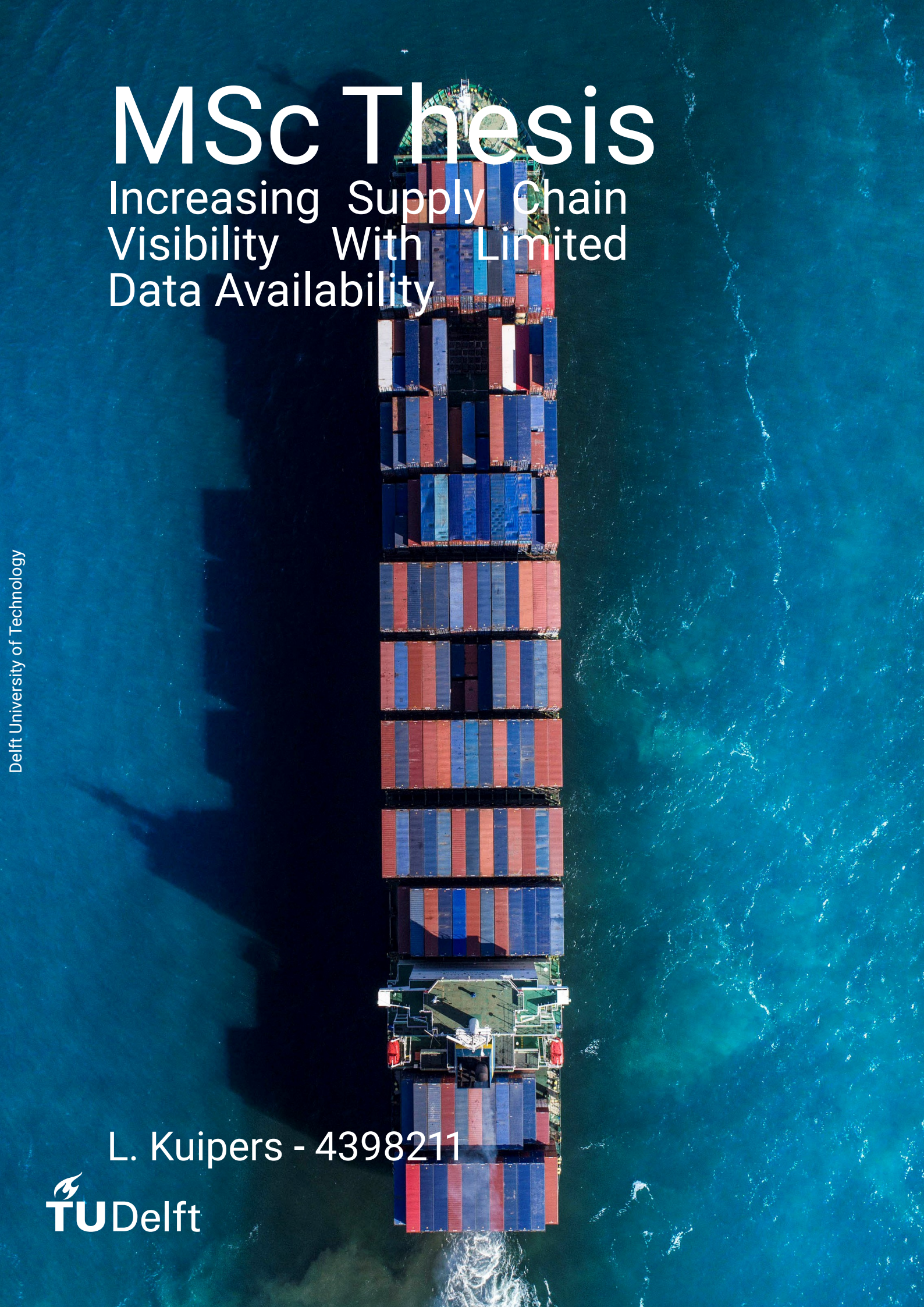
MSc Thesis

Increasing Supply Chain Visibility With Limited Data Availability



L. Kuipers - 4398211

TU Delft

Delft University of Technology

This page is intentionally left blank

Increasing Supply Chain Visibility With Limited Data Availability

Data Assimilation In Discrete Event Simulation

Master thesis submitted to Delft University of Technology

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in **Engineering and Policy Analysis**

Faculty of Technology, Policy and Management

by

Lucas Kuipers

Student Number: 4398211

To be defended in public on August 09 2021

Graduation committee

Chair	Prof.dr.ir. A. (Alexander) Verbraeck	Policy Analysis
First Supervisor	Dr.ir. J.H. (Jan) Kwakkel	Policy Analysis
Second Supervisor	Dr. Y. (Yilin) Huang	Systems Engineering
Daily Supervisor	Ir. I.M. (Isabelle) van Schilt	Policy Analysis
External Supervisor	Dr.ir. F. (Fabio) Zandanel	Metyis

Associated code and models are available at https://github.com/lucaskuipers/Supply-Chain-Data-Assimilation

Executive summary

Being able to timely access and share accurate information to all stakeholders that can influence a supply chain is referred to as *supply chain visibility*. Supply chain visibility is crucial for increasing the resilience of a supply chain as this helps to proactively plan and design interventions to undesirable events. Various methods are available to increase supply chain visibility, often by increasing the accuracy of estimating future events. Many of these methods are reliant on (large volumes of) data, which is problematic because data on supply chains is often not accessible, expensive to acquire, and difficult to process. Therefore, methods to increase supply chain visibility in situations with limited data availability are needed.

Modeling and simulation can be applied to situations with limited data availability, but has a limited predictive power for complex systems. Data assimilation in discrete event simulation (DES DA) is a method to increase the predictive power of modeling and simulation. This method aims to approximate system states based on real-time measurements to increase the accuracy of simulation models.

DES DA has not yet been applied to supply chains and the dependency on data availability still needs to be explored. The goal of this research is to develop a DES DA algorithm that can accurately estimate future events in supply chains and test this algorithm for different levels of data availability. This will give insight in the relation between data availability and the accuracy of the proposed DES DA algorithm.

During the development of this algorithm the typical supply chain challenges of high dimensionality and future state estimation are addressed. The proposed DES DA algorithm shows to have a higher predictive power (accuracy) than the results of a simulation exercise without assimilation. To allow for maximal reproducibility and facilitate future work, all adjustments to the simulation algorithm are extensively described in the body of this thesis.

Analysis of the performance of the proposed algorithm under different levels of data availability shows that limited data availability does not necessarily result in less accurate estimations. Less observations can can result in similar or even more accurate estimations. However, this can only be the case if the missing data is not part of the set of *crucial data sources*.

If the missing data points are part of these *crucial data sources* the accuracy of the data assimilation algorithm detoriates and the algorithm starts to *underfit*. Underfitting occurs when a model uses input variables that are not significant enough to determine a meaningful relationship between the input and output variables.

If limited data availability is the result of omitting data sources that are not part of the *crucial data sources*, the accuracy of the data assimilation algorithm can increase. Using less data points helps combatting the *curse of dimensionality* which results in more accurate estimations.

This research has shown that *crucial data points* for supply chains are the data sources that serve as *arrival sensors* of entities at the system boundaries. The arrival sensors reduce the uncertainty of future state estimations by accurately estimating the number of the entities in the system and the moment of arrival of these entities. Especially the accurate estimation of the moment of arrival reduces the variation in the estimations of the particles and thereby increases the accuracy of the estimation of future events.

For future work, two directions of research are identified. First the *development* of the algorithm can be taken to the next stage by testing this algorithm on data from a physical real system, incorporating structural uncertainty and optimizing the hyperparameters of the algorithm.

The second direction of research should focus on the *implementation* of the algorithm. Research is needed on the establishment of data lakes that combine all different data sources. Next to this the usability of the framework can be improved by finding methods to communicate the shape of posterior probability density functions to supply chain planners and establishing mechanisms that can detect errors in the estimations at an early stage.

Acknowledgements

The past five months have been the most challenging and most satisfactory period of my time at the TU Delft. Five months ago I boarded a train without knowing the destination. Without any prior knowledge on data assimilation Alexander Verbraeck convinced me of doing this research. The route of the train turned out to be *one hell of a ride*. During this period I learned incredibly much about designing my own research, programming, data assimilation, and supply chains. Something that could have never happened without the help of a great group of people around me.

First, I want to express my gratefulness to my graduation committee for their supervision, suggestions, and feedback. I would like to thank Alexander Verbraeck for all the gentle pushes towards the topic of data assimilation, the interesting discussions on the outcomes of the experiments, and for continuously challenging me to bring the research to a higher level. I would like thank Jan Kwakkel for the direct feedback and assistance in shaping the structure of my research and of course great support with all programming questions. I would like to thank Yilin Huang for her positivity and the clear the explanation of challenging data assimilation concepts.

Special thanks go to my daily supervisors Isabelle van Schilt and Fabio Zandanel. Isabelle, it was a pleasure to work under your supervision. Without our biweekly meetings in which we discovered data assimilation in discrete event simulation together, the results would have never been the same. At last I want to thank you for all the time you put in giving me detailed feedback to take this document to the next level.

Fabio, thank you for being the best external supervisor I could ever wish for. Your academic background made it possible to produce an academically relevant research that provides value for Metyis as well. Our daily communication was great, inspiring, and fun. I hope many more graduate interns get the chance to work under your supervision.

Next to this my thanks go out to all other people that surrounded me in the last months. I would like to thank my colleagues at Metyis for the fun (digital) working days. Complaining about the weather in our daily Teams meeting made spending every day in my room a whole lot easier. This also counts for my roommates, hearing me talk about particles every day must have been hard. I would like to thank my parents for their support and for giving me the pressure I like by imposing a strict deadline for my graduation date with your departure to Japan. At last, special thanks go out to Willemijn. I would never have challenged myself this much if you would not have been in my life. Apart from this you were the best getaway possible to escape the house and this research for a few days.

L. Kuipers - 4398211
Delft, July 2021

List of acronyms

AC	At Consolidator
AI	Artificial Intelligence
CTP	Customer Touch Point
DA	Data Assimilation
DC	Distribution Center
DDDAS	Dynamic Data Driven Application Systems
DDDS	Dynamic Data Driven Simulation
DES	Discrete Event Simulation
DES DA	Discrete Event Simulation Data Assimilation
DEVS	Discrete Events System Specification
FIFO	First In First Out
GPS	Global Positioning System
HC	Ho Chi Minh
HP	Hai Phong
IS	Importance Sampling
LHS	Latin Hypercube Sampling
ML	Machine Learning
MS	Modelling and Simulation
PF	Particle Filter
PO	Purchase Order
SMA	Simple Moving Average
SMC	Sequential Monte Carlo
SC	Supply Chain
SCR	Supply Chain Resilience
SCV	Supply Chain Visibility
SZ	Suez
SOP	Sales and Operations Planning
RFID	Radio-frequency identification
RT	Rotterdam
TP	Tanjung Pelepas

Contents

List of Tables

List of Figures

List of Algorithms

1
Introduction

Supply chain (SC) Visibility is an important factor to increase the resilience of supply chains. Many of the techniques to increase SC visibility are heavily reliant on data. But making supply chain data available is challenging. Data assimilation in discrete event simulation (DES DA) have potential to increase SC visibility in situations with low data availability. However, the performance of DES DA in situations with limited data availability is unknown. Therefore, the goal of this research is to develop a DES DA algorithm suitable to increase SC visibility and identify the effect limited data availability has on the performance of the algorithm.

This chapter serves as an introduction to this research. First more insights will be provided on the potential of DES DA to increase SC visibility. Subsequently the knowledge gaps of the current state-of-the-art are identified which shape the goal of this research. Thereafter, the research design and the corresponding research questions will be discussed. At last a reading guide for the remainder of the thesis is provided.

1.1. Need for DES DA to increase supply chain visibility

The increasing globalization and digitization made the number of partnerships between firms and organizations increase rapidly over the past decades. This resulted in a growing complexity of supply chains (Ferdows, 2018). Combined with an increasingly growing focus on efficiency, supply chains have become more and more vulnerable (Silva et al., 2017).

Crises such as COVID-19 showed the reliance of societies on supply chains once again. The limited distribution of protective equipment over hospitals and the current distribution of the various vaccines emphasize how important it is to develop supply chains that are resilient to unexpected events. To this extent visibility on the operations of a supply chain is considered to be crucial, this is explained in the following sections.

1.1.1. How supply chain visibility leads to supply chain resilience

The term resilience comes from the word *resilire*, which is Latin for *to rebound* or *to spring back*. Research to resilience finds its origin in social psychology and materials science. This is extended to various other fields such as ecology and industrial safety, each having a slightly different definition for this term. The common notion in these definitions is the *"ability to successfully accommodate unforeseen environmental perturbations or disturbances"* (Laprie, 2008).

In supply chain management, supply chain resilience (SCR) is referred to as *"the ability to proactively plan and design the Supply Chain network for anticipating unexpected disruptive (negative) events [...] and transcending to a post-event robust state of operations"* (Ponis & Koronis, 2012). Pettit et al. (2019) noted that firms should continuously "balance" their capabilities (results of investments to increase SCR) with vulnerabilities (drivers of SC disruptions) to achieve a state of *balanced resilience*. The general question in this matter is how to manage the portfolio of capabilities as good as possible given the fact that vulnerabilities evolve over time.

To do so, sufficient insight in the vulnerabilities of a supply chain are essential to to allow for the optimization of the deployment of capabilities. Having this insight is defined by Wei & Wang (2010) as *supply chain visibility*. Research shows that having a high degree of visibility enables firms to reconfigure their supply chain resources for greater competitive advantage.

No unique definition for visibility exists, Caridi et al. (2010) identify two main components of visibility based on a literature: 1) The emphasis on information accessibility/exchange and 2) different properties of

information, typically timeliness, usefulness and accuracy. For the remainder of this research visibility is defined as *"The ability to timely access and share relevant and accurate information to all stakeholders that can influence the supply chain"*.

1.1.2. Increasing supply chain visibility with limited data availability

In general one can identify three classes of methods to increase SC visibility: 1) Technological solutions such as Radio-frequency identification (RFID) and blockchain. 2) Artificial Intelligence and Machine Learning Techniques. 3) Modeling and simulation techniques. These techniques are generally dependent on data (see chapter 2 for more insight on this matter). However, making supply chain data available is challenging because of three aspects.

The first aspect that complicates the accessibility of data on supply chains is the *limited availability of data*. As data becomes more and more important in supply chain management, more data on supply chain events (e.g., the loading of containers) becomes available. However, many events remain unrecorded and thereby inaccessible (Martindale et al., 2020).

The second aspect that makes it difficult to make data on supply chains accessible is the *price of supply chain data*. Parts of the supply chain are often handled by second or third parties, if any data is collected these parts of the supply chain, this generally cannot be accessed for free. Examples of data that can be purchased are carrier data points (e.g., TradeLens data which can be purchased from Maersk) or third party data points (e.g., Vessel tracking data which can be purchased from third parties such as VesselTracker)

The third challenging aspect of making supply chain data accessible is the *difficulty of processing data*. Purchased data on supply chains generally comes in raw formats and therefore needs many processing steps to be turned into usable data that can be transformed into information. These processing steps require expertise and knowledge, which is expensive and difficult to acquire.

Summarizing, the limited availability of supply chain data, the price of this data and, the difficulty of processing the data make making supply chain data available a difficult process. Therefore, finding a method that performs well in increasing supply chain visibility in situations with sparse or limited data is essential.

1.1.3. The potential of data assimilation in discrete event simulation

Modeling and simulation techniques are the only method that can be applied in situations with low data availability because domain knowledge can be used to adjust the internal logic of a model instead of data. However, because simulation models are simplifications of reality (Box, 1976), the the ability to *predict* the real behavior of complex systems is limited Darema (2004). This makes data assimilation an interesting solution, because this technique aims to approximate system states based on real-time measurements and thereby increases the accuracy and predictive power of simulation models.

Supply chains are characterized by non-Gaussian and complex behavior because state transitions in supply chains are typically defined by decision rules triggered by events (Xie, 2019). This makes it very difficult to capture this behavior by continuous models. Threfore, discrete event simulation (DES) is the preferred modeling technique to study these types of systems.

Classical data assimilation techniques (see section 2.3.1) are unsuitable for DES models because of their assumption of Gaussian errors. To this extent, sequential Monte Carlo (also known as particle filters) have been developed as an alternative for classical data assimilation techniques. Over the past years sequential Monte Carlo methods have been applied to discrete event simulation. In literature, the application of particle filters to assimilate data into discrete event simulation models is commonly referred to as data assimilation in discrete event simulation (DES DA).

Recent research confirms the potential of DES DA to increase the accuracy and predictive power of DES models in various fields. However, DES DA has not yet been applied to supply chains. Therefor this research will explore if and how DES DA can be applied to increase SC visibility. More details on the suitability, the application and limitations of this technique are given in chapter 2.

1.2. Research goal

As DES DA is a new technique, still many unknowns on this topic exist. Especially related to the application of DES DA to supply chains or more complex systems in general. Section 2.4 gives an overview of the different challenges that originate from the key characteristics of DES. Some of these challenges have already been resolved, others are considered knowledge gaps for future work. In this section, the knowledge gaps that form the foundation of this research are discussed. Based on these knowledge gaps, a research goal and question is composed.

1.2.1. Knowledge gaps and research goal

The first knowledge gap in existing literature on DES DA is the fact that little is known how DES DA an be applied to supply chains or other systems with a high degree of complexity and dimensionality (such as supply chains). Literature shows that applying DES DA to systems with a high degree of dimensionality is challenging since particle filters tend collapse in systems with a high-dimensional state space (see section 2.4.4 for more information).

Two root-causes to this collapse are identified: 1) As the number of dimensions increase, any two points become more similar making distance metrics become less effective (Aggarwal et al., 2001). 2) As the numbers of dimensions grow, data becomes sparse. This requires the numbers of particles to grow sub-exponential with the number of dimensions in the system to make it still approximate a posterior distribution Bengtsson et al. (2008).

The second knowledge gap relates to the effectiveness of DES DA in situations with limited data availability. Cho et al. (2020) and Hu & Wu (2019) show that state estimations by DES DA generally become more accurate if more data is available. However, no research has yet been done on the relation between data availability and the accuracy of DES DA if different types of data are available.

The goal of this research is to fill in these two knowledge gaps. First, this research will discover how the current state-of-the-art DES DA algorithms should be adjusted to allow for the increase of supply chain visibility. One should think of adjustments based on the type of data that is available for supply chains, adjustments based on the output and output of the algorithm, and measures to deal with complexity and a high number of dimensions.

The second goal of this research is to study the effects limited data availability has on the accuracy of DES DA as a method to increase supply chain visibility by providing real-time decision support. Studying these effects will help to identify crucial data points for the application of DES DA to increase supply chain visibility.

1.3. Research design

To reach the previously discussed research goals, a research method is presented. In this section first the general research method (proof of concept) is discussed, after which the use case selection will be further elaborated on. Based on this use case, combined with the aforementioned research goals the scope of the research will be discussed based on which the research questions is determined.

As the goal of this research is to discover if and under what conditions DES DA can be used as a tool to increase supply chain visibility, this research is exploratory in its nature. To bound this exploration, it is decided to perform this research based on a use case which is used as a proof of concept.

Together with an external party, a use case is developed to proof that DES DA method can increase supply chain visibility by satisfying a real decision-support request. This will benefit the generalizability of the developed algorithm and the results since the designed DES DA algorithm is tested on a real supply chain structure. A real use case will also lead to realistic assumptions on both the data availability.

Deploying a use case as proof of concept means that not every component of this research can be generalized. On one hand, a *generalizable* algorithm for data assimilation with supply chains will be developed to contribute to the academic relevance of this thesis. On the other hand, this will be tested on a *case-specific* simulation model of the supply chain.

1.3.1. Defining supply chain visibility as accuracy

For this research, it is decided to work together with a fashion retailer with an international supply chain, further referred to as *stakeholder*. Together with the supply chain planning team a use case is selected to serve as proof of concept:

The stakeholder has put out a request to increase the accuracy of the estimated dates of shipments arriving at the distribution center. An increase in accuracy will help optimizing the inventory planning of the warehouse. This specific request means that the broad definition of supply chain visibility is narrowed down to the increase of accuracy the arrival estimations of shipments. More information on the use case itself is presented in chapter 3.

As supply chain visibility is narrowed down to the accuracy of the arrival estimations of shipment, accuracy will be used in the remainder of this document to assess the degree to which supply chain visibility is increased by DES DA. To determine the accuracy, four different aspects of accuracy are assessed. The aspects are briefly discussed below. More insights on the operationalization and implementation of these aspects is given in chapter 5.

First, two metrics are identified to capture the accuracy of an arrival estimation for an individual shipment. An accurate prediction has a low *absolute distance* between the estimated and the observed arrival day. Next to this a high share of the particles should have estimated the observed arrival date correct which is indicated by a high *percentage*.

Second, the the accuracy of a set of shipment arrival estimations needs to be calculated as well. A high accuracy of a *set* of arrival estimations is indicated by two aspects: An accurate set of estimations indicated by an accurate *weighted average* over the set of shipments and a high level of robustness indicated by a low *standard deviation* for the estimations in this set.

1.3.2. Research questions and methods

Based on the twofold research goal and the selected use case, the following research question is composed:

Research Question: *What is the effect of limited data availability on the accuracy of data assimilation in discrete event simulation on supply chains?*

To answer this question, a set of sub-questions has been composed that collectively help answering the main research question. Each of the sub-research questions will be presented in this section.

The DES DA algorithm that will be constructed does not need to be built from the ground, this can be an adjustment to the current state-of-the-art to make it suitable for supply chain simulation models. To give clarity in the adjustments that are needed, the first sub-question is designed. The goal of this question is to give insight on the different adjustments of the current state-of-the-art DES DA algorithms needed to make the algorithm applicable to supply chains:

1. *How should the current state-of-the-art data assimilation in discrete event simulation algorithms be adjusted for application to supply chains?*

The increase in accuracy of the shipment arrival estimations is as a proxy for the degree to which supply chain visibility is increased. Therefore, the metrics to capture this increase in accuracy should be well defined. The accuracy metrics should be able to capture the average accuracy and the robustness of the accuracy over a set of shipments. The formalization of these metrics will form the answer to sub-question 2:

2. *What accuracy metrics should be used to determine the accuracy of data assimilation in discrete event simulation algorithms for the estimation of future events?*

Based on the identified accuracy metrics, the accuracy of the algorithm can be tested for different levels of data availability. To do so, a set of experiments with different levels of data availability will be designed. The accuracy metrics will be computed for each experiment and form the basis for a comparison of the

different experiment. Based on this comparison, the effect the number of observations that are assimilated into the simulation model has on the accuracy of DES DA forms the answer to sub-question 3:

3. *What is the relation between the number of observations and the accuracy of data assimilation in discrete event simulations on supply chains?*

Literature on machine learning and feature selection shows that if different types of data sources are used as input, some are show to be more important than others Koutroumbas & Theodoridis (2009). Therefore, it is be expected that the relation between the number of observations and the quality of data assimilation is not straightforward. If so, this is an indication that there are data points that are crucial for the accuracy of data assimilation and data points that can be identified as noise.

The results from the analysis resulting from sub-question 3 will be further analyzed to make the distinction between *crucial* data points and *noise*. This distinction should form the answer to sub-question 4:

4. *What sources of data are crucial for the the accuracy of data assimilation with discrete event simulation on supply chains?*

1.4. Structure of this thesis

The thesis consists of eight different chapters each intended to provide argumentation and clarification on the conclusions that are drawn. The first three chapters serve as an introduction to this research with a literature review and a description of the research. The argumentation for the answers to each of the sub-questions can be found in chapter 4 to 7. At last the sub-questions and the main research question are answered in chapter 8. The relation between the sub-questions and the different chapters is visualized in figure 1.1.

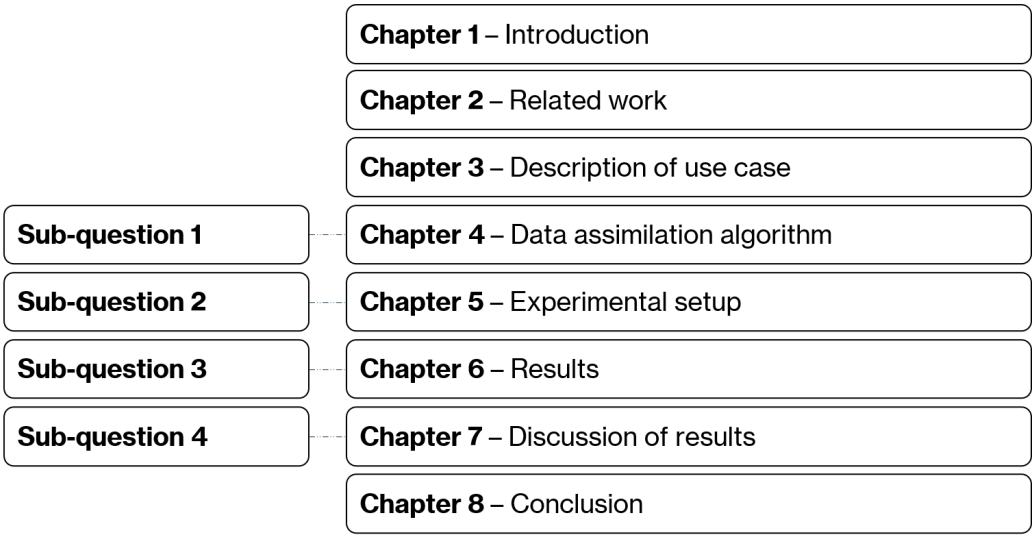


Figure 1.1: Relation between sub-questions and chapters

Chapter 2 is a literature review to give insight in related work. In this chapter, supply chain visibility, discrete event simulation and data assimilation methods are discussed, resulting in an overview of the current state-of-the art and potential challenges for the application of DES DA on supply chains.

As this research is based on a use-case, this use-case will be introduced in chapter 3. In this chapter, the current supply chain operations and planning practices, the decision support request and the data availability for this use case are discussed.

Based on the decision-support request of the use-case and the challenges for applying DES DA to supply chains that are identified a data assimilation algorithm is developed. This algorithm is presented in chapter 4 by discussing each of the functions that form this algorithm to guarantee the reproducibility of this algorithm for future research. This will form the basis for answering sub-question 1.

In chapter 5 the different experiments will be introduced as well as the formalization of the accuracy metrics that are used to assess the accuracy of the data assimilation algorithm. This formalization will form the basis for the answer to sub-question 2.

Subsequently the results of the different experiments will be presented in chapter 6, based on which the relation between limited data availability and the accuracy of the data assimilation algorithm can be identified. This is the answer to sub-question 3.

The results of the experiments will be discussed in chapter 7. This discussion will focus on a comparison with existing literature, a reflection on the generalizability of the results, and discussing the limitations and implications of this research. By discussing the results, the crucial data points are identified. This forms the answer to sub-question 4.

At last, the answers to all sub-questions are presented in chapter 8. These answers together form the answer to the main research question presented in the same chapter.

2
Related work

Chapter 1 introduced the concept of supply chain visibility and how this can be achieved. It shows that DES DA is a technique with a lot of potential to increase SC visibility. Especially when considering real-time decision support. However, DES DA is still an underdeveloped field particularly regarding supply chains. The goal of this chapter is to to provide a more solid academic foundation for these claims. Next to this, a literature review on DES DA is presented to give the reader a better understanding on the current state-of-the art as well as the remaining challenges that form the basis for this research.

First the choice for data assimilation in discrete event simulation (DES DA) as a method to increase SC visibility is discussed. Subsequently more background information on discrete event simulation (DES) will be given, followed by a review of the different data assimilation techniques. Thereafter, the DES DA algorithm will be explained together with the framework of in which this algorithm is generally placed. The last section of this chapter discuss various challenges / knowledge gaps for DES DA which form the basis for the research questions.

2.1. Increasing supply chain visibility

SC visibility has a positive effect on supply chain resilience by helping balancing both the supply chain *capabilities* and *vulnerabilities*. Several methods to increase SC visibility are discussed below. Based on a comparison of the different techniques in situations with low data availability, argumentation is provided on why DES DA is the technique that forms the basis for this research.

2.1.1. Methods to increase supply chain visbility

SC visibility can be improved by various methods. In the the next section the different methods are grouped into three distinguishable classes of methods. Each of these classes will be briefly discussed.

Technological solutions form the first class of methods. These methods deal with the development of technical means to increase visibility. Both Radio Frequency Identification (RFID) as well as blockchain are the two most frequently mentioned developments within this field. Both methods aim to increase visibility by increasing the volume and accuracy of data on the identity, location and status of entities in the supply chain (Angeles, 2005, Caridi et al., 2013, Cole et al., 2019).

The second class of methods are *Artificial intelligence* (AI) and *machine learning* (ML) techniques. AI and ML are rapidly increasing fields of research, becoming more and more dominant within almost every field of application. In SC management, a similar pattern is visible. Baryannis et al. (2019) identify three general fields of application for AI and ML in SC management: the automated *identification* of, *assessment* of and *response* to SC risks.

The third class of methods are *modeling and Simulation (M&S)* techniques that are often used to construct digital counterparts (twins) of the supply chains (Ivanov et al., 2019) or to test different SC configurations on an operational, design or strategic level (Huan et al., 2004) both aiming to increase visibility.

The common denominator for the aforementioned fields is the reliance on data. *Technological solutions* aim to increase the availability and accuracy of data. AI and ML are known to be heavily reliant on large training data sets (Huck, 2019). M&S does not necessarily have to depend on large datasets but does generally require a good understanding of the system under study (Huang, 2013). Over the past years the M&S community made large steps in improving the application of M&S under uncertainty (Deelen et al., 2020), which is caused by sparse data availability (Sankararaman & Mahadevan, 2013). Examples of this are

exploratory modeling and analysis (EMA) Kwakkel (2017) or the deployment of sensitivity analysis (Baroni & Tarantola, 2014).

2.1.2. Suitable methods with low data availability

As not all data is available by default or can be made available when working on supply chains it is decided to limit the scope of the research to M&S techniques. However, traditional M&S techniques show one clear limitation when applied for providing real-time decision support. Traditional simulations are generally conducted in a *what-if setting* with data inputs that are fixed at the time the application commences execution. The result of this is that even elaborate complex models of systems fail to predict the real behavior of those systems (Darema, 2005).

This limitation was acknowledged in an early stage of M&S development. Because simulations require data and business rules to be manually embedded within them by (simulation) experts, continuously reconfiguring a simulation to accommodate real-world changes is often too (labour) expensive (Goodall et al., 2019) and *what-if* simulations can lack accuracy (see also section 2.3.

A response to this limitation was the emergence of *dynamic data driven simulation*, which is an approach where the simulation is parameterised by providing data through a set of data inputs (J. Wang et al., 2011). Which relates to *symbiotic simulation*: a simulation paradigm where a physical system and a simulation system are closely related to each other. In this paradigm the simulation system benefits from real-time measurements about the physical system, provided by corresponding sensors (Aydt et al., 2009). However,When the entire state of the physical system can not directly be observed, these real-time simulation techniques generally lack accuracy.

If unobservable (hidden) states should be estimated, *data assimilation* is a suitable technique. This is an emerging field of research within the field of M&S and refers to *"the process of incorporating observation data into a running model to produce improved estimates of state variables of interest"* (Hu & Wu, 2019). This fits closely to the goal of the research, therefore the research is further scoped to data assimilation for modeling and simulation.

2.1.3. Data assimilation for supply chains

Data assimilation (DA) finds its origin in the field of numerical weather prediction and is further developed within the field of earth sciences. A large share of the current state-of-the-art methods are inspired by and adapted from atmospheric or oceanic data assimilation systems. These conventional DA techniques such as the Kalman filter rely on linear theory and assume Gaussian data distributions (Reichle, 2008). These techniques are referred to as *parametric* techniques because of their reliance on analytic properties (i.e., means and covariance matrices) that can be derived from the numerical models.

Supply chains are nonlinear and complex in its nature (S. Wang, 2010) which limits the usefulness of conventional DA techniques. Besides this it is considered undesirable to represent these systems by means of continuous models. Discrete-event simulation (DES) is better suited for analyzing supply chains because it is able to embody the nonlinear and complex characteristics of supply chains (Blanco et al., 2011). A downside of using DES models is the fact that DES models do not have a numerical model from which analytic properties can be derived. This means that conventional, parametric, DA techniques do not work for DES models.

To this extent, a *non-parametric* data assimilation method is developed based on *Sequential Monte Carlo* (SMC) techniques (Hu & Wu, 2019). SMC techniques, also referred to as Particle Filters (PF) are sample-based methods using Bayesian inference, stochastic sampling and importance resampling to estimate (hidden) system states in an iterative manner (Cho et al., 2020). Instead of using the analytic properties to estimate the probability density functions, these distributions are approximated by a large set of random samples, also known as particles. This is a very new technique, developed over the past years and previous applications have proved its potential in application areas such as wildfire spread simulations (Gu & Hu, 2008, Long & Hu, 2017, Xue et al., 2012) and transport simulations (Hu & Wu, 2019, Xie, 2019). To give

the reader a better understanding of this, first the concept of discrete event simulation will be discussed in the next section.

2.2. Discrete event simulation background

To study and predict the behavior of complex systems, modeling and simulation (M&S) is a desirable method since these types of models are iteratively built until it can represent a real system with sufficient accuracy. Verified and validated simulation models are the rusult of an iterative process of verification and validation until the differences between the real and simulated system are within acceptable bounds. Banks (1998) These models can help to understand system behavior, and are very useful to test different system configurations on operational, design or strategic levels (Huan et al., 2004).

However, even the most elaborate, complex models are not able to model reality perfectly; as supported by the famous saying of George Box (1976) goes: *"All models are wrong"*. This imperfection results in uncertainty which limits the predictive power of most simulation models (Darema, 2004).

However, as Box (1976) claims that all models are wrong, he also claims that *"some models are useful"*. The increase of computational powers in the last decade made the development of *useful* simulation models more important. Simulation models are especially useful to understand systems and evaluate eventual strategies to improve the operations of the systems.

2.2.1. Discrete event system specification

For modeling and simulation, three different *basic system specification formalisms* exist, each suitable to model different types of system systems. The suitability of each formalism to represent a certain system depends on the representation of both the state variables and the time base (continuous or discrete) (Xie, 2019) the different classifications of systems are displayed in figure 2.1.

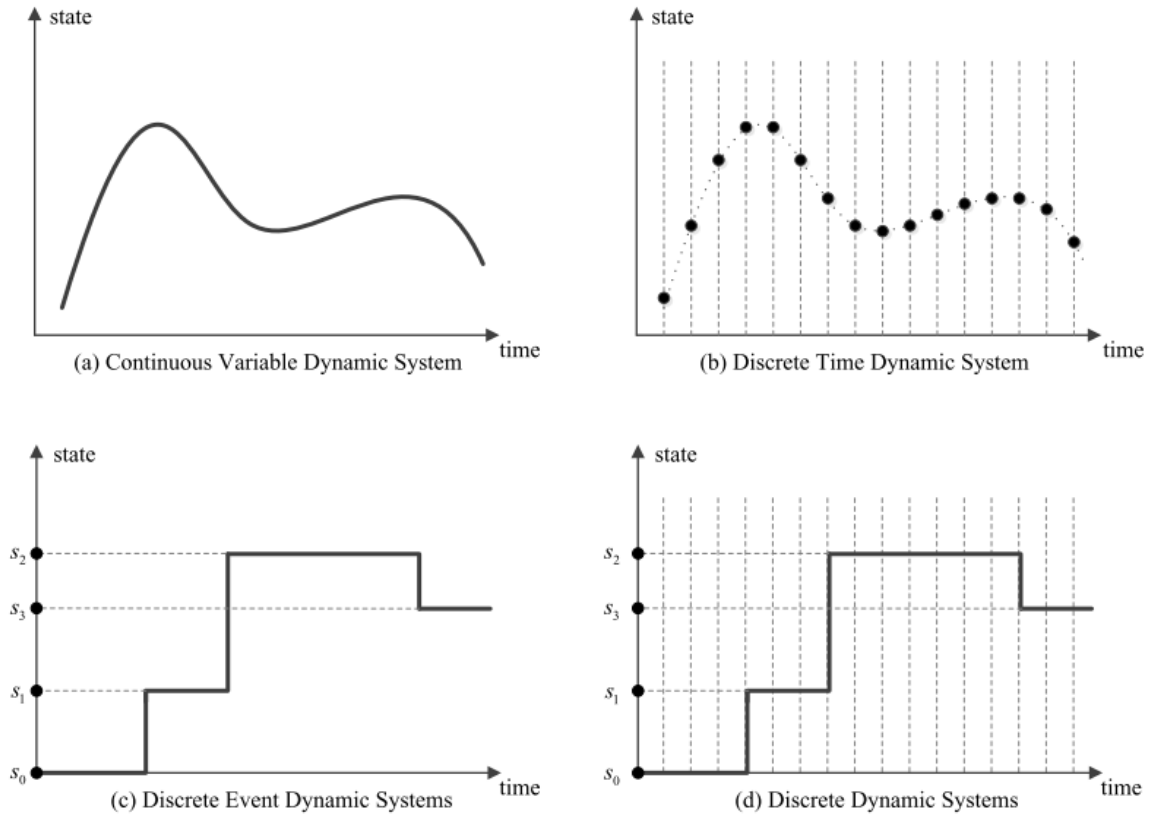


Figure 2.1: System classification according to the representation of time base/state variables, derived from Xie (2019)

The *discrete event system specification (DEVS)* formalism fits to the system classification as shown in figure 2.1 c. In this formalism, a system is represented by piecewise constant state trajectories over a con-

tinuous timebase (Xie, 2019). This means that state changes are only captured at discrete random points in time (instances); such a moment is called referred to as an event (Nance, 1981). The fact that everything that happens between two consecutive instances is omitted can make discrete event simulation very efficient.

For discrete event simulation, various formalisms exist, of which DEVS is the most used one. In DEVS, the system behavior is described on two different levels; an *atomic level* and a *coupled level*. An atomic DEVS model describes the autonomous behavior of a discrete event system as well as how it reacts to input (events) and generate output (events). Atomic models are coupled to form a larger model. More elaborate descriptions of this formalism can be found in Zeigler et al. (2000) and Xie (2019). The most important factor to note from this formalism is that atomic model states remain unchanged even despite events happening on the *coupled* model level (i.e., in other atomic models that are part of the coupled model).

2.2.2. Systems for discrete event simulations

Discrete event simulation is generally used to study the behavior of man-made dynamic systems (Xie, 2019). Examples of systems where DES is effectively applied are healthcare (Günal & Pidd, 2010), factories (Kádár et al., 2004) and supply chains (Ivanov et al., 2019, Tako & Robinson, 2012). The state transitions of these systems are generally defined by decision rules, for which the decisions are triggered by events. This makes the behavior of discrete event simulations highly nonlinear and non Gaussian. This is very difficult to be defined by differential equations and therefore a discrete event approach is needed for these systems. This makes DES the preferred M&S tool to study and predict the behavior of supply chains.

The last important notion on systems that are studied by discrete event simulation is the fact that many of these systems are open system. This means that model components (entities) can flow in and out through the system boundaries which makes the number of components in the system dynamic instead of fixed (Xie & Verbraeck, 2019). One could think of this as the flow of goods through a supply chain; goods arrive at the system boundaries (e.g., after production) and leave the system as well (e.g., after delivery).

Summarizing, discrete event simulations are the most suitable technique to study supply chains. This M&S approach is defined by a set of distinctive characteristics which impact the way data assimilation is performed. The most important characteristics are reiterated:

- The system state is represented by a collection of continuous and discrete state variables
- The system state is changed at events which can take place at random moments (instances) because of the continuous time base
- Between two events in an atomic model component, the atomic model state remains piecewise constant
- System state changes happen based on rules which makes the behavior non-linear and non-gaussian
- Discrete event systems are often open systems with a dynamic number of components

2.3. Data assimilation background

Over the past decades, not only computing power made rapid advancements. Also the possibilities to measure system states in a real-time manner have developed rapidly. The combination of progress in sensors, data management systems and (online) data sharing systems made it possible to (remotely) access larger quantities of more precise data at a faster speed. These developments sparked the development of a new simulation paradigm: *Dynamic Data Driven Application Systems* (DDDAS) which aims to both incorporate additional data into an executing application (simulation model) and also dynamically steer the measurement process (Darema, 2004). This is supposed to mitigate the negative effects of errors in the model by continuously "recalibrating" the model.

Dynamic data driven simulation (DDDS) is related to this paradigm and tries incorporates real-time measurement data into simulation models to improve the accuracy of these models during runtime (Cho et al., 2020). Data assimilation (DA) is used in this approach to approximate system states based on the real-time measurements. To give the reader a good understanding of data assimilation, this section aims to give more insight into the various data assimilation techniques that exist and provides further argumentation on why particle filtering is the technique that fits to discrete event simulation.

2.3.1. Data assimilation techniques

As data assimilation aims to increase the accuracy of dynamical models by incorporating real-time measurements of the real system into the dynamical system model. For this process to work, three components form the foundation (Xie, 2019):

- The *System model* which describes the evolution of a system over time. This is a function that defines the state transition based from time *k-1* to *k*, which takes the state vector *s* of time *k-1* as an input. To this process, system noise (v_{k-1}) is added. This is often defined in a discrete time state space form:

$$s_k = f_k(s_{k-1}) + v_{k-1}, k = 1,2,\ldots \tag{2.1}$$

- The *measurement model* which maps the state (s_k) to the the measurements with a (nonlinear) function g_k and adds measurement noise ϵ_k. This is defined as a discrete time equation as well:

$$m_k = g_k(s_k) + \varepsilon_k, k = 1,2,\ldots, \tag{2.2}$$

- The *data assimilation technique* that provides the state estimation based on the system model, measurement model and, both types of errors.

For different types of system models (and measurement models), different types of data assimilation techniques are more suitable. Generally speaking, two classes of data assimilation techniques are identified.

The first class is referred to as *variational methods*, which aim to minimize a cost function that measures the misfit between the system state, a background state (short-range forecast) and an observation (Xie, 2019).

The second class is commonly referred to as *sequential methods*, which aim to correct estimated states every time an observation becomes available. This class is divided into two groups: parametric and non-paramatric techniques, which both will briefly be discussed in the next sections.

Parametric sequential data assimilation techniques (Kalman Filters)

Parametric sequential data assimilation techniques try to minimize the *analysis error covariance matrix* and the *forecast error covariance matrix*. Since these matrices can be accurately calculated, an optimal filter is found which is the referred to as the Kalman filter, named after its primary developer Rudolf Kalman. Since this filter uses uses a series of measurements observed over timed (sequential), it produces estimates of unknown variables that tend to be more accurate than those based on a single measurement alone. Welch & Bishop (1995) describe this filter as an *"efficient computational (recursive) solution of the least squeares solution."*

However,the Kalman filter is based on two foundations that make it unsuitable for discrete event simulation. At first Kalman filters method is based on an assumption of linearity which definitely does not hold for discrete event simulation. This assumption is relaxed by extended and ensemble Kalman filters. However, the second assumption of Kalman filters is that errors in the system model and the measurement model need to be Gaussian, which is not true for discrete event simulation Xie (2019).

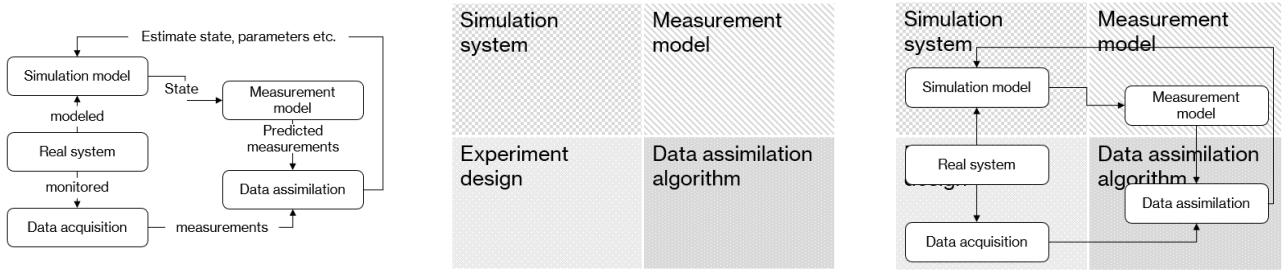
Non-parametric data assimilation techniques (Particle Filters)

The second group of data assimilation techniques do not rely on analytic properties of the model (i.e., mean and covariance matrix) to estimate a conditional distribution of states based on the system model and the measurement model. Instead these methods use a (large) set of weighted samples (particles) to approximate this (posterior) distribution. This is a filtering process based on Bayesian inference, stochastic sampling and importance resampling Cho et al. (2020) and often referred to as *particle filtering* or Sequential Monte Carlo (SMC) methods.

The fact that particle filters (PF's) do not make any assumptions on both the linearity of the model as well as the distribution of the errors makes them suitable for data assimilation in discrete event simulation. This field of application is relatively new and has faced rapid development over the past years. Particle filtering has already shown its potential in application areas such as wildfire spread simulations (Gu & Hu, 2008, Long & Hu, 2017, Xue et al., 2012), transport simulations (Hu & Wu, 2019, Xie & Verbraeck, 2019) and process planning (Cho, 2019, Xie, 2019). For this method to be applicable to discrete event simulation, *dynamic data driven simulation* (DDDS) framework is developed, which will be explained in the next section.

2.3.2. Data assimilation in discrete event simulation framework

A generic DDDAS framework consists of five different components: A real system (1) that is modeled by a simulation model (2) and is monitored by a data acquisition component (3). The simulation model provides a state to the measurement model (4) which subsequently translates this state into estimated measurements to the data assimilation technique (5). Here the measurements and the estimated measurements are compared and subsequently fed back into the simulation model. This framework has graphically been displayed in figure 2.2a. In true DDDAS applications the data assimilation technique is also supposed to dynamically steer the measurement process. Since this is not within the scope of research this is left out of the explanation and the figure.



(a) DDDAS framework based on Xie (2019)

(b) DES DA based on Hu & Wu (2019)

(c) Relation DDDAS and DES DA framework

Figure 2.2: DES DA framework in a DDDAS context

Based on this line of thought, Hu & Wu (2019) proposed a framework for data assimilation in discrete event simulation. This framework is a composition of four interacting parts: A simulation system , a measurement system, the experiment design and the data assimilation algorithm. The first three components will be briefly discussed in this section, after which the data assimilation algorithm (particle filtering) will be described into more detail in the next section. A visualization of this framework and the relation to DDDAS is visualized in figures 2.2b and 2.2c.

Simulation system

The simulation system encompasses the simulation model as shown in figure 2.2 and defines how the system state over time. In principle, a simulating system could just be any DES model, however, several adjustments need to be made to allow for data assimilation according to Hu & Wu (2019). The adjustments are mainly related to the way discrete event simulation systems deal with states.

To allow for data assimilation, it should be specifically stated which states comprise the system state and these states should be stored in the system state vector. Apart from this, a challenge for DES DA comes from the fact that states remain constant between two events. This means that at the time of data assimilation the set of system states does not accurately represent the actual system state of the real system. The problem that arises from this discrepancy is referred to as the *state retrieval problem* (Xie & Verbraeck, 2019). The solution for the state retrieval problem is an interpolation method which is described into more detail in section 2.4.1.

Experiment design

The experiment design defines how the data assimilation experiments are set up and how the effectiveness of the DES DA exercise is evaluated. This encompasses a description the different configurations of the algorithm and the performance metrics, but more importantly a decision on the configuration of the real system and the data acquisition components. According to Hu & Wu (2019), it is desirable to perform experiments on physical systems. However, this is not always practical since physical system can take long periods of time to transform their system state. To mitigate these problems, *identical twin* experiment is presented as a best practice.

In an identical twin experiment, a simulation model is used as substitute of a physical system to take the role of the real system in the framework. This simulation run will first be run and its data will be recorded. This simulation is considered the *"ground truth"*. This process is graphically presented in figure 2.3. This

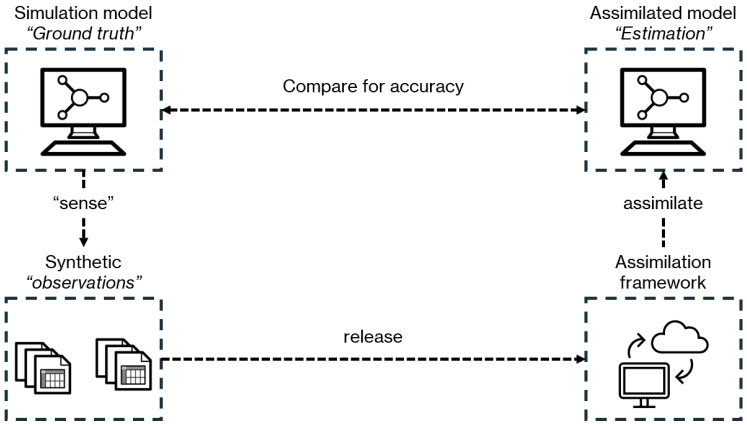


Figure 2.3: Identical twin concept

allows to easily compare the system states of the assimilated system to the "synthetic observations". This has two advantages: 1) The "true system" is and has exactly the same structure as the assimilated system, which means that any deviations between these systems can not by cause by structural differences. 2) This allows for the testing of the impact of platforms not currently deployed (Hu & Wu, 2019).

Measurement model and data assimilation algorithm

The last two component that together form the data assimilation framework are the measurement model and the data assimilation algorithm. These two components form the core of this framework and will therefore be explained into more detail in the next section.

2.3.3. Data assimilation algorithm

As previously described, the goal of data assimilation is to calculate a posterior probability distribution. This is done by *recursively* estimating the state of *dynamic system*, which consists of two models. The first model is the *system transition model* which expresses the transition of system states from time *t-1* to time *t*. The second model is the *measurement model* which defines the mapping from a state variable to a measurement variable. These measurements are subsequently compared with the real observations to give weights to each of the model weights. This is done for a large number of observations (particles). The weighted distribution of the system states of the particles and its weights give the posterior belief of the system state (see equation 2.3).

$$S_t = \left\{ s_t^{(i)}, w_t^{(i)} \mid i = 1, \ldots, N \right\} \tag{2.3}$$

This process is repeated every time new data is assimilated into the system, the time span between these moments is referred to as *t*. In the remainder of this section, each of the steps will be elaborated on in more detail, accompanied with schematic representations of each of the processes. This explanation is based on the *bootstrap filter algorithm* and the framework as provided in Hu & Wu (2019). Different DA algorithms exist, but this one is chosen as an example because of its simple approach of calculating weights and suitability with DES models. The process is graphically displayed in figure 2.4.

At the initialization, N (generally N is a large number) particles are generated. Each particle represents a specific estimate for the system state which is the compound of all system state variables (and therefore a state vector). Each particle is generated by randomly sampling from the ranges of plausible variables for each of the system state variables. In figure 2.4, only the observable system states are shown. The hidden system states are not displayed since these do not affect the data assimilation process itself. In this example two observable system states are displayed. The set of sampled particles s_t is referred to as the *prior believe* of the system state.

The first step of the data assimilation process is the *sampling step*, which runs from the last moment data is assimilated (*t-1*) to the moment new data is assimilated into the system (*t*). Here the *system transition model* performs the state transitions for each of the particles from time *t-1* to time *t* as is expected

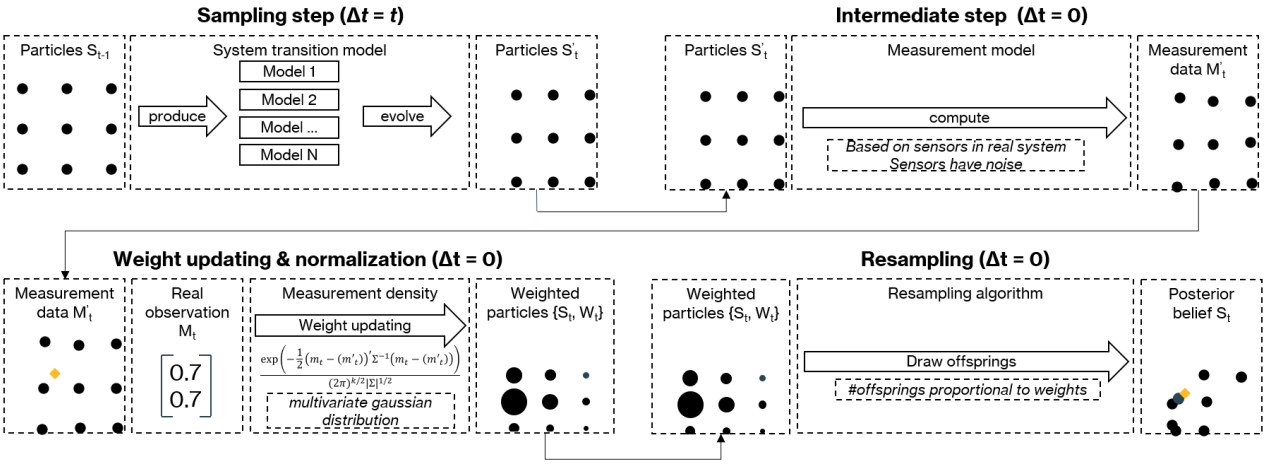


Figure 2.4: One iteration of data assimilation in DES with bootstrap filter algorithm, based on Hu & Wu (2019)

to happen in the real system. In the case of a DES application, the *system transition model* refers to the simulation model. Each particle represents one instance of this simulation model with its own set of parameters. At time t, the simulation is stopped and all (relevant) system states are recorded this is state vector s'_t. Subsequently the state variables that is measured by sensors (the state variables that are observable) are separated and slightly modified in the *intermediate step* based on the *measurement model*. Here the measurable system states are adjusted with an estimation of the expected sensor noise. Generally this is done by adding Gaussian noise to each of the system state variables. For each of the particles the selected set of "measurements" is stored in the vector m'_t

The real observation data (m_t) that comes in, is subsequently compared to the measurement data for each of the particles (each entry in m'_t). Based on he difference between the real observation data and the measurement data, *importance weights* are calculated. Hu & Wu (2019) propose to define a measurement density as a multivariate Gaussian distribution per particle and the real observation data to calculate the weights. The smaller the difference between the measurement data of a particle and the real observation data, the higher the weight that is assigned to this particle. After all weights have computed, the weights are normalized. This is graphically represented by the size of the particles in figure 2.4. As presented in this figure, the closer a particle is to the observation data, the higher the weighth that is assigned to this particle.

At the last step, importance sampling is used to generate N offspring samples. The number of offspring particles each previous particle generates is proportional to its relative *importance weight*. A particle with a high weight can generate a lot of offspring particles. This can lead to a lot of model instances that are exactly the same. To combat this problem, different random seeds should be used for stochastic models and for deterministic models, the system states are perturbed. This is referred to as *particle rejuvenation*. This set of new samples s_t represents the *posterior belief* of the system state and used as the input for a new iteration.

2.4. Challenges for DES DA

The key characteristics of discrete event simulation can make data assimilation in discrete event simulation a challenging subject. Because DES DA this is still an emerging field of research, some of the challenges have already been addressed and some of these are still part of future research. This section gives an overview of the most important challenges and already implemented or suggested directions for research.

2.4.1. State retrieval problem

Xie (2019) describes the a challenge for DES DA called the *state retrieval problem*. The system state of a DES model is the compound of different atomic model component states. These atomic model component states are only updated locally if a new event takes place. This means that at the time of data assimilation the set of system states does not necessarily represent the actual system state accurate and deviations with

the physical system can exist. A solution to this problem is presented by Xie (2019) by introducing an interpolation operation which takes the elapsed time for a state variable into account.

The challenge of implementation of this interpolation operation and the data assimilation itself has already (partly) been resolved by the proposed algorithm of Hu & Wu (2019). This is also implemented in the algorithm developed Xie (2019). Both algorithms are tested on case studies and show that this data assimilation improves the accuracy of modeling and simulation and is robust to errors in the estimation of errors in the *measurement model* (Xie & Verbraeck, 2019). More widespread particles can also make the system more responsive in detecting sudden system changes (Cho et al., 2020).

2.4.2. Variable dimension problem

The state of a discrete event simulation system generally has a variable dimension, which is not a problem by itself but makes DES DA challenging. This variable dimension originates from two of the key characteristics of DES, which will be discussed in the next section. Subsequently the eventual adverse effects on DES DA and the proposed solutions will be discussed. This phenomenon is extensively researched by Xie (2019) and therefore will be only briefly discussed here.

The variable dimension is caused by two characteristics of DES. The first is the fact that events can take place at random moments because of the continuous timebase. This makes the number of events that take place within a *fixed time interval* a random number. If a variable of interest is a sequence of events in a fixed time interval, this means the variable can have different lengths and thereby a variable dimension (for more explanation, see the goldmine case study in Xie (2019)).

The other problematic characteristic is the fact that DES systems are often open systems which make the number of entities in a system variable, especially because the arrival of entities is modelled by (random) events. If the system state is the compound of a set of entity state, the variable number of entities makes the dimension of the system state variable as well (for further elaboration on this topic, see the vehicle trajectory reconstruction case study in Xie & Verbraeck (2019)).

Variable dimensions are problem for data assimilation since these two random processes can cause discrepancies between dimension of the state of the real system and the dimensions of the particles. Xie (2019) proposed a solution to this problem by fixing the dimensions of systems under study to a sufficiently large number with empty state points of entities and eventually discarding them. For open systems one more adjustment is required, sensors at the system boundaries that allow for a reconstruction of the number of entities in the system.

2.4.3. Data dependency

The dependency of DES DA to data availability has previously be researched: Hu & Wu (2019) and Cho et al. (2020) already gave insight in the effects of the number of sensors, number of particles (N) and the time interval on the estimation accuracy of the data assimilation methods. In general the statement that more data results in better accuracy holds (although this relation is not linear).

However, as DES DA is yet to be applied to supply chains chains and specifically shipment arrival estimation, less is known about the effects limited data availability has on the accuracy and effectiveness of this method in this situation. Given the fact that data on supply chains is not always available, can be expensive to acquire and is difficult to process it is an interesting knowledge gap to analyze which types of and what quantity of data can be considered a prerequisite for effective deployment of data assimilation for shipment arrival estimation.

2.4.4. Particle filtering in a high-dimensional state space

As described, particle filters aims to approximate a posterior probability distribution of a state variable of interest by sampling a (high) number of particles and weighing them based on a measurement density (likelihood) resulting from the differences between measurements in the real system and the particles. It is argued that the effectiveness of this method is dependent on the number of dimensions of the state space d.

Systems with a small or moderate d show to be suitable for particle filtering. However, as the dimension of the state space grows, particle filtering *can* become ineffective (Beskos et al., 2017).

The ineffectiveness of particle filtering in a high dimensional state space is the effect is a common enemy of multiple machine learning methods, referred to as the *curse of dimensionality*: A problem caused by the exponential increase in volume associated with adding extra dimensions (Shultz & Fahlman, 2017). This exponential increase in volume brings several implications which are challenging for machine learning models. Two of these implications are of high impact to particle filtering and therefore will be discussed into more detail.

Effectiveness of distance metrics

As particle filtering uses the distance between the particles and the real observation data to assign importance weights to each of the particles (e.g., a multivariate Gaussian distribution as used by Hu & Wu (2019)), the effectiveness of distance metrics in a high dimensional space are of high relevance. Aggarwal et al. (2001) showed that distance metrics can lose their meaning as the number of dimension increases based on a nearest neighbour comparison. This is generally caused by two types of problems.

The first problem is related to the different units of the different dimensions, the distance metric is required to unify the different dimensions. However, if each of the axis have a different unit or are measured on a different scale, a normal distance metric becomes useless. Clustering methods (that rely on distance metrics) have shown to become increasingly accurate after a normalization of the data (Doherty et al., 2004) which has become a best practice for a large set of machine learning methods.

The second problem for the effectiveness of distance metrics in a high-dimensional state-space is the fact that as the number of dimensions on which a distance metric increases, the distance between any two points becomes more similar. The ratio of an object's nearest neighbor over is farthest neighbor approaches to one when increasing the high-dimensional space (Aggarwal et al., 2001). A ratio of one inclines that the nearest and the farthest neighbour have (almost) the same distance and would therefore be assigned same weights, which is problematic for the effectiveness of particle filtering as this would make the posterior distribution exactly the same as the prior distribution.

However, Aggarwal et al. (2001) showed that different distance metrics have different degrees of sensitivity to this problem. In this analysis the focus is on L_k distances (e.g., Manhattan ($k = 1$) or Euclidean distance ($k = 2$)). It is shown that in a high dimensional space, the difference between the furthest and the smallest distance between any two data points ($\mathrm{Dmax}_d^k - \mathrm{Dmin}_d^k$) increases at the rate of $d^{1/k-1/2}$. This means that the smaller k, the more effective a distance metric becomes as is shown in figure 2.5.

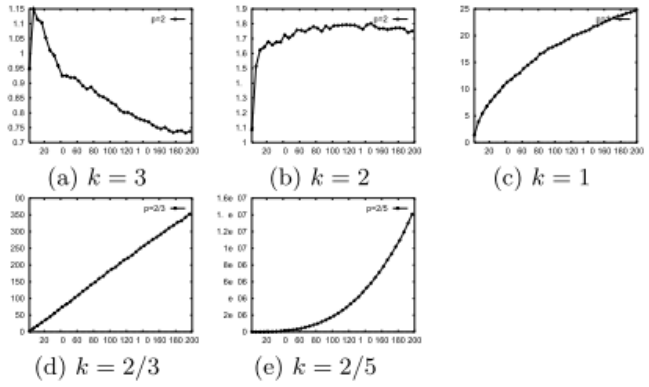


Figure 2.5: | *Dmax - Dmin* | depending on d for different metrics (uniform data) derived from Aggarwal et al. (2001)

To this extent Aggarwal et al. (2001) proposed an extension to the L_k distance, named the *fractional distance metric*. This provides more meaningful results (higher | *Dmax - Dmin* |) and significantly improves the effectiveness of clustering algorithms. Fractional distance metrics are distance metrics with $k < 1$, generally defined by $1/l$ with ideally l as some integer larger than 1. Therefore, the selection of a distance metric is

considered one of the potential solution directions for particle filtering in a high dimensional state space. A critical notion on this statement is that in recent publications (e.g., Mirkes et al. (2020)) the significance of this effect is disputed.

Differentiating in distance metrics is not the only possible path that to a solution for this problem. Kantas et al. (2014) showed that particle filtering can be effective even in for a system with a dimension with an infinite size. As long as the dimension of the information that is used to compute the likelihood function used for the weighing of the particles is small. In other words, as long as the dimension of the space on which the distance metric itself is computed is a (small) subset of the total dimension of the state space of the system, particle filtering is effective. Therefore a direction for particle filtering in systems with a high dimension could be reducing the dimension of the information used to compute the distance between the particles and the measurements of the real system.

Effects of sparse data by increasing dimensions

The second implication of a high dimensional state space is the fact that as the number of dimensions increase, data becomes more sparse. Bengtsson et al. (2008) describe how particle filters with a fixed number of particles tend to collapse as the dimension of the space of interest increases. This is because as the volume of the space represented by the dimensions grows, the number of regions of interest grow quadratic, as shown in figure 2.6. For machine learning models that aim to approximate the posterior distributions based on training data, more dimensions (feature) require more training data to ensure all spaces of interest are being sampled from (Koutroumbas & Theodoridis, 2009).

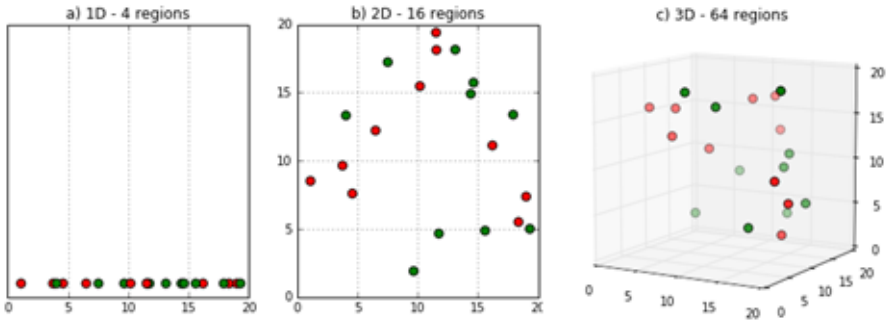


Figure 2.6: Quadratic growth of regions of interest by increasing number of dimensions, derived from DeepAI (2020)

Since particle filtering is not based on learning from training data but on approximating a posterior distribution by assigning importance weights, the number of particles should be increased so that each of the regions of interest is represented by at least one particle. The exponential growth of regions of interest would incline the number of particles (size of the ensemble) has to grow super-exponentially to the system dimension Bengtsson et al. (2008). Since DES DA requires each of the particles to be simulated for during the sampling step (see figure 2.4, an increase of dimensions would increase the computational burden super-exponentially to the system dimension in such a scenario.

This is a problem that can not be solved by a reduction of the number of dimensions that are used as input to compute the likelihood function as proposed by Kantas et al. (2014). This because the complete state space needs to be populated and not only the space used for the likelihood function. However, the fact that a state space of a dynamical system is the set of all *possible* states of the system (Nykam, 2020) does mean that the volume of this space is not only driven by the number of dimensions. An overlooked factor is the range of uncertainty over these dimensions.

This because the state space is seen as a n-dimensional hypercube which volume is dependent on both the number of dimensions as the length of the line segments of the dimensions. The length of the line segments is the range of the uncertainty over the dimensions, since a *possible* state of the system should fall within the ranges of uncertainty on each of the dimensions. Therefore, a system without any form of uncertainty can still hold a high number of dimensions but still have a fairly limited state space.

Based on this reasoning, a (complex) solution direction is proposed to reduce the computational burden of particle filtering in a high dimensional state space is making the number of particles generated at each resampling step inversely related to the degree of uncertainty in the system. This is referred to as *dynamic particle filtering*; a concept that is deployed based on metrics that assess the convergence of algorithms (Elvira et al., 2017) or the innovation error Closas & Fernández-Prades (2011) to dynamically adjust the number of particles.

3
Description of use case

As discussed in chapter 1, the goal of this research is to develop and test the applicability of a particle filter based data assimilation algorithm for discrete event simulations to increase supply chain visibility. To reach this goal, the research is based on use case in collaboration with an external stakeholder.

In this chapter, more context on the current supply chain operations and planning processes and the requested decision support will be given. Next to this, the types of generally available data on supply chains will be discussed.

3.1. Current supply chain operations

As described in the introduction, DES DA is assumed to be particularly helpful in the field of supply chains, given its ability to deal with nonlinear non-Gaussian behavior (Blanco et al., 2011) which typically characterize supply chains (S. Wang, 2010). The potential it shows for providing real-time decision-support by estimating (observable) hidden states (Cho et al., 2020) makes this an interesting method to increase supply chain visibility. For this research the definition of supply chain visibility has been narrowed down to the accurate estimation of future events.

The use case is developed in close collaboration with an external stakeholder: a fashion retailer with an international supply chain. The opportunity to develop and test this algorithm on a real supply chain benefits the future applicability and generalizability of this algorithm. This because the decision-support request, data availability, and potential experiments are delineated in a more precise and realistic manner.

The request by the external stakeholder is to improve the predictability of the arrivals of shipments at their terminal to improve the inventory planning of the warehouse. In the following sections, this request as well as the current supply chain operations will be discussed.

3.1.1. Supply chain structure

For the use case it is decided to focus on one specific subsection of the supply chain. This is the footwear division, which ships over fifteen million *prepacks* of shoes per year divided over more than twelve thousand shipments. The goods are produced by different vendors spread around Southeast Asia and Southern Europe and subsequently shipped via an intermediate port to a terminal where the goods can be temporarily stored before they are called in by the distribution center (DC). The terminal and DC are located in in the Netherlands where they are used to facilitate the distribution of goods over the European, Middle Eastern and African markets.

The flow of goods is initialized by a centrally placed order to one of the vendors for a specified volume of certain products. The vendors are third parties that facilitate the production of the goods. When the vendors have finished the production of the goods as agreed in the order, the goods are delivered by the vendors to the nearest *consolidator*. The consolidators are located at predefined ports and operated by an external booking party that ensures the goods are being shipped to the Netherlands. The stakeholder has requested to put an emphasis on the goods being shipped from Asia to Europe since this encompasses the vast majority of the goods (85% on average annually).

For the Asian part of the supply chain, goods are shipped via air (6%) or water (94%) to the Netherlands. The goods that are shipped via air are the goods that have faced an extreme delay caused by one of the vendors. Thus, the decision to ship these goods via air, is a policy decision and therefore it is decided to leave this out of scope for this forecasting exercise.

Based on these factors, it is decided to solely focus on the goods that are transported by ship from Asia to the Netherlands. The visualization of this part of the supply chain is visualized in figure 3.1.

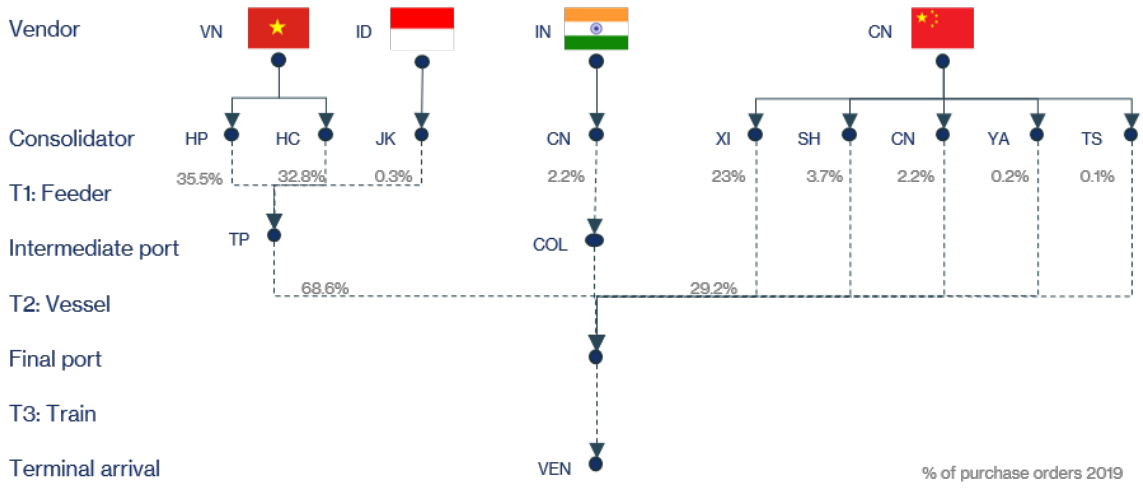


Figure 3.1: Supply chain structure Asia - the Netherlands by ship (flow of shipments: from top to bottom)

3.1.2. Sailing routes

As presented in figure 3.1 the goods that are being shipped from Asia to the Netherlands follow different sailing routes depending on the country of origin of the vendor. From Asia there are nine different ports of loading, where the goods are consolidated by the external booking party and subsequently shipped to the Netherlands. The transport for each of the routes under study is taken care of by one integrated shipping company / carrier (Maersk).

As figure 3.1 shows, four of the nine different routes are characterized by an intermediate port. These intermediate ports are central container terminals located in Asia from which larger vessels can depart for the intercontinental expedition to Europe. At each of the ports (consolidator, intermediate port and final port) the shipments need to be processed (e.g., being cleared by customs) before they can continue to the next stage of transport. Each of the sailing routes (both the feeders and the larger Vessels) are operated by a recurrent schedule of ships departing and arriving at each of the ports. The ships sail fixed routes (except for black swan events such as the blocking of the Suez canal) and once arrived at the final port (Rotterdam), the goods are unloaded and processed. After the processing, the goods are transported by train to the terminal.

3.1.3. Container call-in optimization

The moment the goods arrive at the terminal, the goods are in control of the stakeholder. This terminal is conceptualized as a storage facility for filled containers. These containters are subsequently are called in (one by one) for transport to the distribution center. The capacity of the terminal itself is no constraint for the supply chain operations since there is always enough space to take in new inbound containers.

In contrary, the distribution center itself is limited in storage space, as well as limited *intake and pick-/pack capacity* (Metyis, 2020). Therefore, the decision to call in containers to the distribution center from the Terminal is not based on simple First In First Out (FIFO) rules, but is optimized for.

This inventory optimization is currently based on three factors: The current capacity in the warehouse, the expected outbound orders and planned arrival times of shipments at the terminal. Goods at the terminal are selected based on a certain priority which is the compound of the type, value, season and brand of these goods. Currently substantial deviations in the actual arrival date versus the planned arrival date exist (see figure 3.4). This means the inventory planning of the warehouses is far from its optimal state since there are a high number of early and late arrivals.

An early arrival refers to a shipment that arrives earlier than the planned arrival date. If this is the situ-

ation, containers can not be part of the intake planning of the warehouse yet since these have not been taken account by the inventory optimization. These containers have to remain idle at the terminal until the actual planned arrival date has passed, which is free of charge for a period of two weeks.

Late arrivals result are shipments that arrive later than the planned arrival date. These containers are part of the intake planning of the warehouse, but can not be taken in at their planned date since these have not yet arrived. This results in a sub-optimal warehouse inventory. Empty slots in the warehouse are deemed undesirable and therefore bring costs.

If more accurate arrival estimations could be incorporated into this algorithm, these situations could be prevented from happening or the adverse effects could be reduced. For this the deviations between the actual and planned arrival dates should be reduced (i.e., the planned arrival date should be a more accurate representation of the actual arrival date).

The call-in of containers to the distribution center is being planned for with a granularity of weeks. This means that the planning is created and agreed on on a weekly basis. According to best practices for modeling for decision support (Ibrahim, 2018), the input data that is used for planning date should be determined on a higher level of granularity than the granularity that is planned for. This inclines that the arrival date of the inbound shipments should be planned for with a granularity of *days*.

3.2. Current (seasonal) supply planning process

The current (seasonal) supply planning process is divided into five distinctive processes, which encompass different phases of the supply planning and involve collaboration with different stakeholders. This process is visualized in figure 3.2. All dates for the supply planning are relative to the moment the goods can be in stores; this moment is referred to as the Customer Touch Point (CTP). Based on the desired CTP, all previous steps are planned backwards to create feasible deadlines.

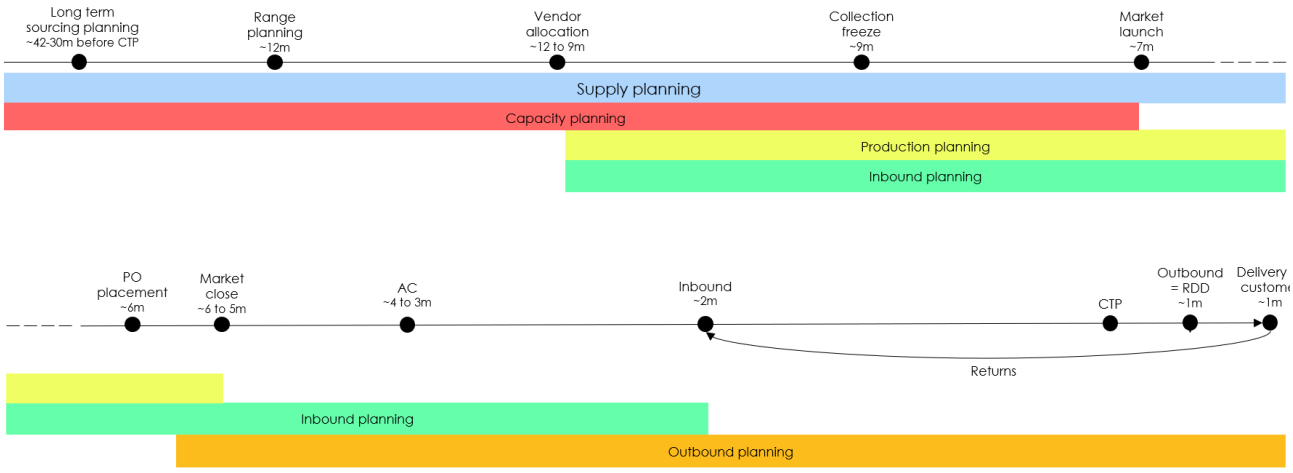


Figure 3.2: Visualization of (seasonal) supply planning process, derived from Metyis (2020)

As shown in figure 3.2 the first two processes take off approximately 42 to 30 months before CTP. The *supply* and *capacity planning* are concerned with the high level planning of *types of goods* that are being ordered as well as the selection of the vendors by which these goods should be ordered. These plans are supposed to be finalized roughly six to seven months before CTP.

A few months before these plans are completely finalized (12 to 9 months before CTP), the *production* and *inbound planning* take off. The result of the production planning is an *agreed AC date*, this is the date (that is agreed on with the vendors) by which the ordered goods are supposed to arrive at one of the consolidators. These dates are the final pieces of information that are needed to finalize the purchase orders (PO's). Once all the PO's are placed, the markets close (five to six months before CTP); this is the end of the

production planning phase.

Based on the preliminary AC dates and later on the agreed AC dates, the *inbound planning* already starts. This process lasts until the orders are delivered at the terminal (approximately 2 months before CTP). This process is concerned with forecasting the terminal delivery date and streamlining the inbound delivery process.

From the moment the markets are closed, the outbound planning process starts as well. This process is concerned with the planning and allocation of the quantities of goods to customers and improving the level of detail in the outbound plan of the supply chain. This research will focus on increasing the accuracy of the *inbound planning* process.

Inbound planning process

Since the inbound planning process is the supply planning process that aims to forecast the terminal delivery date, this process is discussed. In general terms, the result of the inbound planning process is a continuously updated set of Supply Chain planned arrival dates of shipments related to PO's at two different points in the supply chain: arrivals at the consolidator and arrivals at the terminal.

The arrivals at the consolidator are uncertain and difficult to influence for the stakeholder since the arrival dates are merely dependent on the performance of the vendors. As previously described, the final step of formalizing a PO is finding agreement on an AC (at consolidator) date. In the months between the agreement on this date and the actual AC delivery, the AC date is updated in the inbound planning process based on external input on the status of the production.

When the planned AC delivery date is updated, this is referred to as the *planned AC Date* instead of the *Agreed AC Date*. For the majority of the cases the planned AC Date is the same date or later than the agreed AC date since delays do happen and goods are generally not delivered but put on hold if the production process is faster than agreed on (see also figure 3.3).

After the goods have arrived at the consolidator, the AC date changes from planned to actual. The next date that is planned for in the supply chain is the terminal (arrival) date. This date is calculated by taking the average of the duration from AC delivery to terminal delivery for the historical data and adding this to respectively, the agreed, the planned or actual AC dates. An overview of the composition and the related level of uncertainty is given in figure 3.3. Even when dates are flagged as *actual* in the S&OP (Sales and Operations Planning) systems, a small amount of uncertainty remains: errors in the process of booking in actual dates do exist.

Phase	AC Date		Composition estimated terminal date	
	Most up to date source	Visualization of uncertainty	Most up to date composition*	Visualization of uncertainty**
Beginning of production planning – PO placement	Agreed AC Date		Agreed AC Date + transport duration	
Po placement – Update in AC Date	Planned AC Date		Planned AC Date + transport duration	
AC Departure – Terminal Delivery	Actual AC Date		Actual AC Date + transport duration	
Terminal Delivery - CTP	Actual AC Date		Terminal date	

Planned / Actual date
Uncertainty around AC date
Transport duration uncertainty

* transport duration is estimated based on historical data
** transport duration is not updated after AC departure date

Figure 3.3: Composition of and uncertainty around AC and Terminal date

3.2.1. Assessment of inbound planning performance

What stands out from the previous disquisition is the fact that the planned terminal date is not updated after the shipments have departed from the consolidators. This while both the vessel delays as unexpected events in the operations in the ports can cause substantial delays to the planned terminal date according to Hasheminia & Jiang (2017). To see how well the current inbound planning process performs, the deviations

between the planned and actual terminal delivery days are assessed for the different transportation chains (figure 3.4) ‘

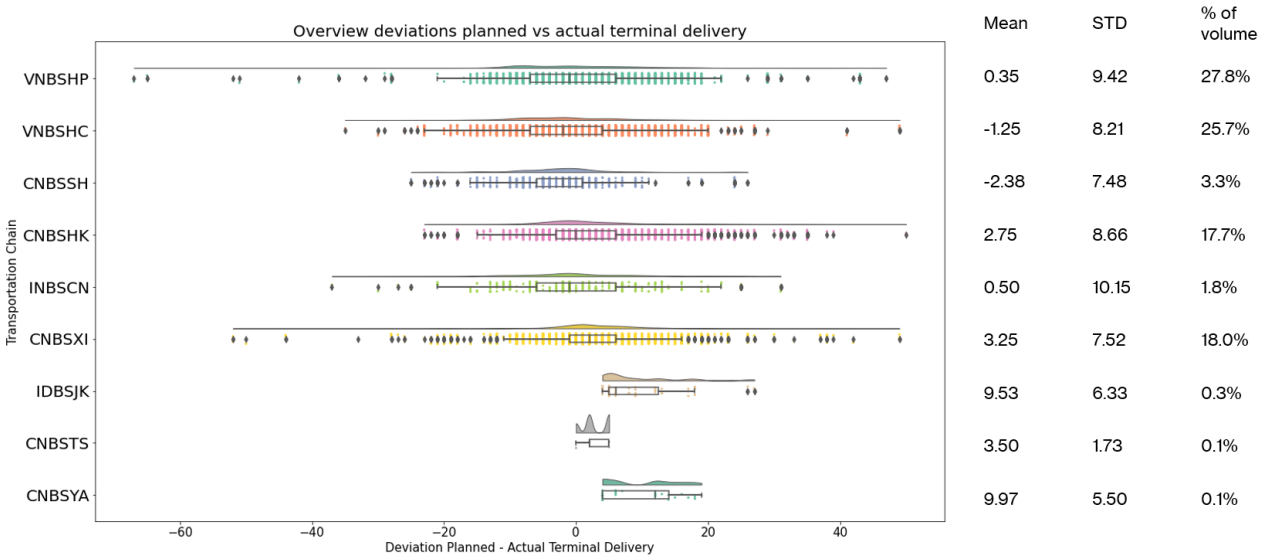


Figure 3.4: Deviation between planned and actual terminal delivery date (days)

Transportation chains are a composition of the country of origin of the vendor (the first two letters of the supply chain indicator), the the mode of transportation (the middle two letters of the supply chain indicator), and the location of the consolidator (last two letters of the supply chain indicator).

In this analysis only the shipping routes between Asia and Europe are taken into account as previously argued for. The analysis confirms the hypothesis that the deviations are (slightly) right-skewed since delays are more common than earlier arrivals (see also figure 3.3). The fact that the transportation chains responsible for the highest volume (more than 89%) all have a standard deviation of more than 7.5 days confirms the idea that the current inbound planning procedure can and should be improved.

3.2.2. Decision support request

The stakeholder has put out a request to increase the predictability of the arrival of shipments at the terminal. An increase in the predictability of the arrival of shipments would allow for the deployment of a call-in optimization algorithm for the distribution center that takes future arrivals into account. This would prevent containers with a high priority but a later arrival date to be blocked by low priority containers while as well reducing the risk for empty slots in the distribution center.

To satisfy this decision support request, the main goal of the data assimilation algorithm is to improve the accuracy of the planned terminal date to allow for the incorportation of future arrivals in the call-in optimization algorithm. This goal can be split in to three sub-goals.

First, the average deviation between the planned arrival date at terminal and the observed arrival date at terminal should be reduced. This means the algorithm should have a high average accuracy.

Second, the accuracy of the algorithm should be robust. This means there should not be too many deviations between shipments regarding accuracy.

Third, the planned terminal date should be accurate before the call-in schedule is calculated. This means the arrival estimations should be accurate at at least a week before the observed terminal date.

3.3. Data availability

The last section regarding the use case refers to the availability of data for the system under study. To eventually apply a data assimilation algorithm, data should be available to be assimilated. Examples of DES DA applications (i.e., the works of Xie (2019) and Hu & Wu (2019)) generally assimilate real sensor-data.

However, the fact that this supply chain is viewed on a higher level, the most important source of data is the documentation of events taking place at certain touch points in S&OP systems or other platforms.

Three different sources of data suitable for assimilation for supply chains are discussed and summarized in table 3.1. Figure 3.5 gives an overview of the different actual and planned data points. The planned dates are placed located at the moment of planning (e.g., the planned terminal date is determined at the moment of departure from the terminal). In chapter A, the other specific data points are discussed in more detail.

	Level of granularity		**Data type**
Data source	**Touch points**	**Content of shipment**	
S&OP Data (SAP)	Low - event based	High - All	Planned & Actual
Carrier Data (Tradelens)	High - event based	Low - Container ID	Actual
Third Party Data (vessel tracker)	High - "continuous"	Low- Vessel ID	Actual

Table 3.1: Different data sources suitable for assimilation and level of granularity

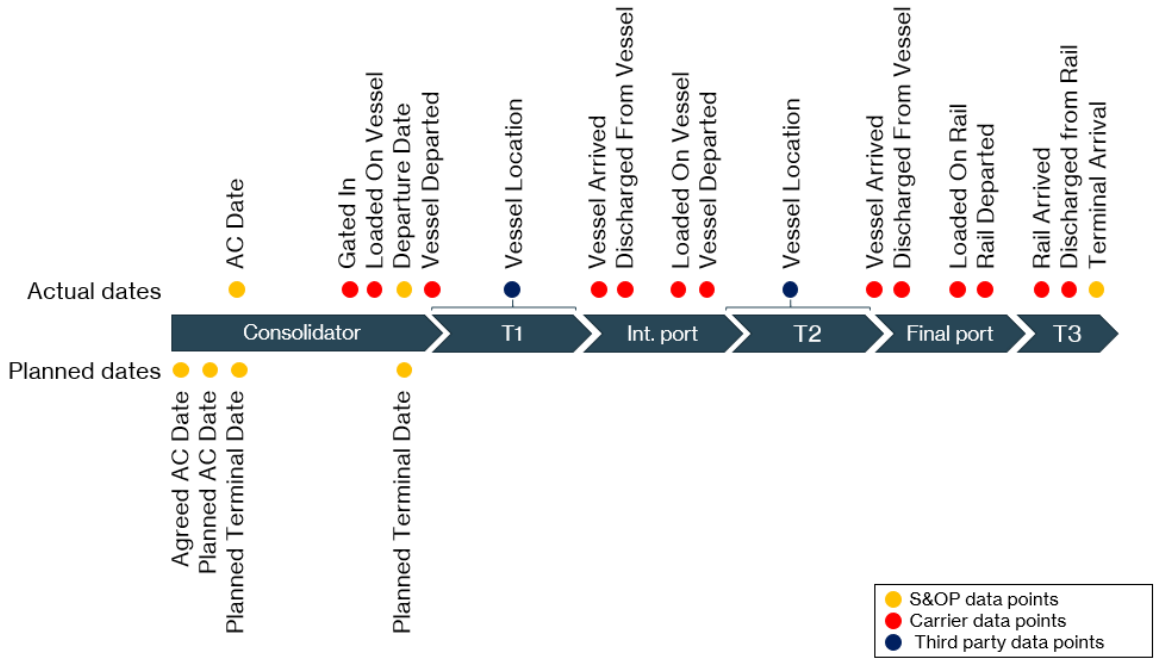


Figure 3.5: Data points of different data sources

The first data source is defined as the *S&OP Data*, this is the data that is accessible via the Sales and Operations Planning (S&OP) software (e.g., SAP software). This data is defined by a low level of granularity with respect to the number of touch points (locations where events are recorded) with event based data. This means that the arrival and departure days of shipments of a relatively low amount of locations are recorded. However, the level of granularity of the content of the shipments is classified as high. All details on the shipments (from quantity, value to product details) are stored. Both the planned (expected) and the actual dates of these touch points are stored in the system. To summarize, the S&OP data contains little data points but a high level of information per data point.

The second source of data is defined as *carrier data*. As shown in figure 3.5, the last known pieces of information from the S&OP data originate from the departure of the shipments from the consolidator and the next known touch point is at the terminal arrival. This is because between these points, the goods are handled by the carrier. Because of the increased technical means such as RFID and blockchain, carriers have found means to increase the visibility in these parts as well.

Based on container ID's, a higher number of touch points are made available via platforms such as Tradelens (Jugović et al., 2019) by carriers. To use this information, the shipments should be carried out by a provider that is attached to such a platform and the digital infrastructure to access this data should be put in place.

Summarizing, this source of data could be used to increase the level of granularity of the number of

touch points per shipment. On the other hand, the level of granularity on the content of the shipment is considered low since it only stores actual dates and the ID of the container.

When reflecting on figure 3.5, the S&OP data and the carrier data make tracking and tracing the movement of the shipments throughout the system possible at a relatively high level of detail. The only remaining "black boxes" are the movements of the shipments between the departure and arrival of vessels.

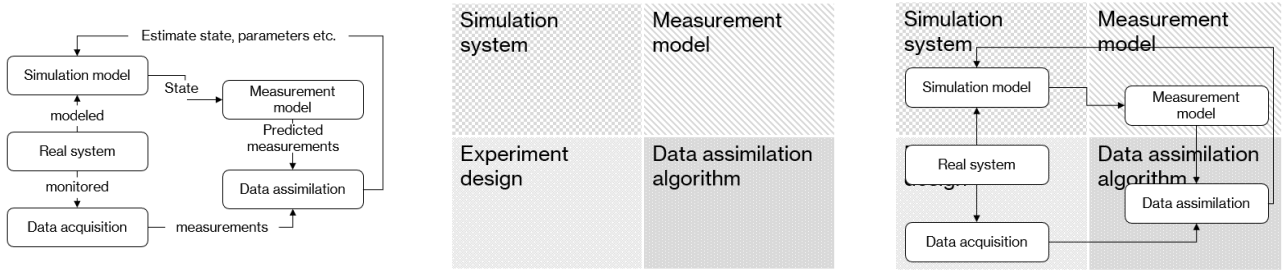
This data is not provided by the aforementioned data sources. For this, several *third party data sources* (e.g., *vessel tracker* or *marinetraffic*) that track global vessel movements based on the global AIS data are used. This data has a high level of granularity since the positions of vessels are recorded in a "continuous" way. These states are not updated based on events but on based on the update interval of the respective API. The level of the granularity regarding the content is low since only the ID of the vessel is stored in these databases.

4
Data Assimilation Algorithm

To test how the accuracy of shipment arrival estimations can be increased by DES DA several steps are taken. A data assimilation system is designed based on the DDDAS framework as presented in figure 4.1.

The crucial adjustments that have been made to the state-of-the-art frameworks to make data assimilation suitable for supply chains made to the *data assimilation algorithm*. This algorithm maps the input (supply chain data and a DES supply chain model) to the output (shipment arrival estimations). The form and volume of the input data as well as the type of output data needed to satisfy the decision support request require specific adjustments to the algorithms that are in place.

The remaining components, the simulation model, the real system, the measurement model, and the data acquisition component, are *implemented* based on the descriptions provided by Xie (2019) and Hu & Wu (2019). These components are discussed in appendix A, because these components did not face any major changes.



(a) DDDAS framework based on Xie (2019)

(b) DES DA based on Hu & Wu (2019)

(c) Relation DDDAS and DES DA framework

Figure 4.1: DES DA framework in a DDDAS context

The remainder chapter aims to give the reader a good understanding of the adjustments made to the algorithm. To do so, the the challenges and that arise from the input and output are discussed first. The solutions to these challenges are presented next. Subsequently all functions that compose the algorithm are presented in pseudo code and explained. This will benefit future implementation of the algorithm to different systems as this is a tutorial for the construction of data assimilation algorithms for high-dimensional open systems that require the estimation of future events.

4.1. Challenges from input data

At the end of the sampling step in each iteration of the particle filtering algorithm, weights are assigned to each of the particles. These weights are the inverse of a distance metric computed by comparing the distance between (a subset of) the system state of the simulation instances to the measurements available on the real system. The goal of the distance metrics is to separate the *good* and the *bad* model instances. Model instances with a low distance to the real system have a higher probability of generating more *offsprings* in the next iteration of the particle filtering exercise than model instances with a high distance to the real system.

A good distance metric should be useful to determine which model instances (particles) have a close resemblance to the real system and which do not have this resemblance. As discussed in section 2.4.4, distance metrics become less effective in a high-dimensional state space.

The number of dimensions of the state space that is used to compute the distance between the particles and the real system is a result of three *drivers of dimensionality* which are discussed in the next section.

4.1.1. Drivers of dimensionality

The sensor data that is available on the supply chain is generated by *events* that occur when shipments (entities) pass certain sensors or when the location of vessels with shipments are updated. This means that the number of data points that are used for a comparison between the particles and the real system after one iteration a function of three factors:

- *The number of shipments* - More shipments (ceteris paribus) equal more entities that pass sensors in a certain interval and thereby cause more events that generate data points. Summarizing, more shipments cause more data points.
- *The number of sensors* - More sensors (ceteris paribus) equal more events in a certain interval since a shipment will face more sensors within the same timespan and therefore more sensors generate more data points.
- *The length of an interval* - Longer time intervals (ceteris paribus) equal more events since more shipments will pass the sensors because more time is available for these events to happen. Therefore longer time intervals will generate more data points

The *data acquisition system* for the supply chain under study embodies the unique characteristic that each data point is linked to a specific shipment (see section A.1). This makes a pairwise comparison of each data point possible, since all events in the real system can be matched to events that take place in the simulation system. This makes it possible to compute a distance between the measurement in the real system and the observation and an observation in a particle.

This process is represented in a simplified way by table 4.1. If one would then compute the relative distance of these particles, this could be done by computing the Manhattan distance over the rows. Each row (which stands for an event) would here be a new dimension for the distance computation.

	real system	**particle 1**		**particle 2**		...	**particle n**	
event	**meas - urement**	**obser- vation**	**dist- ance**	**obser- vation**	**dist- ance**		**obser- vation**	**dist- ance**
sens_1_shipm_1	1	2	1	2	1		2	1
sens_2_shipm_1	4	3	1	4	0		3	1
sens_4_shipm_3	6	4	2	6	0		4	2
sens_1_shipm_2	3	5	2	2	1		5	2
...	...	...	...	...	...		...	...
sens_n_shipm_n	2	6	4	6	4		6	4

Table 4.1: Example of distances computation per time interval

The fact that that: 1) more than a thousand different purchase orders are placed each each year, 2) the number of sensors (touch points) is high and, 3) data can only be assimilated once a day, make the number of events that generate data points per interval *high*. Thus, the number of dimensions over which a distance is computed is high.

A high dimensionality is a the effectiveness and accuracy of the data assimilation algorithm as distance metrics generally become less efficient in higher dimensions (Aggarwal et al., 2001). A decreased efficiency of the distance metric means the quality computation of the weights is compromised as well. Three different distance metrics are proposed that can mitigate this problem.

Weighted distances

Before any of the distance metrics are proposed, one more concept needs to be discussed: the notion of weighted distances. In other data assimilation studies such as the Goldmine model of Xie (2019) or the road traffic of Hu & Wu (2019) the entities are assumed to be equal. However for supply chains, not all entities in a model have an equal level of importance. There are entities that represent shipments with high volume of

goods, this makes these shipments more important to be estimated correct. Other shipments with a small volume are less important. Therefore the weighted distance (dw) is computed by multiplying the computed distance for an entity by the relative volume of that entity:

$$dw_{n,i,j} = \frac{V_i}{\sum_{i=1}^{Ms} V_i} \times d_{n,i,j} \tag{4.1}$$

Ms is the set of all observations in a time interval and V_i is the volume of a shipment.

4.1.2. Proposed distance metrics

Three potential distance metrics are proposed to compute meaningful weights in a high-dimensional state space. Given the time and the scope of this research only one of these metrics is implemented: the *fractional distance metric*. All other distance metrics are suggested for testing in future research in situations with even a larger number of dimensions.

Fractional distances

The distance metric that is tested and therefore will be introduced first is referred to as *fractional distances*. The foundation for this metric comes from the standard *L-norm metrics*. This is an intuitive set of metrics which do not change the number of dimensions. The best known metric of this group is referred to as the *(standard) L1 distance* $dl1_n$, also known as Manhattan distance. This is a simple sum of the distances for each of the observations in an interval which is used as a frame of reference:

$$dl1_n = \sum_{i=1}^{Ms} \sum_{j=1}^{Ms} dw_{n,i,j} \tag{4.2}$$

Fractional distance (dfr_n), are based on the same concept and even belong to the same group of *L-norm metrics*. *Fractional distance* metrics were first described by Aggarwal et al. (2001) (see also section 2.4.4) who showed that L_k norms become more effective distance metrics for smaller k's. This should result in better weights assigned to the particles and in the end for better accuracy results. However, it should be noted that the significance of this increase in accuracy is disputed (Mirkes et al., 2020). The fractional distance for any L is computed as followed:

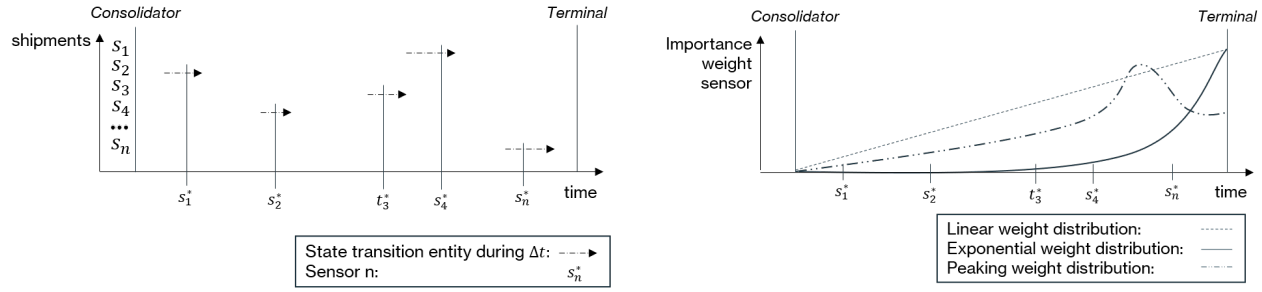
$$dfr_n = \sqrt[\frac{1}{L}]{\sum_{i=1}^{Ms} \sum_{j=1}^{Ms} dw_{n,i,j}^{\frac{1}{L}}} \tag{4.3}$$

Importance weighted metrics

A second solution to this problem is referred to as *Importance Weighted Metrics*. This group of metrics tries to reduce the effect of a high number of dimensions by creating difference in importance of the different sensors. This metric is referred to as the *importance distance* (dir_n). For the previously discussed distance metrics each of the distances are assigned the same weight. However, since the goal of the data assimilation exercise is to forecast shipment arrivals, one could argue that measurements close to the terminal (point of arrival) are more important than measurements far away.

Therefore it is proposed to assign importance weights to the different sensors. The closer a sensor is located to the terminal, the higher the weight that should be assigned to that particle resulting from an entity passing this sensor. Based on this disquisition, and the example presented in figure 4.2a, the distance that come from observations shipment s_n that passed sensor s_n^* during the time interval Δt should get assigned a higher weight than the distance resulting from shipment s_2 that passed sensor s_1^* because entities that pass sensor s_n^* tend to arrive sooner at the terminal than shipments that pass sensor s_1^*.

To achieve this goal, weight distributions should be assigned to each of the sensors, for this three different distributions are proposed. The distributions are graphically represented in figure 4.2b All of the weights are distributed based on the average time it takes for a shipment to reach the sensor:



(a) Different sensor locations

(b) Different weight distributions

Figure 4.2: Visualization of importance weights based on location of sensor

- *Linear weight distribution* - The weights assigned to the sensors are linearly related to the time it takes for a shipment to reach the sensor, represented by the dotted line in figure 4.2b. This will make deviations at the beginning of the supply chain *less* relevant compared to deviations at the end of the supply chain
- *Exponential weight distribution* - The weights assigned to the sensors are exponentially related to the time it takes for a shipment to reach the sensor, represented by the dashed line in figure 4.2b. This will make deviations at the beginning if the supply chain all most irrelevant compared to deviations at the end of the supply chain. This is expected to give more accurate results with a relatively low degree of timeliness (i.e., significantly higher accuracy at t_{-3} than t_{-30}.
- *Peaking weight distribution* - The weights assigned to the sensors are specifically trimmed to steer for accuracy at a certain moment (e.g., on average 2 weeks before arrival), represented by the solid line in figure 4.2b. This is a preferred method when a certain degree of timeliness is preferred regarding the accuracy.

This concept results in the equation for the second metric that could be applied. The *importance distance metric* (di_n), where the L1 distance is multiplied with the weight sw_j associated to the sensor i at which the distance is measured:

$$di_n = \sum_{i=1}^{Ms} \sum_{j=1}^{Ms} dw_{n,i,j} \times sw_j \tag{4.4}$$

Edit distance metrics

The third distance metric computes the distance between two sequences of events. The event-based sensors are triggered during time interval Δt by events of shipments passing these sensors. These events are stored in an event-sequence of sensor-shipment two-tuples (t, e) with e defined as the combination of a sensor and shipment: (s_i^*, s_i). Differences between two sequences is quantified by a so-called *edit distance* (de_n) is implemented by Xie (2019) in the *goldmine model*.

This distance defines the *amount of work that has to be done to convert one sequence to another*. Each transformation that is needed to convert one sequence to another has a certain cost dependent on the type of transformation. Three different transformation types are identified based on Xie (2019):

- *insert* - Inserts an event of type e at time t into the sequence. This cost is dependent of the occurrence (*occ*) of e in a as reference sequence. In the situation of a supply chain each shipment will pass each sensor and therefore the occurrences constant and therefore it is decided to make the volume of the shipment V

$$c(\text{insert}(t, e)) = w(e) \propto \text{V}(e) \tag{4.5}$$

- *delete* - This cost is defined to be the same as the cost of an *insert operation*:

$$c(\text{delete}(t, e)) = w(e) = c(\text{insert}(t, e)) \tag{4.6}$$

- *move* - The operation that moves an existent event *(t, e)* from time t to time t'. The cost of this operation is dependent on the distance in time between t and t'.

$$c\left(\text{move}\left(t, e, t'\right)\right) = v \cdot \left|t - t'\right| \tag{4.7}$$

This aforementioned distance is multiplied with a constant v which should satisfy the following condition: $v\left|t-t'\right| < 2 \cdot w(e), \forall\, t, t', e$ because otherwise first deleting and later on inserting the same event would be less expensive.

For each of the particles, a *particle sequence* exists for a specific time interval Δt which is edited into the *observed sequence* of the real system during time interval Δt. The operations needed for this transformation are stored in the operations sequence $O = \{o_1, o_2, \ldots, o_n\}$ The cost and thereby the *edit distance* is the sum of the cost of the operations stored in O:

$$de_n = c(O) = \sum_{i=1}^{n} c\,(o_i) \tag{4.8}$$

4.1.3. Chosen distance metric

The time and scope of this research do not allow to test all three different distance metrics. Therefore one of the three metrics is chosen for future research. The fractional distance metric is selected for this because of two reasons. First, this distance metric has a solid foundation in literature as Aggarwal et al. (2001) proofed that fractional distance metrics do become more effective in a high-dimensional state-space. Second, because this metric can be applied to all different types of systems without any manual adjustments. The weights in the *importance weighted metrics* need to tuned for each new set sensors and the edit distance metric only works with event-based observations.

4.2. Challenges resulting from the intended output of the algorithm

The output of a particle filtering algorithm not one single number but an estimation of the probability density function of a variable over a certain dimension. For this study, the goal of the created algorithm is to make estimations on the (future) arrival times of inbound shipments. Making estimations of future states differs from estimating actual / current states. How this differs and what the implications for the data assimilation algorithm are is discussed below.

4.2.1. Future state estimation

The fact that the goal of this study is to estimate future arrival times of shipments, has implications for data assimilation algorithm. Especially for the data assimilation algorithm. Generally two fields of application for data assimilation are identified:

- *Actual State Estimation Applications* - Data assimilation is applied to estimate *actual* (current) states. Example of this field of application is the real-time estimation of traffic states on road segments as performed by Hu & Wu (2019).
- *Future State Estimation Applications* - Data assimilation is applied to estimate *future* states. An example of this field of application is the is the estimation of arrival times of trucks in a goldmine (Xie, 2019).

For particle filtering, *Future State Estimation Applications* are more difficult, since future sate estimations impose another requirement on the sampling step (see figure 2.4). Normally, the sampling step is used to transform the system states of each of the particles over time from t_0 to t_1. For DES DA this implies the running of each of the models that represent the particles to time t. After this the states of the particles are transformed by the measurement model and compared to the measurement data. Based on this comparison weights are assigned which are used for the resampling.

Implication on the data assimilation algorithm

For future state estimation, two sets of states should be saved for each assimilation interval (Δ_t). This is visualized in figure 4.3. First, the simulation should be ran from t_0 to t_1 and all *relevant* system states for each of the particles should be saved (which is used to compute the weights of the particles later on). Subsequently each of the particles should be ran for a period of time to estimate the shipment arrival times t_j^* for each of the shipments in the system. These shipment arrival times should be saved with the timestamp t_1 to construct the previously described probability density functions. This has the following implications for the data assimilation algorithm:

- The algorithm should be able to simulate the particles not only for the selected assimilation interval, but also simulate until all shipments that are in the system at t_0 have arrived at the terminal to store the expected arrival times.
- The algorithm should be able to store the expected arrival times with a timestamp and a shipment ID to allow for calculation of accuracy metrics at a later stage.

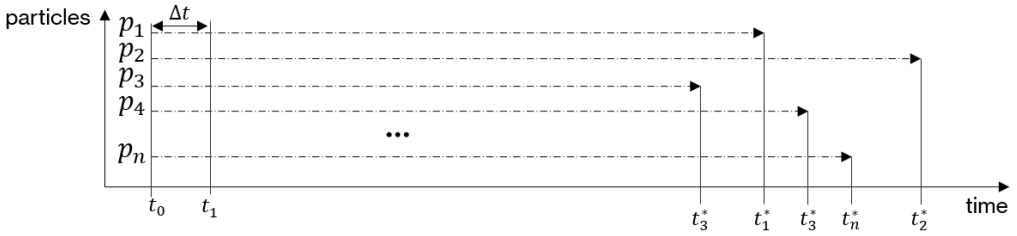


Figure 4.3: Visualization of saving of estimated shipment arrival times of one shipment for one time interval

4.2.2. Arrival estimation clusters

Estimating arrival times has previously been applied by Xie (2019) for a goldmine model. In the goldmine model, this process is extra complex, since arrival times could not be coupled to a specific entity and therefore the data needed one extra processing step: the different estimated arrival times (corresponding to the different particles) are clustered into groups by a k-means clustering algorithm.

The specifics of the supply chain data (see table 3.1) prevent this processing step since each of the data points resulting from each of the sensors can be tied to a specific shipment and thereby the clusters of shipment arrival estimations can be easily separated.

However, this makes the arrival estimation process not fully straightforward. Each of the particles will make a forecast on the arrival times of each of the shipments. These forecasts are recorded together with the weights assigned to these particles. The weights and the forecasts of the shipment arrival times together form the a cluster of shipment arrival forecasts which are represented by a weighted histogram. An example of the weighted histogram is visualized on the left hand side in figure 4.4.

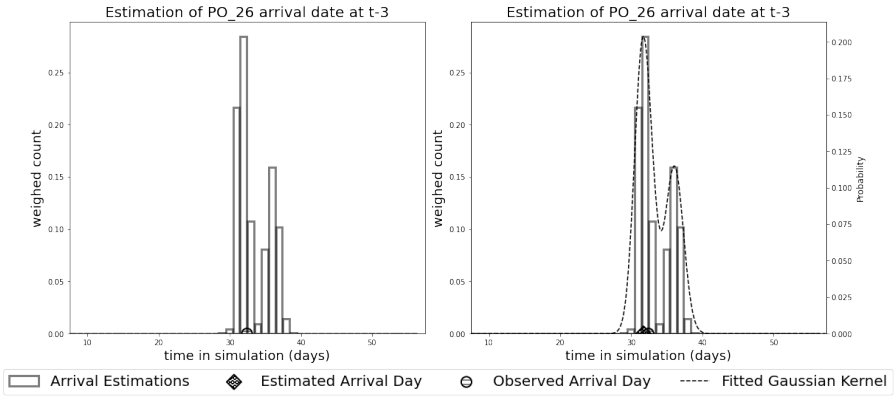


Figure 4.4: Visualization of weighted histogram of shipment arrival forecasts

The basis for this histogram is formed by two sets of data: a set with ground truth arrivals A characterized as tuples: $A = \{(t_1, e_1), (t_2, e_2), \ldots, (t_n, e_n)\}$ with t being an instant in time and e the ID of a shipment. The second data set is the output of the data assimilation algorithm C. This is a set of m clusters with tuples of arrival times belonging to a cluster and corresponding weights that belong to each of the particles. These data points are clustered clustered per shipment: $\left\{C_c \mid C_c = \left\{(t_1^c, w_1), (t_2^c, w_2), \ldots, (t_{n_c}^c, w_c)\right\}\right\}_{c=1}^{m}$.

To go from a cluster of estimations to single estimated arrival date, a probability density function is fitted to this cluster by using a *Gaussian kernel*. Based on the fitted Gaussian kernel, an estimated arrival t_{cj}^*

is extracted. This is the point in time where the probability density function peaks. The fitting of a Gaussian kernel and the estimation of an arrival day is visualized on the right hand side in figure 4.4.

Implication on the data assimilation algorithm
The shapes of the two previously described data sets A and C are the only implication on the data assimilation algorithm:

- After assimilation it should be possible to retrieve the clusters with expected arrival times based on the observed arrival times of shipments.
- For each of the arrival time forecasts by each of the particles, the weight corresponding to that particle should be stored.

4.3. Software implementation

The data assimilation algorithm that will be presented in the following sections is implemented in Python. The annotated code for the algorithm can be found in *Data Assimilation Algorithm.py*. For future research other high-level programming languages could be used as well as long as one prerequisite is met: It should be possible to execute the simulation models that represent the particles in parallel. This decreases the execution time of the data assimilation algorithm dramatically. For this the connection to the simulation environment is crucial, this is discussed below.

4.3.1. Software implementation of the simulation environment

The simulation model is implemented in a discrete event simulation environment. For this research, Salabim is selected as the software package to create the simulation environment. Salabim is an open source object-oriented discrete event simulation software package implemented in python. Salabim allows queue handling, resources and statistical sampling (Ham, 2020). Some real time animation and monitoring options are built in as well, but are not comparable to commercial simulation software packages.

Over the course of this research Salabim has showed some clear benefits compared to other (commercial) simulation packages and one significant downside. These factors are briefly discussed in the following paragraphs; a more elaborate reflection on these factors, solutions for the identified downside and recommendations for the selection of a software package in future research can be found in appendix B.

Using Salabim for this research surfaced large three benefits of this software packages. First, Salabim is an open source simulation package which can tremendously reduced the costs of research, especially for commercial applications. Second, the fact that Salabim is a python-based package eases the process of connecting the simulation models to the data-assimilation algorithm as the same data structures are used. Third, the simulation models that represent the different particles can be executed in parallel by making use of *Multiprocesing*. This significantly decreases the time of executing the particles.

Apart from the aforementioned benefits of using Salabim, there is one important downside on using Salabim. This relates to the saving of model states and creating the *offspring* particles after the resampling step. After the simulations are stopped after the sampling step, instances of the simulation model including their states need to be saved. At the end of the resampling step, the system states of the new (offspring) simulation models at the beginning of the new interval need to be exactly the same as the system states of the original states of the particles at the end of the previous assimilation interval.

To reduce the duration of the data assimilation exercise, the simulation models are executed by multiprocessing. To access (states of) the simulation models in the main process these models should be saved by the *Pickle* package. In the ideal world the model instance and the model states are saved together (presented on the left hand side of figure 4.5). Since Salabim makes use of Generators, this is impossible.

Therefore a different solution is found: At the end of the data assimilation interval all *states* of the simulation models are saved and at the beginning of a new simulation interval the simulation model is *reconstructed* (see the right hand side of figure 4.5). This is a complex procedure, especially because the internal logic of the simulation model needs to be adjusted. Because these adjustments make the data assimilation algorithm and the simulation model unnecessary complex it is recommended to use a different simulation

software package for future research. More information on the implementation of the adjustments needed and the key factors that a future simulation software package should embody are given in appendix B.

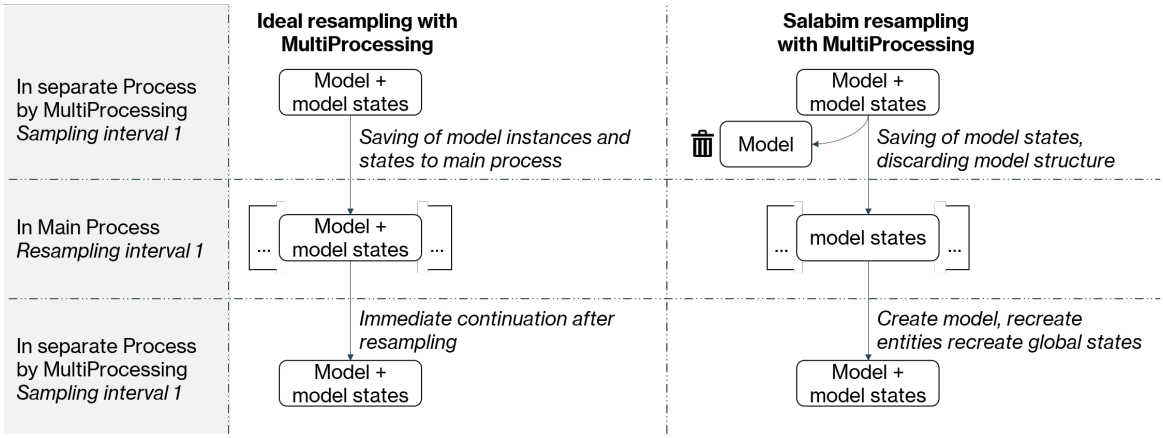


Figure 4.5: Resampling with MultiProcessing and Salabim

4.4. Data assimilation algorithm

The data assimilation algorithm is the centerpiece of the algorithm that maps the input data (measurements) to output data (arrival date estimations) by assimilating the input data into the simulation system. For this a particle filtering approach as described in section 2.3.3 is adjusted and implemented. For consistency and to allow the reader for a better understanding of the algorithm, the discussed algorithm is visualized in figure 4.6.

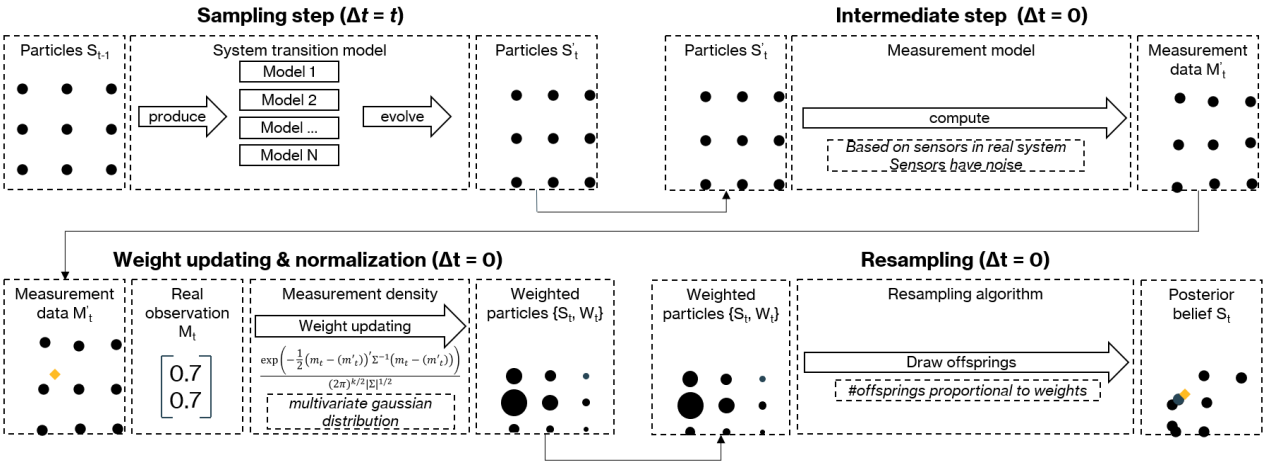


Figure 4.6: Standard data assimilation algorithm (bootstrap filter)

This standard algorithm is transformed based on the implications imposed by the in- and output data. For this newly generated algorithm the existing particle filtering algorithm of Cho (2019) is used as a basis. The constructed particle filtering algorithm consists of six different steps (see algorithm 1) which each are a compound of one or more functions for which the annotated code can be found in *Data Assimilation Algorithm.py* Each of the steps and the corresponding functions are discussed in the sections below, some of the more trivial functions will be briefly discussed while the functions that are special for DES DA in supply chains or will be discussed in more detail.

Algorithm 1: Data assimilation algorithm

Input: Prior belief of system, sensor data
Output: Posterior belief of future system states

```
// Preparation steps
Step 1. - Preparation of the variables
Step 2. - Simulating and measuring the real system
// Data assimilation algorithm steps
while t < 90 do
    if t = 0 then
        Step 3.1 - Sampling - Initialization
    if t != 0 then
        Step 3.1 - Sampling - Advance
    Step 4. - Intermediate step
    Step 5. - Weight updating  normalization
    Step 6. - Resampling
```

4.4.1. Step 1 - Preparation of the variables

In the first step of the algorithm, the variables for the experiments are prepared. In four different functions, the data availability for the different experiments is characterized, the parameters of the ground truth model are set, and the parameters for all the particles are sampled between acceptable ranges. An overview of the order of the different functions is given in pseudocode in algorithm 2.

At first, the two functions `createDataExperiments()` and `initializeTotalDistanceDict()` are used to formalize the different levels of data availability. Depending on the level of data availability, sensors are turned on and off (see also figure 3.3). This is governed by `createDataExperiments()`.

`initializeTotalDistanceDict()` subsequently generates storage room to store the the computed distances for each of the particles. This is used for the normalization of the weights in the function .

Subsequently the parameters and their corresponding ranges for the various models are set, starting with he parameters for the ground truth model by function `setGroundTruthParameters()` (see section A.3 for more insight on the identical twin setup).

As described in section 2.3.3, the goal of the particle filtering exercise is to transform a prior belief of a system into a posterior belief by assimilating measurements. The prior belief is a set of N (generally a large number) particles for which the states / parameters are randomly sampled over the ranges that are deemed to be plausible for this specific system. For this, *Latin hypercube sampling* (LHS) is used. LHS is often used for constructing computer experiments, especially for Monte Carlo simulation because of it uses *orthogonal sampling* to ensure that a set of random numbers is representative of variability over the various dimensions. (Stein, 1987). For this the set of plausible parameter ranges are defined by `setParticleParamRanges()` and the sampling is performed by `sampleLHS()`.

Algorithm 2: Step 1 - Preparation of the variables

Input: Experiment, parameters
Output: Selected set of sensors, parameters of real system; the prior belief of the real system

```
// Define the functions
Function createDataExperiments():
    if experiment == experimentID then
        measurementDimensions = corresponding list of sensors
    return measurementDimensions

Function initializeTotalDistanceDict():
    for sensor in measurementDimensions do
        Create an empty array for storing of the distances
    return totalDistanceDict

Function setGroundTruthParameters():
    for parameter in parameters do
        Set parameter value
        Set parameter unit
        args = parameters
    return args

Function setParticleParamRanges():
    for parameter in parameters do
        Set parameter max and min
        Store max and min in arrays
        minAllowedParams, maxAllowedParams = minArray, maxArray
        spreadOfAllowedParams = maxAllowedParams - minAllowedParams
    return minAllowedParams, maxAllowedParams, spreadOfAllowedParams

Function sampleLHS():
    for particle in N_particles do
        for parameter in parameters do
            sampledParams = LHS sample between minAllowedParams and maxAllowedParams
    return sampledParams

// Call the functions:
createDataExperiments()
initializeTotalDistanceDict()
setGroundTruthParameters()
setParticleParamRanges()
sampleLHS()
```

4.4.2. Step 2 - Running of the real system

After the variables are initialized, the simulation model that represents the real system is executed. The entire real system is simulated before the assimilation algorithm takes place. The states of the model are subsequently saved to the hard drive. The saving of these states to the hard drive has one benefit: this guarantees that each experiment is based on the same ground truth model and eventual adverse effects of randomness are mitigated. For this, four functions are composed. These functions are described presented in pseudocode in algorithm 3.

Algorithm 3: Step 2 - Simulating and measuring the real system

Input: Simulation model, parameters of real system, t, intv, maxSimulationTime
Output: sensorData of real system, ground truth arrivals

```
// Define the functions
Function initializeReal():
    toRun = t + intv
    Create real model with reconstruct = False and groundTruth = True
    Real model parameters = args
    Run Real model to time = toRun
    Interpolate locations of transporters in Real model

Function measureReal():
    for sensor in realSensorData do
        for entity in entities do
            Collect moment in time and entity weight
            Add white Gaussian noise to moment and weight // to mimic sensor imperfections
               of physic systems
    for transporter in realLocationSensorData do
        Collect latitude, longitude, weight and route of transporters
        Add white Gaussian noise to latitude, longitude and weight
    Store all sensorData to hard drive

Function advanceReal():
    toRun = t + intv
    Run model instance to time = toRun
    Interpolate locations of real model instance
Function openRealSensorData():
    Open pickle with the realsensorData corresponding to the current time t

// Call the functions (only if there is no Real data):
if runReal == True then
    initializeReal()
    t = t + intv
    measureReal()
    Clear real sensorData // At each iteration only the sensorData of the current
       time interval is taken into account.
    while t < maxSimulationTime do
        t = t + intv
        advanceReal()
        measureReal()
        Clear real sensorData
    openRealSensorData()
    Store real Terminal arrival dates as .csv // This will allow for the computation of
       accuracy metrics
    t = initialt
```

At first the real model instance is generated by the `initializeReal()` function, this generates an instance of the Salabim simulation model and sets the model parameters to the parameters of the real system

as defined in the `setGroundTruthParameters()` function.

Subsequently the simulation model is executed until the end of the next data assimilation interval. Thereafter the locations of the transporters are interpolated, which is necessary because state changes only happen at events in discrete event simulations (see section 2.2) which means the locations of transporters are only updated at departure and arrival of transporters.

After the interpolation of the locations, the data that is recorded by the various sensors in the Real system are recorded and perturbed by adding white Gaussian noise to mimic the inaccuracies actual sensors have in physic systems. This is done for both the *event-based* sensors as well as the *location* sensors. The perturbed sensor data is stored to the hard drive so this can be opened at any time for an experiment since this is the data that is used to compute the distance metrics between the real system and the particles. After the recording and storage of the sensor data, the dictionaries with sensor data in the simulation model are cleared.

After the initialization the real model is advanced with the function `advanceReal()`, in time steps with the size of the defined assimilation intervals. At each interval the the sensor data is recorded with the function `measureReal()` and subsequently the sensor data is cleared again.

At the end of the simulation, as the *maxSimulationTime* is exceeded, the sensor data is opened with the function `openRealSensorData()`. The sensor data is used to collect all the arrival times at terminal and store these to a .csv file to compute the accuracy metrics after the data assimilation process.

4.4.3. Step 3. - Sampling

In step 3, the actual data assimilation algorithm starts with the *sampling* step. In this step, each of the particles is simulated for the duration of the data assimilation interval.

The implementation of the sampling step has two slightly different forms: 1) A *sampling - initialization* step where the simulation models are generated and run for the *first* data assimilation interval. 2) A *sampling - advance* step where the simulation models are reconstructed based on the previous time step and are run for all assimilation intervals except for the first interval. At initialization, new models are *generated*, for advancing the particles, the models are *reconstructed*.

In the sampling step the system states of each of the particles are transformed over time by the *system transition model*. For DES DA this means running the simulation model that for each of the particles. Because the number of particles N is generally a (very) large number, this imposes a long simulation time.

This is one of the motivations for Salabim, since the execution of simulation models is generally fast in Python and this can be executed on multiple cores by means of MultiProcessing. To facilitate this, two worker functions (*worker_initialize_particles.py* and *worker_advance_particles.py*) are realized which are executed in *MultiProcessing*. In these worker functions the *sampling* step and *intermediate* are deployed which each will be discussed separately. This section focuses *sampling* step.

As one can see in algorithm 4, two versions of the particle are generated at the sampling step. One that is simulated for the duration of the data assimilation interval (p_n) and one for the previously defined duration of the forecast ($p_{forecast}$). The two particles are generated because *all* model states of the particle need to be saved at t_1 to allow for the reconstruction of this particle at the next iteration.

To save the model states the model dictionaries need to be adjusted and therefore the model that is simulated for Δt can not be continued for the duration of $maxForecast$. To combat this problem, a second instance of the particle (referred to as *particle Forecast*) is generated to perform the forecast. This process is visualized in figure 4.7

After the *classic* (see figure 2.4) sampling step is performed by executing the simulation for p_n from t_0 to t_1 all model states are saved (for future reconstruction). Apart from this the *location sensor data* is saved; the location sensor data is the location data of all transporters at t_1.

Subsequently $p_{forecast}$ is generated and simulated for the duration of $maxForecast$ until t_f. After this period, all terminal arrivals (or any other type of forecasted variable) are stored. Similar to the *event-based*

Algorithm 4: Step 3 - Sampling (worker_initialize_particles.py or worker_advance_particles.py)

```
// The first part of the worker function - sampling
Input: pf, t, intv, Model, particleParameterDict, maxForecast
Output: particle states, arrival estimations

// Executed by multiprocessing
Function workerPartOne():
    // Running of particle for the duration of the assimilation interval
    Unwrap args (pf, t, intv, Model, particleParameterDict, maxForecast, etc.)
    if t == 0 + intv then
        Create Particle p_n with reconstruct = False and groundTruth = False // initialization
    if t != 0 + intv then
        Reconstruct Particle p_n with reconstruct = True and groundTruth = False // advance
    p_n_parameters = sampledParameters[particle] // update parameters of model
    Run p_n till t
    Interpolate locations of transporters in p_n
    Save last state of particle as pickle // to allow for reconstruction of the model at a
       later stage

    // Running of a forecast particle
    if t == 0 + intv then
        Create a Particle (p_forecast) model with reconstruct = False and groundTruth = True
           // initialization
    if t != 0 + intv then
        Reconstruct Particle (p_forecast) with reconstruct = True and groundTruth = False // advance
    p_forecast_parameters = sampledParameters[particle] // update parameters of model
    Run p_forecast till t + maxForecast
    Store the estimated arrival times at terminal and the event-based sensor data as a pickle
       // This will allow to open this data in the main process
```

sensor data that is collected over the duration of $maxForecast$.

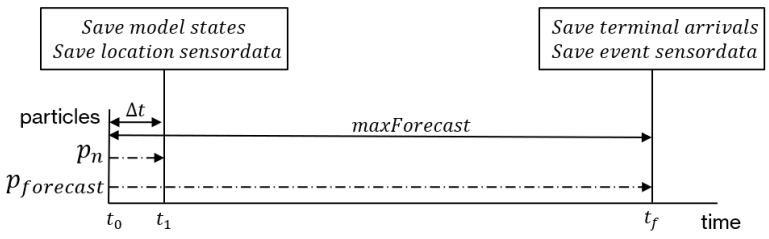


Figure 4.7: Particle and Particle forecast

Arrival sensors

The entities in the simulation model are generated based on the observations of the *arrival sensors*. The concept of an arrival sensor is introduced by Xie (2019) (Road Trajectory model). Arrival sensors are event-based sensors placed at the *system boundaries* of open systems and observe the arrival of entities. In supply chains this can be shipments or transporters.

The benefit of these arrival sensors is that these mitigate the adverse effects of the *variable dimension problem* by helping to accurately estimate the number of entities in open systems Xie2019ASimulations. Next to this, these sensors help to accurately estimate the arrival schedules of the different transporters. An

accurate estimation reduces a large share of the uncertainty in supply chain simulation systems.

In the sampling step, the data points from the arrival sensors is passed to the simulation model in the *realSensorData* argument. Depending on the type of data point (agreed, planned or actual) error terms are added to this data in the *model.readSensorArrivalData* function of the simulation model. Subsequently this data is used to model the arrival of both shipments and transporters in the particles.

4.4.4. Step 4. - Intermediate step

In the fourth step, the relevant system states are transformed into measurements similar to the measurements that are done by the *"sensors"* in the real system. In this so-called *intermediate step*, white Gaussian noise is added in a similar fashion as in the measurement of the real system (`measureReal()`). Just as the sampling step, this is part of the worker function since this has to be performed for each of the particles and therefore is executed by multiprocessing to reduce computing time. For efficiency reasons the distance between the real system and the particles are calculated in the intermediate step.

Algorithm 5: Step 4 - Intermediate step

```
// The second part of the worker function:  the intermediate step
Input: p_n, p_forecast, realSensorData
Output: distance between particles and real system

// Executed by multiprocessing
Function workerPartTwo():
    for observationReal in realsensordata do
        ValueReal, weightReal = observationReal
        if observationReal == event-based then
            Find matching data point (value_RF) based on sensorID  entityID in p_forecast
            if No corresponding observation in particle then
                Value_RF = Missing
            Add white gaussian noise to value_RF
            Calculate absolute distance in time between value_RF & valueReal
            Store distance and weightReal in distance dictionary
        if observationReal == location-based then
            Find matching data point (lat1, lat2) based on sensorID  entityID in p_forecast
            if No corresponding observation in particle then
                Value_RF = Missing
            Add white gaussian noise to lat1 &lat2
            Calculate distance with haversine formula
            Store distance and weightReal in distance dictionary
    for entities in p_n and not in RealSensorData do
        Store as early arrival, distance = max distance // Early arrivals should be punished
    Store distance dictionary as pickle // To make it accessible in the main process
```

After both the simulations of p_n and $p_{forecast}$ are executed, the sensor data resulting from both models is used to compute the distance between each of the data points and the particles. For the event-based sensor data this is the absolute distance in time. For the location-based sensor data this is the absolute distance in location, calculated from the latitude and longtitude by the haversine formula, which takes the curvature of the globe into account (Chopde & Nichat, 2013). Before any distances are calculated, first white gaussian noise is added to the latitude and longitude to mimic the noise sensors in real systems have.

4.4.5. Step 5 - Weight updating and normalization

After the individual distances for each of the measurements are computed, these have to be transformed into one uniform distance which is used to compute weights for the particles. This takes place in the fifth step of the data assimilation algorithm. This step has faced the most adjustments compered to the state-of-the-art algorithms previously used. The adjustments are a solution to the *curse of dimensionality* as discussed in section 2.4.4. Because of these adjustments and the importance of these adjustments, the four different functions that form the basis for this step are discussed into more detail.

Reconstruction of the distances

For each of the particles a set of the distances of each measurement during the data assimilation interval is computed and saved as to the hard drive in the worker function. The goal of the reconstruction step is to generate a uniform set of normalized weights. To this extend, first these different sets should be combined in one overview with *all* distances of all particles. This is executed by the function , described in algorithm 6.

Algorithm 6: Step 5.1 Reconstructing the distances from the worker

Input: particles
Output: Set of particle distances, set of total distances

Function `reconstructTotalDicts()`**:**
- **for** *Particle in N_Particles* **do**
  - Open pickle with set of distances
  - Store set of distances in set of particle distances
  - **for** *distance in set of distances* **do**
    - **if** *sensor of distance in measurementDimensions* **then**
      - Add distance to set of all distances

In this function, two sets of distances are generated. One with the distances sorted per particle and one with all distances measured per sensor. The first set is used for the calculation of the individual particle distances, and the second set is used for the normalization of the measured distances. To perform the normalization, the normalization metrics are computed first.

Normalization of the distances

As thoroughly discussed, some of the experiments embody a high number of sensors which measure event- and location-based data over trajectories with different lengths. Because of the expected implications of the *curse of dimensionality*, fitting a multivariate Gaussian distribution over the various dimensions to determine the particle weights would result in an unreliable outcome. This because the data becomes sparse when the number of dimensions is increased which means the true probability density functions can not be estimated.

Therefore, it is decided to pursue the calculation of the weights by computing a distance metric. However, this cannot be done straightforward. Some of the sensors are therefore expected to have a higher degree of absolute variability and will therefore potentially dominate the distance metrics.

Therefore it is decided to normalize the distances calculated by each of the sensors, a best-practice often applied in other fields of machine learning such as clustering (Ekenel & Stiefelhagen, 2006).

For the normalization it is decided to use *standard normalization*, which makes all data points *mean-centered* and scaled based on the standard deviation. This is done by calculating the standard score (Z) of each distance (x) based on the mean (μ) and the standard deviation (σ) of its corresponding sensor. Since this will eventually be used to compute a total distance the absolute value is taken for the standard scores:

$$Z = \left| \frac{X - \mu}{\sigma} \right| \tag{4.9}$$

In this process, the data is centered (the mean is subtracted) and scaled (divided by the standard deviation). This process and its implications to the distribution of the data is visualized in figure 4.8.

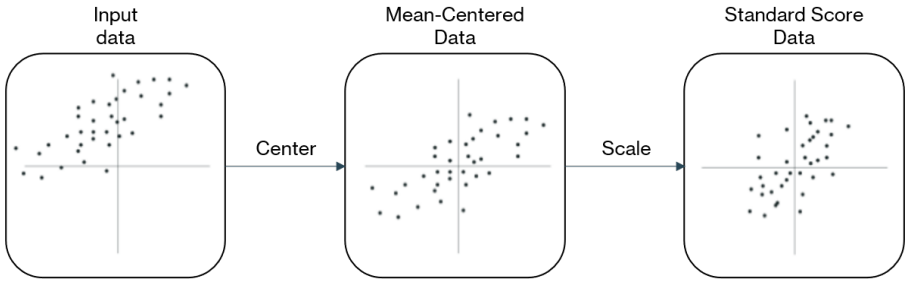


Figure 4.8: Visualization of computation of (non-absolute) standard scores based on Watt et al. (2020)

Unfortunately, unlike in classical machine learning applications where a large training set is available on beforehand, the analytical properties of each of the sensors (mean and standard deviations) can only be retrieved in an iterative fashion. To do this the *simple moving average* (SMA) is deployed, which takes the unweighted average of all distances per sensor to derive the mean and the standard deviation. This is taken care of in the algorithm by the normalizationMetrics function (see algorithm 7).

Algorithm 7: Step 5.2 SMA of normalization metrics

Input: Set of total distances
Output: Set of mean, max, stdev per distance sensor
Function `normalizationMetrics()`**:**
  for *Distance sensor in set of total distances* **do**
    Remove all "missing" values
    Compute mean, stdev and max of set of distances
  return Set with mean, stdev max per distance sensor

After the computation of the normalization metrics, these are used for the actual normalization of the various dimensions of measurements, which will be discussed in the next section.

Computation of weights

Based on the collected measurements on both the real system as the particles and the normalization metrics, the weights of the different particles are calculated. As previously discussed, this is based on a *fractional distance metric* of which the weights are the inverse. To compute the weights two functions are used, which are presented in algorithm 8.

At first, for each of the particles the particle total distance to the observations from the real system are computed. To do so, all standard scores are computed for each of the distances in that dimension are multiplied with the entity weight (see section 4.1). Subsequently these distances per observation are raised to the L'th power (depending on the number that is picked for L to compute the fractional metrics).

Thereafter the total particle distances are computed by summing over the different observations and sensors. The fractional distance is subsequently computed by taking the 1/L'th rooth of the total particle distance. Particle weights are considered to be the inverse of the particle distance, since particles with a *low* distance (high resemblance to the real system) should get assigned a *high* weight. Therefore the weights are computed by dividing one with the fractional distance (as long as the fractional distance is not 0).

To compute the final weights the weights are normalized by dividing them with the total sum of weights to make sure the sum of the total weights adds up to one. This is one with the `normalizeWeights()` function. This pseudocode for this process is given in algorithm 8

Algorithm 8: Step 5.3 Calculation of fractional weights

Input: Set of distances per particle, normalizationmetrics
Output: Normalized weights per particle
Function `computeFractionalWeights()`**:**
- **for** *particle in particles* **do**
  - **for** *dimension in distances belonging to particle* **do**
    - Replace all missing observations with the max observed distance of that dimension
    - Compute standard score (Z) for each of the distances in that dimension
    - Compute distance by multiplying Z-score with entity weight (volume)
    - Raise each distance to the L'th power and sum over the observations
  - Sum over the sensors and take the 1/L'th to calculate the fractional distance
  - **if** *particle Distance > 0* **then**
    - Particle weight = 1 / fractional Distance
  - **if** *particle Distance == 0* **then**
    - Particle weight = 0
- **return** Array of non-normalized particle weights

Function `normalizeWeights()`**:**
- **if** *sum of weights != 0* **then**
  - Normalized weights = weights / sum of weights
- **if** *sum of weights == 0* **then**
  - Normalized weights = 1 / number of particles
- **return** Normalized weights

4.4.6. Step 6 - Resampling

The sixth and last step in the data assimilation algorithm is the *resampling* step. In this step, N offspring samples are generated. The probability of a particle becoming the base of one or more offspring particles is proportional to its previously calculated weight. To preserve variation in the particles, *particle rejuvenation* is applied by making small perturbations into the system states. The process of this is described in algorithm 9.

Algorithm 9: Step 6. Resampling

Input: Particles, particle weights
Output: New, resampled particles
Function `resampleParams()`**:**
- Create a discrete random distribution based on the particle weights
- Draw N new particles form the distribution
- Perform particle rejuvenation by making small perturbations to the particles
- **while** *parameters not in range allowed parameters* **do**
  - Perform particle rejuvenation by making small perturbations to the particles
- **return** resampled particles

In the `resampleParams()` the new particles are drawn from a discrete probability distribution, where the distribution is formed by the particle weights. After the new particles are drawn, small perturbations are made to each of the particles.

An important aspect of this particle rejuvenation is that the perturbations do not result in parameters that are outside of the feasible parameter ranges (e.g., a mode of a distribution smaller than the minimum of the distributions). Therefore the particle rejuvenation is performed as long as the parameters are within the acceptable ranges.

4.4.7. Hyperparameters

For the particle filtering exercise, two parameters are expected to significantly influence the quality of the assimilation algorithm. This is the number of particles N and the length of data assimilation interval Δt. Given the fact that the goal of this research is not to determine the effect and optimal configuration of these parameters, it is decided to select a value for these hyperparameters and keep them constant over all different experiments. The selected values for these hyperparameters is discussed below.

Number of particles N

As discussed in section 2.4.4 a high dimensional state space requires a high number of particles to approximate the posterior distribution over all these dimensions. It is decided to leave the *dynamic particle filtering* out of scope and use a high number of particles instead. Given the computational resources available it is decided to work with 1000 particles.

Length of time interval Δt

The length of a time interval influences the number of dimensions of the distances between the state and measurement vectors (see section 4.1.1) and therefore a smaller time interval could yield more accurate results at a higher computational burden. However, as the S&OP database only contains data points with a granularity of days, the smallest data assimilation interval that can be picked is one day. Therefore Δt is set to one day for all of the experiments.

5
Experimental Setup

This chapter is intended to give the reader insight in how and why the experiments will be performed. As previously explained, the goal of this research is to identify the effects of limited data availability on the accuracy of data assimilation with discrete event simulation.

To serve this goal several experiments are designed, to do so the concept of limited data availability on supply chains is formalized. Based on this, four experiments with different degrees of data availability are designed.

Thereafter, two sets of accuracy metrics will be discussed are used to determine the quality of the data assimilation exercise. Next an experiment to test the effects of randomness on the outcomes of the experiment is presented. Based on this an overview of the different experiments and the intended output of these experiments will be given.

5.1. Experiment input: Data availability

As described in section 2.4.3, more observations generally result in a higher level of accuracy (Cho, 2019). However, as extensively argued in section 4.1, more observations do also result in a higher number of dimensions on which distance metrics should be computed which is expected to result in a lower level of accuracy.

Therefore it is decided to test how different scenarios of data / sensor availability affect the accuracy of the data assimilation exercise. To do this, four different experiments with different levels of *semantic completeness* are designed:

- *Experiment E. 100%* - In this experiment all data points are available. This encompasses all S&OP dates (planned and actual), the carrier data points as well as the vessel locations provided by third parties.
- *Experiment E. 91%* - In this experiment the vessel locations provided by third parties are not available, therefore, these will be omitted from the distance computation. If this experiment would lead to meaningful, accurate results, this would provide valuable insights for the stakeholder. Because this data needs to be acquired and therefore bears costs.
- *Experiment E. 29%* - In this experiment, only a fraction of the carrier data points are available: only the arrival and departure of transport (train and vessels) will be sensed. If this would yield accurate arrival estimations, this would facilitate future implementations of this technique on other (parts) of the supply chains, since less connections to the carrier data need to be made.
- *Experiment E. 25%* - This is the experiment with the lowest level of data availability. In this experiment, only the available S&OP data is assimilated. This group of data points are selected as last remaining data source since these points are crucial for the accuracy metrics (based on the actual terminal arrival) and the entry of entities (AC date). Besides this, this is the easiest to acquire data source for the stakeholder, since this comes from its own data management systems.

5.1.1. Formalization of data availability

The level of data availability relates to the concept of *completeness*; this is *the degree to which a given data collection includes data describing the corresponding set of real-world objects and phenomena* (Huang, 2013). More observations on the operations of the real system indicate a higher level of completeness in this research. To further specify this, two types of completeness are identified.

The first type, *semantic completeness*, does not fit for this research since this is the *degree to which data is of sufficient breadth, depth and scope for the purpose of data use.* And as determining the degree to which the data is of sufficient breadth, depth and scope for the purpose of data use is the goal of this research,

this cannot be determined on beforehand. The second type, *Semantic completeness*, refers to the *degree to which existing values are included in a data collection relevant to the purpose for which the data is stored* (Huang, 2013). Semantic incompleteness is indicated by *missing values* or *missing records*. For the aforementioned experiments, sensors are turned on or off, which results in missing records. Therefore *semantic completeness* is used as definition to formalize the data availability.

No general equation exists to quantify the degree of *semantic completeness*, therefore method to calculate the percentage of semantic completeness in this research is proposed. As literature (i.e., Xie (2019)) shows that a higher number of *observations* generally yields more accurate results, the number of observations for the basis for this equation.

$$Comp_i = 100\% \times \frac{\#\text{observation}_i}{\max\#\text{observations}} \tag{5.1}$$

For each experiment (i) the *rate of semantic completeness* ($Comp_i$) is calculated by dividing the number of observations by the sensors available in the real system for that experiment (#observation $_i$) by the total number of possible observations of all possible sensors in the real system (max#observations). As shown in equation 5.1, the final number is multiplied by 100%.

The different rates of semantic completeness are presented in figure 5.1. The rate of semantic completeness forms the basis for the names of the experiments as well.

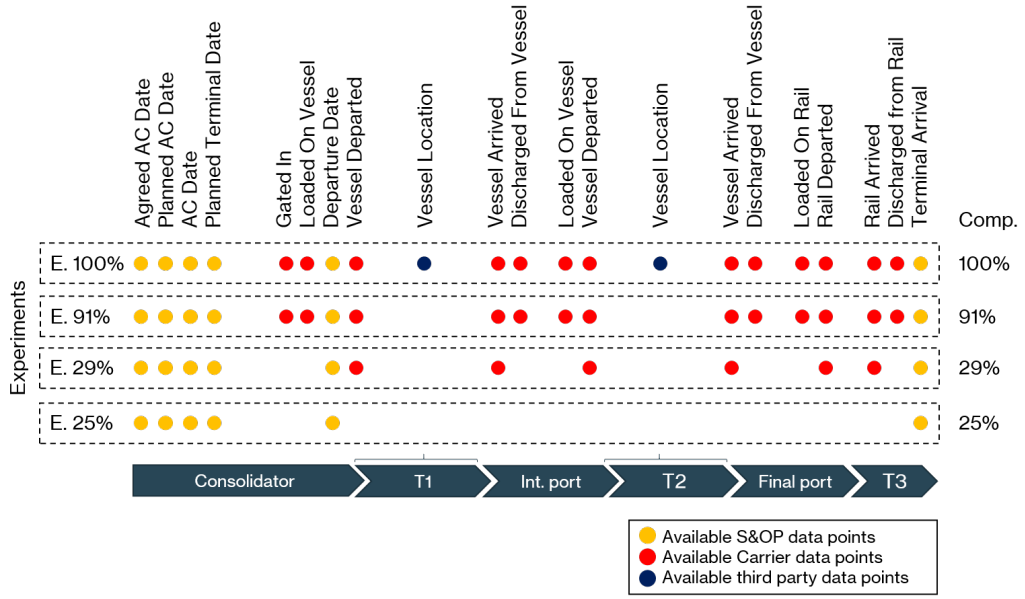


Figure 5.1: Experiments with different rates of semantic completeness

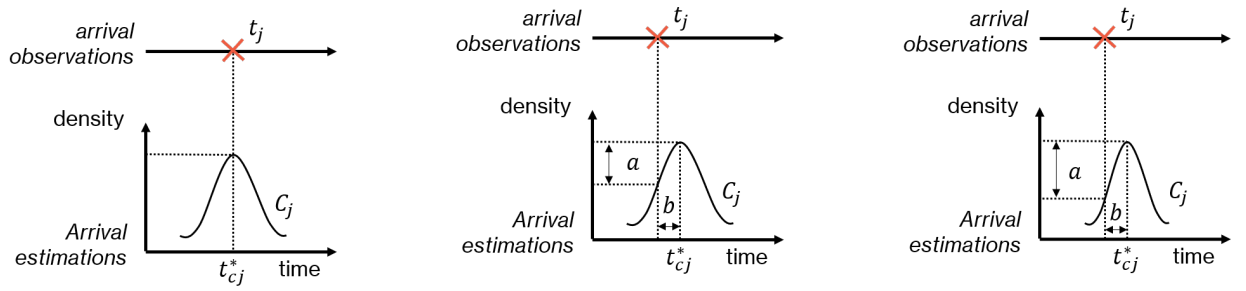
5.2. Experiment output: Accuracy metrics

To determine the effect a reduced level of semantic completeness has on the accuracy of the data assimilation algorithm, the way of determining the accuracy of this algorithm needs to be formalized. As described in section 4.2, the entire distribution of shipment arrivals can eventually be reduced to one estimated day: the point in time where the fitted Gaussian kernel peaks. However, to assess the accuracy of the data assimilation algorithm, it would be foolish to disregard all other information that is embodied by the distribution.

Therefore two characteristics are identified to separate a good estimation by a bad estimation. First, a good estimation is at the right moment in time, i.e., the estimated arrival time is close to the actually observed arrival time. The second characteristic of an accurate prediction is a high proportion of particles that estimated the actual observed arrival time indicated by the height of the probability density function at the observed time of arrival.

5.2.1. Sets of accuracy metrics

To capture these characteristics, two accuracy metrics that previously have been used by Xie (2019) are adopted. To explain the usefulness of these two metrics, three different combinations of observed arrival times and arrival estimations are displayed in figure 5.2.



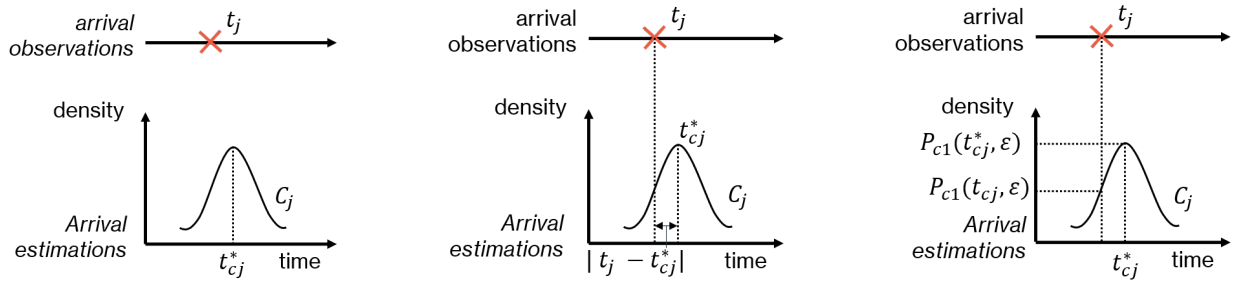
(a) The perfect estimation

(b) Inaccurate estimation with a wide distribution

(c) Inaccurate estimation with a narrow distribution

Figure 5.2: Visualization of different arrival estimations

Figure 5.2a shows the perfect estimation where t_j and $t^*_{c_j}$ are at the exact same instant. If t_j and $t^*_{c_j}$ are not at the exact same instant, the accuracy of the estimation can vary over two dimensions: Relative difference in probability density at t_j versus $t^*_{c_j}$ (displayed with *a)*. And absolute difference in time between t_j and $t^*_{c_j}$ (displayed with *b*). The smaller these differences, the better the estimation.



(a) t_j and $t^*_{c_j}$

(b) $\left|t_j - t^*_{c_j}\right|$

(c) $P_{c_j}\left(t_j, \varepsilon\right)$ and $\left(t^*_{c_j}, \varepsilon\right)$

Figure 5.3: Visualization of arrival estimation accuracy metrics based on Xie (2019)

5.2.2. Aspects of accuracy metrics

To unify the two key characteristics of the accuracy of an *individual cluster of shipment arrival estimations* to a set of clusters resulting from an experiment, two aspects are considered:

- The average accuracy of a set of clusters of shipment arrival estimations should be high.
- The spread in accuracy of a set of clusters of shipment arrival estimations should be low. If this spread is low, the accuracy of the different shipments arrival estimations have a high degree of similarity is a sign of *robustness*.

To embody these aspects in the final outcomes of the experiments, two groups of metrics are designed: First, the *weighted mean based metrics* that capture the average accuracy of a set of clusters of shipment arrival estimations resulting from an experiment. Second, the robustness of the algorithm which is indicated by the *standard deviation based metrics* that capture the spread in accuracy between a set of clusters of shipment arrival estimations.

Weighted mean based metrics

To embody the average accuracy of a set of clusters of shipment arrival estimations, two measures are adopted from the works of Xie (2019). The parameters that form the composition of these metrics are visualized

in figure 5.3. However, the metrics from Xie (2019) are slightly adjusted. As each shipment can contain a different volume of goods, each shipment has a different relative importance. Therefore the volume of goods per shipment (V_j) is used to make this a weighted average based on the versus the total volume V. This eventually leads to the first two accuracy metrics:

- The *weighted mean absolute distance* between the observed arrival (t_j) to the peak of the arrival estimation probability density function ($t^*_{c_j}$). Visualized in figure 5.3b

$$\text{weighted mean absolute distance } = \frac{1}{n_m \times \bar{V}} \sum_{j=i_1}^{i_{nm}} \left| t_j - t^*_{c_j} \right| \times V_j \tag{5.2}$$

- The *weighted mean percentage* of the probability density function at t_j versus the peak of the probability density function at $t^*_{c_j}$. Visualized in figure 5.3c

$$\text{weighted mean percentage} = 100\% \times \frac{1}{n_m \times \bar{V}} \sum_{j=i_1}^{i_{nm}} \frac{P_{c_j}\left(t_j, \varepsilon\right)}{P_{c_j}\left(t^*_{c_j}, \varepsilon\right)} \times V_j \tag{5.3}$$

Standard deviation based metrics

To capture the spread of the accuracy of a set of clusters of shipment arrival estimations, two new measures are developed based the *weighted absolute distance* and the *weighted percentage*. For both metrics, the standard deviation of the weighted observations is calculated. The weights of the observations are considered since it is desirable to have a high degree of similarity between the shipments with the largest volume. This results in the two additional accuracy metrics:

- The *weighted standard deviation of the absolute distances* between the observed arrival (t_j) to the peak of the arrival estimation probability density function ($t^*_{c_j}$). This is computed by the following equation (National Institute of Standards and Technology, 1996):

$$\text{weighted standard deviation of the absolute distances} = \sqrt{\frac{\sum_{i=1}^{n_m} V_j \left(absD_j - WM\_absD\right)^2}{\sum_{i=1}^{n_m} V_i}} \tag{5.4}$$

- The *weighted standard deviation of the mean percentages* of the probability density functions at t_j versus the peak of the probability density functions at $t^*_{c_j}$ computed by the following equation:

$$\text{weighted standard deviation of the mean percentages} = \sqrt{\frac{\sum_{i=1}^{n_m} V_j \left(p_j - WM\_p\right)^2}{\sum_{i=1}^{n_m} V_i}} \tag{5.5}$$

5.2.3. Timeliness of estimations

The third component that needs to be taken into account when considering the accuracy of shipment arrival estimations, is the *timeliness of data*. A correct estimation two days in advance is worth less than an estimation a couple of weeks in advance when considering decision-support since it takes time to adjust the operations of an organization. The four previously described metrics are calculated for four different degrees of timeliness to give insight in this dynamic: Based on the clusters of arrival estimations 30, 14, 7 and 3 days before the observed arrival date.

The four indicated clusters are selected *after* the data assimilation exercise, the procedure for this is presented below. Over the course of the simulation, after each time interval, for each of the shipments, a cluster of shipment arrival estimations will be composed which will form a probability density function. Each of these probability density functions will be stored from the moment the arrival at the consolidator is observed (t_c) until the moment of observed arrival at the terminal (t_j). This makes the number of clusters of shipment arrival estimations (nc) for one shipment: the duration of a shipment in the system divided by the length of a assimilation time interval (Δt):

$$nc = \frac{t_j - t_c}{\Delta t} \tag{5.6}$$

The total data set with clusters subsequently exists of m sets of clusters, each consisting of nc different estimations of arrival times: $\left\{\left\{C_c \mid C_c = \left\{t_1^c, t_2^c, \ldots, t_{n_c}^c\right\}\right\}_{\Delta t}^{nc}\right\}_{c=1}^{m}$.

To determine the final accuracy metrics for a specific shipment, four different clusters with arrival estimations are. Each cluster represent a different degree of timeliness These clusters will be selected based on the number of days before the observed arrival t_j:

- $C_{j,t_{-30}}$: The cluster of estimated arrivals 30 days (t_{-30}) before the actual shipment arrival
- $C_{j,t_{-14}}$: The cluster of estimated arrivals 14 days (t_{-14}) before the actual shipment arrival
- $C_{j,t_{-7}}$: The cluster of estimated arrivals 7 days (t_{-14}) before the actual shipment arrival
- $C_{j,t_{-3}}$: The cluster of estimated arrivals 3 days (t_{-3}) before the actual shipment arrival

In the remainder of this document these different clusters will be referred to as different *degrees of timeliness*. A *high degree of timeliness* indicates a long period between the day of arrival estimation and the observed arrival date. A *low degree of timeliness* indicates a short period between the day of arrival estimation and the observed arrival day. This is visualized in in figure 5.4

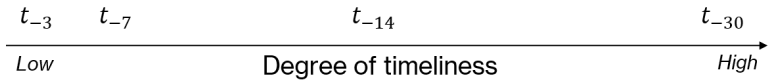


Figure 5.4: Different degrees of timeliness

5.3. Overview of experiments

To perform the experiments a historical three-month period (90 days) will be selected (jan - march 2019). For this period, the data assimilation algorithm will be tested for the different degrees of semantic completeness. Three groups of experiments are identified:

5.3.1. Experiments testing for semantic completeness

At first, four different experiments will be set up to analyze the effect of different degrees of semantic completeness. This will result in four different experiments in which each the data assimilation is ran for a 90 day period starting from the first of January 2019 for different levels of data availability (see figure 5.1).

Over the course of these 90 days, forecasts will be made on the arrival of shipments resulting in the previously discussed data set with clusters of shipment arrival estimations. Based on this data set, the four accuracy metrics will be calculated for four different degrees of timeliness. The annotated code for the data assimilation algorithm used and the execution of the experiments can be found in *Experiment.py*. The code used to calculate the accuracy metrics can be found in notebook *09 Assessment of accuracy.ipynb*.

5.3.2. Base case without assimilation

Apart from the different experiments with different degrees of data availability, one more experiment is designed. This is referred to as the *No Assimilation* experiment (*E. NA*). This experiment is used to determine the predictive capabilities of the simulation model without any form of data assimilation will.

In this experiment, 1000 (same as the number of particles N) simulation models are simulated for 90 days under different combinations of parameters. These parameters have been sampled by a Latin Hypercube sampling over the parameter ranges (i.e., the prior belief on the system). This is comparable to the outcomes of an exploratory modeling exercise. The execution of this experiment can be found in notebook *08 Forecasts without assimilation.ipynb*.

For each of these simulation models, the arrival times of shipments at the consolidator are recorded and clustered as well. Subsequently, fitting a Gaussian kernel to the weighted distributions of the arrival estimations, an estimated arrival day is determined for each shipments.

Based on the estimated and the observed arrival dates the aforementioned accuracy metrics are assessed. These will form a reference point for the increase of accuracy from a prior belief to a posterior belief

by data assimilation, to determine if performing the data assimilation exercise had a beneficial result. The code for this analysis can be found in notebook *09 Assessment of accuracy.ipynb*.

Random seed tests
Since the resampling steps in the data assimilation algorithm involve stochastic sampling, the effect of random seeds should be tested to say something about the validity of the results. To do so the first experiment (experiment *E. 100%*) will be executed with four different random seeds. Subsequently the previously described accuracy metrics will be calculated for each of these random seeds. The outcomes will subsequently be compared by a paired t-test to see if stochasticity significantly influences the outcomes of the data assimilation exercise. The annotated code for the data assimilation algorithm used and the execution of the experiments with different random seeds can be found in *Experiment2_random_seeds.py*

5.4. Presentation of results

To summarize, four experiments with different degrees of data availability will be performed, supplemented by an additional *base case* experiment without data assimilation. The results of these experiments will be presented together, both in tabular form as by a graphical representation. The tabular forms will be discussed in the following section. Thereafter, the form outcomes of the *random seed tests* will be discussed.

Taking into consideration the existence of

- Two different sets of accuracy metrics (*weighted absolute distance* and *weighted percentage*)
- Each having two different aspects (*weighted mean* and *weighted standard deviation*)
- Computed for four different degrees of timeliness (t_{-3}, t_{-7}, t_{-14} and t_{-30})
- Tested for five different experiments of data availability (*no assimilation*, experiment *E. 100%*, experiment *E. 91%*, experiment *E. 29%* and experiment *E. 25%*)

Eventually results in 80 different results that are used to compare the different experiments. These results will be presented by six tables. First, one table is used to give insight the effect of assimilation and subsequently the different experiments with data assimilation are compared in four tables; each for one set of accuracy metrics.

To assess the effect of data assimilation, the experiment without data assimilation is compared to the four experiments with data assimilation. This comparison is based on the estimations with the highest degrees of timeliness (t_{-30}). All four accuracy metrics are presented in the columns and the five different experiments are presented as rows. An empty example of this table is presented in table 5.1.

	Absolute Distance		Percentage	
	Weighted mean	Weighted standard deviation	Weighted mean	Weighted standard deviation
Experiment NA				
Experiment 100%				
Experiment 91%				
Experiment 29%				
Experiment 25%				

Table 5.1: Example table for the presentation of the effect of data assimilation on the absolute distance and percentage

Because the comparison of the different experiments with different levels of data availability is performed for each of the levels of data availability, four tables will be used to present these results of these experiments. These tables are grouped based on the set of accuracy metrics they represent: one set of two tables for the *weighted percentage* and one set for the *weighted absolute distance*. The tables will have the form as presented in table 5.2.
In these table, the different degrees of timeliness are presented as columns and the different experiments as rows.

The second set of experiments refers to the output of the random seed tests. Here for one experiment (*E. 100%*), four experiments with different random seeds will be computed. The accuracy metrics of random

	Weighted mean absolute distance			
	t_{-30}	t_{-14}	t_{-7}	t_{-3}
Experiment E. 100%				
Experiment E. 91%				
Experiment E. 29%				
Experiment E. 25%				

	Weighted standard deviation absolute distance			
	t_{-30}	t_{-14}	t_{-7}	t_{-3}
Experiment E. 100%				
Experiment E. 91%				
Experiment E. 29%				
Experiment E. 25%				

Table 5.2: Example table for the presentation of a set of accuracy metrics for the different experiments

seed 2, 3 and 4 will be compared to the accuracy metrics of random seed 1 (which have been used for the previous outcomes). The comparison will based on an independent t-test, because it can not be assumed that each shipment will have the same accuracy, the overall accuracy is relevant in this experiment.

For each of the experiments the computed p-value for the set of computed *absolute distances* (p_{absD}) and the p-value for the set of computed *percentages* (p_p) will be presented. This will be computed for each degree of timeliness (presented in the column). An example table is presented by table 5.3

	t_{-30}		t_{-14}		t_{-7}		t_{-3}	
	p_{absD}	p_p	p_{absD}	p_p	p_{absD}	p_p	p_{absD}	p_p
Seed 1 vs 2								
Seed 1 vs 3								
Seed 1 vs 4								

Table 5.3: Example table for the presentation of the results of the random seed tests

Apart from these tabular representations, the results of the experiments will be presented visually as well. The weighted accuracy per shipment will be part of these visualizations to support further analysis of the results presented in the tables.

6
Results

The results of the different experiments are presented and analyzed in this chapter. The effect of data assimilation is presented first, thereafter the effect of limited data availability on accuracy of the shipment arrival estimations is discussed. At last the results of the random seed test are presented.

Each of the sections starts with a presentation of the aggregated results which are subsequently analyzed based on the accuracy of the arrival estimations of separate shipments. The code for the analysis analysis in this chapter can be found in notebook *09 Assessment of accuracy.ipynb* and notebook *11 Random seed analysis.ipynb*.

6.1. Overview of experiments

The different experiments with different levels of data availability are discussed first to give the reader a good understanding of how the results relate to the different levels of data availability. Four experiments with data assimilation are designed to test the effect of different levels of data availability. These experiments are presented in figure 6.1 and labeled based on the level of semantic completeness (percentage of available observations).

Experiment *E. 100%* is used as reference point with all data sources used for observations. Experiment *E. 24%* is the experiment lowest level of data availability with only 24 percent of the data points available. Apart from these four experiments a fifth experiment is performed without data assimilation, this experiment is referred to as *E. NA* (No Assimilation).

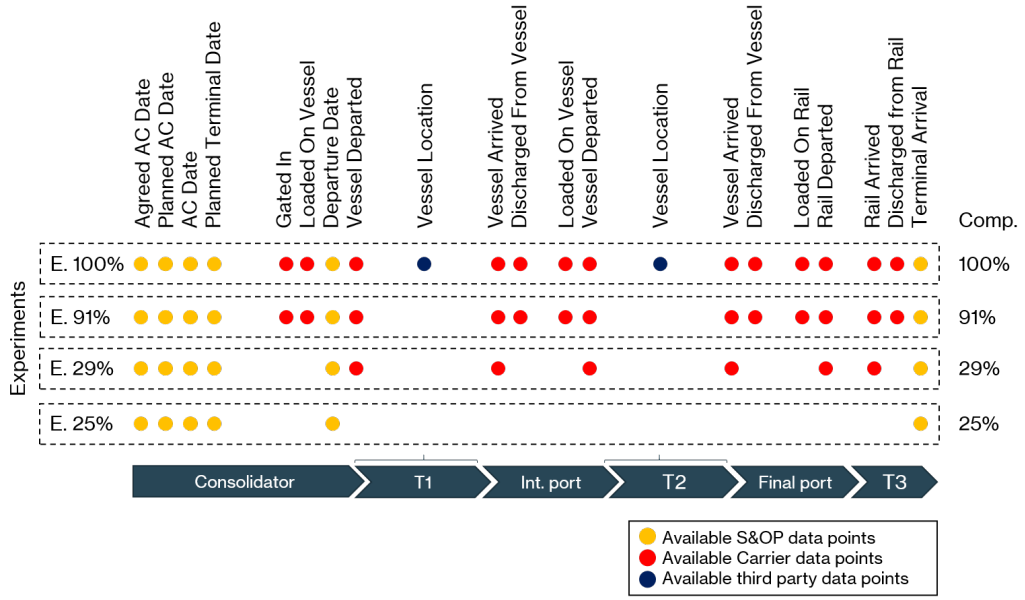


Figure 6.1: Experiments with different levels of semantic completeness

The four metrics to assess accuracy of the shipment arrival estimations over a set of shipments are discussed in this section as well. The formalization of, and the argumentation for these metrics is extensively discussed in section 5.4. This section serves as a brief summary to increase the interpretability of the results.

An accurate estimation of the arrival of a shipment is indicated by two metrics. First, the estimated arrival date should have a low *absolute distance* (in days) to the observed arrival date. Second, a high share of the particles should have estimated the observed future event right, which is indicated by a high *percentage*.

Over a set of arrival estimation, the accuracy over the different shipments should follow two patterns. First, the *average* accuracy over the different shipments should be high. Second, there should be little variation between the accuracy of the different shipments (i.e., the algorithm should be robust), which is indicated by a low *standard deviation*.

Because the shipments represent different volumes, each shipment is assigned a weight proportional to the volume of the shipment. The data assimilation algorithm is intended to prioritize shipments with a higher weight. Therefore, the weight of the shipments is part of the accuracy metrics as well. For each set of shipments, the *weighted average* and the *weighted standard deviation* are calculated for both accuracy metrics ofindividual shipments. This results in four accuracy metrics that are presented in table 6.1.

	Weighted Mean	**Weighted Standard Deviation**
Absolute Distance	Weighted mean absolute distance (low)	Weighted standard deviation of the absolute distances (low)
Percentage	Weighted mean percentage (high)	Weighted standard deviation of the percentages (low)

Table 6.1: Accuracy metrics for shipment arrival estimations by DES DA. Desired value between brackets.

For the estimation of a future event it is also highly relevant *when* this estimation is done. Therefore, the four accuracy metrics are collected for four different *degrees of timeliness*. The closer to the observed event an estimation is performed the lower the degree of timeliness of this estimation. The four degrees of timeliness used in this research are represented in figure 6.2.

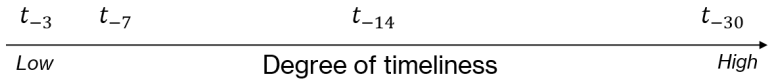


Figure 6.2: Different degrees of timeliness

6.2. Effect of data assimilation

To identify the effect of data assimilation, the experiment without data assimilation (*E. NA*) is compared to the four different experiments with data assimilation. This comparison is based on the *weighted mean* and the *weighted standard deviation* of the *absolute distance* and the *percentage*. For each of the experiments with data assimilation, the arrival estimations with a high degree of timeliness (t_{-30}) are used. The results of the experiments are presented in table 6.2.

	Absolute Distance		Percentage	
	Weighted mean	Weighted standard deviation	Weighted mean	Weighted standard deviation
Experiment NA	8.5	9.6	60.2%	38.9%
Experiment 100%	4.7	4.0	70.7%	23.8%
Experiment 91%	4.7	3.4	61.6%	23.0%
Experiment 29%	4.7	5.2	67.7%	18.8%
Experiment 25%	4.4	2.5	66.3%	21.1%

Table 6.2: Effect of data assimilation on the absolute distance and percentage

To confirm that that data assimilation increases the accuracy of arrival estimations, three aspects of increasing accuracy should be visible in the data: First the *weighted mean absolute distance* should be re-

duced, second the *weighted mean percentage* should be increased, and third the *weighted standard deviation* of both the absolute distance and the percentage should be reduced. All trends are clearly visible in table 6.2 and will be analyzed below.

6.2.1. Analysis of the effect of data assimilation

To understand the aggregated results presented in table 6.2, the accuracy scores *per shipment* are visualized in figure 6.3 and 6.4. Analyzing these figures helps to understand the effect of data assimilation.

The accuracy scores for the different shipments are shown in dark blue; the weight of the shipment is indicated by the size of the scatters. The aggregated accuracy metrics per experiment are shown as well. The weighted mean is shown by the yellow crosses and the weighted standard deviation is represented by the interquartile ranges visualized by box plots.

In the following sections both figures will be analyzed to understand the three different aspects of increasing accuracy presented in table 6.2.

Weighted absolute distance

The reduction of both the *weighted mean* and the *weighted standard deviation* of the absolute distances per shipment is clearly visible in figure 6.3. The weighted means of the absolute distances are lower for each of the experiments with data assimilation than the experiment without data assimilation.

The reduction of the weighted standard deviation is also clearly visible when comparing the box plots. Figure 6.3 shows that the interquartile ranges of the weighted absolute distances are (much) smaller for the experiments with data assimilation than the interquartile ranges of the experiments without data assimilation.

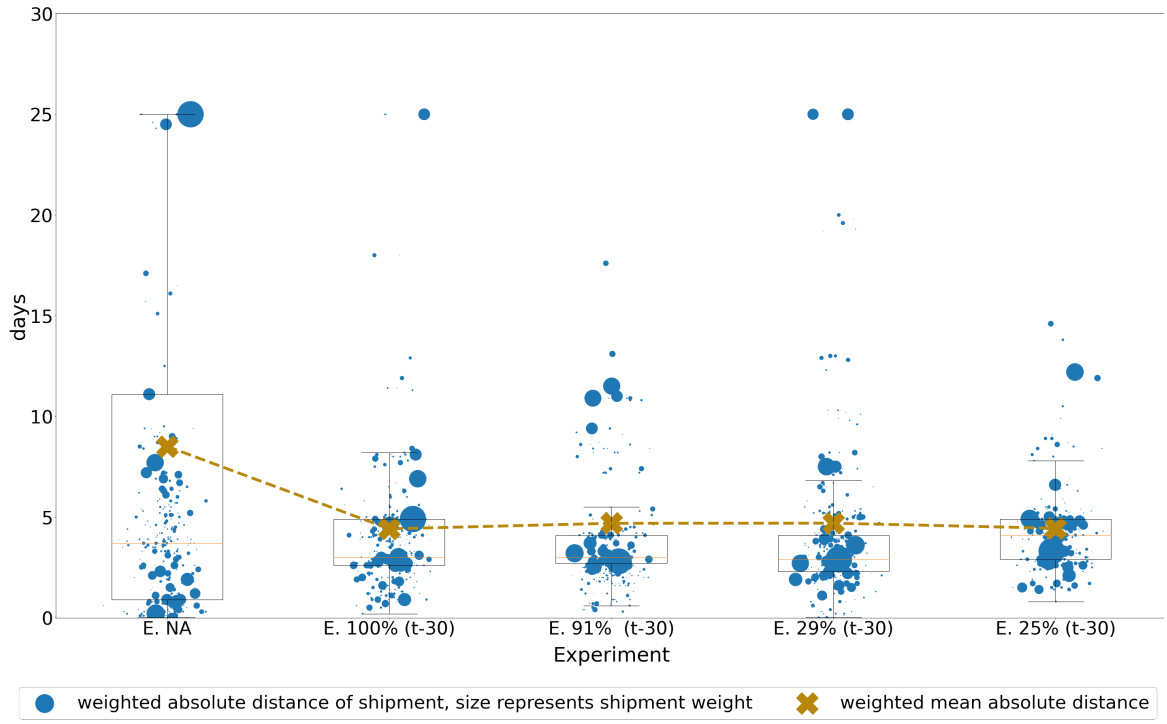


Figure 6.3: Effect of data assimilation on the weighted absolute distance

A detailed inspection of the absolute distance per shipment in figure 6.3 shows that the reduction of both the *weighted mean* and the *weighted standard deviation* of the absolute distances should be contributed to three effects.

First, the *absolute number of outliers* is reduced by data assimilation. Each of the experiments with data assimilation (e.g., E. 100%) have less shipments with an absolute distance of more than 20 days.

Second, the *weighted number of outliers* is reduced by data assimilation. For each of the experiments with data assimilation, the shipments with the highest weight have a relatively low absolute distance. While

for the experiment without data assimilation, the outliers are the shipments with a high weight. This shows that the data assimilation algorithm is able to *prioritize* the shipments with a high weight (volume).

Third, the *spread in absolute distances* of the different shipment arrival estimations is reduced. Figure 6.3 clearly shows that the absolute distances of the shipments are closer together for the experiments with data assimilation than the experiment without data assimilation.

Weighted percentage

The similar desired effects of data assimilation on the weighted percentage are shown in figure 6.4. Data assimilation increases the *weighted mean* of the percentages. This indicates that on average, there was a higher confidence in the observed arrival day (i.e., more simulation models predicted the right arrival day).

The smaller interquartile ranges for the experiments with data assimilation indicate that the *weighted standard deviation* of the percentages of the different shipment is reduced.

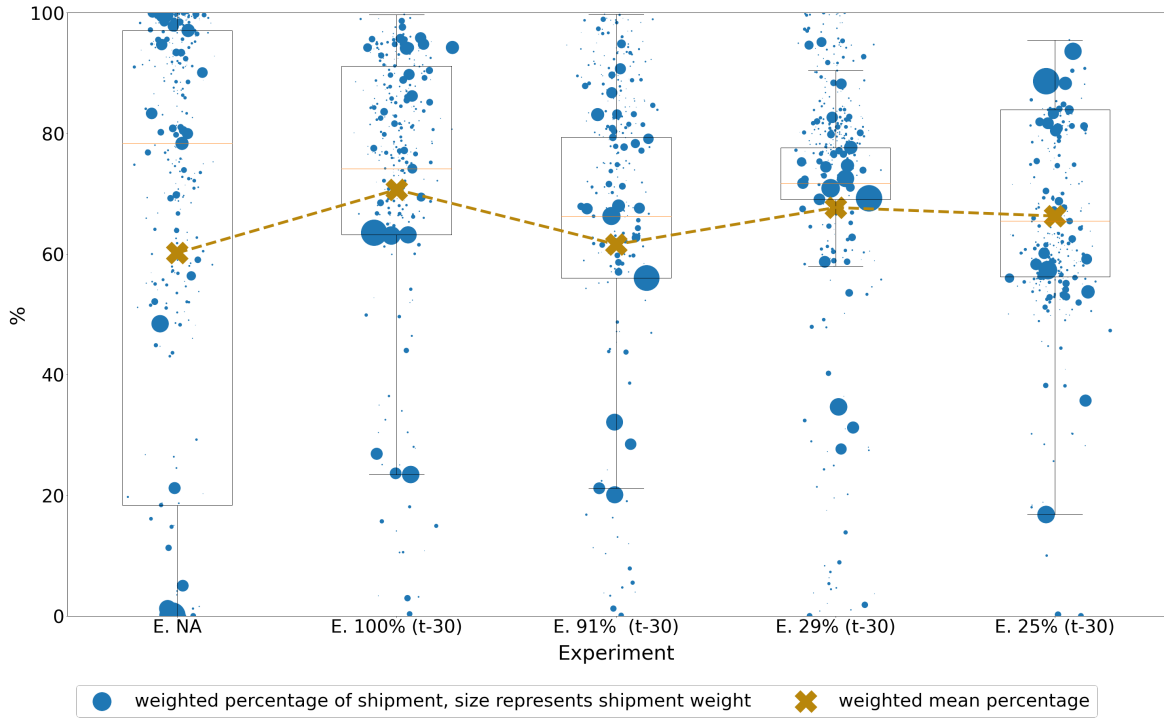


Figure 6.4: Effect of data assimilation on the weighted percentage

Analysis of figure 6.4 shows similar effects on the percentage per shipment as for figure 6.3. Data assimilation reduces the *absolute number of outliers* as well as the *weighted absolute number of outliers*. Figure 6.4 clearly shows a decrease of (large) shipments for which the observed arrival date was completely unexpected (i.e., a percentage smaller than 10 percent).

At last the percentages of the different shipments are closer together for the experiments with data assimilation. The smaller interquartile ranges show that the *spread in percentages* is decreased.

6.2.2. Summary of the effect of data assimilation

Based on the previous sections it is concluded that data assimilation makes shipment arrival estimation more accurate as all the results four accuracy metrics changed in the desired direction. This is the result of three effects that separate the experiments with data assimilation from the experiment without data assimilation: First, data assimilation reduces the absolute number of outliers. Second, data assimilation is able to prioritize shipments with a high weight, and third, data assimilation reduces the spread between the different shipments.

Important to note is that for this analysis only the estimations with the largest degree of timeliness (t_{-30}) are considered. Therefore the effect of different degrees of timeliness will be discussed in the remainder of this chapter.

6.3. Effect of limited data availability on the accuracy of data assimilation

Now the positive effect of data assimilation is identified, the remainder of this chapter will focus on a comparison of the four different experiments with data assimilation. Each experiment is performed for a different level of data availability (semantic completeness) and accuracy metrics are collected for four different degrees of timeliness. The effect of limited data availability on the weighted absolute distance will be discussed first, after which the effect on the weighted percentage will be analyzed.

6.3.1. Weighted absolute distance

The results of the different experiments regarding the weighted absolute distance are shown in table 6.3. Four different trends are visible in this table:

First, the relation *more observations equals more accurate results* does not hold. The experiment with only 29 percent of the observations clearly outperforms the experiment with 91 percent of the observations. This implies that some data points are more important than others.

Second, no clear relation between data availability and robustness is identified. The weighted standard deviation differs per experiment and degree of timeliness.

Third, a lower degree of timeliness (e.g., t_{-3}) does not always result in more accurate arrival estimations. The experiments with 91 and 25 percent of the observations available show an opposite trend: A lower degree of timeliness yields a higher weighted mean absolute distance.

Fourth, a reducing degree of timeliness does decrease the weighted standard deviation of the absolute distances for almost every experiment. Thus, the closer to the actual arrival date an arrival estimation is produced, the more robust this estimation is.

	Weighted mean absolute distance			
	t_{-30}	t_{-14}	t_{-7}	t_{-3}
Experiment E. 100%	1.6	1.6	1.0	1.0
Experiment E. 91%	1.7	1.7	2.4	2.3
Experiment E. 29%	2.0	2.0	1.1	0.9
Experiment E. 25%	4.3	4.3	5.9	10.3

	Weighted standard deviation absolute distance			
	t_{-30}	t_{-14}	t_{-7}	t_{-3}
Experiment E. 100%	4.0	2.3	1.2	1.4
Experiment E. 91%	3.4	1.6	2.5	2.3
Experiment E. 29%	5.2	2.9	1.3	0.6
Experiment E. 25%	2.5	0.9	1.3	1.1

Table 6.3: Effect of limited data availability on weighted absolute distance

To understand the four identified trends, the data behind the aggregated accuracy metrics is further analyzed. For each degree of timeliness, the weighted absolute distances per shipment are compared based on the visualization presented in figure 6.5. The shipments are shown as dark blue scatters; the size of the scatter represents the weight (volume) of the shipment and the absolute distance is represented by its position on the y-axis. The weighted mean of the shipments is presented by a yellow cross.

Because it is difficult to interpret the weighted standard deviations of the different shipments solely based on the scatters of the shipments, this is represented by the interquartile ranges of the (weighted) box plots. A larger interquartile range implies a higher weighted standard deviation.

Analysis of trend 1: The effect of data availability on the weighted mean absolute distance

The first trend observed in table 6.3 is clearly visible in figure 6.5: More data does not necessarily result in a lower weighted absolute distance (i.e., more accurate results). This is concluded based on the fact that experiment *E. 29%* clearly outperforms experiment *E. 91%* (with respect to the weighted mean of the absolute distances) wile using only one third of the observations.

This is an unexpected trend compared to literature, therefore more understanding on the relation between data availability and the quality of data assimilation is needed. This understanding is provided by an analysis of the clusters of arrival estimations provided by the particles. A clusters of arrival estimations is presented by weighted histograms of the arrival estimations per particle, to which a Gaussian kernel is fitted (for more information see chapter 5).

These weighted histograms and fitted Gaussian kernels are visualized for four example shipments (one per experiment). These examples are representative for the sets of shipment arrival estimations in each experiment. Extra weighted histograms are presented in appendix C.

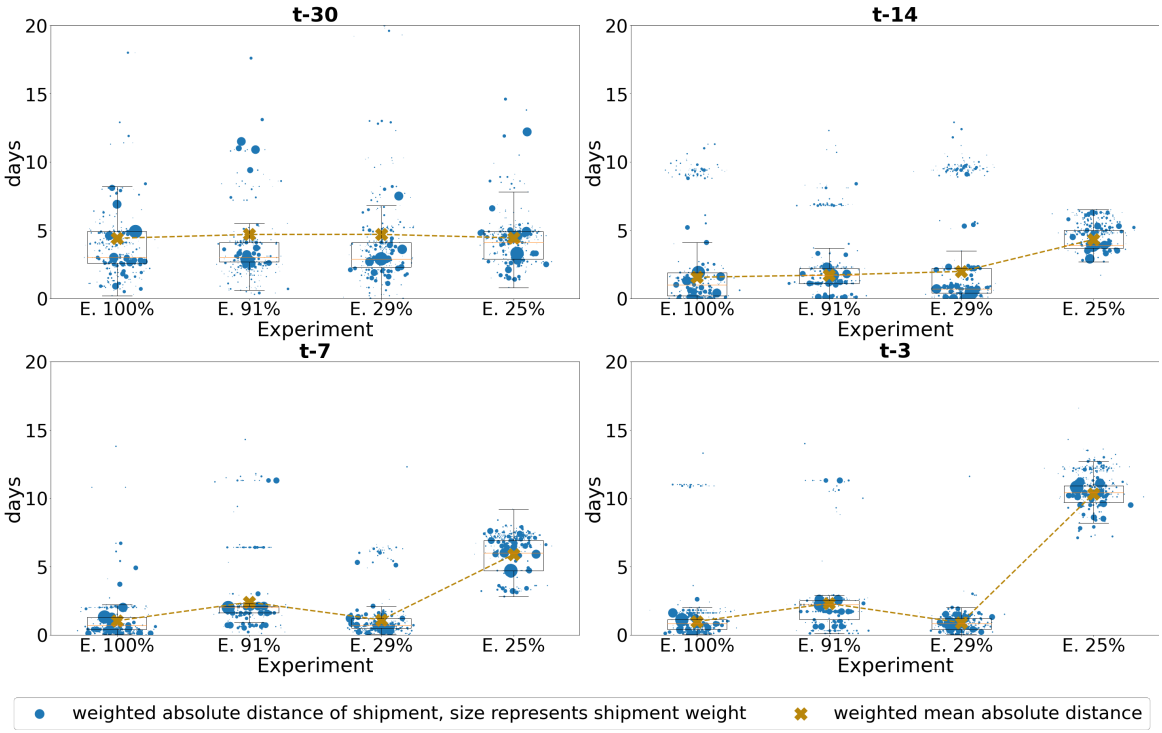


Figure 6.5: Spread of weighted absolute distance per degree of timeliness

First the weighted histograms and fitted Gaussian kernels of experiment *E. 100%* are compared to experiment *E. 91%*. The only difference between these experiments in terms of data availability is the omission of the *third party data points* (see figure 6.1).

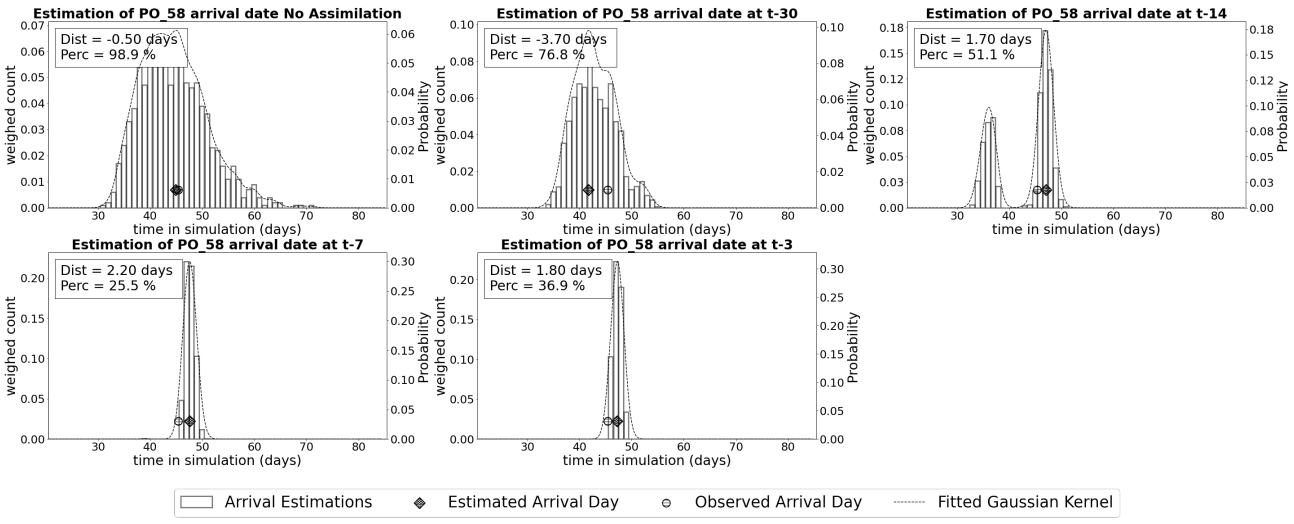


Figure 6.6: Weighted histograms of arrival estimations of particles of shipment PO_58, experiment *E. 100%*

The comparison of the weighted histograms and the fitted Gaussian kernels presented in figure 6.6 and 6.7 show two factors:

First, the data assimilation algorithm is able to reduce the spread in the arrival estimations of the different particles for both experiments. As one can clearly see in figure 6.6 and 6.7, as the degree of timeliness is reduced, the estimations of the particles become increasingly which makes the fitted kernels more narrow.

Second, a difference between the two experiments is visible for estimations with a lower degree of timeliness. Here both peaks become narrow, but the peaks in experiment *E. 91%* have a consistent higher absolute distance between the estimated arrival day and the observed arrival day than the peaks of experiment *E. 100%*.

This indicates that the data points lost between the two experiments play an important role in increas-

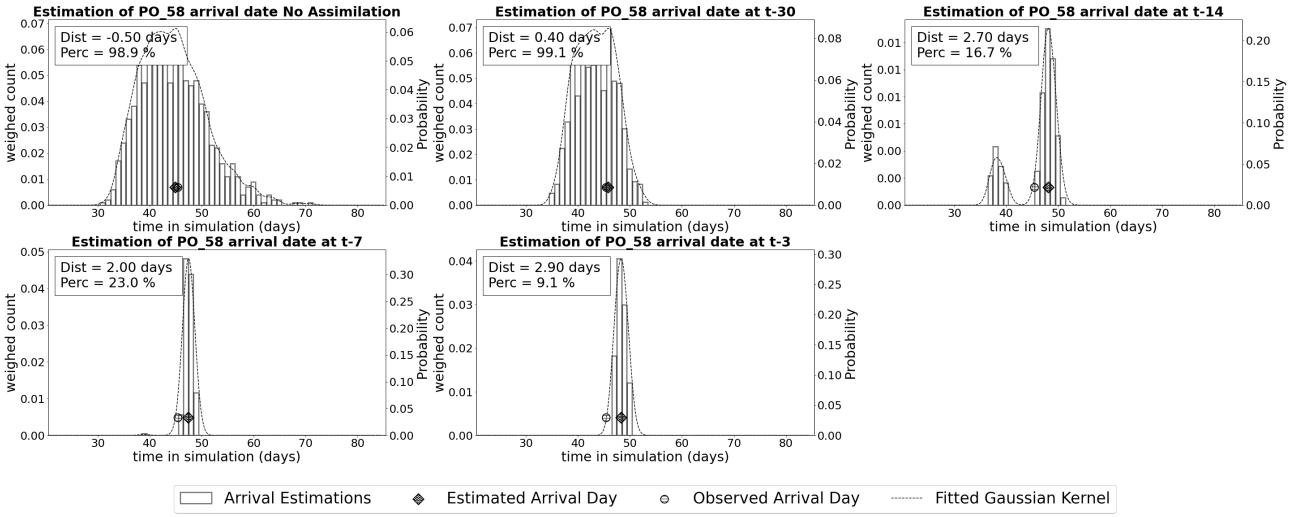


Figure 6.7: Weighted histograms of arrival estimations of particles of shipment PO_58, experiment *E. 91%*

ing the accuracy of the estimations in the final steps of the supply chain. This is also visible in figure 6.5: shipments have a higher absolute distance in experiment *E. 91%* compared to experiment *E.100%*

Second, the weighted histograms of experiment *E. 29%* (figure 6.8) are compared to those of the two aforementioned experiments. The shipment arrival estimations of the different particles in this experiment show similar behavior to those of *E. 100%*. As the degree of timeliness is reduced, the kernels presented in figure 6.8 narrow down with a high level of precision and the absolute distance between the estimated and observed arrival dates is reduced. This is in line with the results presented in figure 6.5

The fact that the absolute distance is consistently lower for shipment arrival estimations in experiment *E. 29%* than *E. 91%* indicates that the observations omitted between these two experiments are not crucial for the accuracy of the shipment arrival estimations. The opposite is true, these data points have a detoriating effect on the accuracy and should be identified as *noise*.

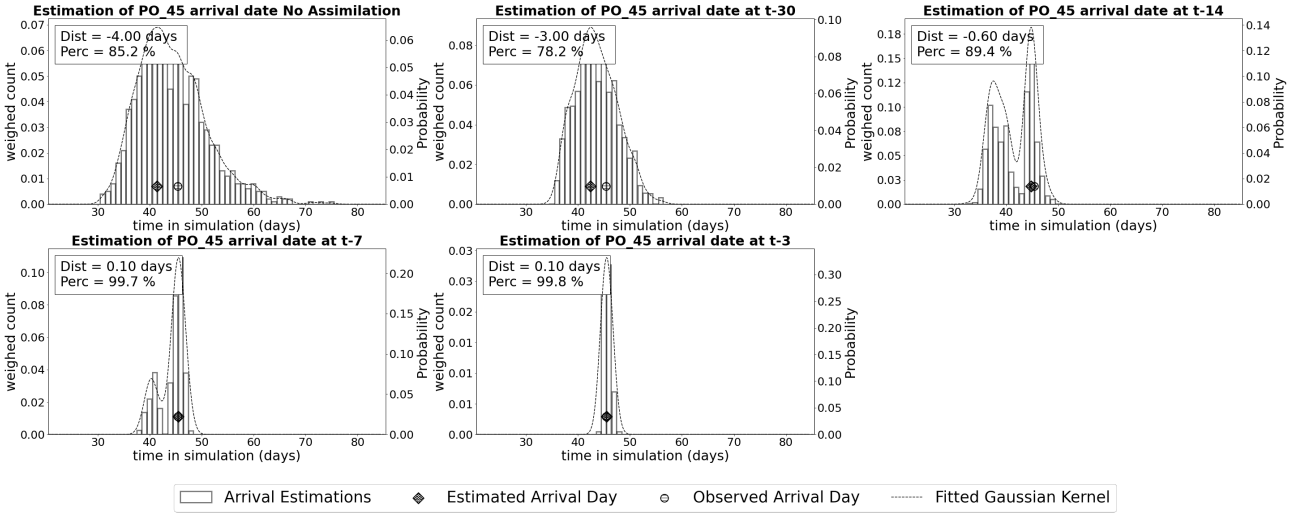


Figure 6.8: Weighted histograms of arrival estimations of particles of shipment PO_45, experiment *E. 29%*

At last, the weighted histograms with the arrival estimations of a single shipment for experiment *E. 25%* are compared to the other three experiments. Figure 6.9 shows an incredible contrast to the other three experiments.

The data available in this experiment is clearly insufficient to narrow down the fitted Gaussian kernels. The pattern of a *wide range* of *consistently wrong* arrival estimations of the particles returns in all of the individual shipment arrival estimations. This pattern, in combination with the clearly higher weighted mean

absolute distance for *E. 25%* (see figure 6.5) is a clear indication that the four percent of the observations that are omitted between *E. 29%* and *E. 25%* are crucial for the accuracy of shipment arrival estimations.

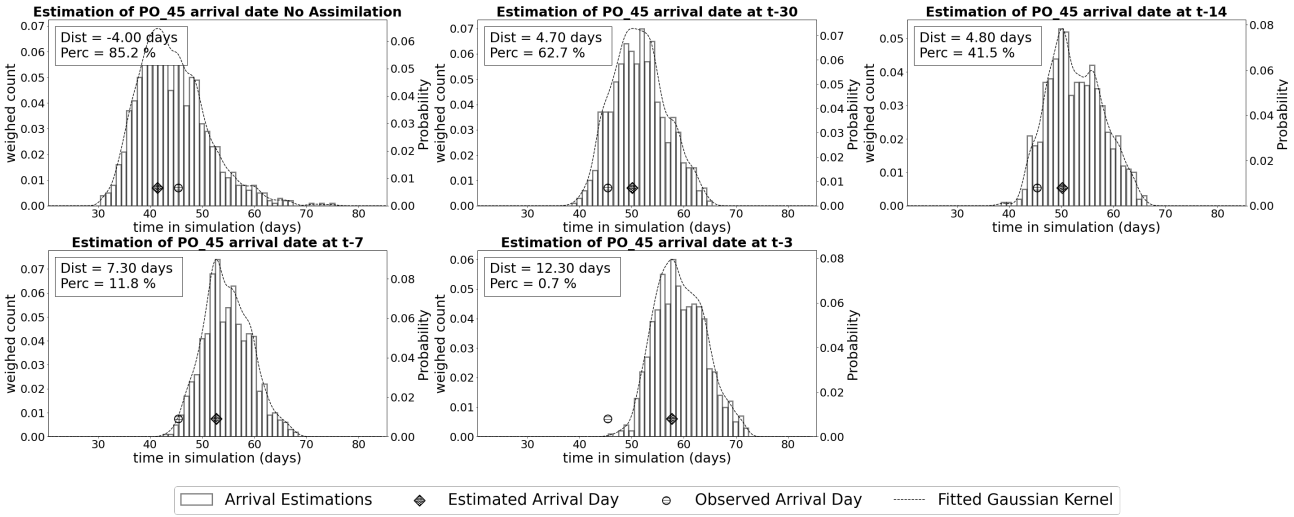


Figure 6.9: Weighted histograms of arrival estimations of particles of shipment PO_45, experiment *E. 25%*

To summarize, the level of data availability is reduced in three steps and the weighted mean of the absolute distance does not increase proportionally. This is a sign that not all data points are of equal importance. Three groups of data points are identified:

First, the data points that are omitted between experiment *E. 100%* and *E. 91%* help making the assimilation algorithm more accurate for results with a lower degree of timeliness. These are the *third party data points* (vessel locations).

Second, the data points that are omitted between experiment *E. 91%* and experiment *E. 29%* are classified as *noise*. Omitting them increases the accuracy of the assimilation algorithm as experiment *E. 29%* yields a lower weighted mean absolute distance than experiment *E. 91%*. These data points are the touch points of the individual *shipments* in the *carrier data*.

Third, the four percent of the data points that are available to the algorithm in experiment *E. 29%* but omitted in experiment *E. 25%* are *crucial* for the data assimilation algorithm. These are the touch points of the *transporters* (aggregated shipments) in the *carrier data*.

Analysis of trend 2: The effect of data availability on the weighted standard deviation of the absolute distances

Analysis of table 6.3 shows no clear relation between the level of data availability and the weighted standard deviation of the absolute distance (i.e., the robustness of the estimations). Further inspection of the weighted absolute distance of individual shipments in figure 6.5 confirms this trend. All experiments have varying interquartile ranges based on which no clear relation is identified.

A comparison of the weighted histograms presented in figure 6.6 to 6.9 shows two effects that influence the weighted standard deviations of the absolute distances:

The first effect is based on three of the experiments (experiments *E. 100%, E. 91% and E. 29%*). For these experiments, up to 34 percent of the arrival estimations are the result of a phenomenon identified as the *two-peak estimation*.

An example of this phenomenon is presented in figure 6.10, where the estimation at t_{-14} clearly shows two peaks, of which one accurately estimates the observed arrival day and one misses the observed arrival date. Because the second (wrong) peak in figure 6.10 was higher, the wrong estimated arrival day is chosen.

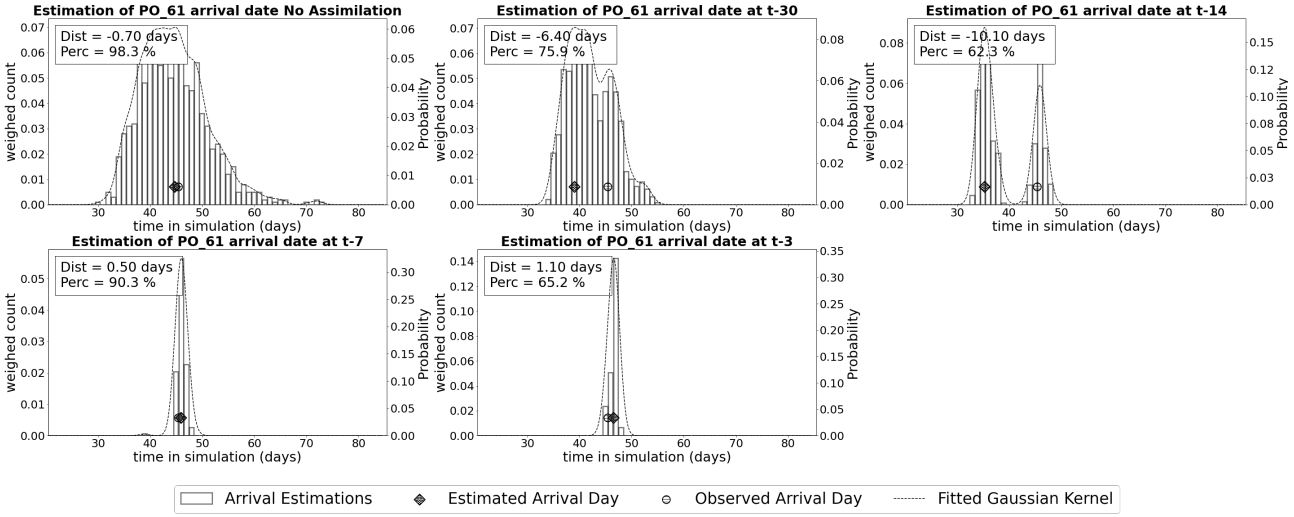


Figure 6.10: Weighted histograms of arrival estimations of particles of shipment PO_61, experiment *E. 100%*

This results in a high absolute distance, despite a high number of "right" particles. Analysis of all weighted histograms shows that about 50 percent of the *two-peak* estimations result in a *right* estimations and for the other 50 percent the estimated arrival date is based on the *wrong* peak. This makes the *weighted standard deviation* over all the estimations relatively high, while a large portion of the particles produced an accurate prediction.

The second effect relates to the arrival estimations in experiment *E. 25%*. Table 6.3 shows a very low standard deviation for this experiment. This *could* be a sign of robustness. Analysis of the weighted histograms confirms this robustness, however this should *not* be seen as a desirable result. The algorithm is simply very robust in producing inaccurate estimations.

Analysis of trend 3: The effect of the degree of timeliness on the weighted mean absolute distance

Table 6.3 shows that reducing the degree of timeliness does not always result in more accurate results. This unexpected trend is visualized in figure 6.11 where it is clearly visible that for experiment *E. 91%* and *E. 25%* a lower degree of timeliness does not result in more accurate results.

Figure 6.11 gives more insight in the contribution of the different shipments to these trends. First it is visible that omitting information on the location of the vessels in the system (from *E. 100%* to *E. 91%*) makes the algorithm incapable of becoming more accurate for a lower degree of timeliness. This confirms the previously made claim that the data on the location of the vessels is important for increasing the accuracy for a low degree of timeliness if a high number of data sources (dimension) is used.

Second, is clear that solely relying on S&OP data points (experiment *E. 25%*) results in even more inaccurate shipment arrival estimations. Based on this data it is impossible for the particles to correct their estimation and the accuracy of the results detoriates as time proceeds. This is clearly visible in the weighted histograms presented in figure 6.9. For each lowered degree of timeliness the weighted histogram shifts right and the arrival estimation becomes increasingly inaccurate.

Analysis of trend 4: The effect of the degree of timeliness on the weighted standard deviation of the absolute distances

A decreasing degree of timeliness generally results in a lower standard deviation of the weighted absolute distance. In other words, the arrival estimations closer to the observed arrival date are more robust. This is visible in figure 6.11 by the decreasing interquartile ranges of the box plots for each of the experiments.

Analysis of the weighted histograms and fitted Gaussian kernels as presented in figure 6.6 to 6.10 helps understanding this behavior. As the degree of timeliness is reduced, the particles are guided by the resampling process of the data assimilation algorithm. The result of this is more agreement in the estimated

Figure 6.11: Spread of weighted absolute distance per experiment

arrival date. The peaks of the fitted Gaussian kernels become more narrow, which results in lower variability of the arrival estimations itself. This translates into a lower variability (i.e., weighted standard deviation) of the absolute distance between the estimated and observed arrival dates.

However, as previously discussed the decrease in weighted standard deviation for *E. 25%* should not be considered desirable as this is just an indicator of consistent inaccurate results.

6.3.2. Weighted percentage

The effects of different levels of data availability on the weighted percentage are discussed in the subsequent sections. Table 6.4 shows the weighted mean and the weighted standard deviation over the different experiments. Three out of the four trends that have been identified for the weighted absolute distance are visible for the weighted percentage as well.

	Weighted mean percentage			
	t_{-30}	t_{-14}	t_{-7}	t_{-3}
E. 100%	70.7%	77.9%	79.9%	82.2%
E. 91%	61.6%	62.1%	50.5%	45.2%
E. 29%	67.7%	82.6%	88.7%	89.1%
E. 25%	66.3%	62.0%	25.5%	2.2%

	Weighted standard deviation percentage			
	t_{-30}	t_{-14}	t_{-7}	t_{-3}
E. 100%	23.8%	25.2%	22.1%	18.3%
E. 91%	23.0%	27.2%	25.3%	28.1%
E. 29%	18.8%	22.0%	13.9%	9.4%
E. 25%	21.1%	11.6%	13.4%	2.4%

Table 6.4: Effect of limited data availability on the weighted percentage

The first clearly visible trend is the fact that the relation *more observations equals more accurate results* also does not hold for the weighted percentage. Second, a clear relation between the data availability and the weighted standard deviation of the percentages can also not be identified. Third, when studying the effect of the timeliness of the estimations similar trends are visible regarding the weighted mean percentage. Experiment *E. 100%* and *E. 29%* follow an expected trend: a lower degree of timeliness results in more accurate results. The opposite trend is visible for experiment *E. 91%* and *E. 25%*. A lower degree of timeliness results in a lower weighted mean percentage for these experiments.

Next to these similarities, one interesting difference versus the weighted absolute distances is visible in table 6.4. Where a lower degree of timeliness results in a lower weighted standard deviation of the weighted absolute distance (i.e., more robust results), this trend does not exist for the weighted percentage. The weighted standard deviation even increases for some of the experiments (e.g., *E.91%.*)

These four trends will be reflected on in the following sections. As the trends are mainly in line with the trends identified for the weighted absolute distance, the reflection will brief and focused on the differences compared to the trends of the weighted absolute distance.

To support this reflection, the results presented in table 6.4 are presented in visual form in figure 6.12. Here the percentage of the different shipments is be visualized in a similar way as in the previous parts of this chapter. The weighted percentage per shipment is presented by dark blue scatters, where the size of the scatter represents the weight of the shipment. The weighted mean percentage is presented with a yellow cross and the standard deviation is represented by the size of the interquartile ranges of the box plots.

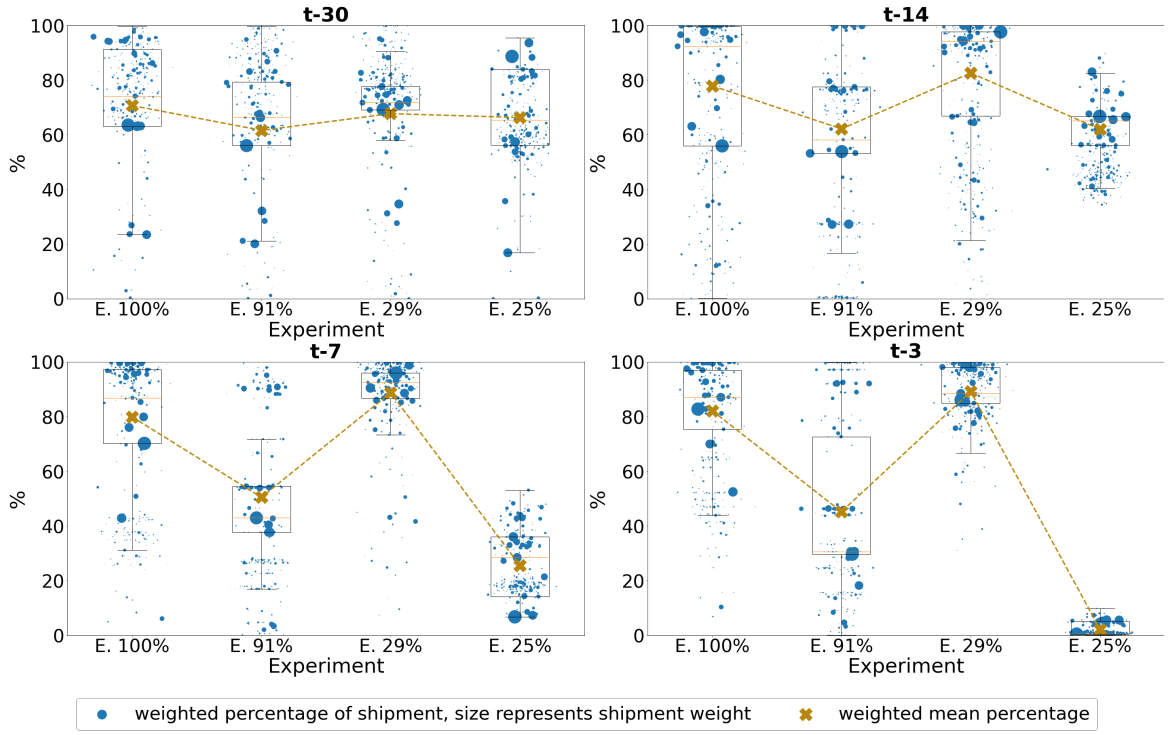


Figure 6.12: Spread of weighted percentage per degree of timeliness

Analysis of trend 1: The effect of data availability on the weighted mean percentage

The lack of the relation *more observations equals more accurate results* becomes even more visible in figure 6.12. For every degree of timeliness, experiment *E. 91%* and *E. 25%* have a lower weighted average percentage than experiment *E. 100%* and *E. 29%*.

Analysis of the weighted histograms and fitted Gaussian kernels as presented in figure 6.6 to 6.10 gives the same insights as presented for the weighted absolute distance and are therefore not separately discussed.

Analysis of trend 2: The effect of data availability on the weighted standard deviation of the percentages

Both table 6.4 and figure 6.12 show no clear relation between the availability of data and the weighted standard deviation of the percentages for each of the shipments. For each of the experiments, (except experiment *E. 25%*) a relatively high degree of variation is observed.

Analysis of the estimations of the different particles (see figure 6.6 to 6.10) provide reasoning for the high variation between the percentages of the shipments. As the particles find agreement due to the importance resampling of the data assimilation algorithm, the fitted Gaussian kernels become more narrow. This results

in situations where an estimated arrival day that is very close to the observed arrival date is still missed by all of the particles. The result of this is a very low very low percentage.

An example of this is presented in figure 6.6 for the estimation at t_{-3}. This estimation has an absolute distance of just 1.8 days and a percentage of 36.9%. A large contrast to the estimation at t_{-30}, where a doubled absolute distance (3.7 days) results in a higher percentage (76.8%).

In other words, a small deviation in the observed arrival date can result in a very high deviation of the percentage of an estimation due to the narrow peaks.

For experiment *E. 25%* the situation regarding the weighted standard deviation is the same for the percentage as the absolute distance. The algorithm becomes very consistent in inaccurate estimations with a weighted mean percentage of 2.2% and a weighted standard deviation of 2.4% for the lowest degree of timeliness (t_{-3}).

Analysis of trend 3: The effect of the degree of timeliness on the weighted mean percentage

The third trend for the weighted percentage is in line with the weighted absolute distance as well. Reducing the degree of timeliness does not always result in more accurate results. For experiment *E. 91%* and *E. 25%* the opposite trend is visible: a lower degree of timeliness results in a lower weighted mean percentage. This trend is even more visible in figure 6.13.

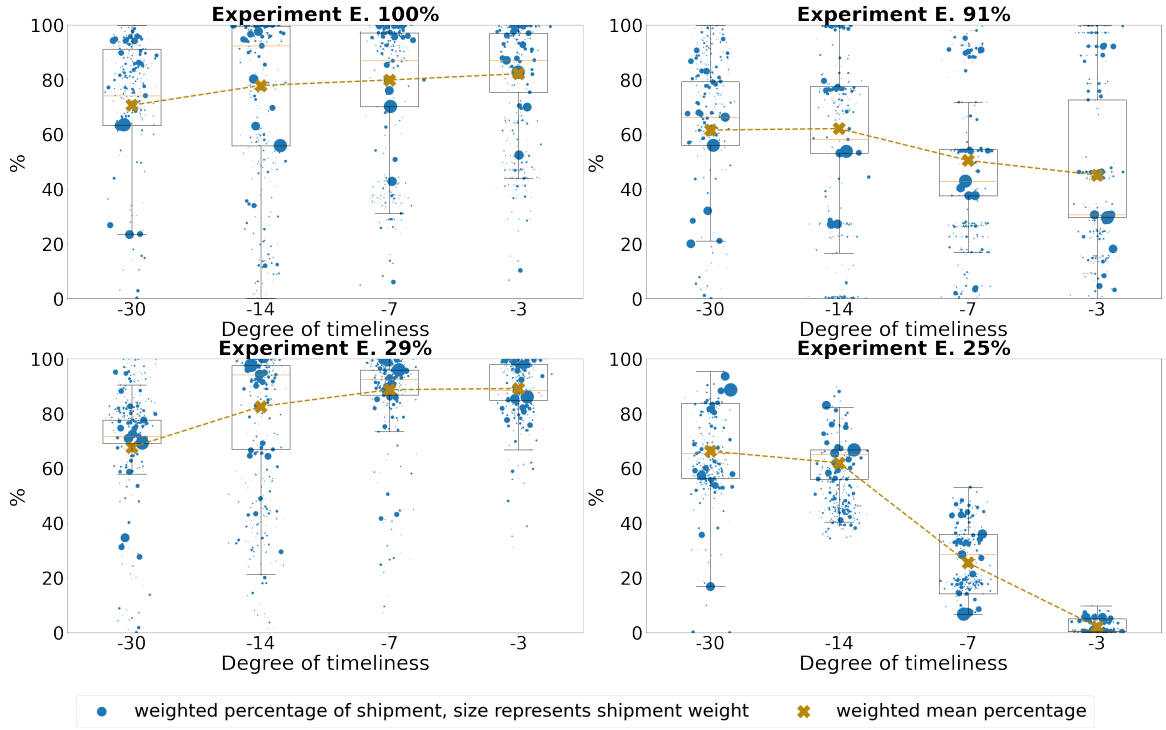


Figure 6.13: Spread of weighted absolute percentage per experiment

Analysis of the weighted histograms and fitted Gaussian kernels as presented in figure 6.6 to 6.10 help understanding these unexpected results. For experiment *E. 91%* this is explained by the increasing uniformity in estimated arrival date of the particles. As figure 6.5 shows, the mean absolute distance does not decrease together with the degree of timeliness for experiment *E. 91%* while the mean percentage does decrease. A constant absolute distance combined with more narrow peaks of the fitted Gaussian kernel does always imply a lower percentage.

The explanation of the results from experiment *E. 25%* is even more straightforward. Figure 6.9 shows that the fitted Gaussian kernels do not become more narrow, but the fitted kernel continuously shifts right. This results in a reducing weighted percentage for each of the shipments.

Analysis of trend 4: The effect of the degree of timeliness on the weighted standard deviation of the percentages

Figure 6.13 and table 6.4 embody one unexpected result. As the degree of timeliness is reduced, the weighted standard deviation of the percentages is not decreased. It even increases for many of the experiments. This is an unexpected but explainable result.

As discussed in the Analysis of trend 3, a decreasing degree of timeliness results in more uniform estimations by the particles. This results in more narrow fitted Gaussian kernels. Therefore, small variations between the observed and estimated arrival date result in large variations in the percentage for that shipment. This effect becomes larger as the degree of timeliness is reduced, resulting in an increase of the weighted standard deviation.

This explanation is supported by an analysis of the weighted histograms and fitted Gaussian kernels for experiment *E. 25%*. An example of this is given in figure 6.9, which shows that for this experiment the peaks do not become more narrow over time. For this experiment the standard deviation should not increase as much as the other experimetns. Figure 6.13 and table 6.4 confirm this.

6.4. Seed analysis

To confirm the previously presented results, a random seed analysis is performed. Three runs with different random seeds have been executed for experiment *E. 100%*. For each extra experiment the accuracy metrics (*weighted absolute distance* and *weighted percentage*) per cluster of shipment arrival estimations are compared. The outcomes of this analysis are visually presented in figure 6.14

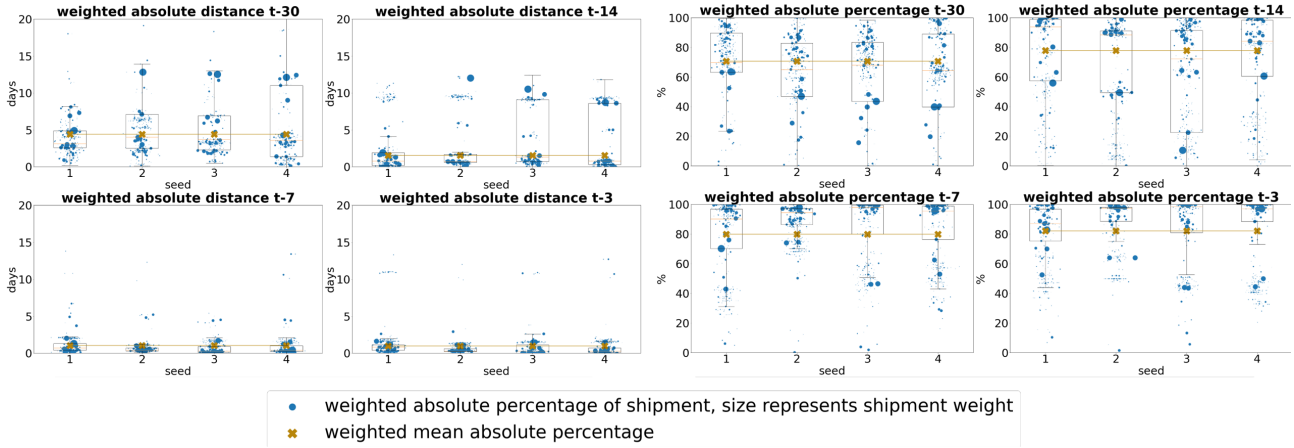


Figure 6.14: Results of experiment *E.100%* with different random seeds

Analysis of results of seed analysis

Figure 6.14 shows that the differences for the *weighted mean* based metrics are small. However, for different random seeds small differences are visible regarding the weighted standard deviations on both accuracy metrics. This implies that randomness can affect the robustness of the data assimilation algorithm. However, the differences do not appear to be significant.

To confirm this hypothesis, the different experiments have been compared by means of an independent t-test. The p-values are presented in table; p_{absD} refers to the p-values for the weighted absolute distances and p_p refers to the p-values for the weighted percentages. A glance at table 6.5 shows that for t_{-14} the distribution for the *weighted absolute distances* can differ significantly. The same accounts for the distributions of the *weighted percentages* between random seed one and two. Since the spread in percentages is merely high for all random seeds, this is no problem for future implementation of the data assimilation algorithm.

	t_{-30}		t_{-14}		t_{-7}		t_{-3}	
	p_{absD}	p_p	p_{absD}	p_p	p_{absD}	p_p	p_{absD}	p_p
Seed 1 vs 2	0.018	1.000	0.000	1.000	1.000	0.000	1.000	0.000
Seed 1 vs 3	0.385	0.989	0.001	0.996	0.963	0.458	1.000	0.485
Seed 1 vs 4	0.664	0.998	0.002	0.853	0.986	0.290	1.000	0.290

Table 6.5: P-values for random seed test

6.5. Summary of results

The results show a clear positive effect of data assimilation on the accuracy of shipment arrival estimations on all accuracy metrics. Next to this, the results show that the effect of limited data availability on the weighted mean of the accuracy metrics is not proportional to the number of observations. Especially for arrival estimations with a low degree of timeliness. This implies some data points are more important than others.

Three groups of data points are identified. First, the third party data points that contain the location data of the vessels. These help increasing the accuracy for situations with a low degree of timeliness. Second, the carrier data points regarding the event-based observations of individual shipments. These are considered *noise* as they have a negative effect on the accuracy of the data assimilation exercise. The third group of data points are the *crucial* data points; the carrier data points with the events of transporter departing and arriving at locations. If these data points are omitted, the accuracy of the data assimilation exercise detoriates.

In contrast to the weighted mean, limited data availability does not show to have a clear impact on the *robustness* of the data assimilation algorithm. This shows to be a function of the degree of timeliness; generally the closer an estimation is to the observed arrival day, the more robust the accuracy of that estimation is.

The last interesting finding relates to the effect of the degree of timeliness on the weighted mean of the accuracy metrics. The data assimilation algorithm yields less accurate results for a reducing degree of timeliness for experiment *E. 91%* and *E. 25%*.

7
Discussion of results

In the following chapter, the results of the research will be discussed. First the effects of data assimilation visible in the results will be discussed and compared to existing literature. Subsequently the generalizability of the results to other (types of) supply chains will be discussed based on a comparison of the case study to other supply chain systems. At last the limitations of the research will be discussed which will help both placing the results in a wider perspective for future improvements as well as giving directions for future work.

7.1. Discussion of results

The results that have been presented in the previous chapter will be discussed and compared to other theories and results found in literature. This discussion will help uncovering the findings that are a contribution to the current state-of-the-art of data assimilation in discrete event simulation literature. Next to this, comparing the results to other streams of literature will provide a basis for generalizing the results of the case study to different types of systems and data assimilation exercises.

7.1.1. Effect of assimilation

The first aspect that is discussed is the effect of data assimilation. The results presented in table 6.2 and figure 6.3 and 6.4 show the positive effect on the accuracy of data assimilation. Data assimilation shows to positively affect both the weighted mean and the weighted standard deviation of both accuracy metrics.

The confirmation of the positive effect of data assimilation shows that DES DA is a solution to one of the pitfalls of traditional modeling and similation; the lack of predictive power (Darema, 2005). The insight that, even in situations with limited data availability, data assimilation has a positive effect on the accuracy is a contribution to DES DA literature. The research of Xie (2019) already showed the improvement of accuracy of data assimilation regarding arrival estimations, however the dependency on data availability was only moderately tested.

7.1.2. Effect of data availability on the accuracy of the data assimilation algorithm

The fact that more observation data does not necessarily result in more accurate results is unexpected and interesting. Hu & Wu (2019) studied the effect of reducing the number of sensors that are used to collect the observation data. They state that *"generally speaking, more sensors provide more observation data and thus lead to more accurate data assimilation results"* (p. 21). However in their research all sensors collect the same data and are evenly spaced and therefore no observation is more important than others. For supply chains, the opposite is true, some observations are more important than others as different types of observations are collected.

The finding that not all sensor data is of equal importance is interesting and new to DES DA literature but can be explained by machine learning literate. In machine learning, the importance of different features is a widely studied topic, especially in the field of clustering. Literature on this topic helps to understand why the arrival estimations of the experiment with 91 percent of the observations available are less accurate than the results of the experiment with 29 percent of the observations available.

Feature selection is the process of selecting a subset of relevant features to use as input for a model (Koutroumbas & Theodoridis, 2009). It is reasoned that features that are *irrelevant* or *redundant* can be omitted without losing the crucial information (Watt et al., 2020). Often feature selection is applied to avoid

the adverse effects of the *curse of dimensionality* (Bengtsson et al., 2008).

The notion of *irrelevant* or *redundant* information and the curse of dimensionality explains why the experiment with 29 percent of the observations available yields more accurate estimations than the experiment with 91 percent of the observations. The data points omitted between these experiments are *irrelevant* and should be classified as *noise*. The increase of the accuracy by omitting these data points is explained by the mitigation of the effects of the *curse of dimensionality*.

The link to the research stream of feature selection supports the identification of three different groups of data points. First, the third party vessel location data points are *beneficial* for the accuracy in a high-dimensional data set and help mitigating the curse of the dimensionality. Second the carrier data points are classified as *noise* since omitting these data points yields better results due to the reduced effects of the curse of dimensionality. The third group of data points are the carrier data points related to the departure and arrival of transporters (vessels and trains). This group of data points is *crucial* since the accuracy of the arrival estimations implodes after omitting them.

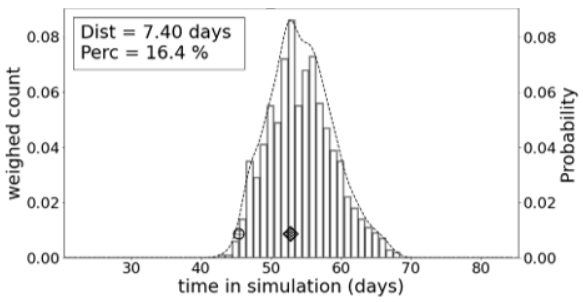
To transfer these findings to other systems with different sensors and behavior, it is necessary to know *a priori* which sensors in a system are part of the *crucial* sensors and which sensors should be identified as *noise*. In machine learning, this is referred to as feature selection, which is the process of finding the optimal set of features. This is generally done by removing the features with a high correlation to other features (since this is a sign of redundancy) and evaluating different subsets of the remaining set based on different accuracy metrics. (Koutroumbas & Theodoridis, 2009).

As DES DA is generally applied in situations where historical data is not available, traditional methods of feature selections such as *filtering* and *wrapping* Sánchez-Maroño & Alonso-etanzos (2007) do not work. Another criterion for the identification of the *crucial* sensors is proposed instead.

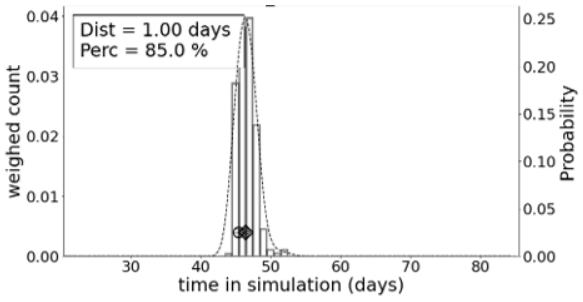
The *crucial* sensors in this research are the the carrier data points related to the departure and arrival of transporters (vessels and trains). These sensors serve as *arrival sensors* for the entities at the system boundaries. Arrival sensors are no new concept by itself as this concept first is applied by Xie (2019), who used them to accurately estimate the number of entities in an open system to combat the variable dimension problem. Next to this, this research has shown that for supply chains these arrival sensors are crucial in reducing the uncertainty of the arrival estimations as well.

The results of this research show that using arrival sensors reduces the largest share of uncertainty in the simulation model. This reduction of uncertainty is caused by more than just accurately estimating the number of entities in an open system. The arrival sensors sensors allow an accurate estimation of the arrival schedules of the transporters and thereby reduce the variation the arrival estimations of the particles.

A visualization of this effect is given in figure 7.1. Figure 7.1a clearly shows that there is still a high level of uncertainty (indicated by the wide peak) in the arrival estimations if no arrival sensors are used. While figure 7.1b shows that the uncertainty can dramatically be reduced by using these arrival sensors.



(a) Weighted histogram of shipment arrival estimations without arrival sensors



(b) Weighted histogram of shipment arrival estimations with arrival sensors

Figure 7.1: Effect of using arrival sensors on the variation of the shipment arrival estimations

To summarize, because sensors have different levels of importance in supply chains, more observations do not necessarily yield more accurate estimations. Based on a reflection on the different types of data, a method to determine the crucial sensors for DES DA with open systems *a priori* is identified: The crucial sensors are the ones that can serve as *arrival sensors* at the system boundaries of an open system.

7.1.3. Effect of the degree of timeliness on the accuracy of the data assimilation algorithm

The results show that for two out of the four experiments (*E. 100%* and *E. 29%*) the accuracy of the shipment arrival estimation increases as the date of estimation becomes closer to the observed arrival dates. This is in line with literature on forecasting by modeling and simulation. An example of this is given by Rosenhead (1989), who describes this effect as the *uncertainty trumpet*: The further the time horizon of a forecast expands, the bigger the range of possible futures gets due to inherent uncertainties in a system that cannot be captured by simulation models. In other words, estimations of arrivals closer in time should be more accurate.

This notion of the timeliness of an estimation affecting the accuracy of an estimation is new in DES DA literature. However, in other data assimilation exercises this is a well known phenomenon. An example of this effect in Monte Carlo simulations that is well-known to the public is the (long-term) weather forecast. Here an initial condition produces an increasingly wide range of different potential futures due to the uncertainties in the models, which results in a plume shaped temperature prediction (Zhao et al., 2018). This plume shape indicates that estimations with a low degree of timeliness embody less uncertainty and are therefore generally more accurate.

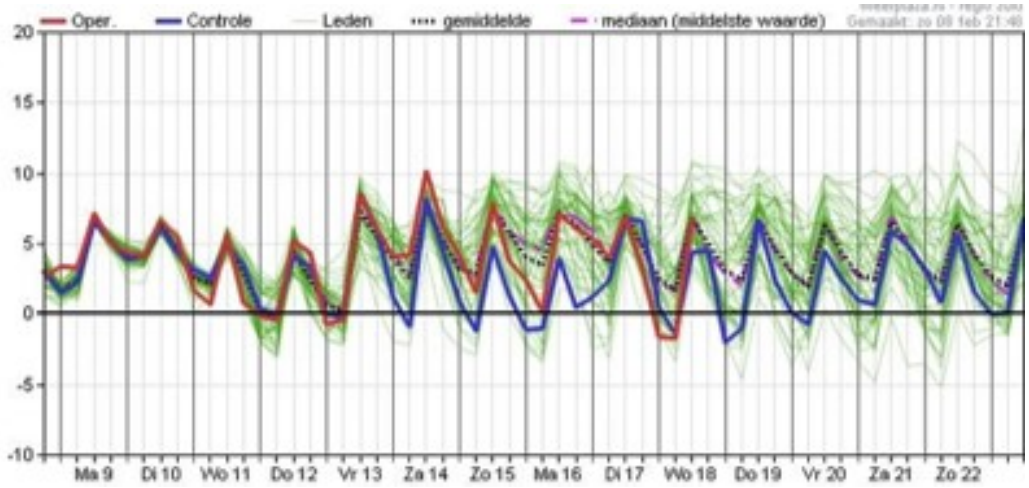


Figure 7.2: Example of a plume shape weather prediction derrived from KNMI.nl

Thus, the literature on forecasting by modeling and simulation provides an explanation for two out of the four experiments. However, the two other experiments show the opposite trend by becoming less accurate as the degree of timeliness is reduced.

The results of experiment *E. 91%* stay about the same as the degree of timeliness is reduced. This could be an effect of the *curse of dimensionality*. The high volume of *irrelevant* or *redundant* observations make it difficult to compute meaningful distance metrics and thereby assign high weights to the particles that can estimate the arrival of the shipments correct.

To understand why the accuracy of the arrival estimations for experiment *E. 25%* even decreases as the degree of timeliness is reduced, the concept of *underfitting* is discussed. Underfitting occurs when a model uses input variables that are not significant enough to determine a meaningful relationship between the input and output variables (Koutroumbas & Theodoridis, 2009).

In experiment *E. 25%* there is a very large period in time between the last observation on a shipment (departure from the consolidator) and the actual arrival of a shipment (see figure 7.3). In this period of time on average 30 iterations of the data assimilation algorithm are executed in which the weights of the particles are updated every time. This decreases the relevance of the weights with respect to the shipments later on in the supply chain and thereby reduces the accuracy of the arrival estimations with a low degree of timeliness. This effect is visible in figure 6.9 by the right-shifting weighted histogram and fitting Gaussian kernel.

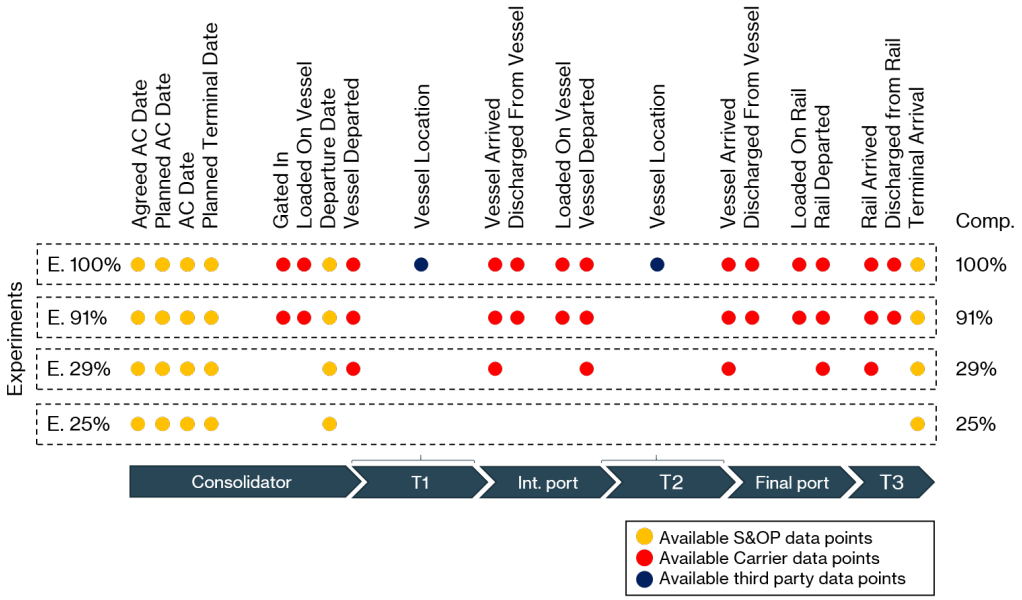
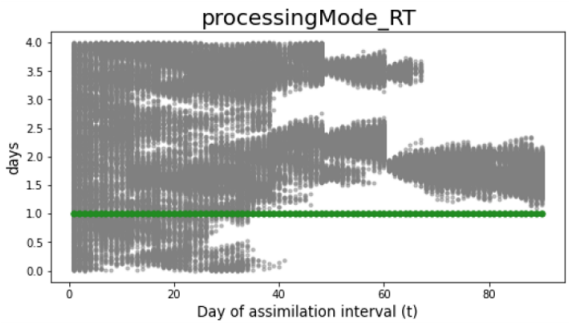
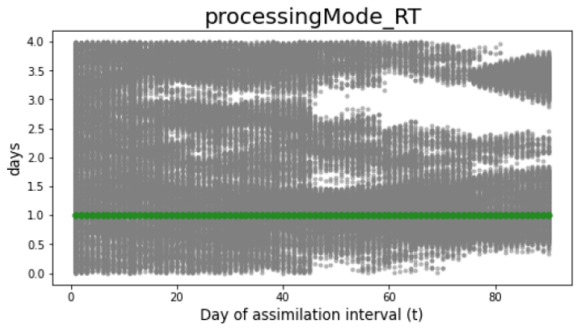


Figure 7.3: Experiments with different levels of semantic completeness

Next to the right shifting weighted histograms, *underfitting* is also clearly visible in the data assimilation dashboard for experiment *E. 25%* presented in appendix C. Where the particles in the other experiments converge, the convergence of the particles is limited of *E. 25%*. This means the observations are not significant enough to meaningfully steer the particles in the state space. A snapshot the data assimilation dashboards for experiment *E. 29%* and *E. 25%* is presented in figure 7.4 to show the effect of *underfitting*.



(a) Example without underfitting (experiment *E. 29%*)



(b) Example of underfitting (experiment *E. 25%*)

Figure 7.4: Snapshot of data assimilation dashboards experiment *E. 29%* and *E. 25%*. States of variables presented in grey, variables of real system in green.

Summarizing, the effect the degree of timeliness has on the accuracy of the data assimilation algorithm is dependent on the data availability. If there is a large share of significant data points (features) in the observations that are used for the computation of the weights of the particles (*E. 100%* and *E. 29%*), a decreasing degree of timeliness yields more accurate results. If too many data points are classified as noise (*E. 91%*), this effect disappears. And if there are not enough significant data points, *underfitting* occurs and a decreasing degree of timeliness even results in less accurate estimations.

7.1.4. Comparison of data assimilation to other techniques

The high level of accuracy achieved in this research shows that DES DA is highly promising as a technique to increase supply chain visibility by estimating future events. One alternative class of techniques for the *automatic* estimation of future events based on data is available: machine learning models. In this section DES DA is compared to (supervised) machine learning to understand the potential for actual implementation of DES DA.

Supervised machine learning models are a method of data analysis that *automates analytical model building* (Koutroumbas & Theodoridis, 2009). Machine learning models are generally constructed (commonly referred to as trained) by learning from a labeled data set (Watt et al., 2020). For the example of estimating shipment arrivals this *historical data set* is composed by observations of events or locations in the supply chain, with the observed arrival days as labels. Automatically building analytical models is different than DES DA which make DES DA desirable in some situations which are discussed first. Thereafter, the main benefit of supervised machine compared to DES DA is discussed.

Machine learning models need a historical data set for training before any (accurate) results can be produced (Watt et al., 2020). DES DA only needs real-time data and some prior knowledge on the system to provide accurate shipment arrival estimations. This difference makes DES DA the preferred technique in situations where historical data is not available; for example after the establishment of a new supply chain route or after new data is acquired.

The second main benefit of DES DA compared to machine learning models is the form of the output of DES DA. The output of the data assimilation algorithm takes the form of a *probability density function*, while machine learning models generally output a single value or a set of classes with assigned probabilities. Even complex neural networks need several adjustments to generate complete probability densities (van Gerven & Bohte, 2017). To understand he benefit of probability density functions as output, the concept of the *two-peak estimation* is discussed again.

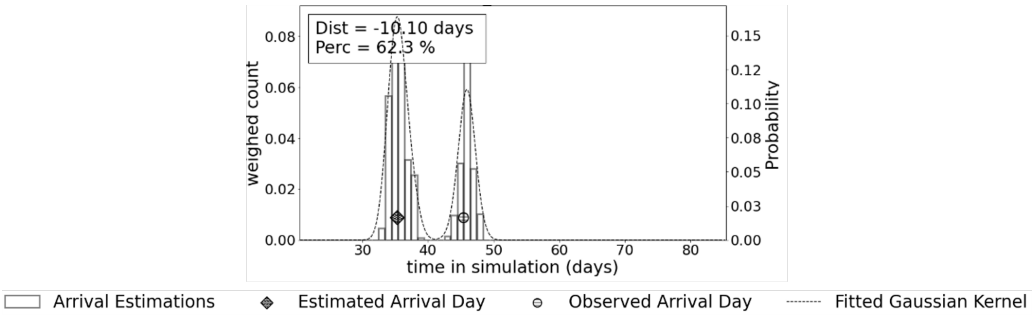


Figure 7.5: Example of a two-peak estimation

The results showed many situations where two estimated arrival days are (almost) equally probable, resulting in two peaks in the probability density function of estimated arrival days. An example of this is given in 7.5 and is referred to as a *two-peak estimation*. In these situations machine learning models generally communicate one of these two peaks or even worse, the average day between these two peaks. DES DA allows to actually communicate the existence of these two peaks. Based on this planners could to design robust interventions that can deal with arrivals on both estimated peaks. This is a large advantage of DES DA compared to machine learning models.

Despite the aforementioned advantages of DES DA, there is one scenario where machine learning models are preferred over data assimilation. This is the scenario where a large, well-maintained historical labeled data set is available. Machine learning models would be preferred in these situations these models are generally easier to implement as they are automatically trained. DES DA requires the building of a validated simulation model that accurately represents the real system which is a time consuming process.

7.2. Discussion of the generalizability of the results

The research questions are focused the application of data assimilation in discrete event simulation on supply chains in general, while this research is based on one specific use case. Therefore, the generalizability of the results to other supply chains is discussed.

The generalizability of the supply chain of the use-case to other supply chains will be discussed first. Thereafter the used data sources will be discussed to see how these relate to other supply chains.

7.2.1. Generalizability of the supply chain under study

The results for the use-case can be generalized to supply chains in general if the supply chain of the use-case embodies characteristics that are typical for supply chains for which similar decision support requests can be expected. These requests are typical for supply chains with a low level of visibility and a high level of uncertainty (Ivanov et al., 2019).

Therefore, the level of visibility and uncertainty of the supply chain in this research is assessed to ensure that the results from the use-case can be generalized to other supply chains and are not too optimistic. For the supply chain under study three drivers are identified that underpin the low level of supply chain visibility and the high level of uncertainty.

The fact that this supply chain is a *multi-tier* supply chain is the first driver of low visibility and uncertainty. This means that there are multiple supplier to buyer relationships within one supply chain (Jia et al., 2019). In the supply chain under study there are three distinctive tiers present. The vendors buy raw materials from their suppliers and subsequently sell the produced goods to the stakeholder, which thereafter sells its products to its (wholesale) customers. This dependency on other parties reduces the supply chain visibility for the stakeholder and increases the uncertainty in the shipment arrival times.

The second driver is the fact that the transport in the supply chain is carried out by an external party; a carrier. This means that the stakeholder can not make any policy decisions to steer the arrival dates after the shipments have left the consolidator. This increases the uncertainty for the stakeholder and reduces the supply chain visibility.

Third, the uncertainty and variation in shipment arrival dates is high because multiple sequential modes of transports are used to transport the shipments from the consolidator to the terminal. First the shipments are transported with a feeder to an intermediate port, subsequently with a vessel to the final port and at last with a train to the terminal. Each of these modes of transports has its own (uncertain) schedule. Research on planning optimization (Sohoni et al., 2011) shows that the connectivity of sequential schedules can have a high impact on the variation of the arrival times and thereby drives the nonlinearity and uncertainty of a system. The presence of nonlinearity is clearly visible by the binomial distribution shown in figure 7.5.

To summarize, because the supply chain under study embodies three important factors that are common for supply chains with a low degree of visibility, the results of this research can be generalized to other supply chains with a similar decision support request. If any of these factors are not present in that supply chain, the results are expected to have an even higher degree of accuracy.

7.2.2. Generalizability of data sources

A second concern for the generalizability or the results lays in the type of data sources (sensors) that are used for this analysis. These data sources should not be unique for the supply chain under study but commonly used in other (commercial) supply chain as well. As described in chapter 3, this was one of the criteria for selecting the data sources in the first place. Therefore this should impose no problems for the generalizability of the results; this is discussed in below.

The first data source used in this research is the S&OP data. This is the internal supply chain and operations data with the agreed, planned and actual delivery dates of the shipments at the consolidator and the terminal. A market analysis by McKinsey & Company (2016) has shown that the majority of the publicly listed companies with an international supply chain manage this data in tools such as SAP.

The second data source used is the carrier data. For this supply chain this data is available via TradeLens, a Maersk platform on which event-based observations on the transport of shipments are available. Since

2019 five out of the six largest global ocean carriers joined the TradeLens platform (TradeLens, 2019) capturing over 61 percent of the world liner fleet (Statista, 2021). This indicates that carrier data is accessible in many (international) supply chains.

The last data source used is for this research is the third party location data of the vessels. This data encompasses the lattitude and longitude of vessels retrieved from AIS satellite data. This data can be acquired via a wide variety of platforms such as *vesselfinder.com*, *vesseltracker.com* and *marinetraffic.com*. Therefore this data is accessible for many if not all international supply chains that use vessels to transport their goods.

To summarize, the data sources used in this research are expected to impose no problems for the generalizability of the findings of this research to other (commercial) supply chains. Each of the data sources is either considered a best practice or widely available for international supply chains.

7.3. Discussion of the limitations of the research

Given the time and scope of this research, not every aspect of the algorithm is tested or optimized for. These limitations of the research can affect the findings of the research. Therefore these limitations of this research and their implications are discussed. First, three limitations of the research that can have a negative impact on the accuracy of the algorithm are discussed. Second, two limitations regarding the generalizability of the results are discussed.

7.3.1. Limitations affecting the accuracy of the algorithm

The first set of limitations that is discussed are the those that could have an impact on the reported performance (accuracy) of the data assimilation. These limitations relate to the optimization of the performance of the data assimilation algorithm. Because this research focuses on testing the effects of limited data availability on the accuracy of DES DA, the optimization of the algorithm is left out of scope.

Hyperparameters of the algorithm

Given the time and scope of the research is decided to test one version of the data assimilation algorithm for different levels of data availability and keeping the hyperparameters of the algorithm constant over the experiments. For particle filtering algorithms (which are used for DES DA) two hyperparameters have significant impact on the performance of the algorithms. Tuning the hyperparameters can result in more accurate results. Both parameters are discussed to investigate the potential effects of tuning these parameters.

The first hyperparameter is the *number of particles (N)*. For DES DA each particle is an instance of a simulation model. Therefore, increasing the number of particles increases the computational burden. This generally results in a longer duration of the execution of the experiments. On the other hand, previous results on DES DA indicated that more particles generally yield more accurate results (Cho, 2019, Xie, 2019). Therefore a balance between the computational burden and the accuracy of the outcomes of the experiments should be found.

The second hyperparameter that is kept constant over the various experiments is the *length of the data assimilation interval* (Δt). Cho et al. (2020) showed that smaller data assimilation intervals generally yield more accurate results.

The different data sources used in this research are only updated once every day, which means that new data becomes available with an interval of one day. The data assimilation interval can not be smaller than the interval by which data becomes available. Thus, the the smallest assimilation interval possible is used. Therefore the length of the data assimilation interval used in this research is expected to have no limiting effect on the presented results, although it could be an interesting direction of further research to investigate the effect of increasing the data assimilation interval to the accuracy of the algorithm to confirm this hypothesis.

Distance metrics

In the research three different distance metrics are proposed to combat the curse of dimensionality. Given the time and scope of this research only one of these metrics is used for the different experiments. This is the *fractional distance* as this metric has the best foundation in literature (i.e., Aggarwal et al. (2001)). Analysis

of the results has shown that this distance metric performs well for this system although although it could not be guaranteed that the other distance metrics can result in a higher level of accuracy. Therefore further research would be needed to assess the effect of different distance metrics.

Identical twin

The last limitation of this research is the fact that the data assimilation algorithm is developed and tested based on an identical twin concept due to the lack of availability of historical carrier and third party data. Working with an identical twin generally results in more accurate results than assimilating data from a physical system since the model structure used for the particles is always 100% valid (Xie, 2019).

Therefore working with the historical data of an actual physical system would give more realistic results in absolute sense. However, as the goal of this research was to compare the relative performance of the algorithm under different levels of data availability, these results can still be deemed valid.

Estimated effect on results

It is expected that a further tuning of the hyperparameters can increase the accuracy of each of the experiments. Especially increasing the number of particles (N) is expected to drive this positive effect. However, this will also increase the computational burden. This could make the results of the different experiments more uniform, but as it is expected that this will positively impact the accuracy of all experiments this should have no implications for the findings of this report.

On the other hand the implementation of this algorithm with an actual real system instead of an identical twin as surrogate is expected to decrease the absolute accuracy of the arrival estimations. However, the relative importance of the different data sources is expected to remain the same.

7.3.2. Limitations for the generalizability of the results

In general the findings of this research can be generalized to other supply chains because the supply chain structure and the data sources used for this research are common to other supply chains for which these types of decision requests can be expected. However, as the scope of this research was limited, there are situations in which the results can not be generalized. These situations are situations for which the the algorithm has not been tested or developed for these types.

Structural uncertainty

For the system under study it was assumed that no *structural uncertainty* existed. This makes the current algorithm inapplicable to supply chains with structural uncertainty Structural uncertainty is present when we are uncertain about the model output because we are uncertain about the functional form of the model (Kwakkel et al., 2015). For supply chains structural uncertainty is present when it is uncertain which route or mode of transport shipments take to go from a to b. This is uncertain since shipments could be transported by different carriers or carriers could decide take different routes.

Incorporating structural uncertainty would make the data assimilation more complex because of two factors. First, structural uncertainty generally results in nonlinearity and binomial distributions in the estimations of future events. It can be challenging to capture these nonlinear dynamics by data assimilation. However, because the current data assimilation is already able to capture these nonlinearities, shown by the estimation of binomial distributions as presented in figure 7.5, the effects of this factor are expected to be limited.

The second factor is the the *variable dimension* problem (Xie, 2019) which is expected to impose the biggest challenge and needs future research. As shipments in the real system can take different routes, shipments in the real system could pass completely different sensors than shipments particles (simulation models). This results in a variable dimension between the observations of the real system and the measurements from the particles which makes computing a distance for each of the particles a very complex challenge.

Black swan events

The data assimilation algorithm is developed to support decision making in supply chains for *normal operations*. In other words, it is not tested how the algorithm would react to so-called *black swan* events such as the blocking of the Suez Canal.

This makes the conclusions that will be drawn on the effect of data availability on the accuracy of DES DA valid for supply chains in normal operations. However, the predictive capabilities of the proposed algorithm in these *black swan* events can not be guaranteed and could be an interesting field of future research.

7.4. Implication of research

This research is a basis for the future development of DES DA algorithms for supply chains. The main implication of this research is the notion that, for supply chains, more observations do not necessarily result in more accurate results. An unexpected trend since this is different than previous observations of Xie (2019) and Hu & Wu (2019) who noted that more observations generally result in more accurate results. Machine learning literature on feature selection supports these results. Thus, for future DES DA research it is crucial take into account that some observations are more important than others. The crucial data sources are the data sources that can serve as *arrival sensors* at the system boundaries.

The second academic contribution to the state-of-the-art DES DA research is the development of a DES DA algorithm that can provide accurate estimations of future events based on high volumes of both event-based and location-based data, which is typical for supply chains. The provided code and extensive description of the different functions that form the algorithm are intended to increase the reproducibility of this research and facilitate future research to further develop this algorithm.

The third contribution to the current state-of-the-art DES DA algorithms is the fact that the presented algorithm uses *weighted observations*: the volume of a shipment determines the importance of an observation and shipments with high volumes should be prioritized by the algorithm. In other research all entities and observations are considered equal. The results clearly show that the algorithm is able to prioritize the large shipments, which benefits the applicability of this algorithm.

Apart from these academic implications, the practical implications of this research are significant as well. After presenting the results to different teams of the stakeholder, the stakeholder decided to further prioritize on retrieving the crucial data points to potentially start implementing DES DA.

A second practical implication of this research, is the insight that is given in the effectiveness of Salabim as a simulation software package to create the simulation environment. As thoroughly discussed in chapter 4, the characteristics of Salabim made the reconstruction of the particles in the resampling step an unnecessary difficult process. Therefore it is recommended to use a different simulation software package for future research. The requirements for a future package are described in appendix B.

7.5. Future work

As stated, this research should be used as a basis for the future development of data assimilation in discrete event simulation algorithms. Based on the identified limitations of this research two fields for future research are identified. First, it is suggested to focus on the *development* of data assimilation algorithm. Second, it the *implementation* of data assimilation algorithms needs more research. For each of these fields several potential topics will be discussed.

7.5.1. Algorithm development

One of the limitations of this research is the fact that this research is based on the *identical twin* concept. This means that the simulation models used to represent the real system in the particles are always completely valid as these are the exact same model. using an *identical twin* is desirable for research to assess the quality of algorithms because this guarantees that none of the results are caused by an unsatisfactory validation and verification of the simulation models.

However, as these algorithms are developed for application to real systems, more research should be focused on the performance of DES DA algorims based data coming from an actual real system. To perform this research it is suggested to collaborate with a stakeholder that has access to the historical data from all data sources identified in this research. This data should be already available and can not be collected real-time supply chain planning cycles generally take multiple months which makes running multiple ex-

periments a timely exercise.

The second topic of for future work regarding the development of DES DA algorithms is the incorporation of structural uncertainty. As discussed structural uncertainty can make DES DA very challenging because of the variable dimension problem that surfaces if entities take different routes in each of the particles. To mitigate the adverse effect of the variable dimension problem adjustments to the computation of the weights should be made for which more research is needed.

To incorporate structural uncertainty, the current simulation model can simply be extend by adding to different routes and modes of transport (e.g., transport via air). To do this research it is suggested to collaborate with the current stakeholder as a part of the current supply chain has already been modeled and different routes and modes of transport are present.

At last it a set of follow up experiments is suggested to test how the performance of the presented algorithm could further be optimized. As previously discussed the main improvement is expected in tuning the different hyperparameters, testing different distance metrics and testing more combinations of input data.

For these follow up experiments it is recommended to take the data assimilation algorithm that has been developed during this research and extend this with the two distance metrics that are suggested but have not been tested in this research. A full factorial design with varying distance metrics, number of particles and different levels of data availability should be the basis for a sensitivity test comparable to the research of Cho (2019). Performing this sensitivity test will provide a good overview of the ideal configuration of the DES DA algorithm for this type of supply chains.

7.5.2. Algorithm implementation

Apart from the suggestions for future work on the development of DES DA algorithms it is suggested to do future research on the implementation of DES DA to supply chains. This field of research has a rather practical character. Three topics that need to be addressed before the proposed data assimilation algorithm can be implemented on the real supply chain are discussed.

At first more research should be done on how the different types of data can be made available to the algorithm in case of implementation on an actual supply chain. In this research, all data points that are assimilated into the simulation models are the result of a simulation model itself and are generated by the data assimilation algorithm.

To implement this algorithm based on the real-time observations of an actual supply chains, a data lake should be constructed in which the observations (from different data sources) can be stored in such a way that the data assimilation algorithm can collect these observations and assimilate them into the simulation model.

Another point of concern when implementing the proposed algorithm is the detection of particles drifting off. There could be situations where none of the particles represent the behavior of the real system. For the supply chain operations and planning team it is crucial that these situations are detected before the shipments arrive.

This could be done by imposing a threshold condition on the distance metrics that are calculated for each of the particles. If all particles exceed this threshold, none of the particles represents the behavior of the real system and a *rollback and resampling* operation needs to be performed to allow the particles to match the real system again. More research on how to establish such a threshold is needed.

At last it is suggested to explore if more advanced methods could be used to unify the clusters of shipment arrival estimations into one final shipment arrival estimation. Currently this is done at by taking the peak of the fitted Gaussian kernel, however analysis of the different clusters shows there are many situations where the clusters result in two peaks (see appendix C). An example of a *two-peak estimation* is given in figure C.7.

This research should not only focus at the detection of patterns such as the *two-peak estimations* but

also on the communication of these results to the planners. Notifying the supply chain planners of two (almost) equally probable estimated arrival dates allows these planners to design robust inventory planning strategies that can cope with both estimated arrival dates. However, current S&OP planning tools do not allow for multiple planned dates. Therefore, the communication of estimated arrival dates should be improved as well.

8
Conclusion

In this chapter, the final research question as proposed in chapter 1 will be answered: *What is the effect of limited data availability on the accuracy of data assimilation in discrete event simulation for supply chains?* To support the answer on this final question, first the four sub-questions will be answered. Based on these insights, the main research question will be answered.

8.1. Answers to sub-questions

To answer the main research question, four sub-questions are identified that together help answering the main research question. In the following sections, each of these sub-questions are discussed and answered. The answers are intended to be applicable to data assimilation in discrete event simulation algorithms for supply chains in general.

1. *How should the current state-of-the-art data assimilation in discrete event simulation algorithms be adjusted for application to supply chains?*

Applying data assimilation in discrete event simulation is different for supply chains than many other field of applications because of the challenges that rise with the volume and shape of the input data as well as the required output data.

The challenge imposed by the input data comes from the fact that data available on supply chains is generally event-based. Supply chains generally embody a high number of shipments and high number of touch points for which events are recorded. This results in a high number of observations and thus a high level of dimensionality of the data that is used to compute the weights of the different particles. Due to the *curse of dimensionality*, fitting a multivariate Gaussian distribution to determine the weights as applied by Cho (2019) does not work.

Therefore the data assimilation algorithm is adjusted and a single distance metric for each particle computed. Three different metrics are proposed that show potential to combat the adverse effects of the curse of dimensionality. The *fractional distance* as proposed by (Aggarwal et al., 2001) is implemented and results in accurate arrival estimations.

The second challenge for the application of DES DA to supply is caused by the required output data comes. In supply chains, the hidden state that should be estimated is generally a future state (i.e., a forecast of future events). Therefore, the data assimilation algorithm is adjusted: After the states of a particle are recorded at the end of an assimilation interval, each particle is simulated for an extra period of time to make this future state estimation.

2. *What accuracy metrics should be used to determine the accuracy of data assimilation in discrete event simulation algorithms for the estimation of future events?*

An estimation of a future event by data assimilation in discrete event simulation is considered accurate if two requirements are satisfied: First, the *absolute distance* between the estimation of a future event and the observed future event should be low. Because a low absolute distance indicates the right prediction is made and the information based on which policies are implemented is right. Second, a high share of the particles should have estimated the observed future event right, which is indicated by the *percentage*. The percentage is an important indicator because the output of a DES DA algorithm is a probability density

function and not a simple number.

To aspects of accuracy are proposed to assess the accuracy over a set of estimations of future events. First, the *average* accuracy over the set of estimations should be high. Second, there should be little variation between the accuracy of the different future estimations (i.e., the algorithm should be robust). Little variation is is indicated by a low *standard deviation*.

If different types of future events are estimated, each event should get assigned a weight. The data assimilation algorithm is intended to prioritize accurate estimations of events with a higher weight. Therefore, the weight of the events is part of the accuracy metrics as well.

This results in four recommended accuracy metrics that should be used to determine the accuracy of DES DA algorithms for the estimation of future events. These accuracy metrics are presented in table 8.1

	Weighted Mean	**Weighted Standard Deviation**
Absolute Distance	Weighted mean absolute distance (low)	Weighted standard deviation of the absolute distances (low)
Percentage	Weighted mean percentage (high)	Weighted standard deviation of the percentages (low)

Table 8.1: Accuracy metrics for the estimation of future events by DES DA. Desired value between brackets.

For the estimation of a future event it is also highly relevant *when* this estimation is done. Therefore, the *degree of timeliness* of an estimation is introduced in this Thesis. The closer to the observed event an estimation is performed the lower the degree of timeliness of this estimation. It is recommended to collect the accuracy metrics for different degrees of timeliness in future research.

3. *What is the relation between the number of observations and the accuracy of data assimilation in discrete event simulations on supply chains?*

The results show that the relation identified in previous DES DA research, *more observations results in more accurate results*, does not hold for the average accuracy of shipment arrival estimations in supply chains. The experiment with 29 percent of the observations (*E. 29%*) produces more accurate results than the experiment with 91 percent of the observations available (*E. 91%*). This shows that using of less observations can actually help to combat the *curse of dimensionality* and thereby produces in more accurate results.

However, the results also clearly show that *less is more* is also not true as the experiment with 91 percent of the observations available (*E. 91%*) yields less accurate results than the experiment with 100 percent of the observations available.

Therefore, there is no clear relation between the number of observations and the average accuracy of data assimilation. Some observations clearly have more impact than others. Next to this, the results have shown that no clear relation between the level of data availability and the robustness (weighted standard deviation) of the arrival estimations exists.

4. *What sources of data are crucial for the the accuracy of data assimilation with discrete event simulation on supply chains?*

Three groups of data points are identified based on the results and the discussion on the results. First, the *third party vessel location data* points are *beneficial* for the accuracy in a high-dimensional data set. Second the *carrier data points* can be seen as *noise* since omitting these data points yields better results due to the reduced effects of the curse of dimensionality. The third group of data points are the c*arrier data points* related to the departure and arrival of transporters (vessels and trains). This group of data points is *crucial* since the accuracy of the arrival estimations implodes after omitting them.

In machine learning literature the crucial sources of data are the data sources that help explain the highest level of variability (i.e., uncertainty) in a system. For this research the same pattern is identified. the *crucial* data sources are the the carrier data points related to the departure and arrival of transporters (vessels and trains). The effect of these data sources on the accuracy of the data assimilation is high because these data sources also serve as *arrival sensors* for entities in the system (see also Xie (2019)). Using the arrival sensors allows for the reduction the largest share of uncertainty in the simulation model by an accurately estimating of the arrival schedules of the transporters and thereby reduce the variation the arrival estimations of the particles.

To summarize, the *crucial data points* are the data sources that serve as *arrival sensors*. The arrival sensors help reducing the uncertainty in the supply chain by accurately estimating the number of entities and the arrival schedules of these entities in an open system.

8.2. Answer to research question

Research Question: *What is the effect of limited data availability on the accuracy of data assimilation in discrete event simulation on supply chains?*

This research shows that limited data availability on supply chains does not necessarily has to result in less accurate estimations by DES DA. A lower number of observations can result in just as accurate or even more accurate results. However, this can only be the case if the omitted data is not part of the set of *crucial data sources*. The crucial data sources are the data sources that serve as *arrival sensors* which help to reduce most of the uncertainty in simulations of open systems.

If the omitted data points are part of these *crucial data sources* the accuracy of the data assimilation algorithm detoriates. This results in inaccurate estimations, especially for estimations with a low degree of timeliness.

However, if limited data availability is the result of omitting data sources that are not part of the *crucial data sources*, the accuracy of the data assimilation algorithm can increase. Less data points help combatting the *curse of dimensionality* and result in more accurate estimations.

Bibliography

Aggarwal, C. C., Hinneburg, A., & Keim, D. A. (2001). On the surprising behavior of distance metrics in high dimensional space. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1973*, 420–434. doi: 10.1007/3-540-44503-x{_}27

Angeles, R. (2005). Rfid Technologies : Supply-Chain Applications and Implementation Issues. *Information Systems Management*, *22*(1), 51–65. doi: 10.1201/1078/44912.22.1.20051201/85749.7

Aydt, H., Turner, S. J., Cai, W., & Low, M. Y. H. (2009). Research issues in symbiotic simulation. *Proceedings - Winter Simulation Conference*, 1213–1222. doi: 10.1109/WSC.2009.5429419

Banks, J. (1998). *Handbook of Simiulation*. New York: Wiley-Interscience.

Baroni, G., & Tarantola, S. (2014). A General Probabilistic Framework for uncertainty and global sensitivity analysis of deterministic models: A hydrological case study. *Environmental Modelling and Software*, *51*, 26–34. doi: 10.1016/j.envsoft.2013.09.022

Baryannis, G., Validi, S., Dani, S., & Antoniou, G. (2019). Supply chain risk management and artificial intelligence: state of the art and future research directions. *International Journal of Production Research*, *57*(7), 2179–2202. doi: 10.1080/00207543.2018.1530476

Bengtsson, T., Bickel, P., & Li, B. (2008). Curse-of-dimensionality revisited: Collapse of the particle filter in very large scale systems. *Probability and Statistics: Essays in Honor of David A. Freedman*, *2*, 316–334. doi: 10.1214/193940307000000518

Beskos, A., Crisan, D., Jasra, A., Kamatani, K., & Zhou, Y. (2017). A stable particle filter for a class of high-dimensional state-space models. *Advances in Applied Probability*, *49*(1), 24–48. doi: 10.1017.apr.2016.77

Blanco, E. E., Yang, X., Gralla, E., Godding, G., & Rodriguez, E. (2011). Using discrete-event simulation for evaluating non-linear supply chain phenomena. In *Proceedings - winter simulation conference* (pp. 2255–2267). Phoenix Arizona. doi: 10.1109/WSC.2011.6147937

Box, G. (1976). Science and Statistics. *Journal of the American Statistical Association*, *71*(356), 791–799. doi: 10.1641/B570910

Caridi, M., Crippa, L., Perego, A., Sianesi, A., & Tumino, A. (2010). Measuring visibility to improve supply chain performance: A quantitative approach. *Benchmarking*, *17*(4), 593–615. doi: 10.1108/14635771011060602

Caridi, M., Perego, A., & Tumino, A. (2013). Measuring supply chain visibility in the apparel industry. *Benchmarking*, *20*(1), 25–44. doi: 10.1108/14635771311299470

Cho, Y. (2019). *Strategic usage of Real-time data for Dynamic Data Driven Simulation* (Doctoral dissertation, TU Delft). Retrieved from `https://repository.tudelft.nl/islandora/object/uuid%3Ab8aaa3b2-eb6f-408b-88cd-5df6d8f90625`

Cho, Y., Huang, Y., & Verbraeck, A. (2020). *Strategic use of data assimilation for dynamic data-driven simulation* (Vol. 12142 LNCS). Springer International Publishing. doi: 10.1007/978-3-030-50433-5{_}3

Chopde, N., & Nichat, M. (2013). Landmark Based Shortest Path Detection by Using A* and Haversine Formula. *International Journal of Innovative Research in Computer and Communication Engineering*, *1*(2), 298–302. doi: 10.1145/4568552

Closas, P., & Fernández-Prades, C. (2011). Particle filtering with adaptive number of particles. *IEEE Aerospace Conference Proceedings*. doi: 10.1109/AERO.2011.5747439

Cole, R., Stevenson, M., & Aitken, J. (2019). Blockchain Technology: Implication for operations and supply chain management. *Supply Chain Management: an iInternational Journal*, *4*(24), 469–483. doi: 10.1108/SCM-09-2018-0309

Darema, F. (2004). Dynamic data driven applications systems: A new paradigm for application simulations and measurements. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3038*, 662–669. doi: 10.1007/978-3-540-24688-6{_}86

Darema, F. (2005). Grid computing and beyond: The context of dynamic data driven applications systems. *Proceedings of the IEEE*, *93*(3), 692–697. doi: 10.1109/JPROC.2004.842783

Deelen, J. V., Kuipers, L., Meijnema, M., & Meggelen, D. V. (2020). Methods To Reduce Uncertainty For Forecasting By Modeling and Simulation - A Review. , 1–25. Retrieved from `www.brightspace.tudelft.nl`

DeepAI. (2020). *What is the curse of dimensionality?* Retrieved from `https://deepai.org/machine-learning-glossary-and-terms/curse-of-dimensionality`

Doherty, K., Adams, R., & Davey, N. (2004). Non-Euclidean Norms and Data Normalisation. *ESANN'2004 Proceedings-European Symposium on Artificial Neural Networks*(April), 181–186. Retrieved from `https://uhra.herts.ac.uk/bitstream/handle/2299/118/102247.pdf?sequence=1`

Ekenel, H. K., & Stiefelhagen, R. (2006). Analysis of local appearance-based face recognition: Effects of feature selection and feature normalization. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2006*. doi: 10.1109/CVPRW.2006.29

Elvira, V., Miguez, J., & Djurie, P. M. (2017). Adapting the Number of Particles in Sequential Monte Carlo Methods Through an Online Scheme for Convergence Assessment. *IEEE Transactions on Signal Processing*, *65*(7), 1781–1794. doi: 10.1109/TSP.2016.2637324

Ferdows, K. (2018). Keeping up with growing complexity of managing global operations. , *38*(2), 390–402. doi: 10.1108/IJOPM-01-2017-0019

Goodall, P., Sharpe, R., & West, A. (2019). A data-driven simulation to support remanufacturing operations. *Computers in Industry*, *105*, 48–60. Retrieved from `https://doi.org/10.1016/j.compind.2018.11.001` doi: 10.1016/j.compind.2018.11.001

Gu, F., & Hu, X. (2008). TOWARDS APPLICATIONS OF PARTICLE FILTERS IN WILDFIRE SPREAD SIMULATION Feng. In S. Mason, R. Hill, L. Monch, O. Rose, T. Jefferson, & J. Fowler (Eds.), *Proceedings of the 2008 winter simulation conference* (pp. 2852–2860). Miami, FL.

Günal, M. M., & Pidd, M. (2010). Discrete event simulation for performance modelling in health care: A review of the literature. *Journal of Simulation*, *4*(1), 42–51. doi: 10.1057/jos.2009.25

Ham, R. V. D. (2020). salabim Documentation. Retrieved from `https://www.salabim.org/`

Hasheminia, H., & Jiang, C. (2017). Strategic trade-off between vessel delay and schedule recovery: an empirical analysis of container liner shipping. *Maritime Policy and Management*, *44*(4), 458–473. doi: 10.1080/03088839.2017.1298867

Hu, X., & Wu, P. (2019). A data assimilation framework for discrete event simulations. *ACM Transactions on Modeling and Computer Simulation*, *29*(3). doi: 10.1145/3301502

Huan, S. H., Sheoran, S. K., & Wan, G. (2004). A review and analysis of supply chain operations reference (SCOR) model. *Supply Chain Management*, *9*(1), 23–29. doi: 10.1108/13598540410517557

Huang, Y. (2013). *Automated Simulation Model Generation* (Doctoral dissertation, TU Delft). doi: 10.4233/uuid:dab2b000-eba3-42ee-8eab-b4840f711e37

Huck, N. (2019). Large data sets and machine learning: Applications to statistical arbitrage. *European Journal of Operational Research*, *278*(1), 330–342. doi: 10.1016/j.ejor.2019.04.013

Ibrahim, O. (2018). *Design and Investigation of a Decision Support System for Public Policy Formulation* (Vol. 12) (No. October). Retrieved from `http://urn.kb.se/resolve?urn=urn:nbn:se:su:diva-159437`

Ivanov, D., Dolgui, A., Das, A., & Sokolov, B. (2019). Digital Supply Chain Twins: Managing the Ripple Effect, Resilience, and Disruption Risks by Data-Driven Optimization, Simulation, and Visibility. *International Series in Operations Research and Management Science*, *276*, 309–332. doi: 10.1007/978-3-030-14302-2{_}15

Jia, F., Gong, Y., & Brown, S. (2019). Multi-tier sustainable supply chain management: The role of supply chain leadership. *International Journal of Production Economics*, *217*(July 2017), 44–63. doi: 10.1016/j.ijpe.2018.07.022

Jugović, A., Bukša, J., Dragoslavić, A., & Sopta, D. (2019). The possibilities of applying blockchain technology in shipping. *Pomorstvo*, *33*(2), 274–279. doi: 10.31217/p.33.2.19

Kádár, B., Pfeiffer, A., & Monostori, L. (2004). Discrete event simulation for supporting production planning and scheduling decisions in digital factories. *Proceedings of the 37th CIRP international seminar on manufacturing systems*, 444–448. doi: 10.3390/1035854

Kantas, N., Beskos, A., & Jasra, A. (2014). Sequential Monte Carlo Methods for High-Dimensional Inverse Problems: A Case Study for the Navier–Stokes Equations. *SIAM/ASA Journal on Uncertainty Quantification*, *2*(1), 464–489. doi: 10.1137/130930364

Koutroumbas, K., & Theodoridis, S. (2009). *Pattern Recognition* (e-book ed.). Cambridge, Massachusetts: Academic Press. doi: https://doi.org/10.1016/B978-1-59749-272-0.X0001-2

Kwakkel, J. H. (2017). *The Exploratory Modeling Workbench: An open source toolkit for exploratory modeling, scenario discovery, and (multi-objective) robust decision making* (Vol. 96). doi: 10.1016/j.envsoft.2017.06.054

Kwakkel, J. H., Haasnoot, M., & Walker, W. E. (2015). Developing dynamic adaptive policy pathways: a computer-assisted approach for developing adaptive strategies for a deeply uncertain world. *Climatic Change*, *132*(3), 373–386. doi: 10.1007/s10584-014-1210-4

Laprie, J.-C. (2008). From dependability to resilience. *38th Annual IEEE/IFIP International Conference On Dependable Systems and Networks*, Fast abstracts. doi: 10.495/340022

Long, Y., & Hu, X. (2017). Spatial partition-based particle filtering for data assimilation inwildfire spread simulation. *ACM Transactions on Spatial Algorithms and Systems*, *3*(2). doi: 10.1145/3099471

Martindale, W., Duong, L., Hollands, T., & Swainson, M. (2020). Testing the data platforms required for the 21st century food system using an industry ecosystem approach. *Science of the Total Environment*, *724*, 137871. doi: 10.1016/j.scitotenv.2020.137871

McKinsey & Company. (2016). *Big data and the supply chain: The big-supply-chain analytics landscape (Part 1)*. Retrieved from `https://www.mckinsey.com/business-functions/operations/our-insights/big-data-and-the-supply-chain-the-big-supply-chain-analytics-landscape-part-1`

Metyis. (2020). *SCENARIO PLANNING TOOL - SPT* (Tech. Rep. No. September). Amsterdam: Metyis.

Mirkes, E. M., Allohibi, J., & Gorban, A. (2020). Fractional norms and quasinorms do not help to overcome the curse of dimensionality. *Entropy*, *22*(10), 1–31. doi: 10.3390/e22101105

Nance, R. E. (1981). The Time and State Relationships in Simulation Modeling. *Communications of the ACM*, *24*(4), 173–179. doi: 10.1145/358598.358601

National Institute of Standards and Technology. (1996). Weighted Standard Deviation. *Dataplot Reference Manual*, *11*, 1. Retrieved from http://itl.nist.gov/div898/software/dataplot/refman2/ch2/weightsd.pdf

Novikov, A. (2019, 5). *Creating sea routes from the sea of AIS data.* Retrieved from https://towardsdatascience.com/creating-sea-routes-from-the-sea-of-ais-data-30bc68d8530e

Nykam, D. (2020). *State Space Definition.* Retrieved from https://mathinsight.org/definition/state_space

Pettit, T. J., Croxton, K. L., & Fiksel, J. (2019). The Evolution of Resilience in Supply Chain Management: A Retrospective on Ensuring Supply Chain Resilience. *Journal of Business Logistics*, *40*(1), 56–65. doi: 10.1111/jbl.12202

Ponis, S., & Koronis, E. (2012). Supply Chain Resilience: Definition Of Concept And Its Formative Elements. *The Journal of Applied Business Research*, *28*(5), 921–930. doi: 10.9774/gleaf.9781783533527{_}4

Reichle, R. H. (2008). Data assimilation methods in the Earth sciences. *Advances in Water Resources*, *31*(11), 1411–1418. doi: 10.1016/j.advwatres.2008.01.001

Rosenhead, J. (1989). *Rational analysis for a problematic world: Problem structuring methods for complexity, uncertainty and conflict.* Chichester, UK: Wiley. doi: https://doi.org/10.1057/jors.1989.150

Sánchez-Maroño, N., & Alonso-etanzos, A. (2007). Filter Methods for Feature Selection - A Comparative Study. In *International conference on intelligent data engineering and automated learning.* (pp. 178–187). Berlin, Heidelberg: Springer. doi: 10.1007/978-3-540-77226-2

Sankararaman, S., & Mahadevan, S. (2013). Distribution type uncertainty due to sparse and imprecise data. *Mechanical Systems and Signal Processing*, *37*(1-2), 182–198. doi: 10.1016/j.ymssp.2012.07.008

Shultz, T. R., & Fahlman, S. E. (2017). *Encyclopedia of Machine Learning and Data Mining.* doi: 10.1007/978-1-4899-7687-1

Silva, N., Ferreira, L. M. D., Silva, C., Magalhães, V., & Neto, P. (2017). Improving Supply Chain Visibility With Artificial Neural Networks. *Procedia Manufacturing*, *11*(June), 2083–2090. doi: 10.1016/j.promfg.2017.07.329

Sohoni, M., Lee, Y.-c., Klabjan, D., Sohoni, M., Lee, Y.-c., & Klabjan, D. (2011). Linked references are available on JSTOR for this article : Robust Airline Scheduling Under Block-Time Uncertainty. *Transport Science*, *45*(4), 451–464. doi: 10.1287/trsc.1100.0361

Statista. (2021). *Leading ship operator's share of the world liner fleet as of June 21, 201.* Retrieved from https://www.statista.com/statistics/198206/share-of-leading-container-ship-operators-on-the-world-liner-fleet/

Stein, M. (1987). Large sample properties of simulations using latin hypercube sampling. *Technometrics*, *29*(2), 143–151. doi: 10.1080/00401706.1987.10488205

Tako, A. A., & Robinson, S. (2012). The application of discrete event simulation and system dynamics in the logistics and supply chain context. *Decision Support Systems*, *52*(4), 802–815. doi: 10.1016/j.dss.2011.11.015

TradeLens. (2019). *TradeLens adds major ocen carriers Hapag-Lloyd and Ocean Network Express.* Retrieved from https://www.tradelens.com/press-releases/hapag-lloyd-and-ocean-network-express

van Gerven, M., & Bohte, S. (2017, 12). Editorial: Artificial Neural Networks as Models of Neural Information Processing. *Frontiers in Computational Neuroscience*, *11*(7553), 436–444. doi: 10.3389/fncom.2017.00114

Wang, J., Chang, Q., Xiao, G., Wang, N., & Li, S. (2011). Data driven production modeling and simulation of complex automobile general assembly plant. *Computers in Industry*, *62*(7), 765–775. doi: 10.1016/j.compind.2011.05.004

Wang, S. (2010). *Modeling supply chain dynamics with calibrated simulation using data fusion* (Doctoral dissertation, Arizona State University). doi: 10.1.1.477.2280

Watt, J., Borhani, R., & Katsaggelos, A. (2020). *Machine learning refined: foundations, algorithms,* (2nd Editio ed.). Cambridge, Massachusetts: Cambridge Univerity Press. Retrieved from `https://www.cambridge.org/highereducation/books/machine-learning-refined/0A64B2370C2F7CE3ACF535835E9D7955#overview`

Wei, H. L., & Wang, E. T. (2010). The strategic value of supply chain visibility: Increasing the ability to reconfigure. *European Journal of Information Systems*, *19*(2), 238–249. doi: 10.1057/ejis.2010.10

Welch, G., & Bishop, G. (1995). *An introduction to the Kalman Filter*. doi: 10.1109/LAWP.2018.2818058

Xie, X. (2019). *Data Assimilation in Discrete Event Simulation* (Doctoral dissertation, Delft University of Technology). doi: https://doi.org/10.4233/uuid

Xie, X., & Verbraeck, A. (2019). A particle filter-based data assimilation framework for discrete event simulations. *Simulation*, *95*(11), 1027–1053. doi: 10.1177/0037549718798466

Xue, H., Gu, F., & Hu, X. (2012). Data assimilation using sequential Monte Carlo methods in wildfire spread simulation. *ACM Transactions on Modeling and Computer Simulation*, *22*(4). doi: 10.1145/2379810.2379816

Zeigler, B. P., Praehofer, B., & Kim, T. (2000). Introduction to Systems Modeling Concepts. In *Theory of modeling and simulation* (2nd ed., pp. 3–25). New York: Academic Press. doi: 10.1016/b978-0-12-813370-5.00009-2

Zhao, J., Duan, Y., & Liu, X. (2018). Uncertainty analysis of weather forecast data for cooling load forecasting based on the monte carlo method. *Energies*, *11*(7), 18. doi: 10.3390/en11071900

Appendices

A
Data acquisition system, simulation model, measurement model and real system

The four remaining components of the DES DA framework are discussed in this appendix. These components are tailored to the use-case, however a large share of the aspects of the data sources, processes in the simulation model and implementation of the measurement model could be used to other (parts of) supply chains.

In the following sections, first the *data acquisition system* will be discussed. Here the key characteristics of the data sources ("sensors") in the real system will be discussed. Subsequently the *simulation model* and the *measurement system* will be discussed together. At last the simulation model that will be used as a surrogate for the *real system* will be elaborated on.

A.1. Data acquisition system

The data acquisition system refers to sensors that measure (partial) system states that are assimilated by the data assimilation algorithm into the particles. In other DA examples this data comes from actual sensors (e.g., Hu & Wu (2019)). For supply chains, the data available on the arrival of shipments generally does not directly come from sensors but from data warehouses such as SAP or Tradelens (see section 3.3 for more information). All data points on the supply chain under study are presented in figure fig: touch points data sources app

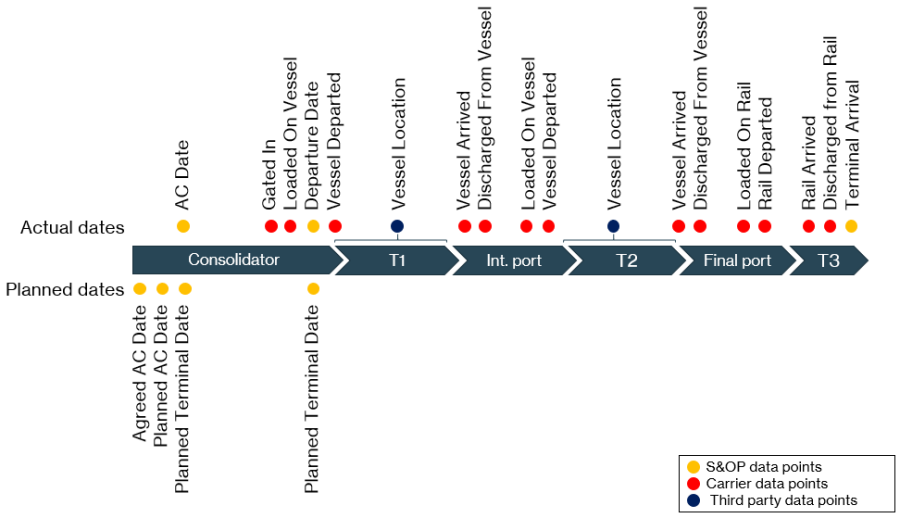


Figure A.1: Overview of data points of different data sources

The observations in the data warehouses can be treated as if they came from actual sensors which observe events or the locations of entities. These observation and contain errors which are the result of measurement errors in actual sensors, mistakes in planning certain dates or simply human mistakes on entering the dates. Therefore assumptions are made on the size and shape of the distribution of these errors.

Three different types of measurements are identified that each have its own type of error distribution and the shape of the distributions is similar to the distribution of the uncertainty as presented in 3.4. *Agreed* and *planned* tend to have a higher degree of uncertainty and are positively skewed because of the tendency

of planned dates to take place more often at a later date than an earlier date. An overview of the error assumptions has been given in table A.1.

All sensors are have *measurement intervals* of one day and the sensors that measure the moments on which events take place have a granularity of days. Since it has been decided to adopt an *identical twin* approach, the sensors data is be extracted from the simulation model that represents the *real system*.

Type of data point	Assumed size of error	Assumed shape of distribution
Agreed data points	Large	Positive Skew
Planned data points	Medium	Positive Skew
Actual data points	Low	Symmetrical

Table A.1: Overview of different error assumptions per type of data point

The the fact that the sensor data comes from data warehouses brings a unique opportunity. Instead of just measuring events that take place, each event or location that is measured is coupled to a shipment. Either via the orderID, the containerID or the vesselID (see also table 3.1). This makes the computation of as well distance as accuracy metrics more accurate as already has been discussed in section 4.2.

A.2. Simulation model and measurement model

The next two components addressed in this section are the simulation model and the measurement model. This simulation model should simulate the flow of goods trough in the real system in a sufficient accurate manner to make estimations on the arrival times of shipments. As previously argued (see section 2.2), discrete event simulation is the modelling tool of choice.

The measurement model are the adjustments made to the model to extract the measurements that are part of the data acquisition system from each of particles. Extracting these measurements by the measurement model allows to compute the distance of the particles to the real system.

Both components will be described in the following sections. The simulation model will be further delineate by a description of the system boundaries, the model variables and the model structure. The measurement model is implemented in the model structure and is discussed together with the structure. The annotated code for the simulation model and the measurement model is presented in *SC_Model*.

A.2.1. System boundaries of the simulation model

To create a valid simulation model, the system boundaries should be clearly delineated. For this a two-step approach is adopted: First one of the vendor countries as presented in figure A.2 is selected for further analysis and subsequently the conceptual system boundaries are defined. It is decided to limit the scope of this research to one of the inbound supply chain branches to deliver the proof of concept, which can be scaled up in future work.

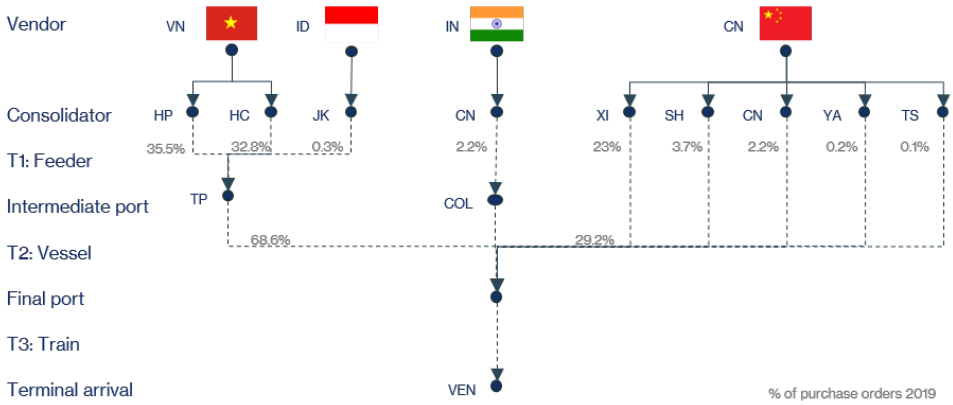


Figure A.2: Supply chain structure Asia - the Netherlands by ship (flow of goods: from top to bottom)

As figure A.2 shows, four different vendor countries exist. Two of them (Indonesia and India) have just one port of loading (consolidator) , and the two others (China and Vietnam) have multiple consolidators. The most suitable branch for this proof of concept is considered to be the most complex branch with the highest need for accuracy. Therefore the different branches are assessed based on four criteria:

- *The number of consolidators* - The more consolidators there are present, the more complex the supply chain structure becomes, since shipments will come in from two different sources with different locations.
- The number of intermediate ports - The more intermediate ports, the more complex the supply chain structure becomes. This because goods will need to be cross-docked here which adds complexity and decision-rules into the supply chain.
- *The number of shipments* - The more shipments, the higher the number of relevant dimensions for computing the distance metrics in the data assimilation algorithm (see section 4.1.1 for more insight on this topic). This could be problematic for the particle filtering exercise, which makes this a desirable feature for the proof of concept selection.
- *The standard deviation between planned and actual terminal delivery date* - The higher the standard deviation between the planned and actual terminal delivery date is, the higher the need for decision-support / data assimilation.

Vietnam is selected as *vendor country* for the proof of concept because this country has multiple consolidators, the goods pass an intermediate port, a large share of the shipments is produced in Vietnam and the deviation between he planned and actual terminal dates is the highest.

The scope of the model is limited from the arrival of the goods at the consolidator, to the delivery of the goods at the terminal. This based on two reasons: First, there is no data nor visibility on the (production) processes at the vendors which make data assimilation impossible. Second, after arrival of the goods at the terminal, the flow of goods is controlled by the stakeholder. This makes forecasting further behavior unnecessary.

At last, the conceptual system boundaries are defined. The focus of this simulation model is on forecasting the flow of shipments trough the supply chain from the consolidator to the terminal. The goal is hereby to forecast in a granularity of days and therefore it is acceptable to simplify some of the processes that take place at both the ports as well as during transportation (e.g., omitting the concepts of acceleration and deceleration for the vessels).

A.2.2. Model variables

For the simulation model a set of 24 variables is used. These are the parameters of probability distributions used to sample the interarrival times and speed of the transporters as well as the processing- and loading times of the shipments at the various ports. The 24 variables are divided in four groups which each will be discussed briefly.

The first group of variables are the modes of the triangular distributions of interarrivaltimes of transporters (feeders, vessels, trains) at the various points of departure (Ho Chi Minh (HC), Hai Phong (HP), Tanjung Pelepas (TP), Rotterdam (RT)). An overview of the parameters is given in table A.2. Data analysis has shown that these arrivals are irregular and should be represented by a triangular distribution (see notebook *02 Creating Shipping Schedules.ipynb*).

Variable	Real Value	Uncertainty Range	Unit
feederHCMode	4	1 - 7	Days
feederHPMode	4	1 - 7	
vesselTPMode	7	1 - 7	
TrainRTMode	1	0.5 - 2	

Table A.2: Modes of interarrival times

The second group of the variables is the The mode of the triangular distribution from which the time of loading and unloading of shipments at a port of departure and arrival (Ho Chi Minh, Hai Phong, Tanjung Pelepas, Rotterdam, Venlo (VN)) is sampled. The uncertainty range for the loading times in Venlo is smaller because a train is unloaded instead of an container vessel which generally takes a longer period of time.

Variable	Real Value	Uncertainty Range	Unit
feederLoadingTimeHCMode	0.2	0 - 0.5	Days
feederLoadingTimeHPMode	0.2	0 - 0.5	
shipLoadingTimeTPMode	0.2	0 - 0.5	
trainLoadingTimeRTMode	0.1	0 - 0.2	

Table A.3: Mode of triangular distribution of loading times

The third group of the variables is the mode of the triangular distribution of the speed of the transporters(feeders, vessels, trains) between two destinations (Ho Chi Minh, Hai Phong, Tanjung Pelepas, Rotterdam, Venlo). This speed is sampled on a daily basis. The mode of this triangular distribution is the only uncertain variable as the minimum and the maximum speed of transporters are known. An overview of these variables is given in table A.5.

Variable	Real Value	Uncertainty Range	Unit
speedFeederHC_TPMode	27.75	18.5 - 46.25	km / hr
speedFeederHP_TPMode	27.75	18.5 - 46.25	
speedVesselTP_RTMode	35.15	18.5 - 46.25	
speedTrainRT_VNMode	35	25 - 40	

Table A.4: Mode of triangular distribution of speed of transporters

The fourth group of variables relates to the processing times at the consolidators, ports and eventual the terminal. These processing times are modelled by means of a *skewed normal distribution* which is formed by three parameters. The *mode*, the *skewness*, and the *size* of the distribution, which is comparable to the standard deviation of a normal distribution. These parameters are considered uncertain and described in table

Variable	Real Value	Uncertainty Range	Unit
processingMode_HC	1	0.5 - 4	Days
processingMode_HP	1	0.5 - 4	
processingMode_TP	1	0.5 - 4	
processingMode_RT	1	0.5 - 4	
processingSkewness_HC	12	1.5 - 2.5	Days
processingSkewness_HP	2	1.5 - 2.5	
processingSkewness_TP	2	1.5 - 2.5	
processingSkewness_RT	2	1.5 - 2.5	
processingSize_HC	1	1.5 - 1.5	Days
processingSize_HP	1	1.5 - 1.5	
processingSize_TP	1	1.5 - 1.5	
processingSize_RT	1	1.5 - 1.5	

Table A.5: Parameters for skewed normal distributions for the processing times

A.2.3. Model structure

Based on the description of the supply chain operations as provided in chapter 3 and the system boundaries, a model structure is generated. This model structure is implemented in *Salabim*. In the model, entities are generated for each shipment as well as for the various modes of transport. For the processes of these entities, servers (hold statements in Salabim) are used for the different terminal processes and the transport between the various ports or terminals. If all entities have been processed, but no mode of transport is available, the entities are placed into a Queue, which are emptied once the transporting entities have arrived.

The arrival of the entities is based on schedules, shipments on the planned AC (At Consolidator) dates. The arrival of the transport modes is based on the available sailing / train schedules. The processing duration and the transport speed (which is the key determinant for transport speed) are sampled from distributions that have been constructed based on the available data (see section A.2.2). The sampling of these processing times will benefit the variability in the data assimilation in a later stage.

A.2.4. Model components and processes

In the following section, the previously described model structures and the corresponding processes are described in more detail. The annotated code for the simulation model can be found in SC_model.py.

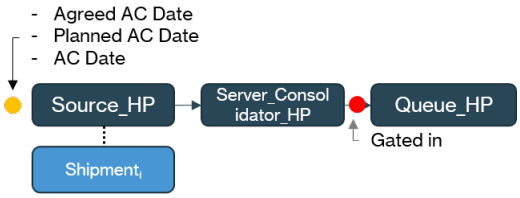
Important to note is that these descriptions relate to the simulation models used as *system transition model* (i.e., the simulation models for the particles). All adjustments for the model used as surrogate for the real system are discussed in section A.3.

Consolidator processes

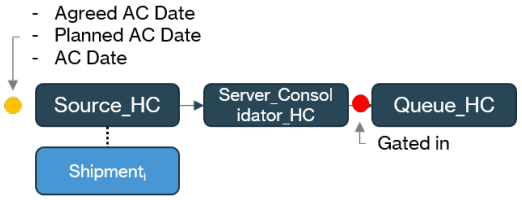
The first model structure and group of processes that are addressed are the processes at the two consolidators. These consolidators are placed at the system boundaries and shipments enter the systems at the moment of delivery by one of the vendors. The different model processes and the corresponding measurement data is visualized in figure A.3.

Arrival of shipments

The first process at the consolidators is the arrival of the shipments in Hai Phong and Ho Chi Minh. This is formalized in the model by means of two *componentGenerators* (Source_HP and Source_HC). Shipment



(a) Consolidator Hai Phong



(b) Consolidator Ho Chi Minh

Figure A.3: Consolidator processes

entities are generated based on the agreed, planned, and (actual) AC Dates. These data points are displayed by the orange dots in figure A.3. At the moment of generation, each of the shipments gets assigned its (planned) characteristics (e.g., ID, route and volume) and is subsequently declared ready for processing.

To increase the reusability of this model for different supply chains, a universal class has been generated (*model.shipmentGenerator*). The two instances of the class are defined at initialization and its location-specific behavior is adjusted by variables such as the departure schedule and the routes that should be assigned to the shipments.

The detection of the actual arrival of shipments is based on the concept of arrival detectors (see the vehicle trajectory model of Xie (2019) for more insight on this topic). At the beginning of each data assimilation interval, information is available on the actual and future (agreed and planned) shipment arrivals. These datasets are generated in notebook *03 Creating AC Arrival Sensor Data.ipynb*. Depending on the type of data point (Agreed, Planned or Actual) error terms are added to this data in the *model.readSensorArrivalData* function. More information on the size and shape of these error terms can be found in table A.1.

Processing at consolidator and Queue

After the shipments have been generated, their behavior is governed depending on their route and status. This is governed by the function *shipment.process*. This starts with the processing at the consolidator (Server _Consolidator in figure A.3), formalized in the function *shipment.processingAtConsolidator*.

Here each of the shipment is held for a period of time, sampled from a skewed normal distribution, for which the Mode, Skewness and the Size are determined by the data assimilation algorithm. An important note here is that the sampled time cannot be below 0 (as negative processing times do not exist). If a negative value is drawn, the value is sampled again.

After processing, the shipments are ready for departure. For this two FIFO (First In First Out) queue's are generated, as the shipments enter the queue and wait for the arrival of a feeder for transport to the next port.

Measurement model consolidator

At the arrival of the shipments at the system boundary the first event-based data point will be generated. To retrieve this data points, a sensor function (*model.sensor*) has been written which is based on four components. This sensor function mimics the behavior of the S&OP and carrier data warehouses. For each sensor the ID of the entity triggering the sensor (e.g., shipment), the time and the volume corresponding to the entity are recorded and stored by the the ID (location) of the sensor. Table A.6 provides an overview of the sensor ID's corresponding to this part of the model.

	EntityType	**SensorID**
AC Date HP	Shipment	1HP01_shipment_gated_in
AC Date HC	Shipment	1HC01_shipment_gated_in

Table A.6: SensorID's at consolidator

Transport to intermediate port

The second part of the component is the transportation of the shipments from the consolidators to the intermediate port in Malaysia (Tanjung Pelepas). This process is visualized in figure A.4. For the transport of shipments throughout the model a second class of entities is generated: *Transporters*. These transporters are generated and can take entities from a queue and transport them via predefined routes to a subsequent destination.

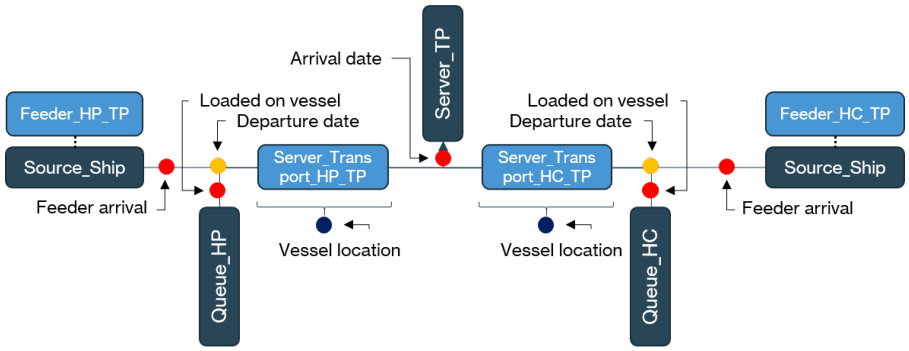


Figure A.4: Transport from consolidator to intermediate port

Arrival of feeders

The transport between the consolidators and the intermediate port is taken care of by so-called feeders. These are freight vessels with a lower capacity and a lower speed than the transcontinental vessels. To increase reproducability of this model, all transporters (feeders, vessels, trains) are defined by one class and their individual behavior is governed by parameters such as type, route and speed. The generation of the transporter entities is governed by location specific instances (*Source_HP* and *Source_HC*) of the *Transporter Generator* class.

The analysis of historical Maersk data (see notebook *02 Creating Shipping Schedules.ipynb*) has shown that the arrival of feeders is a random process, where planned dates are (almost) never met. Therefore it has been decided to model this as a process with random interarrivaltimes drawn from a triangular distribution of which the mode is determined by the data assimilation algorithm (see section A.2.2). Only actual (feeder arrival) dates are (when available in the selected scenario) assimilated into the simulation model.

Loading and unloading of the shipments

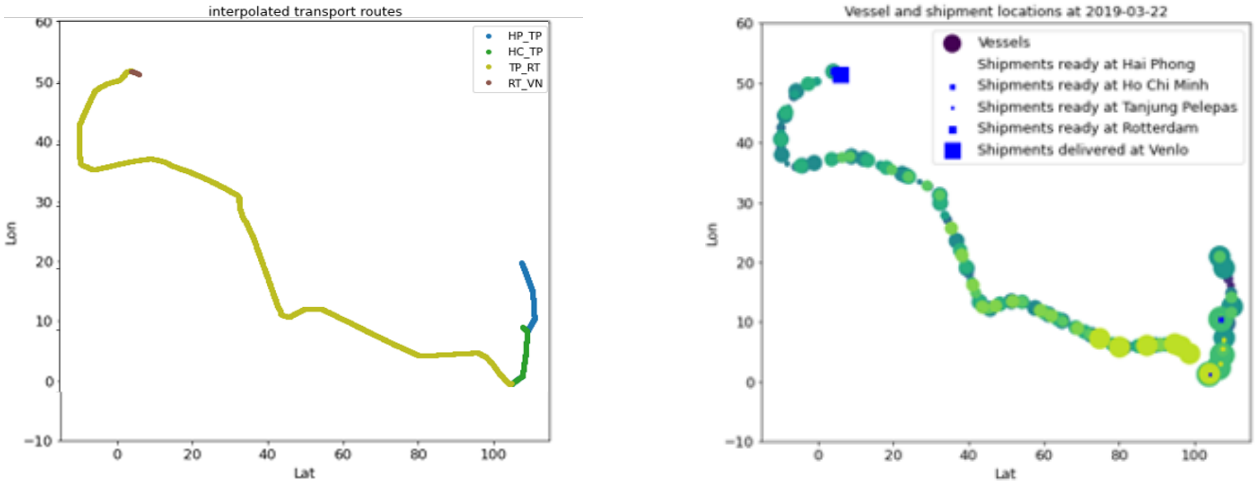
If a feeder arrives at the consolidator, the shipment entities are loaded on the feeder. For this a loading time dependent on the port of departure is sampled from a triangular distribution for which the mode is estimated by the data assimilation algorithm (see section A.2.2). The loading time is assumed to be irrelevant of the number of shipments of the stakeholder that are loaded since these are considered to be only a (relatively) small proportion of the total number of containers that are loaded on board. Based on the agreements the stakeholder has with the carrier, it can be assumed that all containers that are ready for shipment are loaded on board. The unloading of the vessels works similar to the loading.

Moving of feeders

After the loading the feeders will start sailing to the next destination. DES models are the compound of different atomic model states and these atomic model states are only updated locally if a new event takes place (see section 2.4.1). This would generally imply that the location of a transporter would only needed to be updated at departure and arrival.

However, this is considered to be undesirable since the Global Positioning System (GPS) information of vessels can be made available by third parties (see section3.3). This could be crucial data for the accuracy of the data assimilation study and therefore this needs to be available in the simulation models as well. An extensive data analysis has been performed on a dataset of historical AIS data between port provided by Novikov (2019). This analysis can be found in notebook *05 Creating coordinates for transporters.ipynb*.

Based on a sparse dataset with coordinates of routes between numbered ports, first the right ports have been identified after which the routes between these ports are interpolated to a dense set of latitude and longitude combinations. Subsequently the distance between the nearest neighbour combinations of sets are computed to create a dataset with the all latitudes and longitudes assigned to the cumulative distance on the route. In this dataset the entire transport route is divided in different sections as can be seen in figure A.5a.



(a) Coordinates of different transport routes

(b) Visualization of movement of vessels over route

Figure A.5: Transporter routes and coordinates

The actual movement of the formalized by sampling a distance sailed per day from a triangular distribution, for which the mode is estimated by the data assimilation algorithm. After each day the cumulative distance of the feeder is calculated and the nearest match with the previously discussed dataset is looked up to retrieve the latitude and longitude of the feeders. The effects of this method is a step-wise movement, which has been visualized in figure A.5. Each shade of green represents a vessel, where the size of the scatters represents the volume of shipments on that respective vessel.

Measurement model transport to intermediate port

As can be seen in figure A.4, nine different sources of data are located in this part of the supply chain. Of these sources, the moment of arrival of the feeder, the moment of loading of the shipment, the departure date of the vessel and the arrival of the feeder are event-based sensors. This data is collected with the *model.sensor* function as well. An overview of the corresponding SensorID's is given in table A.7.

	EntityType	**SensorID**
Feeder Arrival HP	Transporter	2HP04_transporter_out_arrived
Feeder Arrival HC	Transporter	1HC04_transporter_out_arrived
Loaded On Vessel HP	Shipment	2HP05_shipment_loaded
Loaded On Vessel HC	Shipment	1HC05_shipment_loaded
Departure Date HP	Transporter	2HP06_transporter_out_departed
Departure Date HC	Transporter	1HC06_transporter_out_departed
Arrival Date TP	Transporter	3TP02_transporter_in_arrived

Table A.7: SensorID's transport to intermediate port

Location-based sensor data

One set of measurements has not been discussed yet; the location-based sensor data. These are the locations of the vessels. Because of the previously discussed state retrieval problem (see section 2.4.1) the locations of transporters is not always updated at the end of a data assimilation interval.

To this extent, a location interpolation function (*model.interpolateLocation*) has been designed that interpolates the location of all active transporters based on the previous location, time sailed and speed. Subsequently the locations of all active vessels are assigned to the transporter ID and stored in a global dataset with the location data of all vessels.

Intermediate port processes

After the arrival of the feeder at the intermediate port the goods can be unloaded and processed at the intermediate port. These processes are similar to the loading and processing of the goods at the terminal; the mode of the processing time is estimated by the data assimilation process as well. After the processing the shipments are ready for departure and enter a FIFO queue. These processes have been visualized in figure A.6.

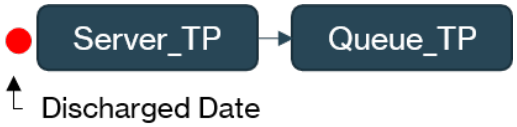


Figure A.6: Intermediate port processes

Measurement model intermediate port processes

As displayed in figure A.6, only one source of sensor data is located at the intermediate port. This is the sensor triggered at the unloading of the shipments at the intermediate port. This is implemented in the same way as the retrieval of all other event-based sensor data. The characteristics of this sensor are summarized in table A.7

	EntityType	**SensorID**
Discharged Date TP	Transporter	3TP3_shipment_unloaded

Table A.8: SensorID's intermediate port

Transport to final port processes

At the intermediate port, intercontinental fright vessels depart to bring the shipments to the final port (Rotterdam). This transport is set up based on the same characteristics (functions) as the transport from the consolidators to the intermediate port. The vessels arrive based on a triangular distribution with a mean that is estimated by the data assimilation algorithm. The processes are visualized in figure A.7.

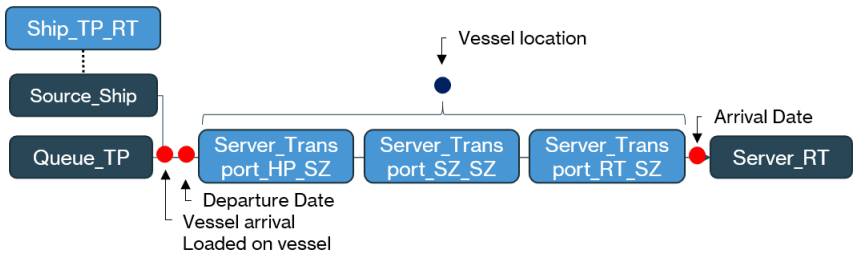


Figure A.7: Transport to final port processes

Different route segments

The moving of the vessels is formalized the same way as described for the feeders, with three main differences. First, the vessels follow a different route (see figure A.5b). Second, the vessels have a different (higher) speed than the feeders which is sampled by the data assimilation algorithm. The third difference is the most crucial difference, this relates to the different route segments.

The route from Tanjung Pelepas (Malaysia) to Rotterdam has been cut into three different segments. The first segment is from Tanjung Pelepas to the Suez (SZ) Canal, the second is the Suez Canal itself and the

third is the Suez Canal to Rotterdam. It has been decided to separate these segments since the speed of the vessels is not the same for these segments; therefore the average speed of the vessels will be lowered for the Suez Canal.

Measurement model transport to final port processes

As one can see in figure A.7 four different potential sources of data to be assimilated can be identified. First, the moments of arrival of the vessels at the intermediate port, the loading of the shipments on the vessel and the departure of the vessel are recorded by event-based sensors. Subsequently the location of the vessel is recorded and at arrival at the final port (Rotterdam), an event-based sensor is triggered again. This is all implemented exactly the same as for the feeders; an overview of the event-based sensor ID's is given in table A.9.

	EntityType	SensorID
Vessel Arrival TP	Transporter	3TP04_transporter_out_arrived
Loaded on vessel TP	Shipment	3TP05_shipment_loaded
Departure Date TP	Transporter	3TP06_transporter_out_departed
Arrival RT	Transporter	4RT02_transporter_in_arrived

Table A.9: SensorID's transport to final port

Processes in country of destination

After the vessel has arrived in the final port, the goods are in the country of destination. Here the goods are offloaded of the vessel, processed and subsequently transported by train to the terminal. The processing of the shipments and the entering of the queue at the final port is modelled by the same building blocks as the processing of the shipments at the other ports. Therefore this is not further elaborated on.

The same accounts for the transport of the shipments to the terminal by train; the only significant difference here is that the location of the trains is not recorded since this is not available in the real system. The order of the processes is visualized in figure A.8.

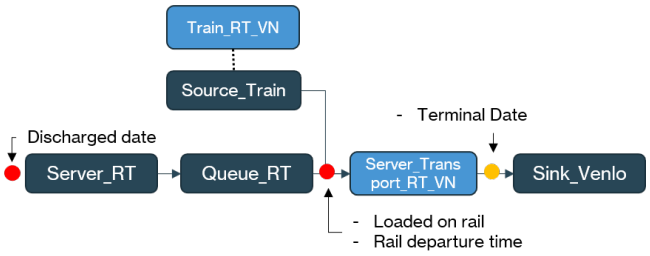


Figure A.8: Processes in country of destination

Arrival at terminal

After the arrival of the train at the terminal, the shipments are unloaded and deactivated. At this moment the time of delivery is recorded separately (apart from the normal sensor data) which eventually is used to make estimations on the arrival times.

Measurement model country of destination

In the country of destination, five different event-based sensors can be triggered. As previously discussed, no location data on the trains is available in the real system and therefore this is not modelled in the simulation model as well. An overview of the event-based sensor ID's is given in table A.10.

	EntityType	SensorID
Discharged date RT	Transporter	4RT03_shipment_unloaded
Rail arrival time RT	Transporter	4RT_04_transporter_out_arrived
Loaded on rail RT	Shipment	4RT_05_shipment_loaded
Rail departure time RT	Transporter	4RT_06_transporter_out_departed
Rail arrival Time	Transporter	5VN02_transporter_in_arrived
Terminal Date VN	Transporter	5VN03_shipment_unloaded

Table A.10: SensorID's country of destiantion

A.3. Real system

As previously described, a simulation model will be used as a surrogate for the real system in a identical twin setup. This because the supply chain planning cycle encompasses multiple months and therefore it is infeasible to test and develop the proposed framework with an actual physical system. However, the identical twin model will be based on real-world data.

This real system is an instance of the previously described simulation model, with the global state variable groundTruth as True instead of False (which is used for the particles). Because of this difference, the behavior of this simulation model is different than that from the particles. This difference is found in three major aspects: 1) The arrival of shipments is based on a schedule, 2) The schedule of transporters is based on a schedule, and 3) The model parameters are fixed. Each of these aspects will be discussed in the following sections.

A.3.1. Arrival of shipments in the real system

Instead of basing the he arrival of shipments on sensor data with different levels of errors, the arrival of shipments in the model used for the real system is based on historical data. Based on the S&OP data of the stakeholder, a schedule has been generated containing the arrival days, locations, ID's and volumes of shipments. This schedule is split based on the location of the consolidators and subsequently used to generate the shipment entities at the right moment in time. The processing times at the consolidator are sampled and added to this schedule as well. The process for creating this schedule can be found in notebook *04 Creating the ground truth shipment arrival and AC processing times.ipynb*.

A.3.2. Arrival of transporters in the real system

For the arrival of transporters (feeders, vessels and trains) a similar approach is adopted; for each of the different transporter generators a schedule is used as input. However, no historical data on transporter arrival within the same time frame of the historical S&OP data was available. To generate this missing data, historical Maersk data is to create a new artificial schedule.

As the goal of this research is to provide a proof of concept, this is considered to be no threat to the validity of the results. More insight on the process of creating these schedules can be found in notebook *02 Creating Shipping Schedules.ipynb*.

A.3.3. Real model parameters

The model parameters that steer the behavior of the system are estimated by the data assimilation algorithm. These parameters are fixed in the model that is used as surrogate for the real system. These parameters are set after iteratively testing and running the model until the variability in the shipment delivery days at terminal was comparable with the variability as shown in figure 3.4.

B
Reflection on Salabim as simulation environment

The discrete event simulation model that is used for this research has been implemented in Salabim. Salabim is a good discrete event simulation environment for traditional what-if simulations or even exploratory modelling. However, using this environment for a DES DA exercise in combination with *MultiProcessing* turned out to be challenging since Salabim makes use of generators. The different benefits and the major downside of using Salabim for DES DA will be discussed in this chapter, combined with the solution to the identified challenge.

B.1. Benefits of Salabim for DES DA

Using Salabim as simulation software to create the simulation environment for the particles and the real system had several benefits to other simulation environments. Each of the benefits will be discussed in the this section. These benefits can be a reason to use Salabim in future research or should be present in another simulation software package.

First, a large benefit of Salabim is the fact that this software is *open source software*. Many other commercial simulation software packages such as Simio require a license to be purchased. These licenses can be very expensive, especially for commercial application. Apart from this the fact that Salabim is open source means that the code is freely available and can be if adjusted if desired.

The second benefit of Salabim is the fact that this is a Python-based package. This makes connecting the simulation models to the data assimilation algorithm a relatively straightforward process since the data structures (e.g., lists, dictionaries) match. Simulation packages such as Simio generally write data to .csv files first which complicates connecting the simulation environment to the data assimilation algorithm.

The third significant benefit of Salabim is the compatibility with *MultiProcessing* for a faster execution of the data assimilation algorithm. DES DA generally requires a high number of particles to approximate a posterior probability density function. Because each particle is represented by an instance of a simulation model, the time it takes to execute a simulation model is very important. This time can be tremendously increased by running multiple simulation models in parallel. The *MultiProcessing* package in Python is used to do so and because Salabim is a Python-based package the different simulation models can be executed in parallel.

B.2. Downside of using Salabim for DES DA

Using Salabim over the course of this research surfaced a large downside this simulation software package as well: saving simulation models (particles) that have been executed to reconstruct the *offspring particles* again after the resampling step is very difficult.

As the particles are executed by multiProcessing to reduce the duration of the data assimilation exercise, the instances of the models including their states need to be pickled to make them accessible in the main frame. Because Salabim makes use of Generators, it is impossible to save the actual instances of the model since these cannot be *Pickled*.

Therefore a solution is found: At the end of the data assimilation interval all *states* of the simulation models are saved and at the beginning of a new simulation interval the simulation model is *reconstructed*.

This way of working only allows to save the model states, all model components and processes cannot be saved (because of the presence of generators). Therefore all model components need to be reconstructed at the start of a new assimilation interval and the original states need to be reinstated. For the fixed model

components (e.g., *shipmentGenerators*) this is no problem. However, for the entities (e.g., *shipments*) this is more challenging. It needs to be assured that every reconstructed entity starts at the exact same process and state as it left. This makes the internal logic of the model more complex.

To summarize, generating offspring particles is a very complex procedure since the instances of the simulation models cannot be saved together with the model states if MultiProcessing is used. Since MultiProcessing is crucial for performing these experiments within an acceptable period of time another solution is found. However, as the internal logic of the simulation model is adjusted, the complexity of the simulation model is significantly increased. Therefore it is recommended not to use Salabim for future research. If no alternative is found, the adjustments to the simulation model are discussed below so Salabim still can be used in future research.

B.2.1. Adjustments to the simulation model

To facilitate this, three significant adjustments have been made to the simulation model: 1) Extra processes have been added to save the states of each of the model components and the global model states. 2) Extra processes have been added to allow for the reconstruction of the simulation model. 3) The internal logic of the model processes have been altered to make sure regenerated entities follow the right logic. These adjustments are presented below.

Saving of model states

At the end of an assimilation interval, all model states need to be saved. The model state can be divided in three groups which are consecutively saved. First a set of all transporters and its states is generated. For all transporters, a state vector of all states that belong to that transporter is coupled to its ID. These ID - state vector tuples are stored together. Subsequently the same procedure is applied to all shipments in the system. At last all global model states (e.g., sensor data and locations of transporters) are stored together. These three sets of states are stored as a pickle. This has been formalized in the *model.saveLastState* function, for which the pseudo code has been presented in algorithm 10.

Algorithm 10: Saving of model states

Input: Simulation model
Output: Sets of state vectors of simulation model

Function `saveLastState()`**:**

```
// Retrieve transporter state vectors
for transporter in transporters do
    for State in transporter.__dict__ do
        if State contains no Generators then
            Add state to state vector
    Couple state vector to transporterID and add to set of transporter states
// Retrieve shipment state vectors
for shipment in shipments do
    for State in shipment.__dict__ do
        if State contains no Generators then
            Add state to state vector
    Couple state vector to shipmentID and add to set of shipment states
// Retrieve global states
for State in global states do
    if State contains no Generators then
        Couple state to state ID and add to set of global
```

Reconstruction of the simulation model

At the beginning of a new assimilation interval, the resampled models that represent the *offspring particles* are reconstructed. This starts with an *empty* simulation model for which *no* components (e.g., generators) have been created. Subsequently the simulation time is advanced to the time of the start of the data assimilation interval. At that moment the function *model.setModelStates* is called to recreate all the saved model entities and their states as well as the global model states. The pseudocode for this function is presented in algorithm 11

As this function is called, two additional generators are created. This are different generators than the generators that create the entities during the simulation. These generators allow to create new entities and replace their states with the original saved state vectors. After all original entities have been generated, the additional generators are deactivated. Thereafter, all saved global states are copied from the saved global state vector.

The result of these steps is a simulation model with all previous states and entities. To finish this step, all fixed model components are generated as they were by the function *model.generateComponents*. This finalizes the process of reconstructing the simulation model.

Algorithm 11: Reconstructing the model

Input: Sets of state vectors of simulation model
Output: Simulation model

Function `setModelStates()`**:**
- Create a transporter reconstruction source (model.transporterGenerator2)
- `// In the transporter reconstruction source:`
- **for** *transporter in saved set of transporter states* **do**
  - Generate new transporter and replace states with saved sate vector
- Create a shipment reconstruction source (model.shipmentGenerator2)
- `// In the shipment reconstruction source:`
- **for** *shipment in saved set of transporter states* **do**
  - Generate new shipment and replace states with saved sate vector
- **for** *saved state in saved global state vector* **do**
  - Replace model state with saved state

Adjustment of internal logic of model processes

As the entities are regenerated, the entities should pick up exactly the same processes as they were executing at the moment they were saved. For example, shipments that were being processed at the consolidator should continue this process for the remaining duration. For this a set of extra state variables is assigned to each of the regenerated entities. In each of the processes additional logic is added to make sure the entities do not start over with their processes but continue their processes. More insight of the verification and validation of this can be found in the notebook *07 Verification of reconstruction of the model .ipynb*

B.3. Recommendations for picking a simulation package for future DES DA

Based on the experiences of this research it is recommended to use a *different* simulation package for future DES DA research. The process of saving model states, reconstructing the simulation models and adjusting the internal logic of the models makes connecting the simulation environment to the data assimilation algorithm an unnecessarily difficult process.

To select a discrete event simulation software package for future research it is recommended to prioritize on the following four factors in descending order of importance:

1. It should be possible to execute multiple instances of the simulation model in parallel.

2. It should be possible to save instances of the simulation model together with their states at any moment, closing them and opening and starting them again in multiprocessing.
3. The simulation software package should be based on the same data structures as the high-level programming language in which the data assimilation algorithm has been written.
4. The simulation software package should be open source.

C
Additional results of experiments

In the results section, examples of the most occurring types of clusters of shipment arrival estimations have been analyzed. All visualizations of the set of arrival estimations can be found in the folder *3_Output1_Figures3_Arrival_Estimation_Histograms*. To give the reader extra understanding of the different typologies of clusters of shipment arrival estimations per experiment, the different typologies that are visible in each experiments are discussed. The data assimilation dashboards with the traces of the particles, the particle weights and the duration of the sampling step are presented as well.

C.1. Experiment *E. 100%*

Experiment *E. 100%* is the experiment with all data points available. The results presented in chapter 6 show that the data assimilation algorithm can achieve a high level of accuracy given these data points. Next to this, the results get more accurate as the degree of timeliness is reduced.

C.1.1. Shipment arrival estimation clusters

Three different types of weighted histograms are visible in the results of experiment *E. 100%*: The major of the shipment arrival estimations clusters are characterized by an increasing confidence (narrower peak) and accuracy (lower absolute distance) for just one peak. The second and third type are examples of the *two-peak estimation* discussed in chapter 6 and 7. The first shows an increasing accuracy and the other shows a decreasing accuracy. An overview of the occurrence of each type is given in table C.1. On the next page, each type of shipment arrival estimation clusters is discussed based on one example.

	Count	**Share**
One-Peak	230	72.1%
Two-Peak high accuracy	62	19.4%
Two-Peak low accuracy	27	8.5%

Table C.1: Types of arrival estimation clusters experiment *E. 100%*

One peak, high confidence and accuracy

The majority of the weighted histograms of clusters of shipment arrival estimations look similar to the example presented in figure C.1. As the degree of timeliness is reduced, the particles become more uniform in their shipment arrival estimations and the peak of the weighted histogram becomes more narrow. This is a sign of increasing confidence. In the shipment arrival estimations for this type only one accurate peak is visible, where the top of the peak matches the observed arrival date closely.

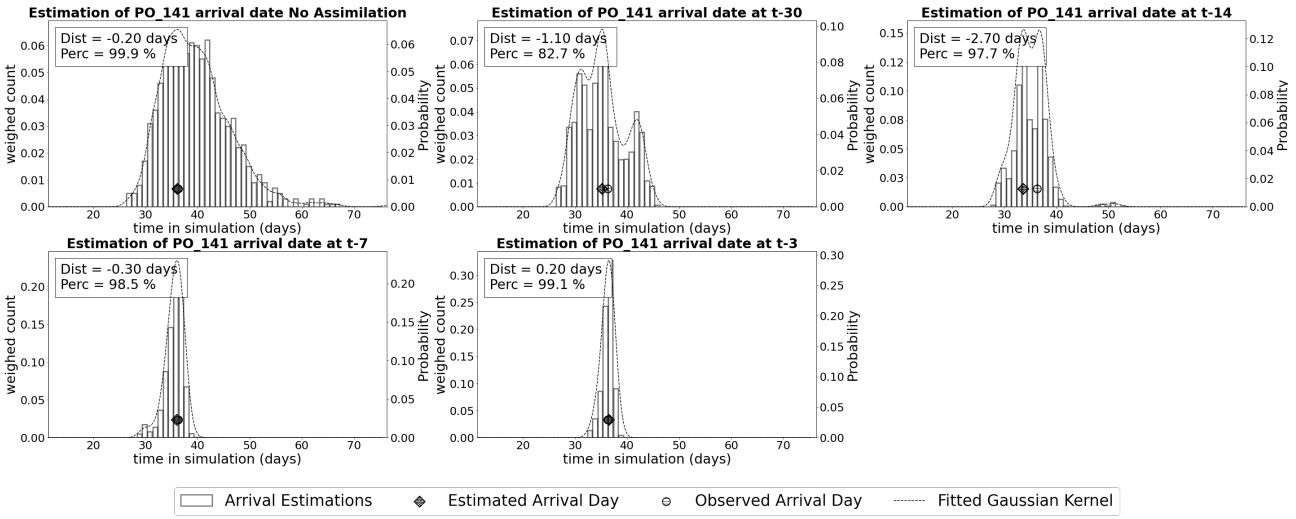


Figure C.1: Weighted histograms of arrival predictions of particles of shipment PO_141, experiment *E. 100%*

Two peaks, high confidence and accuracy

The second type of weighted histograms show an increase in the confidence in the estimated arrival dates as the peaks become increasingly narrow as the degree of timeliness is reduced. However two peaks are observed which is an indicator of two arrival dates becoming (almost) equally probable. An example of this is presented in figure C.2. The highest peak determines the estimated arrival date, in this set the highest peak matches the observed arrival date and therefore the accuracy is still increased over time.

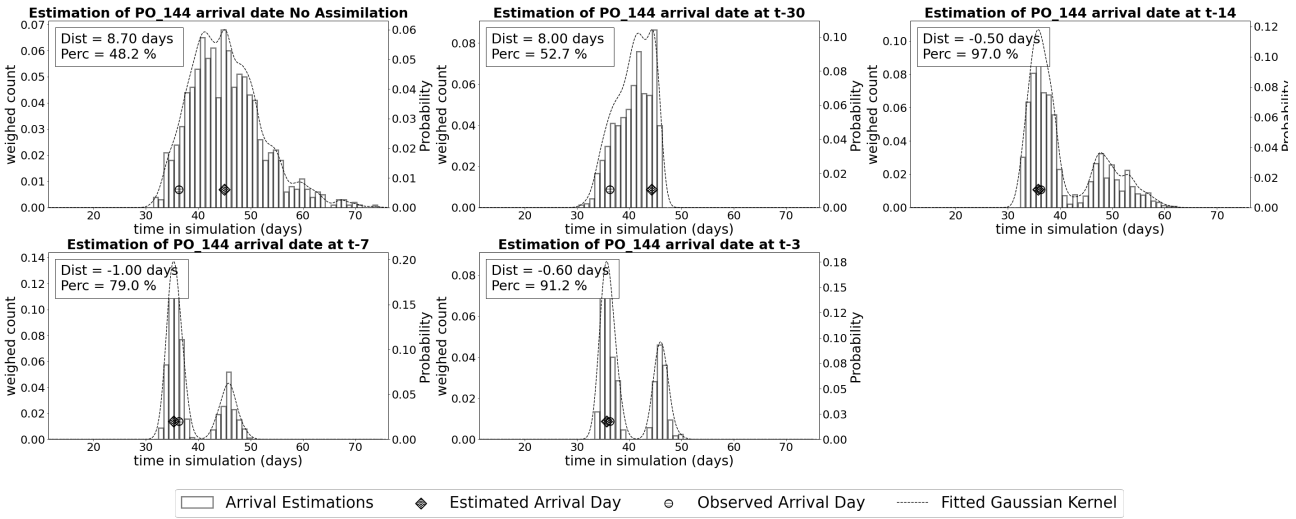


Figure C.2: Weighted histograms of arrival predictions of particles of shipment PO_144, experiment *E. 100%*

Two peaks, high confidence and low accuracy

The results of experiment *E. 100%* have shown to be highly accurate, however there are some shipments with a relatively high absolute distance. These are the result of a *two-peak estimation* where the highest peak does not match the observed arrival date. What is interesting to see is that none of the observed arrival dates were completely missed by the particles as the secondary (smaller) peak matches the observed arrival date. An example of this is presented in figure C.7.

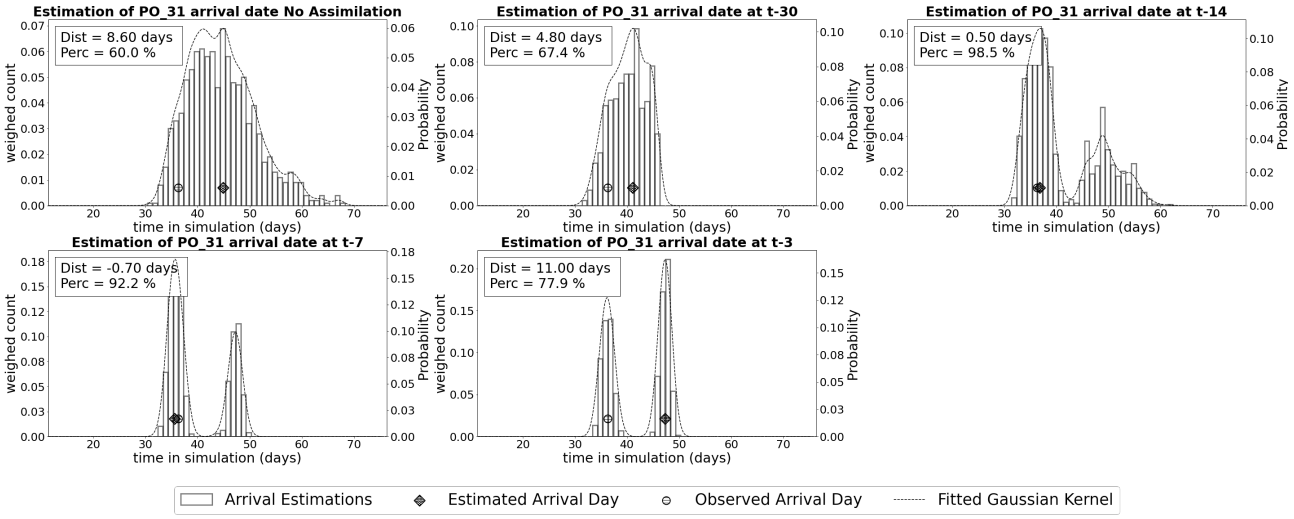


Figure C.3: Weighted histograms of arrival predictions of shipment PO_31, experiment *E. 100%*

C.1.2. Data assimilation dashboard experiment *E. 100%*

The data assimilation dashboard for experiment *E. 100%* is presented in figure C.4. Here the traces over the uncertain state variables are presented in grey. The actual values for these variables are presented in green. The data in experiment *E. 100%* shows to be able to steer the particles in the right direction. This is confirmed by the traces of the particle weights, in each data assimilation interval, sufficient measurement come in to provide a high variation in the weights.

Figure C.4: Data assimilation dashboard E. 100%

C.2. Experiment *E. 91%*

91 percent of all available data is present in experiment *E. 91%*, only the *third party* vessel location data points are omitted. The results presented in chapter 6 show that experiment *E. 91%* yields less accurate results compared to *E. 100%* especially for estimations with a low degree of timeliness.

C.2.1. Shipment arrival estimation clusters

Analysis of weighted histograms for experiment *E. 91%* provides more insight on why the the results show a larger weighted average absolute distance in experiment *E. 91%* compared to *E. 100%*. The types of weighted histograms are the same for both experiments, there are confident *one-peak estimations*, and two types of *two-peak estimations*. One type where the highest peak matches the observed arrival date (accurate estimations) and one where the highest peak does not match the observed arrival date (inaccurate estimations).

However, the difference compared to *E. 100%* is found in two factors. First, the *one-peak estimations* are less accurate than those of *E. 100%* and the share of inaccurate *two-peak estimations* is higher for experiment *E. 91%* than *E. 100%*. This is presented in table C.2. Examples of each of the types of shipment arrival estimation clusters are presented below.

	Count	**Share**
One-Peak	211	66.1%
Two-Peak high accuracy	48	15.0%
Two-Peak low accuracy	60	18.8%

Table C.2: Types of arrival estimation clusters experiment *E. 91%*

One peak, high confidence and medium accuracy

The majority of the clusters of shipment arrival estimations look similar to the weighted histograms of estimations for shipment PO_22 presented in figure C.5. As the degree of timeliness is reduced the peak of the estimations become increasingly narrow indicating an increase of confidence in the estimated arrival day. However, compared to *E. 100%* the top of the peak generally has a higher absolute distance to the observed arrival date.

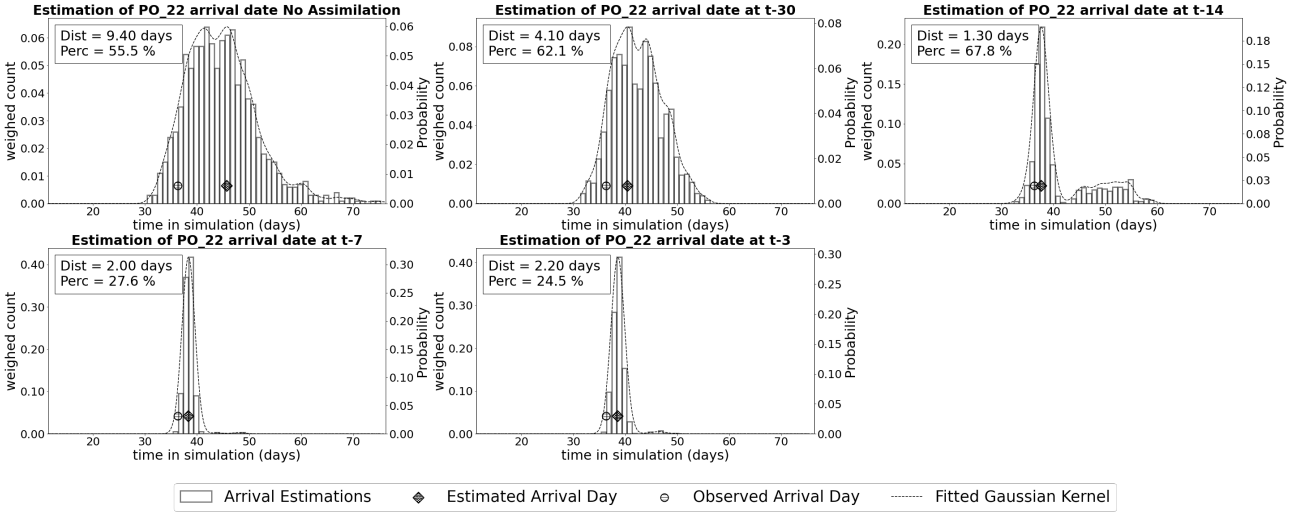


Figure C.5: Weighted histograms of arrival predictions of particles of shipment PO_22, experiment *E. 91%*

Two peaks, high confidence and high accuracy

An example of the second type of clusters is presented in figure C.6. Here the confidence in the arrival estimations is increased (visible by the narrow peaks), however two peaks of estimated arrival dates are visible. This type of estimations differentiates itself from the other *two-peak estimations* because the top of the highest peak matches the observed arrival date, resulting in a high accuracy. The share of this type of arrival estimations is low for experiment *E. 91%*.

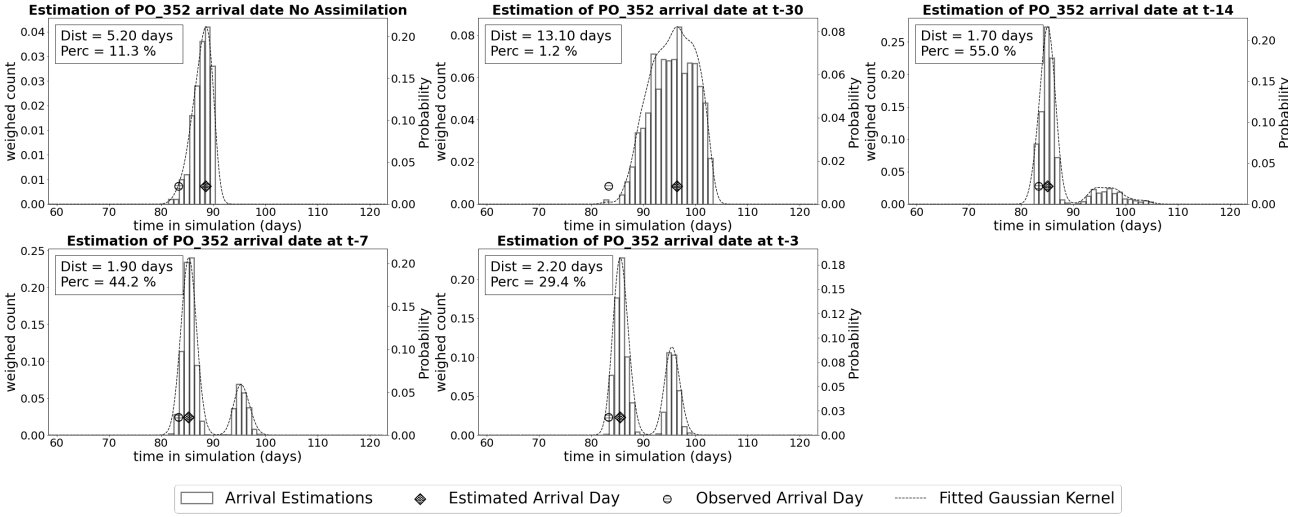


Figure C.6: Weighted histograms of arrival predictions of particles of shipment PO_352, experiment *E. 91%*

Two peaks, high confidence and low accuracy

The third set of clusters can also be identified by two peaks with an increasing level of confidence. However, here the highest peak does not match the observed arrival date resulting in a high absolute distance (i.e., a low accuracy). The share of this type compared to the *two-peak estimations* with a high accuracy is high for experiment *E. 91%*.

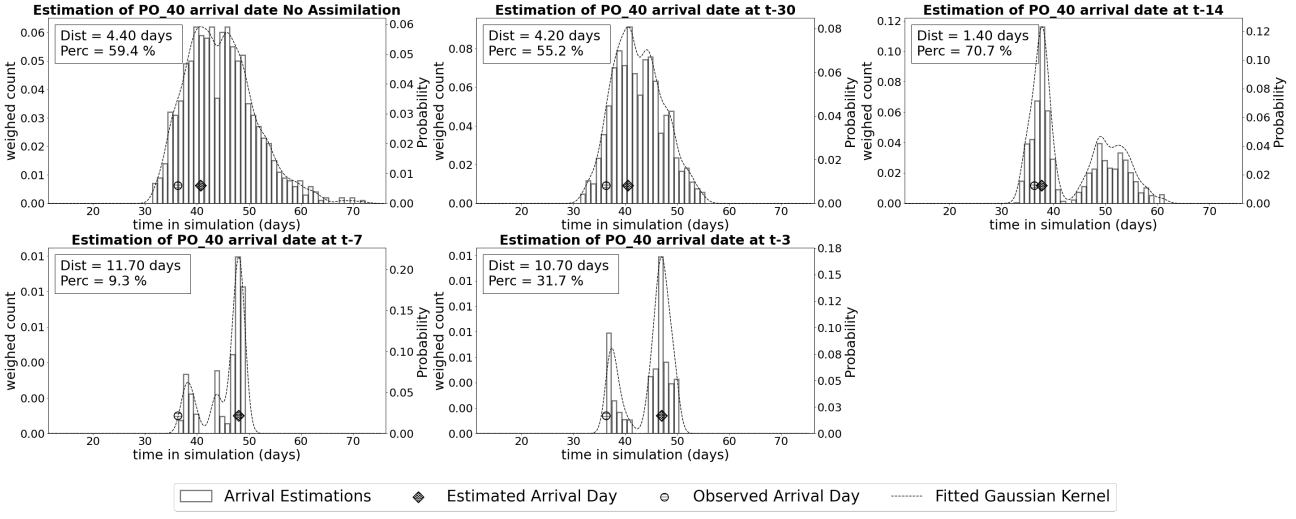


Figure C.7: Weighted histograms of arrival predictions of particles of shipment PO_40, experiment *E. 91%*

C.2.2. Data assimilation dashboard experiment *E. 91%*

The data assimilation dashboard for experiment *E. 91%* is presented in figure C.8. The traces over the uncertain state variables are presented in grey. The actual values for these variables are presented in green.

It is clearly visible that the data points lost between experiment *E. 100%* and *E.91%* were important for maintaining sufficient spread between the particles. As is visible in the spread of the particle weights, there are multiple moments in time (i.e., t=28 or t = 68) where the majority weights are assigned to a few particles resulting in the drifting off of the particles. This reduces the variability in the set of potential futures that can be explained by the particles. Which is an explanation for the reducing accuracy of arrival estimations with a low degree of timeliness.

Figure C.8: Data assimilation dashboard E. 91%

C.3. Experiment *E. 29%*

In experiment *E.29%* less than one third of the available data points are used. Interesting enough the results show to very accurate, comparable to the results of experiment *E. 100%* where all available observations are used. As the degree of timeliness is reduced, the results of experiment *E. 29%* become more accurate as well.

C.3.1. Shipment arrival estimation clusters

Analysis of the weighted histograms for experiment *E. 25%* shows very similar types of shipment arrival estimation clusters as experiment *E. 100%*. A very large share of the clusters show to have an increasing level of confidence over time by one peak and a small subset of the cluster shows two peak of estimated arrival dates. Similar to experiment *E. 100%*, the largest share of the *two-peak estimations* are accurate. An overview is given in table an example of each type is discussed below.

	Count	**Share**
One-Peak	241	75.5 %
Two-Peak high accuracy	60	18.8 %
Two-Peak low accuracy	18	5.6 %

Table C.3: Types of arrival estimation clusters experiment *E. 29%*

One peak, high confidence and accuracy

The vast majority of the shipment arrival estimation clusters for experiment *E. 29%* result in one peak on the weighted histograms. The top of the peaks match the observed arrival dates, resulting in accurate estimated arrival days. What is interesting to see, that compared to the previously discussed experiments, the peaks become increasingly narrow however with a lower speed. Especially for the estimations for t_{-7} the particles still represent a large set of potential futures. An example of this type of arrival estimations is given in figure C.9.

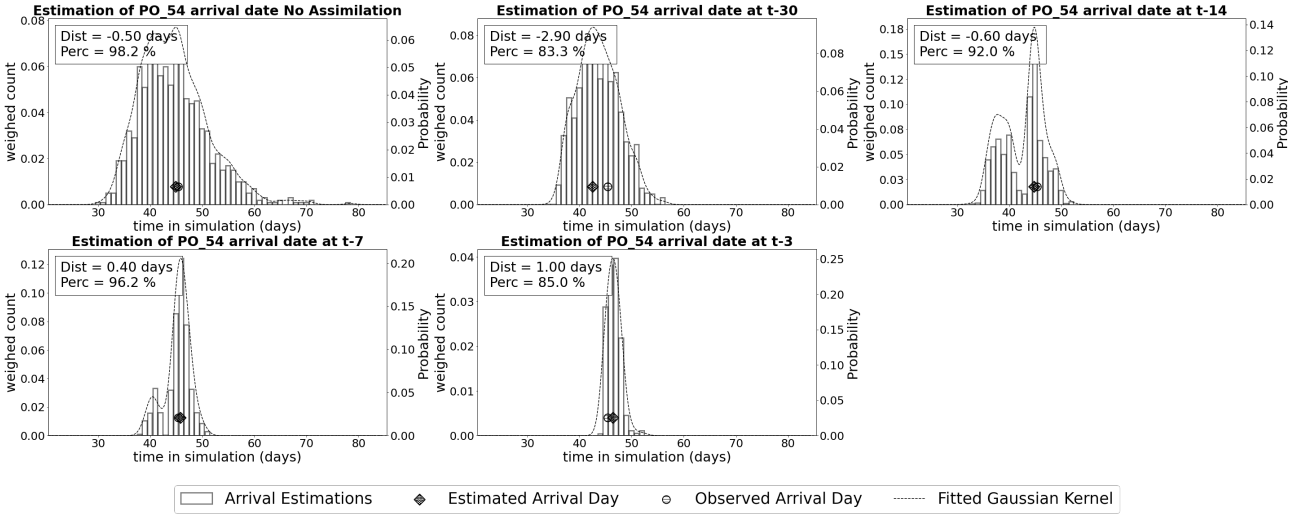


Figure C.9: Weighted histograms of arrival predictions of particles of shipment PO_54, experiment *E. 29%*

C.3.2. Two peaks, high confidence and accuracy

The share of the *two-peak estimation*s is surprisingly low in experiment *E. 29%*. Only a small share of the arrival estimations are the result of a weighted histograms with two peaks and if so, the accurate peak is generally much higher than the inaccurate peak. An example of this type is given in figure C.10.

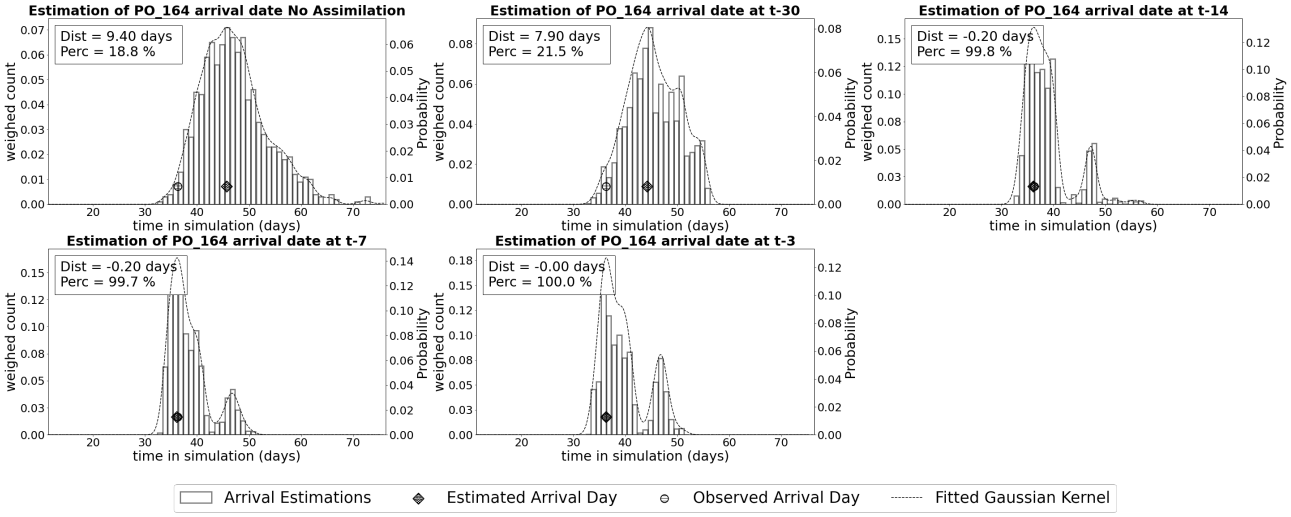


Figure C.10: Weighted histograms of arrival predictions of particles of shipment PO_164, experiment *E.29%*

C.3.3. Two peaks, high confidence, low accuracy

The last type of estimations is the group of *two-peak estimations* with a low accuracy. This type embodies the smallest share of the arrival estimations, an example of this is given in figure C.11. The low accuracy is the result of two peaks, where the largest peak does not correspond to the observed arrival date.

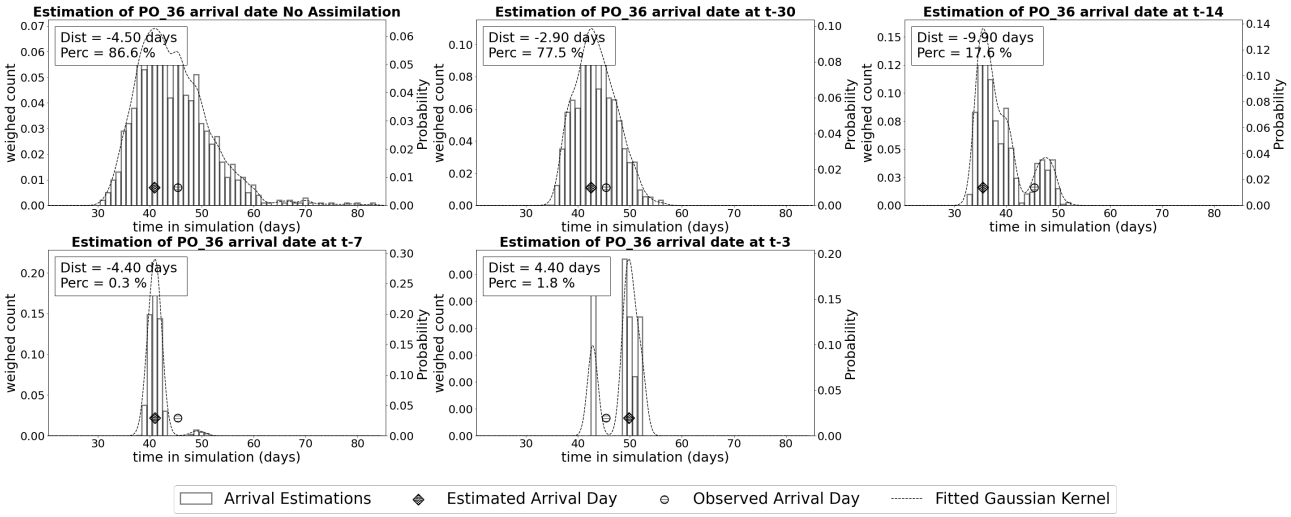


Figure C.11: Weighted histograms of arrival predictions of particles of shipment PO_36, experiment *E.29%*

C.3.4. Data assimilation dashboard experiment *E. 29%*

To finalize the overview of the results of experiment *E. 29%*, the data assimilation dashboard is presented in figure C.12. Here the traces over the uncertain state variables are presented in grey. The actual values for these variables are presented in green.

It is interesting to see that the particles are clearly steered by the data assimilation algorithm in this experiment. However, not for all parameters the particles are steered into the right direction. Given the accurate results, this could be an indication that the algorithm prioritizes the most important parameters that explain the highest variation.

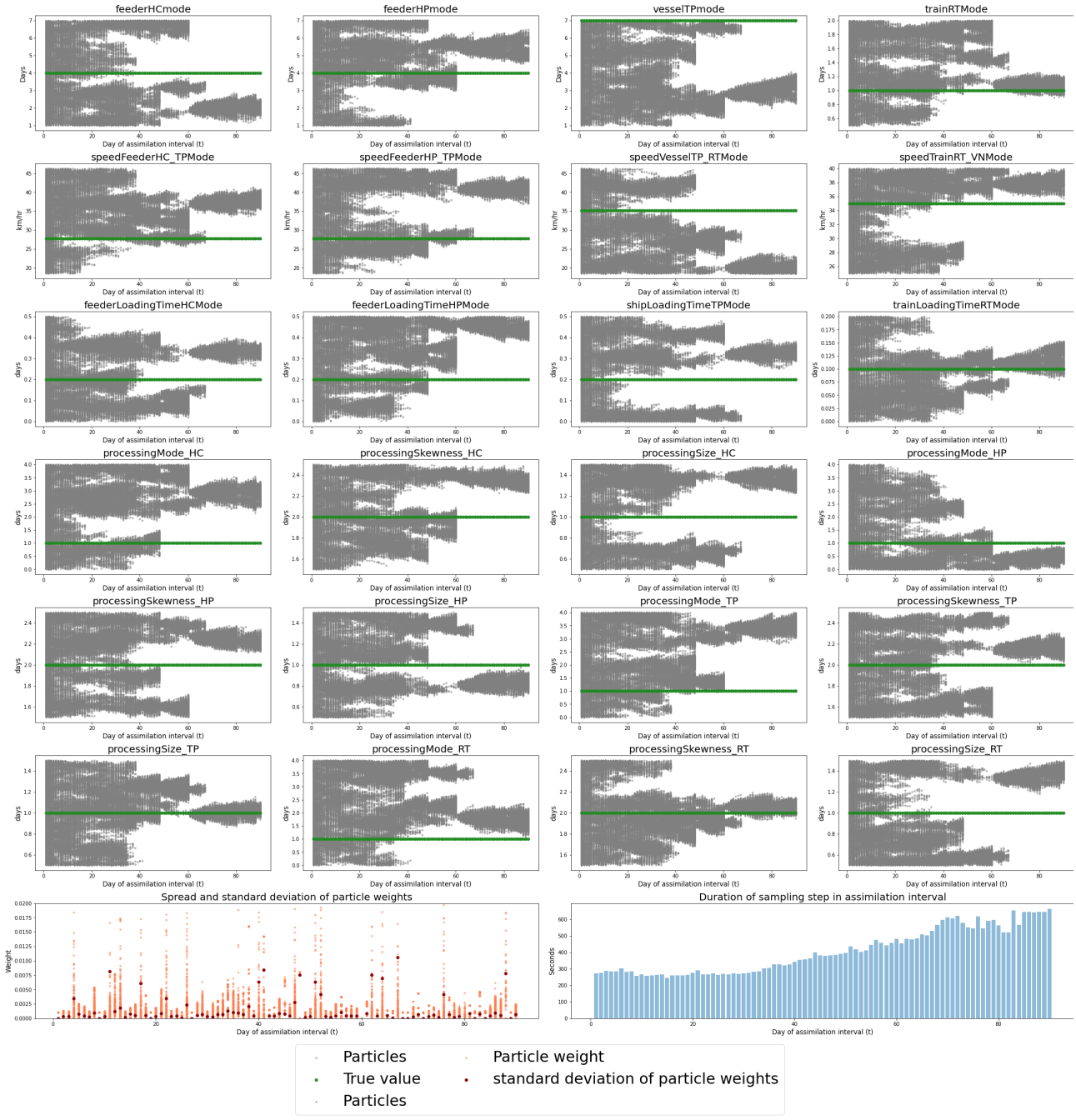


Figure C.12: Data assimilation dashboard E. 29%

C.4. Experiment *E. 25%*

The last experiment is the experiment with only 25 percent of all available observations (S&OP data points) used. The results presented in chapter 6 show that these data points are clearly insufficient to produce accurate shipment arrival estimations. The results have a lower level of accuracy than all other experiments.

C.4.1. Shipment arrival estimation clusters

Analysis of the weighted histograms for experiment *E. 25%* shows completely different clusters of shipment arrival estimations than all the other experiments. For this experiment only on type of clusters is present, an estimation with one peak with no increase in confidence and a low accuracy. An example of this type is presented below.

C.4.2. One peak, low confidence, low accuracy

The weighted histograms belonging to PO_41 presented in figure C.13 is exemplaric for experiment *E. 25%*. For each of the shipments the accuracy is reduced together with the degree of timeliness. The algorithm become very consistent in producing inaccurate estimations. It is also clearly visible that the 25 percent of the data available in this experiment is insufficient for the data assimilation process since the peaks to not become more narrow over time,

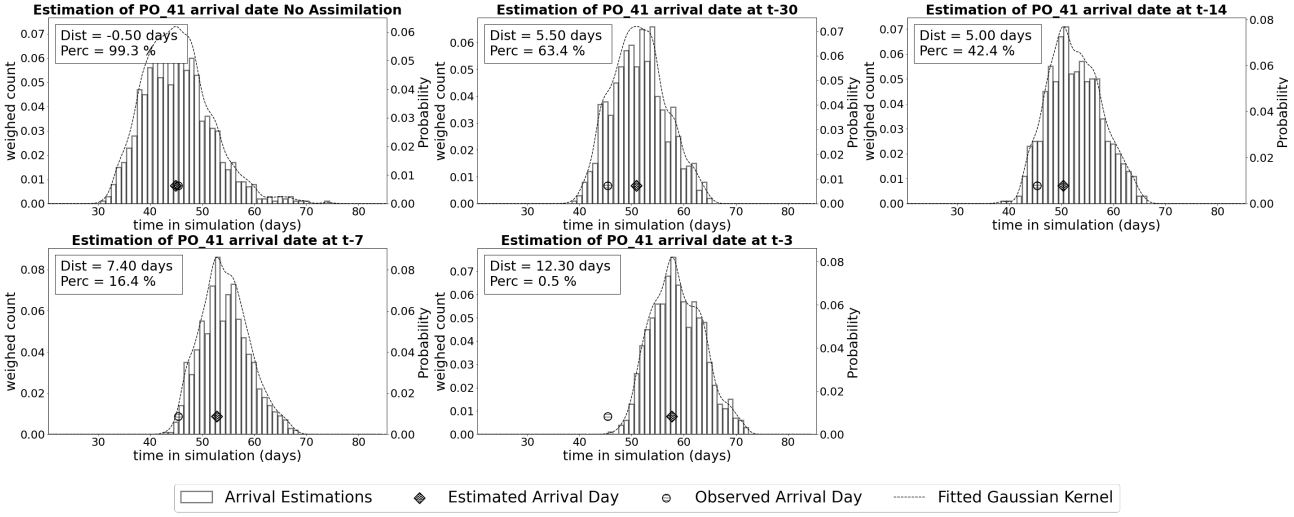


Figure C.13: Weighted histograms of arrival predictions of particles of shipment PO_41, experiment *E. 25%*

C.4.3. Data assimilation dashboard experiment *E. 25%*

At last the the data assimilation dashboard for experiment *E. 25%*is presented in figure C.12. The traces over the uncertain state variables are presented in grey. The actual values for these variables are presented in green.

The observation that the 25 percent of the data available in this experiment is insufficient to steer the particles becomes even more obvious in this dashboard. No significant weights are assigned to the particles in the vast majority of the data assimilation interval which is visible by the low standard deviation of the weights. Apart from this the particles to not converge in the state space, which is visible by the large spread of the particles.

Figure C.14: Data assimilation dashboard E. 25%