_detected = False
865
866     i = 0
867     while i < len(data):
868
869         current_row = data.iloc[i]
870         current_time, current_location, current_loctype = current_row['timestamp'], current_row['location'], current_row['loctype']
871
872         #wanneer starten met cleaning?
873         if previous_vop is None and not in_cleaning:
874             if current_loctype == 'VOP' and current_location in skipped_vops:
875                 i += 1
876                 continue
877             elif current_loctype != 'VOP':
878                 i += 1
879                 continue
880             else:
881                 pass
882
883         if not in_cleaning:
884             if current_loctype == 'VOP':
885                 if current_location in temporary_skipped_vops:
886                     i += 1
887                     continue
888                 else:
889                     #start cleaning
890                     in_cleaning = True
891                     start_time = current_time
892                     current_vop = current_location
893                     i += 1
894                     continue
895             else:
896                 i += 1
897                 continue

```

```

897 #als je aan het schoonmaken bent ...
898 if in_cleaning:
899     #als V4 op CDF, dan zien als 1 grote periode
900     if vehicle_id == 'V4' and vop_location_dict.get(current_vop) == 'CDF':
901         if (vop_location_dict.get(current_location) == 'CDF') or (current_location == 'A20') or (pd.isna(current_location)):
902             i += 1
903             continue
904         else:
905             end_time = data.iloc[i-1]['timestamp']
906             (last_vop_end_time, previous_vop, temporary_skipped_vops, skipped_vops, time_gap_detected, i) = decide_if_end_cleaning(snow_period, current_vop,
907                 ↪ start_time, end_time, vehicle_id, break_times, cleaning_periods, travel_times, cycli,
908                     minimum_cleantime, maximum_cleantime,
909                     ↪ max_travel_time_within_bay,
910                     ↪ max_travel_time_within_centrum,
911                     ↪ max_travel_time_outside_centrum,
912                     ↪ constant_within_bay_travel_time,
913                     ↪ time_range_check, time_gap_detected, last_vop_end_time,
914                     ↪ previous_vop, temporary_skipped_vops, skipped_vops,
915                     ↪ i, data, VGRS_data, df_ciss)
916
917             in_cleaning = False
918             start_time, current_vop = None, None
919             # temporary_skipped_vops.clear()
920             continue
921
922 # Kijk vooruit in de tijd om te zien of het schoonmaken eindigt
923 lookahead_data = data.iloc[i:i + lookahead_window]
924 future_locations = list(zip(lookahead_data['location'], lookahead_data['loctype'])) if not lookahead_data.empty else []
925 valid_future_locations = [loc for loc in future_locations if pd.notna(loc[0]) and pd.notna(loc[1])]
926
927 #check timestep continuity
928 if i > 0:
929     previous_time = data.iloc[i - 1]['timestamp']
930     time_gap = (current_time - previous_time).total_seconds()
931     if time_gap > time_gap_threshold:
932         print(f"grote timegap aanwezig, {vehicle_id}, om {previous_time} - {current_time}")
933         # Zoek naar de laatste geldige rij waar de locatie overeenkomt met current_vop -> deze range vanwege de logica verderop
934         i_timegap = i
935         for j in range(i - 1, max(i - lookahead_window - 1, -1), -1):
936             if data.iloc[j]['loctype'] == 'VOP' and data.iloc[j]['location'] == current_vop:
937                 last_valid_index = j
938                 break
939
940     end_time = data.iloc[last_valid_index]['timestamp']
941     (last_vop_end_time, previous_vop, temporary_skipped_vops, skipped_vops, time_gap_detected, i) = decide_if_end_cleaning(snow_period, current_vop,
942         ↪ start_time, end_time, vehicle_id, break_times, cleaning_periods, travel_times, cycli,
943             minimum_cleantime, maximum_cleantime,
944             ↪ max_travel_time_within_bay,
945             ↪ max_travel_time_within_centrum,
946             ↪ max_travel_time_outside_centrum,
947             ↪ constant_within_bay_travel_time,
948             ↪ time_range_check, time_gap_detected, last_vop_end_time,
949             ↪ previous_vop, temporary_skipped_vops, skipped_vops,
950             ↪ i, data, VGRS_data, df_ciss)
951
952     in_cleaning = False
953     start_time, current_vop = None, None
954     time_gap_detected = True
955     if i == i_timegap:
956         i = i + 1
957     continue
958 else:
959     pass
960
961 if current_loctype != 'VOP' or current_location != current_vop:
962     # Als de auto volledig naar een nieuwe polygon gaat en niet terugkeert naar de huidige VOP
963     if all(future_loc != (current_vop, 'VOP') for future_loc in valid_future_locations):
964         end_time = data.iloc[i-1]['timestamp']
965         (last_vop_end_time, previous_vop, temporary_skipped_vops, skipped_vops, time_gap_detected, i) = decide_if_end_cleaning(snow_period, current_vop,
966             ↪ start_time, end_time, vehicle_id, break_times, cleaning_periods, travel_times, cycli,
967                 minimum_cleantime, maximum_cleantime,
968                 ↪ max_travel_time_within_bay,
969                 ↪ max_travel_time_within_centrum,
970                 ↪ max_travel_time_outside_centrum,
971                 ↪ constant_within_bay_travel_time,
972                 ↪ time_range_check, time_gap_detected, last_vop_end_time,
973                 ↪ previous_vop, temporary_skipped_vops, skipped_vops,
974                 ↪ i, data, VGRS_data, df_ciss)
975
976     in_cleaning = False
977     start_time, current_vop = None, None
978     continue
979
980 #Als auto nog teurg naar current VOP gaat, check of ie niet ergens verweg is geweest
981 else:
982     next_vop_index = next(index for index, loc in enumerate(valid_future_locations) if loc == (current_vop, 'VOP'))
983
984     # Controleer tussenliggende punten tot aan de volgende current_vop
985     end_cleaning_triggered = False
986     for intermediate_loc, intermediate_loctype in valid_future_locations[:next_vop_index]:
987         too_far_count = 0
988         threshold_count = 5
989
990         if intermediate_loctype == 'VOP':
991             if vop_location_dict.get(current_vop) == vop_location_dict.get(intermediate_loc): #binnen de baai
992                 too_far_count = 0
993             else:
994                 too_far_count += 1
995             if too_far_count >= threshold_count:
996                 print(f"Tussenliggende locatie {intermediate_loc} te ver weg tijdens terugkeer naar {current_vop}.")
997                 end_cleaning_triggered = True
998
999         # Einde schoonmaak vanwege te verre tussenliggende punten
1000         end_time = data.iloc[i-1]['timestamp']
1001         (last_vop_end_time, previous_vop, temporary_skipped_vops, skipped_vops, time_gap_detected, i) = decide_if_end_cleaning(snow_period,
1002             ↪ current_vop, start_time, end_time, vehicle_id, break_times, cleaning_periods, travel_times, cycli,
1003                 minimum_cleantime, maximum_cleantime,
1004                 ↪ max_travel_time_within_bay,
1005                 ↪ max_travel_time_within_centrum,
1006                 ↪ max_travel_time_outside_centrum,
1007                 ↪ constant_within_bay_travel_time,

```



```

982                                                                 time_range_check, time_gap_detected, last_vop_end_time,
983                                                                 ↪ previous_vop, temporary_skipped_vops, skipped_vops,
984                                                                 ↪ i, data, VGRS_data, df_ciss)
985
986         in_cleaning = False
987         start_time, current_vop = None, None
988         break
989     elif intermediate_loctype in ('TAXIWAY', 'RUNWAY'):
990         #sws te ver
991         too_far_count += 1
992         if too_far_count >= threshold_count:
993             print(f"Tussenliggende locatie {intermediate_loc} te ver weg tijdens terugkeer naar {current_vop}.")
994             end_cleaning_triggered = True
995
996         # Einde schoonmaak vanwege te verre tussenliggende punten
997         end_time = data.iloc[i-1]['timestamp']
998         (last_vop_end_time, previous_vop, temporary_skipped_vops, skipped_vops, time_gap_detected, i) = decide_if_end_cleaning(snow_period,
999                                                                 ↪ current_vop, start_time, end_time, vehicle_id, break_times, cleaning_periods, travel_times, cycli,
1000                                                                 minimum_cleantime, maximum_cleantime,
1001                                                                 ↪ max_travel_time_within_bay,
1002                                                                 ↪ max_travel_time_within_centrum,
1003                                                                 ↪ max_travel_time_outside_centrum,
1004                                                                 ↪ constant_within_bay_travel_time,
1005                                                                 time_range_check, time_gap_detected, last_vop_end_time,
1006                                                                 ↪ previous_vop, temporary_skipped_vops, skipped_vops,
1007                                                                 ↪ i, data, VGRS_data, df_ciss)
1008
1009         in_cleaning = False
1010         start_time, current_vop = None, None
1011         break
1012     else:
1013         #dus als de loc type taxilane is --> baseer op max aantal polygons binnen een baai van taxilane
1014         if calculate_polygon_distance(intermediate_loc, current_vop, polygon_graph) > polygon_threshold:
1015             too_far_count += 1
1016             if too_far_count >= threshold_count:
1017                 print(f"Tussenliggende locatie {intermediate_loc} te ver weg tijdens terugkeer naar {current_vop}.")
1018                 end_cleaning_triggered = True
1019
1020             # Einde schoonmaak vanwege te verre tussenliggende punten
1021             end_time = data.iloc[i-1]['timestamp']
1022             (last_vop_end_time, previous_vop, temporary_skipped_vops, skipped_vops, time_gap_detected, i) = decide_if_end_cleaning(snow_period,
1023                                                                 ↪ current_vop, start_time, end_time, vehicle_id, break_times, cleaning_periods, travel_times, cycli,
1024                                                                 minimum_cleantime, maximum_cleantime,
1025                                                                 ↪ max_travel_time_within_bay,
1026                                                                 ↪ max_travel_time_within_centrum,
1027                                                                 ↪ max_travel_time_outside_centrum,
1028                                                                 ↪ constant_within_bay_travel_time,
1029                                                                 time_range_check, time_gap_detected, last_vop_end_time,
1030                                                                 ↪ previous_vop, temporary_skipped_vops, skipped_vops,
1031                                                                 ↪ i, data, VGRS_data, df_ciss)
1032
1033         in_cleaning = False
1034         start_time, current_vop = None, None
1035         break
1036     else:
1037         too_far_count = 0
1038
1039         if not end_cleaning_triggered:
1040             i += 1
1041             continue
1042
1043         else:
1044             i += 1
1045             continue
1046
1047     #zorgt ervoor dat de break times alleen gelden als er daarna nog vops worden schoongemaakt.
1048     while break_times and cleaning_periods and break_times[-1]['start_time'] >= cleaning_periods[-1]['start_time']:
1049         print("Laatste break start_time is later dan laatste cleaning_period start_time. Verwijder de laatste break_time:", break_times[-1])
1050         break_times.pop()
1051
1052     travel_times_df = pd.DataFrame(travel_times)
1053     cleaning_periods_df = pd.DataFrame(cleaning_periods)
1054     break_times_df = pd.DataFrame(break_times)
1055     cycli_df = pd.DataFrame(cycli)
1056
1057     return travel_times_df, cleaning_periods_df, break_times_df, cycli_df
1058
1059 # In[ ]:
1060
1061 # SNOW DAYS
1062 all_travel_times = []
1063 all_cleaning_periods = []
1064 all_break_times = []
1065 all_cycli = []
1066
1067 selected_years = [2024, 2025]
1068 df = df_rg
1069 df_ciss = ciss_data_snow
1070 df_VGRS = VGRS_data
1071
1072 df_selected_years = df[df['timestamp'].dt.year.isin(selected_years)]
1073
1074 for start_datum, eind_datum in snow_periods:
1075     if start_datum.year not in selected_years and eind_datum.year not in selected_years:
1076         continue
1077     print(f"Initiele Sneeuwperiodes: {start_datum} - {eind_datum}")
1078
1079     #check of de time van snow periods wel gelijk is aan CISS
1080     ciss_data_to_check = df_ciss[
1081         (df_ciss['start_datetime'].dt.date >= start_datum.date()) &
1082         (df_ciss['start_datetime'].dt.date <= eind_datum.date())
1083     ]
1084
1085     if ciss_data_to_check.empty:
1086         continue
1087     ciss_data_to_check = ciss_data_to_check.sort_values(by='start_datetime').reset_index(drop=True)
1088
1089     #select start and end time based on CISS data
1090     ciss_start_time = ciss_data_to_check.iloc[0]['start_datetime']
1091     ciss_end_time = ciss_data_to_check.iloc[-1]['end_datetime']

```

```

1077 #kies tussen CISS en LHSP
1078 start_time = min(start_datum, ciss_start_time)
1079 end_time = max(eind_datum, ciss_end_time)
1080 snow_period = (start_time, end_time)
1081 print(f"Sneeuwperiode met ciss: {start_time} - {end_time}")
1082
1083 for vehicle_id in df_selected_years['vehicle_registration'].unique():
1084     print(f"Voertuig {vehicle_id}")
1085
1086     #Ciss data
1087     filtered_ciss_data = df_ciss[
1088         (df_ciss['tractor_movement'] == vehicle_id) &
1089         (df_ciss['start_datetime'] >= start_time) &
1090         (df_ciss['end_datetime'] <= end_time)]
1091
1092     if filtered_ciss_data.empty:
1093         print(f"Geen CISS-data voor voertuig {vehicle_id} in deze periode, skipping...")
1094         continue
1095     filtered_ciss_data = filtered_ciss_data.sort_values(by='start_datetime').reset_index(drop=True)
1096
1097     #Casper data
1098     df_filtered = df_selected_years[
1099         (df_selected_years['timestamp'] >= start_time) &
1100         (df_selected_years['timestamp'] <= end_time) &
1101         (df_selected_years['vehicle_registration'] == vehicle_id)]
1102
1103     if df_filtered.empty:
1104         print(f"Voertuig {vehicle_id} heeft geen data in deze periode, skipping...")
1105         continue
1106     df_filtered = df_filtered.sort_values(by=['timestamp']).reset_index(drop=True)
1107     # print(df_filtered[['location', 'timestamp']])
1108
1109     #VGRS-data
1110     VGRS_filtered = df_VGRS[
1111         (df_VGRS['Uitvoering door'] == vehicle_id) &
1112         (df_VGRS['Begin'] >= start_time) &
1113         (df_VGRS['Begin'] <= end_time)]
1114
1115     if VGRS_filtered.empty:
1116         VGRS_filtered = pd.DataFrame()
1117     else:
1118         VGRS_filtered = VGRS_filtered.sort_values(by='Begin').reset_index(drop=True)
1119
1120     travel_times_df, cleaning_periods_df, break_times_df, cycli_df = determine_cleaning_capacity(snow_period, df_filtered, filtered_ciss_data, VGRS_filtered,
1121     ↪ vehicle_id,
1122                                     lookahead_window, minimum_cleantime, maximum_cleantime, max_travel_time_within_bay,
1123                                     ↪ max_travel_time_within_centrum,
1124                                     max_travel_time_outside_centrum, constant_within_bay_travel_time, time_range_check,
1125                                     ↪ time_gap_threshold, polygon_threshold, polygon_graph)
1126
1127     all_travel_times.append(travel_times_df)
1128     all_cleaning_periods.append(cleaning_periods_df)
1129     all_break_times.append(break_times_df)
1130     all_cycli.append(cycli_df)
1131
1132 #Combineer alle dataframes
1133 final_travel_times = pd.concat(all_travel_times, ignore_index=True)
1134 final_cleaning_periods = pd.concat(all_cleaning_periods, ignore_index=True)
1135 final_break_times = pd.concat(all_break_times, ignore_index=True)
1136 final_cycli = pd.concat(all_cycli, ignore_index=True)
1137
1138 # In[ ]:
1139
1140 get_ipython().system('jupyter nbconvert --to script Radardata_for_overleaf.ipynb')
1141
1142 # In[ ]:
1143
1144
1145
1146
1147

```

D.2. Simulation model

```

1 #!/usr/bin/env python
2 # coding: utf-8
3
4 # In[1]:
5
6
7 import os
8 import pandas as pd
9 import numpy as np
10 import matplotlib.pyplot as plt
11 import seaborn as sns
12 import matplotlib.dates as mdates
13 from datetime import timedelta, date, time
14 import math
15 import matplotlib.dates as mdates
16 import matplotlib.gridspec as gridspec
17 from collections import defaultdict
18 import matplotlib.patches as patches
19 from matplotlib.patches import FancyBboxPatch, BoxStyle
20
21
22 # In[ ]:
23
24
25 base_directory = "/Users/laradegeus/Library/Mobile Documents/com-apple-CloudDocs/TIL/THESIS/THESISdata/"
26

```

```

27 file_name1 = 'flight_schedule_08_01_2025.csv'
28 VOP_information_file_name = "VOP_information.xlsx"
29
30 #flight schedule
31 file_path2 = os.path.join(base_directory, file_name1)
32 df_flight_schedule = pd.read_csv(file_path2, sep=",")
33
34 #VOPtypes
35 file_path3_1 = os.path.join(base_directory, VOP_information_file_name)
36 vop_information = pd.read_excel(file_path3_1)
37
38
39 # In[169]:
40
41
42 #make dictionary of VOP types for later use
43 vop_type_dict = vop_information.set_index('VOP')['type'].to_dict()
44 vop_type2_dict = vop_information.set_index('VOP')['type2'].to_dict()
45
46 #make dictionary of VOP location for later use
47 vop_location_dict = dict(zip(vop_information['VOP'], vop_information['location']))
48
49 vop_location2_dict = dict(zip(vop_information['VOP'], vop_information['location2']))
50
51 #make dictionary of VOP location type (Centrum or not) for later use
52 vop_locationtype_dict = dict(zip(vop_information['VOP'], vop_information['locationtype']))
53
54
55 # In[ ]:
56
57
58 df_flight_schedule['DateTime Scheduled'] = pd.to_datetime(df_flight_schedule['DateTime Scheduled']).dt.tz_localize(None)
59 df_flight_schedule['VOP'] = df_flight_schedule['GateRampIDRamp'].str.split('|').str[0]
60 df_flight_schedule['VOP_type'] = df_flight_schedule['VOP'].map(vop_type_dict)
61 df_flight_schedule['location_type'] = df_flight_schedule['VOP'].map(vop_location_dict)
62
63 nan_rows2 = df_flight_schedule[df_flight_schedule.isna().any(axis=1)]
64 nan_df2 = df_flight_schedule[df_flight_schedule.isna().any(axis=1)]
65 df_flight_schedule = df_flight_schedule.dropna(how='any')
66
67
68 # In[ ]:
69
70
71 #kleur for plots
72 donker_blaauw = (0.0078, 0.0706, 0.3059) #021259
73 midden_blaauw = (0.282, 0.518, 0.894) #4877E4
74 koning_blaauw = '#072ac8'
75
76
77 # ## INPUT variabelen
78
79 # In[ ]:
80
81
82 #SITUATION SPECIFIC INPUTS --> determine these based on the upcoming conditions!
83
84 #kies welk scenario: low - middle - high
85 scenario = 'middle'
86
87 #start-end of period of the inbound flights you want to see
88 start_window = pd.to_datetime('2025-01-08 00:00')
89 end_window = pd.to_datetime('2025-01-08 23:59')
90
91 #period of snow
92 sneeuw_perioden = [
93     {'start': '2025-01-08 03:00', 'end': '2025-01-08 11:00'}
94 ]
95
96 #cleaning necessary until
97 stop_cleaning = pd.to_datetime('2025-01-12 23:59') #default
98
99 #intensity of snow, in total accumulation cm
100 sneeuw_accumulatie = [
101     {'start': '2025-01-08 03:00', 'end': '2025-01-08 05:00', 'total_accumulation': 'low'},
102     {'start': '2025-01-08 05:00', 'end': '2025-01-08 10:00', 'total_accumulation': 'high'},
103     {'start': '2025-01-08 10:00', 'end': '2025-01-08 11:00', 'total_accumulation': 'low'},
104 ]
105
106 #how many teams are active?
107 teams_during_snow = 4
108 extra_after_snow = 2
109
110 #determined runway clearing scenario's during sector briefing
111 determined_runway_scenarios = [
112     {'start': '2025-01-08 03:00', 'end': '2025-01-08 06:00', 'scenario': 'C'},
113     {'start': '2025-01-08 06:00', 'end': '2025-01-08 07:00', 'scenario': 'D'},
114     {'start': '2025-01-08 07:00', 'end': '2025-01-08 08:00', 'scenario': 'E'},
115     {'start': '2025-01-08 08:00', 'end': '2025-01-08 09:00', 'scenario': 'D'},
116     {'start': '2025-01-08 09:00', 'end': '2025-01-08 10:00', 'scenario': 'E'},
117     {'start': '2025-01-08 10:00', 'end': '2025-01-08 11:00', 'scenario': 'D'},
118 ]
119
120 determined_deicing_capacity = [
121     {'start': '2025-01-08 07:00', 'end': '2025-01-08 10:00', 'restricted_deicing_capacity': 12},
122     {'start': '2025-01-08 10:00', 'end': '2025-01-08 11:00', 'restricted_deicing_capacity': 24},
123     {'start': '2025-01-08 11:00', 'end': '2025-01-08 15:00', 'restricted_deicing_capacity': 26},
124 ]
125
126 #beyond runway capacity restrictions, additional restrictions for de-icing or vop cleaning
127 additional_capacity_restrictions = [
128     {'start': '2025-01-08 06:30', 'end': '2025-01-08 07:00', 'restricted_capacity': 12},
129     {'start': '2025-01-08 10:00', 'end': '2025-01-08 12:00', 'restricted_capacity': 12}
130 ]
131
132 #shifts scheduled
133 first_shift_start = '2025-01-08 02:00'
134
135
136 # In[ ]:
137

```

```

138
139
140
141 #time slots of inbound flights
142 TIME_SLOT_MINUTES = 30
143
144 #runway capacity scenarios
145 runway_scenario_map = {
146     'E': 10,
147     'D': 17,
148     'C': 35,
149     'B': 68}
150
151 #clean time of one runway
152 runway_clean_time = 40
153
154 #afhandeltijd van vop --> voordat er weer een nieuw vliegtuig op kan komen
155 afhandeltijd = {
156     'NABO': 50,
157     'WIBO': 75}
158
159 #capacity per type of cleaning --> low, middle, high
160 capacity_data_low = [
161     {'route_type': 'Within bay/platform', 'vop_type': 'NABO', 'snow_accumulation': 'low', 'capacity_per_hour': 7.2},
162     {'route_type': 'Within bay/platform', 'vop_type': 'NABO', 'snow_accumulation': 'high', 'capacity_per_hour': 5.7},
163     {'route_type': 'Within bay/platform', 'vop_type': 'WIBO', 'snow_accumulation': 'low', 'capacity_per_hour': 5.5},
164     {'route_type': 'Within bay/platform', 'vop_type': 'WIBO', 'snow_accumulation': 'high', 'capacity_per_hour': 5.5},
165     {'route_type': 'Within zone', 'vop_type': 'NABO', 'snow_accumulation': 'low', 'capacity_per_hour': 3.3},
166     {'route_type': 'Within zone', 'vop_type': 'NABO', 'snow_accumulation': 'high', 'capacity_per_hour': 3.3},
167     {'route_type': 'Within zone', 'vop_type': 'WIBO', 'snow_accumulation': 'low', 'capacity_per_hour': 3.3},
168     {'route_type': 'Within zone', 'vop_type': 'WIBO', 'snow_accumulation': 'high', 'capacity_per_hour': 3.3},
169 ]
170
171 capacity_data_middle = [
172     {'route_type': 'Within bay/platform', 'vop_type': 'NABO', 'snow_accumulation': 'low', 'capacity_per_hour': 4.9},
173     {'route_type': 'Within bay/platform', 'vop_type': 'NABO', 'snow_accumulation': 'high', 'capacity_per_hour': 3.7},
174     {'route_type': 'Within bay/platform', 'vop_type': 'WIBO', 'snow_accumulation': 'low', 'capacity_per_hour': 3.4},
175     {'route_type': 'Within bay/platform', 'vop_type': 'WIBO', 'snow_accumulation': 'high', 'capacity_per_hour': 3.4},
176     {'route_type': 'Within zone', 'vop_type': 'NABO', 'snow_accumulation': 'low', 'capacity_per_hour': 2.2},
177     {'route_type': 'Within zone', 'vop_type': 'NABO', 'snow_accumulation': 'high', 'capacity_per_hour': 2.2},
178     {'route_type': 'Within zone', 'vop_type': 'WIBO', 'snow_accumulation': 'low', 'capacity_per_hour': 2.2},
179     {'route_type': 'Within zone', 'vop_type': 'WIBO', 'snow_accumulation': 'high', 'capacity_per_hour': 2.2},
180 ]
181
182 capacity_data_high = [
183     {'route_type': 'Within bay/platform', 'vop_type': 'NABO', 'snow_accumulation': 'low', 'capacity_per_hour': 3.2},
184     {'route_type': 'Within bay/platform', 'vop_type': 'NABO', 'snow_accumulation': 'high', 'capacity_per_hour': 1.9},
185     {'route_type': 'Within bay/platform', 'vop_type': 'WIBO', 'snow_accumulation': 'low', 'capacity_per_hour': 2.0},
186     {'route_type': 'Within bay/platform', 'vop_type': 'WIBO', 'snow_accumulation': 'high', 'capacity_per_hour': 2.0},
187     {'route_type': 'Within zone', 'vop_type': 'NABO', 'snow_accumulation': 'low', 'capacity_per_hour': 1.8},
188     {'route_type': 'Within zone', 'vop_type': 'NABO', 'snow_accumulation': 'high', 'capacity_per_hour': 1.8},
189     {'route_type': 'Within zone', 'vop_type': 'WIBO', 'snow_accumulation': 'low', 'capacity_per_hour': 1.8},
190     {'route_type': 'Within zone', 'vop_type': 'WIBO', 'snow_accumulation': 'high', 'capacity_per_hour': 1.8},
191 ]
192
193 capacity_scenarios = {
194     'low': capacity_data_low,
195     'middle': capacity_data_middle,
196     'high': capacity_data_high,
197 }
198
199 #hoeveel routes zijn binnen de baai, en hoeveel van baai naar baai
200 route_type_ratio = 0.58 # % within bay
201
202 #break periods
203 break_duration = 35 #minutes
204 break_after_snow_start = 1800
205 shift_duration = 480 #minutes
206
207
208 # ## CODE ZONDER CLEANED STANDS
209
210 # In[ ]:
211
212
213 #MODEL INPUT
214 df_flight_schedule = df_flight_schedule[
215     (df_flight_schedule['DateTime Scheduled'] >= start_window) &
216     (df_flight_schedule['DateTime Scheduled'] <= end_window)]
217
218 capacity_df = pd.DataFrame(capacity_data)
219
220 time_index = pd.date_range(start=start_window, end=end_window, freq=f'{TIME_SLOT_MINUTES}min')
221 df_snow = pd.DataFrame({'slot': time_index})
222
223 #sneeuw
224 def is_snow(x):
225     for periode in sneeuw_periodes:
226         start = pd.to_datetime(periode['start']).time()
227         end = pd.to_datetime(periode['end']).time()
228         if start <= x.time() < end:
229             return True
230     return False
231 df_snow['snow'] = df_snow['slot'].apply(is_snow)
232 first_snow_slot = df_snow[df_snow['snow'] == True]['slot'].min()
233 last_snow_slot = df_snow[df_snow['snow'] == True]['slot'].max()
234
235 #hoeveel teams
236 df_teams = df_snow.copy()
237 df_teams['n_teams'] = 0
238 df_teams.loc[df_teams['slot'] >= first_snow_slot, 'n_teams'] = teams_during_snow
239
240 for i in range(1, len(df_teams)):
241     if df_teams.loc[i-1, 'snow'] == True and df_teams.loc[i, 'snow'] == False:
242         # vanaf hier extra team
243         df_teams.loc[i:, 'n_teams'] = teams_during_snow + extra_after_snow
244         break
245
246 # break times --> less teams available
247 df_teams['n_teams_available'] = df_teams['n_teams']
248 total_window_minutes = (end_window - start_window).total_seconds() / 60

```

```

249 n_shifts = math.ceil(total_window_minutes / shift_duration)
250 start_shift_times = [pd.Timestamp(first_shift_start) + pd.Timedelta(minutes=i * shift_duration) for i in range(n_shifts)]
251
252 for shift_start_time in start_shift_times:
253     break_start_time = shift_start_time + pd.Timedelta(minutes=break_after_snow_start)
254     teams_at_break_start = df_teams.loc[df_teams['slot'] == break_start_time, 'n_teams'].values
255     if len(teams_at_break_start) > 0:
256         total_break_minutes = teams_at_break_start[0] * break_duration
257         rounded_break_minutes = math.ceil(total_break_minutes / TIME_SLOT_MINUTES) * TIME_SLOT_MINUTES
258         break_end_time = break_start_time + pd.Timedelta(minutes=rounded_break_minutes)
259         mask = (df_teams['slot'] >= break_start_time) & (df_teams['slot'] < break_end_time)
260         df_teams.loc[mask, 'n_teams_available'] -= 1
261     else:
262         total_break_minutes = 0
263
264 # Stop flights in een time slot
265 def round_to_slot(dt, minutes):
266     discard = pd.Timedelta(minutes=dt.minute % minutes,
267                             seconds=dt.second,
268                             microseconds=dt.microsecond)
269     return dt - discard
270
271 df_flight_schedule['slot'] = df_flight_schedule['DateTime Scheduled'].apply(lambda x: round_to_slot(x, TIME_SLOT_MINUTES))
272
273 df_flight_schedule_snow = pd.merge(df_flight_schedule, df_snow[['slot', 'snow']], on='slot', how='left')
274 df_flight_schedule_snow['needs_cleaning'] = False
275 df_flight_schedule_snow = df_flight_schedule_snow.sort_values(['slot'])
276
277 #-----
278 #inbound capaciteit aanpassen adhv runway capaciteit
279
280 def assign_capacity_restrictions(df, determined_runway_scenarios, runway_scenario_map, clean_minutes):
281     #runway capacity
282     df['runway_capacity'] = None
283
284     for periode in determined_runway_scenarios:
285         start_dt = pd.to_datetime(periode['start'])
286         end_dt = pd.to_datetime(periode['end'])
287         current_scenario = periode['scenario']
288         runway_capacity = runway_scenario_map[current_scenario]
289         df.loc[(df['slot'] >= start_dt) & (df['slot'] < end_dt), 'runway_capacity'] = runway_capacity
290
291     # Na laatste sneeuwslot: overgang naar scenario B
292     last_scenario = determined_runway_scenarios[-1]['scenario']
293     last_end = pd.to_datetime(determined_runway_scenarios[-1]['end'])
294     current_time = last_end
295
296     scenario_order = ['E', 'D', 'C', 'B']
297     current_index = scenario_order.index(last_scenario)
298
299     # Ga omhoog in de scenario-ladder tot B
300     for scenario in scenario_order[current_index:]:
301         if scenario == 'B':
302             break
303         else:
304             next_time = current_time + pd.Timedelta(minutes=clean_minutes)
305             next_capacity = runway_scenario_map[scenario]
306
307             df.loc[(df['slot'] >= current_time) & (df['slot'] < next_time), 'runway_capacity'] = next_capacity
308             current_time = next_time
309
310     #deicing capacity
311     df['deicing_cap_restriction'] = None
312     for deicing_restriction in determined_deicing_capacity:
313         start_deicing_restriction = pd.to_datetime(deicing_restriction['start'])
314         end_deicing_restriction = pd.to_datetime(deicing_restriction['end'])
315         restricted_deicing_cap = deicing_restriction['restricted_deicing_capacity']
316
317         df.loc[(df['slot'] >= start_deicing_restriction) & (df['slot'] < end_deicing_restriction), 'deicing_cap_restriction'] = restricted_deicing_cap
318
319     #additional restricted capacity --> voor vops
320     df['additional_cap_restriction'] = None
321     for restriction in additional_capacity_restrictions:
322         start_restriction = pd.to_datetime(restriction['start'])
323         end_restriction = pd.to_datetime(restriction['end'])
324         restricted_cap = restriction['restricted_capacity']
325
326         df.loc[(df['slot'] >= start_restriction) & (df['slot'] < end_restriction), 'additional_cap_restriction'] = restricted_cap
327     return df
328
329
330
331 def limit_flights_by_runway_capacity(df):
332     df['keep'] = True
333     removed_flights = []
334     removed_df = pd.DataFrame(columns=df.columns.drop('keep'))
335
336     for slot, group in df.groupby('slot'):
337         if pd.isna(group['runway_capacity'].iloc[0]) and pd.isna(group['deicing_cap_restriction'].iloc[0]) and pd.isna(group['additional_cap_restriction'].iloc[0]):
338             continue # geen beperking
339
340         runway_cap = group['runway_capacity'].dropna().iloc[0] if not group['runway_capacity'].dropna().empty else None
341         deicing_cap = group['deicing_cap_restriction'].dropna().iloc[0] if not group['deicing_cap_restriction'].dropna().empty else None
342         additional_cap = group['additional_cap_restriction'].dropna().iloc[0] if not group['additional_cap_restriction'].dropna().empty else None
343
344         # Combineer beperkingen
345         all_caps = [c for c in [runway_cap, deicing_cap, additional_cap] if c is not None]
346         adjusted_cap = min(all_caps) if all_caps else None
347         df.loc[df['slot'] == slot, 'adjusted_total_capacity'] = adjusted_cap
348
349         if adjusted_cap is None:
350             continue
351
352         #allowed capacity, rekening houdend met oneven getallen
353         hourly_capacity = int(adjusted_cap)
354         slots_per_hour = int(60 / TIME_SLOT_MINUTES)
355         base = hourly_capacity // slots_per_hour
356         extra = hourly_capacity % slots_per_hour # vaak 0 of 1
357
358         slot_minute = pd.to_datetime(slot).minute
359         slot_index_in_hour = slot_minute // TIME_SLOT_MINUTES

```

```

360
361     allowed = base + (1 if slot_index_in_hour < extra else 0)
362     df.loc[df['slot'] == slot, 'allowed_per_slot'] = allowed
363
364     if len(group) <= allowed:
365         continue
366
367     # Te veel? → schrappen volgens 95% NABO, 5% WIBO
368     n_remove = len(group) - allowed
369     nabo = group[group['VOP_type'] == 'NABO']
370     wibo = group[group['VOP_type'] == 'WIBO']
371
372     n_nabo = math.ceil(n_remove * 0.95)
373
374     drop_indices = []
375
376     # Verwijder eerst zoveel mogelijk NABO
377     if len(nabo) >= n_nabo:
378         drop_indices.extend(nabo.head(n_nabo).index.tolist())
379     else:
380         drop_indices.extend(nabo.index.tolist())
381
382     # Daarna eventueel WIBO
383     remaining_remove = n_remove - len(drop_indices)
384     if remaining_remove > 0 and len(wibo) >= remaining_remove:
385         drop_indices.extend(wibo.head(remaining_remove).index.tolist())
386     elif remaining_remove > 0:
387         drop_indices.extend(wibo.index.tolist())
388
389     df.loc[drop_indices, 'keep'] = False
390     removed_flights.extend(drop_indices)
391     removed_df = df.loc[removed_flights].drop(columns='keep')
392
393     return df[df['keep']].drop(columns='keep'), removed_df
394
395 df_flight_schedule_adjusted = assign_capacity_restrictions(df_flight_schedule_snow, determined_runway_scenarios, runway_scenario_map, runway_clean_time)
396 adjusted_flight_schedule, removed_flights = limit_flights_by_runway_capacity(df_flight_schedule_adjusted)
397 adjusted_flight_schedule = adjusted_flight_schedule.sort_values(by='DateTime Scheduled')
398
399 #-----
400
401 # Voor elke VOP apart bijhouden
402 already_cleaned = set()
403 last_occupied_time = {}
404 reassigned_gates = []
405
406 #check of VOP binnenkort nodig is
407 def is_vop_reserved_soon(vop, current_time, afhandeltijd_vop, df):
408     end_time = current_time + afhandeltijd_vop
409     future_flights = df[(df['DateTime Scheduled'] > current_time) &
410                        (df['DateTime Scheduled'] <= end_time) &
411                        (df['VOP'] == vop)]
412     return not future_flights.empty
413
414 def is_valid_gate(vop_schengen, flight_schengen):
415     if flight_schengen == 'Schengen':
416         return vop_schengen in ['Schengen', 'Beide', 'Niet van toepassing']
417     else: # Non-Schengen
418         return vop_schengen in ['Non-Schengen', 'Beide', 'Niet van toepassing']
419
420 def to_minutes(t):
421     return t.hour * 60 + t.minute
422
423 for idx, row in adjusted_flight_schedule.iterrows():
424     vop = row['VOP']
425     slot = row['slot']
426     snow = row['snow']
427     vop_type = row['VOP_type']
428     afhandeltijd_vop = pd.Timedelta(minutes=afhandeltijd_vop[vop_type])
429     flight_schengen_status = vop_type2_dict.get(vop)
430
431     if slot < first_snow_slot:
432         adjusted_flight_schedule.at[idx, 'needs_cleaning'] = False
433         continue
434
435     if slot >= stop_cleaning:
436         adjusted_flight_schedule.at[idx, 'needs_cleaning'] = False
437         continue
438
439     if snow and slot != last_snow_slot:
440         # reset schoonstatus
441         already_cleaned.discard(vop)
442         adjusted_flight_schedule.at[idx, 'needs_cleaning'] = True
443
444     elif vop in already_cleaned:
445         #vop is al schoon
446         last_occupied_time[vop] = slot
447     else:
448         #niet schoon dus check voor betere vop optie
449         candidate_vops = [c for c in already_cleaned if
450                          vop_type_dict.get(c) == vop_type and
451                          is_valid_gate(vop_type2_dict.get(c), flight_schengen_status) and
452                          to_minutes((last_occupied_time.get(c) + afhandeltijd_vop).time()) <= to_minutes(slot.time()) and
453                          not is_vop_reserved_soon(c, slot, afhandeltijd_vop, adjusted_flight_schedule)]
454
455         if candidate_vops:
456             selected_vop = candidate_vops[0] # je kunt hier ook op earliest available sorteren
457             adjusted_flight_schedule.at[idx, 'VOP'] = selected_vop
458             adjusted_flight_schedule.at[idx, 'needs_cleaning'] = False
459             last_occupied_time[selected_vop] = slot
460             reassigned_gates.append((idx, vop, selected_vop, slot))
461         else:
462             adjusted_flight_schedule.at[idx, 'needs_cleaning'] = True
463             already_cleaned.add(vop)
464             last_occupied_time[vop] = slot
465
466 #demand per slot per VOP-type
467 df_cleaning = adjusted_flight_schedule[adjusted_flight_schedule['needs_cleaning'] == True]
468
469 all_slots = pd.DataFrame({'slot': time_index}).assign(key=1).merge(
470     pd.DataFrame({'VOP_type': ['WIBO', 'NABO'], 'key': 1}),
471     on='key').drop(columns='key')

```

```

471
472 df_cleaning_demand = (
473     df_cleaning
474     .groupby(['slot', 'VOP_type'])
475     .agg(
476         flight_list=('Main Flt Nr', list),
477         count=('Main Flt Nr', 'count'),
478     )
479     .reset_index()
480 )
481
482 df_cleaning_demand = pd.merge(
483     all_slots,
484     df_cleaning_demand,
485     on=['slot', 'VOP_type'],
486     how='left'
487 )
488
489 df_cleaning_demand['flight_list'] = df_cleaning_demand['flight_list'].apply(lambda x: x if isinstance(x, list) else [])
490 df_cleaning_demand['count'] = df_cleaning_demand['count'].fillna(0).astype(int)
491
492 df_cleaning_demand = pd.merge(df_cleaning_demand, df_snow[['slot', 'snow']], on='slot', how='left')
493 df_cleaning_demand = pd.merge(df_cleaning_demand, df_teams[['slot', 'n_teams_available']], on='slot', how='left')
494
495 #add snow accumulation
496 df_accumulation = pd.DataFrame(sneeuw_accumulatie)
497 df_accumulation['start'] = pd.to_datetime(df_accumulation['start'])
498 df_accumulation['end'] = pd.to_datetime(df_accumulation['end'])
499
500 df_cleaning_demand['snow_accumulation'] = None
501 for _, row in df_accumulation.iterrows():
502     mask = (df_cleaning_demand['slot'] >= row['start']) & (df_cleaning_demand['slot'] < row['end'])
503     df_cleaning_demand.loc[mask, 'snow_accumulation'] = row['total_accumulation']
504 df_cleaning_demand['snow_accumulation'] = df_cleaning_demand['snow_accumulation'].fillna('low')
505
506 #-----
507 def custom_round(val):
508     frac = val - math.floor(val)
509     if frac < 0.4:
510         return math.floor(val)
511     else:
512         return math.ceil(val)
513
514 # CAPACITY, CLEANED VOPS & QUEUE
515 def get_hourly_cap(vop_type, snow_accumulation, count):
516     cap_within_bay = capacity_df[
517         (capacity_df['vop_type'] == vop_type) &
518         (capacity_df['snow_accumulation'] == snow_accumulation) &
519         (capacity_df['route_type'] == 'Within bay/platform')
520     ]['capacity_per_hour'].values
521
522     cap_cross_bays = capacity_df[
523         (capacity_df['vop_type'] == vop_type) &
524         (capacity_df['snow_accumulation'] == snow_accumulation) &
525         (capacity_df['route_type'] == 'Within zone')
526     ]['capacity_per_hour'].values
527
528     if len(cap_within_bay) == 0 or len(cap_cross_bays) == 0:
529         print(f" Geen match voor {vop_type}, {snow_accumulation}")
530         return 0
531
532     ratio = 1.0 if count == 0 else route_type_ratio
533
534     return ratio * cap_within_bay[0] + (1-ratio) * cap_cross_bays[0]
535
536
537 def clean_future_slots(df, slot, vop_type, remaining_capacity):
538     cleaned_list = {'WIBO': [], 'NABO': []}
539     other_type = 'WIBO' if vop_type == 'NABO' else 'NABO'
540
541     next_slots = sorted(df['slot'].unique())
542     current_index = next_slots.index(slot)
543     next_index = current_index + 1
544
545     while remaining_capacity > 0.01 and next_index < len(next_slots):
546         next_slot = next_slots[next_index]
547
548         #kijk of er nog VOPs van hetzelfde type zijn
549         match_main = (df['slot'] == next_slot) & (df['VOP_type'] == vop_type)
550         if match_main.any():
551             idx_main = df.loc[match_main].index[0]
552             flight_list_main = df.at[idx_main, 'adjusted_flight_list']
553
554             if flight_list_main:
555                 clean_now = min(remaining_capacity, len(flight_list_main))
556                 clean_flights = custom_round(clean_now)
557                 cleaned_list[vop_type] += flight_list_main[:clean_flights]
558                 df.at[idx_main, 'adjusted_flight_list'] = flight_list_main[clean_flights:]
559                 remaining_capacity -= clean_now
560
561             if remaining_capacity > 0.01:
562                 #probeer andere type in hetzelfde slot
563                 match_other = (df['slot'] == next_slot) & (df['VOP_type'] == other_type)
564                 if match_other.any():
565                     idx_other = df.loc[match_other].index[0]
566                     flight_list_other = df.at[idx_other, 'adjusted_flight_list']
567
568                     if flight_list_other:
569                         clean_now = min(remaining_capacity, len(flight_list_other))
570                         clean_flights = custom_round(clean_now)
571                         cleaned_list[other_type] += flight_list_other[:clean_flights]
572                         df.at[idx_other, 'adjusted_flight_list'] = flight_list_other[clean_flights:]
573                         remaining_capacity -= clean_now
574
575         next_index += 1
576
577     return cleaned_list
578
579 def dynamic_weighted_cleaning(slot, slot_df, df, queue_per_type, queue_list_per_type, slack_capacity_per_type):
580     results = []
581     VOP_TYPES = ['WIBO', 'NABO']

```

```

582
583
584 # Voeg queue toe
585 slot_df['queue_in'] = slot_df['VOP_type'].map(queue_per_type)
586 slot_df['total_demand'] = slot_df.apply(lambda row: len(row['adjusted_flight_list']) + len(queue_per_type[row['VOP_type']]), axis=1)
587 total_demand = slot_df['total_demand'].sum()
588 n_teams = slot_df['n_teams_available'].iloc[0]
589
590 if slot >= stop_cleaning:
591     total_demand = 0
592
593 if total_demand > 0:
594     #verplaats slack als nodig
595     demand_per_type = slot_df.groupby('VOP_type')['total_demand'].sum().to_dict()
596     for vop_type, slack in list(slack_capacity_per_type.items()):
597         if demand_per_type.get(vop_type, 0) == 0 and slack != 0:
598             other = 'WIBO' if vop_type == 'NABO' else 'NABO'
599             # verplaats slack
600             slack_capacity_per_type[other] += slack
601             slack_capacity_per_type[vop_type] = 0
602
603 # Bereken bijdrage aan gewogen capaciteit
604 slot_df['cap_contrib'] = slot_df.apply(
605     lambda row: get_hourly_cap(row['VOP_type'], row['snow_accumulation'], row['total_demand']) * row['total_demand'], axis=1)
606 total_cap_contrib = slot_df['cap_contrib'].sum()
607 weighted_avg_cap_per_team = total_cap_contrib / total_demand
608 total_capacity = (weighted_avg_cap_per_team / (60 / TIME_SLOT_MINUTES)) * n_teams
609
610 cap_for_type = {}
611 initial_capacity = {}
612 slack_capacity = {}
613 for _, row in slot_df.iterrows():
614     vop_type = row['VOP_type']
615     demand_share = row['total_demand'] / total_demand
616     capacity_for_type = demand_share * total_capacity
617     initial_capacity[vop_type] = capacity_for_type
618     cap_for_type[vop_type] = capacity_for_type + slack_capacity_per_type.get(vop_type, 0.0)
619     slack_capacity[vop_type] = cap_for_type[vop_type]
620     slack_capacity_per_type[vop_type] = 0.0
621
622 cleaned_queue_count = dict.fromkeys(VOP_TYPES, 0)
623 cleaned_queue_list = {vop_type: [] for vop_type in VOP_TYPES}
624
625 cleaned_cross_count = {}
626 cleaned_cross_list = {}
627
628 #eigen queue
629 for vop_type in VOP_TYPES:
630     q = queue_per_type[vop_type]
631     n = custom_round(min(cap_for_type[vop_type], len(q)))
632     cleaned_queue_count[vop_type] = n
633     cleaned_queue_list[vop_type] = q[:n]
634     cap_for_type[vop_type] -= n
635     queue_per_type[vop_type] = q[n:]
636
637 # cross-type queue cleanup
638 for vop_type in VOP_TYPES:
639     other = 'NABO' if vop_type == 'WIBO' else 'WIBO'
640     q_other = queue_per_type[other]
641     if cap_for_type[vop_type] > 0 and len(q_other) > 0:
642         n2 = custom_round(min(cap_for_type[vop_type], len(q_other)))
643         cleaned_cross_count[(vop_type, other)] = n2
644         cleaned_cross_list[(vop_type, other)] = q_other[:n2]
645         cap_for_type[vop_type] -= n2
646         queue_per_type[other] = q_other[n2:]
647     else:
648         cleaned_cross_count[(vop_type, other)] = 0
649         cleaned_cross_list[(vop_type, other)] = []
650
651 #na queue
652 for _, row in slot_df.iterrows():
653     vop_type = row['VOP_type']
654     other = 'NABO' if vop_type == 'WIBO' else 'WIBO'
655     cleaned_from_queue = cleaned_queue_list[vop_type]
656     cleaned_from_cross = cleaned_cross_list[(vop_type, other)]
657
658 remaining_capacity = cap_for_type[vop_type]
659 adjusted_flight_list = list(row['adjusted_flight_list'])
660 cleaned_from_currentflights = []
661 queue_new = []
662
663 # clean huidige vluchten met wat er nog over is
664 if remaining_capacity > 0 and adjusted_flight_list:
665     n_clean_from_currentflights = custom_round(min(remaining_capacity, len(adjusted_flight_list)))
666     cleaned_from_currentflights = adjusted_flight_list[:n_clean_from_currentflights]
667     queue_new = adjusted_flight_list[n_clean_from_currentflights:]
668     remaining_capacity -= n_clean_from_currentflights
669 else:
670     queue_new = adjusted_flight_list
671
672 extra_cleaned_list = []
673 other_extra_cleaned_list = []
674 if remaining_capacity > 0 and not row['snow']:
675     additional_cleaned_list = clean_future_slots(df, slot, vop_type, remaining_capacity)
676     extra_cleaned_list = additional_cleaned_list[vop_type]
677     other_extra_cleaned_list = additional_cleaned_list[other]
678     slack = remaining_capacity - len(extra_cleaned_list) - len(other_extra_cleaned_list)
679     if len(other_extra_cleaned_list) > 0:
680         slack_capacity_per_type[other] += slack
681     else:
682         slack_capacity_per_type[vop_type] += slack
683 elif remaining_capacity <= 0.4:
684     # capaciteit volledig gebruikt: slack = init-cap gebruikt
685     used_capacity = (
686         len(cleaned_from_queue) +
687         len(cleaned_from_cross) +
688         len(cleaned_from_currentflights))
689     slack_capacity_per_type[vop_type] += (slack_capacity[vop_type] - used_capacity)
690
691 else:
692     #sneeuwt, en er is nog capacity, dat wordt geen slack

```



```

693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803

pass

# bouw de totale cleaned_list en queue_list
cleaned_list = cleaned_from_queue + cleaned_from_currentflights
total_cleaned = (len(cleaned_from_queue)
                 + len(cleaned_from_currentflights)
                 + len(extra_cleaned_list))
total_queue = queuelist_per_type[vop_type] + queue_new

results.append({
    'slot': slot,
    'VOP_type': vop_type,
    'queue_in': row['queue_in'],
    'total_demand': row['total_demand'],
    'capacity': initial_capacity[vop_type],
    'capacity_slack': slack_capacity[vop_type],
    'cleaned_count': len(cleaned_list),
    'cleaned_list': cleaned_list,
    'other_cleaned_queue_count': len(cleaned_from_cross),
    'other_cleaned_queue': cleaned_from_cross,
    'extra_cleaned_list': extra_cleaned_list,
    'other_voptypes_cleaned_list': other_extra_cleaned_list,
    'total_cleaned': total_cleaned,
    'queue_list': total_queue,
    'queue_out': len(total_queue),
    'flight_list': row['flight_list'],
    'adjusted_flight_list': adjusted_flight_list,
    'active_teams': n_teams,
    'snow': row['snow'],
    'snow_accumulation': row['snow_accumulation']
})

# Update queue voor volgende slot
queue_per_type[vop_type] = len(total_queue)
queuelist_per_type[vop_type] = total_queue

else:
# total_demand == 0 → kijk naar future demand
future_slots = df[df['slot'] > slot]
future_demand = future_slots.groupby('slot')['adjusted_flight_list'].apply(lambda lists: sum(len(list) for list in lists))
future_slots_with_demand = future_demand[future_demand > 0]

if not future_slots_with_demand.empty:
# Er is nog toekomstige vraag
next_slot = future_slots_with_demand.index[0]
next_slot_df = df[df['slot'] == next_slot].copy()
next_slot_df['total_demand'] = len(next_slot_df['adjusted_flight_list'])
total_demand_next = next_slot_df['total_demand'].sum()

next_slot_df['cap_contrib'] = next_slot_df.apply(
    lambda row: get_hourly_cap(row['VOP_type'], row['snow_accumulation'], row['total_demand']) * row['total_demand'], axis=1)
total_cap_contrib = next_slot_df['cap_contrib'].sum()

weighted_avg_cap_per_team = total_cap_contrib / total_demand_next
total_capacity = (weighted_avg_cap_per_team / (60 / TIME_SLOT_MINUTES)) * n_teams

demand_per_type = next_slot_df.groupby('VOP_type')['total_demand'].sum().to_dict()
for vop_type, slack in list(slack_capacity_per_type.items()):
    if demand_per_type.get(vop_type, 0) == 0 and slack != 0:
        other = 'WIBO' if vop_type == 'NABO' else 'NABO'
        # verplaats slack
        slack_capacity_per_type[other] += slack
        slack_capacity_per_type[vop_type] = 0

# En dan capaciteit verdelen volgens de verhouding in de volgende slot
for idx, row in slot_df.iterrows():
    vop_type = row['VOP_type']
    other_type = 'WIBO' if vop_type == 'NABO' else 'NABO'
    adjusted_flight_list = row['adjusted_flight_list']
    queue_list = queuelist_per_type[row['VOP_type']]

    demand_share = next_slot_df[next_slot_df['VOP_type'] == vop_type]['total_demand'].sum() / total_demand_next
    capacity_for_type = demand_share * total_capacity
    # print(slack_capacity_per_type[vop_type])
    cap_for_type = capacity_for_type + slack_capacity_per_type.get(vop_type, 0.0)
    slack_capacity_per_type[vop_type] = 0.0

    total_cleaned = 0
    extra_cleaned_list = []
    other_extra_cleaned_list = []
    remaining_capacity = cap_for_type

    if remaining_capacity > 0 and not row['snow']:
        additional_cleaned_list = clean_future_slots(df, slot, vop_type, remaining_capacity)
        extra_cleaned_list = additional_cleaned_list[vop_type]
        other_extra_cleaned_list = additional_cleaned_list[other_type]

    #slack
    slack = remaining_capacity - len(extra_cleaned_list) - len(other_extra_cleaned_list)
    if len(other_extra_cleaned_list) > 0:
        slack_capacity_per_type[other_type] += slack
    else:
        slack_capacity_per_type[vop_type] += slack

#bouw de totale cleaned_list en queue_list
total_cleaned = len(extra_cleaned_list) + len(other_extra_cleaned_list)
total_queue = []

results.append({
    'slot': slot,
    'VOP_type': vop_type,
    'queue_in': row['queue_in'],
    'total_demand': row['total_demand'],
    'capacity': capacity_for_type,
    'capacity_slack': cap_for_type,
    'cleaned_count': 0,
    'cleaned_list': [],
    'other_cleaned_queue_count': 0,
    'other_cleaned_queue': [],
    'extra_cleaned_list': extra_cleaned_list,

```

```

804         'other_voetypes_cleaned_list': other_extra_cleaned_list,
805         'total_cleaned': total_cleaned,
806         'queue_list': total_queue,
807         'queue_out': len(total_queue),
808         'flight_list': row['flight_list'],
809         'adjusted_flight_list': adjusted_flight_list,
810         'active_teams': n_teams,
811         'snow': row['snow'],
812         'snow_accumulation': row['snow_accumulation']
813     })
814
815     # Update queue voor volgende slot
816     queue_per_type[vop_type] = len(total_queue)
817     queue_list_per_type[vop_type] = total_queue
818
819     else:
820     # total_demand == 0: fallback logic for no demand
821     for _, row in slot_df.iterrows():
822         vop_type = row['VOP_type']
823         slack_capacity_per_type[vop_type] = 0.0
824         results.append({
825             'slot': slot,
826             'VOP_type': vop_type,
827             'queue_in': queue_per_type[vop_type],
828             'total_demand': 0,
829             'capacity': 0,
830             'capacity_slack': 0,
831             'cleaned_count': 0,
832             'cleaned_list': [],
833             'other_cleaned_queue_count': 0,
834             'other_cleaned_queue': [],
835             'extra_cleaned_list': [],
836             'other_voetypes_cleaned_list': [],
837             'total_cleaned': 0,
838             'queue_list': [],
839             'queue_out': 0,
840             'flight_list': [],
841             'adjusted_flight_list': [],
842             'active_teams': n_teams,
843             'snow': row['snow'],
844             'snow_accumulation': row['snow_accumulation']
845         })
846
847         queue_per_type[vop_type] = 0
848         queue_list_per_type[vop_type] = []
849
850     return results, df, queue_per_type, queue_list_per_type, slack_capacity_per_type
851
852
853 # =====
854
855 results = []
856 queue_per_type = {'WIBO': 0, 'NABO': 0}
857 queue_list_per_type = {'WIBO': [], 'NABO': []}
858 slack_capacity_per_type = {'WIBO': 0.0, 'NABO': 0.0}
859 df = df_cleaning_demand.copy()
860 df['adjusted_count'] = df['count']
861 df['adjusted_flight_list'] = df['flight_list']
862
863 for slot in sorted(df['slot'].unique()):
864     slot_df = df[df['slot'] == slot].copy()
865     results_per_slot, df, queue_per_type, queue_list_per_type, slack_capacity_per_type = dynamic_weighted_cleaning(slot, slot_df, df, queue_per_type,
866     ↳ queue_list_per_type, slack_capacity_per_type)
867     results.extend(results_per_slot)
868
869 df_dynamic_cleaning = pd.DataFrame(results)
870 df_dynamic_cleaning['other_voetypes_cleaned_list'] = df_dynamic_cleaning['other_voetypes_cleaned_list'].apply(lambda x: x if isinstance(x, list) else [])
871 df_dynamic_cleaning['queue_length'] = df_dynamic_cleaning['queue_list'].apply(len)
872
873 # MODEL OUTPUT
874
875 bar_width = TIME_SLOT_MINUTES / 1440 # bijv. 30 / 1440 = 0.02083
876
877 color1 = koning_blauw
878 color2 = midden_blauw
879 queue_color = 'white'
880
881 def get_color(val):
882     if val == 0:
883         return '#e0e0e0'
884     elif val <= 3:
885         return '#ffcccc'
886     elif val <= 6:
887         return '#ff9999'
888     else:
889         return '#ff6666'
890
891 effective_cleaned_dict = defaultdict(float)
892
893 for _, row in df_dynamic_cleaning.iterrows():
894     slot = row['slot']
895     own_type = row['VOP_type']
896     other_type = 'WIBO' if own_type == 'NABO' else 'NABO'
897
898     effective_cleaned_dict[(slot, own_type)] += row['total_cleaned']
899     effective_cleaned_dict[(slot, other_type)] += len(row['other_voetypes_cleaned_list']) + row['other_cleaned_queue_count']
900
901 # Maak een nieuw DataFrame van de dictionary
902 df_effective_cleaned = pd.DataFrame([
903     {'slot': slot, 'VOP_type': vop_type, 'effective_cleaned': cleaned}
904     for (slot, vop_type), cleaned in effective_cleaned_dict.items()])
905
906 cleaned_pivot = df_effective_cleaned.pivot(index='slot', columns='VOP_type', values='effective_cleaned').fillna(0)
907 queue_pivot = df_dynamic_cleaning.pivot(index='slot', columns='VOP_type', values='queue_length').fillna(0)
908 capacity_per_slot = df_dynamic_cleaning.groupby('slot')['capacity_slack'].sum()
909
910 queue_total = df_dynamic_cleaning.groupby('slot')['queue_length'].sum()
911 all_slots = queue_total.index
912
913 cleaned_pivot = cleaned_pivot.reindex(all_slots, fill_value=0)

```

```

914 queue_pivot = queue_pivot.reindex(all_slots, fill_value=0)
915 capacity_per_slot = capacity_per_slot.reindex(all_slots, fill_value=0)
916
917 # Figure en subplots
918 fig = plt.figure(figsize=(22, 9))
919 gs = gridspec.GridSpec(2, 2, width_ratios=[6, 1], height_ratios=[3, 0.2], wspace=0.05, hspace=0.1)
920
921 ax1 = fig.add_subplot(gs[0, 0]) # barplot
922 ax2 = fig.add_subplot(gs[1, 0], sharex=ax1) # heatmap
923 ax_kpi = fig.add_subplot(gs[0, 1]) # KPI-dashboard
924
925
926 # Barplots (boven)
927 ax1.bar(all_slots, cleaned_pivot['WIBO'], width=bar_width, color=color1,
928        label='WIBO cleaned', edgecolor='white', linewidth=0.5, align='edge')
929
930 ax1.bar(all_slots, cleaned_pivot['NABO'], width=bar_width, color=color2,
931        bottom=cleaned_pivot['WIBO'], label='NABO cleaned', edgecolor='white', linewidth=0.5, align='edge')
932
933 queue_wibo_bottom = cleaned_pivot['WIBO'] + cleaned_pivot['NABO']
934 ax1.bar(all_slots, queue_pivot['WIBO'], width=bar_width, color=queue_color,
935        bottom=queue_wibo_bottom, hatch='///', edgecolor=color1, label='WIBO queue', linewidth=0.5, align='edge')
936
937 queue_nabo_bottom = queue_wibo_bottom + queue_pivot['WIBO']
938 ax1.bar(all_slots, queue_pivot['NABO'], width=bar_width, color=queue_color,
939        bottom=queue_nabo_bottom, hatch='///', edgecolor=color2, label='NABO queue', linewidth=0.5, align='edge')
940
941 # Capaciteitlijn (nu juist uitgelijnd)
942 ax1.step(all_slots, capacity_per_slot.values, label='Total capacity',
943        color=donker_blauw, linewidth=2, where='post')
944
945 # Sneeuwperiode(s)
946 for i, periode in enumerate(sneeuw_perioden):
947     ax1.axvspan(pd.to_datetime(periode['start']),
948               pd.to_datetime(periode['end']),
949               color='lightblue', alpha=0.3,
950               label='Snow period' if i == 0 else None)
951 from matplotlib.ticker import MaxNLocator
952 total_heights = (
953     cleaned_pivot[['WIBO', 'NABO']].sum(axis=1)
954     + queue_pivot[['WIBO', 'NABO']].sum(axis=1))
955
956 y_max = math.ceil(total_heights.max()) + 1
957 ax1.set_ylim(0, y_max)
958 ax1.yaxis.set_major_locator(MaxNLocator(integer=True))
959
960 ax1.set_ylabel('Number of aircraft stands')
961 ax1.legend()
962 ax1.grid(axis='y')
963
964 ax1.tick_params(
965     axis='x',
966     which='both',
967     bottom=False,
968     labelbottom=False
969 )
970
971 # Heatmap: Queue
972
973 rounded_queue = queue_pivot[['NABO', 'WIBO']].applymap(custom_round)
974 n_timeslots_with_queue = (rounded_queue.sum(axis=1) > 0).sum()
975 max_total_queue = rounded_queue.sum(axis=1).max()
976
977 for slot, row in queue_pivot.iterrows():
978     x = mdates.date2num(slot)
979
980     nabo_val = row['NABO']
981     wibo_val = row['WIBO']
982
983     # Bovenste helft: NABO
984     rect_nabo = patches.Rectangle((x, 0.5), bar_width, 0.5, color=get_color(nabo_val), ec='white')
985     ax2.add_patch(rect_nabo)
986     ax2.text(x + bar_width / 2, 0.75, str(int(nabo_val)) if nabo_val > 0 else '-',
987            ha='center', va='center', fontsize=9)
988
989     # Onderste helft: WIBO
990     rect_wibo = patches.Rectangle((x, 0), bar_width, 0.5, color=get_color(wibo_val), ec='white')
991     ax2.add_patch(rect_wibo)
992     ax2.text(x + bar_width / 2, 0.25, str(int(wibo_val)) if wibo_val > 0 else '-',
993            ha='center', va='center', fontsize=9)
994
995 for slot, row in queue_pivot.iterrows():
996     x = mdates.date2num(slot)
997
998     nabo_val = row['NABO']
999     val_nabo_rounded = custom_round(nabo_val)
1000     wibo_val = row['WIBO']
1001     val_wibo_rounded = custom_round(wibo_val)
1002
1003     # Bovenste helft: NABO
1004     rect_nabo = patches.Rectangle((x, 0.5), bar_width, 0.5, color=get_color(nabo_val), ec='white')
1005     ax2.add_patch(rect_nabo)
1006     ax2.text(x + bar_width / 2, 0.75, str(int(val_nabo_rounded)) if val_nabo_rounded > 0 else '-',
1007            ha='center', va='center', fontsize=9)
1008
1009     # Onderste helft: WIBO
1010     rect_wibo = patches.Rectangle((x, 0), bar_width, 0.5, color=get_color(wibo_val), ec='white')
1011     ax2.add_patch(rect_wibo)
1012     ax2.text(x + bar_width / 2, 0.25, str(int(val_wibo_rounded)) if val_wibo_rounded > 0 else '-',
1013            ha='center', va='center', fontsize=9)
1014
1015 ax2.xaxis_date()
1016 ax2.xaxis.set_major_locator(mdates.MinuteLocator(interval=30)) # Ticks op elk half uur
1017 ax2.xaxis.set_major_formatter(mdates.DateFormatter('%H:%M'))
1018
1019 ax2.set_xticks([mdates.date2num(t) for t in all_slots]) # ticks exact op slot-begin
1020 ax2.set_xticklabels([t.strftime('%H:%M') for t in all_slots]) # eventueel labels
1021
1022 ax2.set_ylim(0, 1)
1023 ax2.set_yticks([0.25, 0.75])

```

```

1025 ax2.set_yticklabels(['Wibo', 'Nabo'], fontsize=10)
1026
1027 for label in ax2.get_xticklabels():
1028     label.set_rotation(45)
1029     label.set_ha('right')
1030
1031 # KPI Dashboard
1032
1033 #1. max queue in a slot
1034 max_queue = queue_total.max()
1035
1036 #2. max slots that one aircraft is in queue
1037 from collections import Counter
1038 all_queued = [flight
1039               for lst in df_dynamic_cleaning['queue_list']
1040               for flight in lst]
1041 cnt = Counter(all_queued)
1042 flight_max, n_slots_max = cnt.most_common(1)[0]
1043
1044 #3. Total slots with queue
1045 num_slots_with_queue = (queue_total > 0).sum()
1046
1047 #4. total aircraft that has been in queue
1048 all_queued_per_type = defaultdict(list)
1049 for _, row in df_dynamic_cleaning.iterrows():
1050     vop = row['VOP_type']
1051     for flight in row['queue_list']:
1052         all_queued_per_type[vop].append(flight)
1053
1054 unique_in_queue = {
1055     vop: len(set(flights))
1056     for vop, flights in all_queued_per_type.items()}
1057
1058 unique_wibo = unique_in_queue.get('WIBO', 0)
1059 unique_nabo = unique_in_queue.get('NABO', 0)
1060
1061 #5. How many flights wait more than 30 minutes
1062 num_waited_multiple = sum(1 for _, c in cnt.items() if c > 1)
1063 icon = ""
1064 kpi_text = [
1065     # ("Queued aircraft:")
1066     ("Narrow body", f"{unique_nabo} "),
1067     ("Wide body", f"{unique_wibo}"),
1068     ("Max queue slot", f"{max_queue} "),
1069     ("Max waiting time", f"{n_slots_max} slots"),
1070     (">1 slot in queue", f"{num_waited_multiple}"),
1071 ]
1072
1073 pad_x = 0.04
1074 pad_y = 0.04
1075 round_style = BoxStyle.Round(pad=0.05, rounding_size=0.2)
1076
1077 rect = FancyBboxPatch(
1078     (-pad_x, -pad_y),
1079     1 + 2*pad_x,
1080     1 + 2*pad_y,
1081     transform=ax_kpi.transAxes,
1082     boxstyle=round_style,
1083     facecolor='#f0f0f0',
1084     edgecolor='none',
1085     zorder=0
1086 )
1087 ax_kpi.add_patch(rect)
1088 ax_kpi.axis('off')
1089
1090 icon = ""
1091 icon_size = 16
1092 text_color = donker_blaauw
1093 y = 1 - pad_y + 0.005
1094
1095 x_label = pad_x * 1.2
1096 x_icon = (1 - pad_x/2) - 0.02
1097 x_value = x_icon - 0.08
1098
1099
1100 ax_kpi.text(pad_x, y, "Aircraft in queue",
1101            fontsize=12, fontweight='bold',
1102            color=text_color, va='top', zorder=1)
1103 y -= 0.08
1104
1105 # 1) Aircraft in queue
1106 ax_kpi.text(x_label, y, "• Narrow body", fontsize=11, va='top', color=text_color, zorder=1)
1107 ax_kpi.text(x_value, y, f"{unique_nabo}", fontsize=11, ha='right', va='top', color=text_color, zorder=1)
1108 ax_kpi.text(x_icon, y, icon, fontsize=icon_size, ha='center', va='center',
1109            color=text_color, rotation=90, zorder=1)
1110 y -= 0.07
1111
1112 ax_kpi.text(x_label, y, "• Wide body", fontsize=11, va='top', color=text_color, zorder=1)
1113 ax_kpi.text(x_value, y, f"{unique_wibo}", fontsize=11, ha='right', va='top', color=text_color, zorder=1)
1114 ax_kpi.text(x_icon, y, icon, fontsize=icon_size, ha='center', va='center',
1115            color=text_color, rotation=90, zorder=1)
1116 y -= 0.12
1117
1118 # 2) Queue
1119 ax_kpi.text(pad_x, y, "Queue", fontsize=12, fontweight='bold', va='top', color=text_color, zorder=1)
1120 y -= 0.08
1121
1122 ax_kpi.text(x_label, y, "• Max/slot ", fontsize=11, va='top', color=text_color, zorder=1)
1123 ax_kpi.text(x_value, y, f"{max_queue}", fontsize=11, ha='right', va='top', color=text_color, zorder=1)
1124 ax_kpi.text(x_icon, y, icon, fontsize=icon_size, ha='center', va='center',
1125            color=text_color, rotation=90, zorder=1)
1126 y -= 0.07
1127
1128 ax_kpi.text(x_label, y, "• Total slots", fontsize=11, va='top', color=text_color, zorder=1)
1129 ax_kpi.text(x_value, y, f"{num_slots_with_queue}", fontsize=11, ha='right', va='top', color=text_color, zorder=1)
1130 y -= 0.12
1131
1132 # 3) Waiting time
1133 ax_kpi.text(pad_x, y, "Waiting time", fontsize=12, fontweight='bold', va='top', color=text_color, zorder=1)
1134 y -= 0.08
1135

```

```
1136 ax_kpi.text(x_label, y, "* Max slots",      fontsize=11, va='top', color=text_color, zorder=1)
1137 ax_kpi.text(x_value, y, f"{n_slots_max}",    fontsize=11, ha='right', va='top', color=text_color, zorder=1)
1138
1139
1140 y -= 0.07
1141
1142 ax_kpi.text(x_label, y, ">1 slot in queue",    fontsize=11, va='top', color=text_color, zorder=1)
1143 ax_kpi.text(x_value, y, f"{num_waited_multiple}",  fontsize=11, ha='right', va='top', color=text_color, zorder=1)
1144 ax_kpi.text(x_icon, y, icon,                  fontsize=icon_size, ha='center', va='center',
1145             color=text_color, rotation=90, zorder=1)
1146
1147 plt.tight_layout()
1148 plt.show()
1149
1150
1151 # In[ ]:
1152
1153
1154 get_ipython().system('jupyter nbconvert --to script MODEL_for_overleaf.ipynb')
1155
1156
1157 # In[ ]:
1158
1159
1160
1161
```