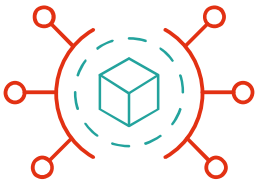


New ICT infrastructure and reference architecture to support Operations in future PI Logistics NETworks

D2.13 Intelligent Optimization of PI Containers and PI Means in PI Nodes (v2)

Document Summary Information

Grant Agreement No	769119	**Acronym**	ICONET
Full Title	New **ICT** infrastructure and reference architecture to support **O**perations in future PI Logistics **NET**works		
Start Date	01/09/2018	**Duration**	30 months
Project URL	https://www.iconetproject.eu/		
Deliverable	D2.13 Intelligent Optimization of PI Containers and PI Means in PI Nodes (v2)		
Work Package	WP2		
Contractual due date	31/01/2020	**Actual submission date**	30/06/2020 (Resubmitted)
Nature	Other	**Dissemination Level**	Public
Lead Beneficiary	IBM		
Responsible Authors	Gabriele Ranco (IBM), Kieran Flynn (IBM), Gordon Doyle (IBM)		
Contributions from	CLMS, ELU		

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Grant Agreement No 769119.

Disclaimer

Copyright message

Table of Contents

List of Figures

List of Tables

Glossary of terms and abbreviations used

Abbreviation / Term	Description
API	Application Programming Interface
BPP	Bin Packing Problem
DI	Digital Internet
GA	Grant Agreement
GRPP	Guillotine Rectangular Packing Problems
Intra/Inter-AS Goods	Intra/Inter-Autonomous Systems Goods
IoT	Internet of Things
NFP	Network Flow Problem
NOLI Model	New Open Logistics Interconnection Model
NP	Non-Polynomial
OLI Model	Open Logistics Interconnection Model
OSI Model	Open Systems Interconnection Model
PI	Physical Internet
PoA	Port of Antwerp
SBPP	Single Bin Packing Problem
SOFM	Self-Organising Feature Map
SOM	Self-Organizing Map
SPP	Strip Packing Problem
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem Living Lab 1

1 Revised Executive Summary

As part of the development process of this deliverable's resubmission IBM have reviewed the feedback provided by the project officer in detail. As such we have identified several key areas that required improved demonstration of the research and development undertaken as part of this task. Hence, this deliverable serves as an update demonstrating more clearly the following:

1. Acquisition of data from Living Labs
2. Expansion on the number of use cases addressed
3. How the proposed solutions go beyond current State of the Art
4. That proposed solutions use the most up to date and modern technologies and approaches
5. That Machine Learning techniques are employed where meaningful and feasible
6. That the ultimate value proposition of the research to the PI Concept and the business cases within the Living Labs has been showcased.

The ability to more clearly demonstrate these contributions has been enabled by further increase of data availability from the living labs subsequent to the initial submission of this deliverable. Living Lab 1 is most closely aligned with this deliverable as the Port of Antwerp is a large site with many internal operations that can benefit from optimization techniques. Living Labs 2, 3 and 4 use cases are more appropriately aligned with the value proposition of the other services in the PI Service stack. That having been said, these living labs also provide an excellent opportunity to expand the work of this deliverable beyond Living Lab 1 by discussing with the Living Lab partners additional challenges with regards efficiencies in their internal operations. IBM pursued this avenue with Living Lab 2, 3 and 4 and received datasets from all three. Although datasets have been received from LL3 and LL2, they are not immediately suitable for machine learning approaches, we do believe that optimization is a relevant technology for LL3 use cases and as datasets become available we plan to carry out similar analysis to address warehouse operations and to create association maps between products to predict stockouts. As already mentioned, IBM have engaged with partners in Living Lab 1 and Living Lab 4 to discuss the format and content of datasets relevant to their internal operations in areas we believe optimizations could be applied. LL1 has provided extensive information about train and wagon movements within Belgium, but specific details regarding containers on these trains is not available due to data protection and privacy controls. Synthetic data was used where needed to mitigate this problem. LL4 provided a number of datasets relating to the movements within warehouses for example date and timestamped orders for storing, retrieving as well as some parameters regarding the product within.

With a number of datasets available to us, an overall machine learning strategy and approach was defined for this deliverable, namely the ensemble approach. This approach sees the use of multiple models trained from diverse datasets. The weaknesses or inaccuracies in a single model will be overcome by the strengths in another, and vice versa. By combining multiple models with a diverse set of strengths and weaknesses the overall results have a greater chance of accuracy and success.

These datasets were successfully used to evaluate Machine Learning (ML) Algorithms that optimize warehouse and port operations by reducing distance travelled by picking operations. This is of particular interest in ICONET as picking operations are common to almost all logistics nodes (and therefore PI Nodes) – picking pallets in a warehouse, picking containers in a port or rail yard, picking items in a retail distribution centre etc. Therefore, the research in this area is widely applicable and reusable. Although current State of the Art is already addressing optimising picking route selection, further research is required into the optimisation of storage locations to compliment optimised route selection. This deliverable reports advancements to the state of the art in this area. A reduction in travel time and distance

for route selection reduces Co2 emissions, reduces cost and increases efficiency within the PI node. A number of ML algorithms have been evaluated, and it will be shown how Self Organised Maps for Clustering is considered the best match for the use cases in ICONET. Although some datasets are still pending, initial evaluations show that approaches already researched will also be applicable in LL3.

In the area of loading and composition, the bin packing algorithm was originally researched, as this technology is widely adopted in the logistics industry. To explore potential alternatives not found in the state-of-the-art IBM have conducted further research in this area and have introduced multi-dimensional bin packing enhanced by reinforcement learning techniques. This allows algorithms to learn from past failures and successes without the need for pre-training or pre-processed datasets.

The business value of how these techniques can be adopted in real world scenarios is described in the implementation section below showing how reinforcement learning and neural networks can be applied in the Port of Antwerp shunting yard. To increase efficiency and clustering techniques they can be applied in the container storage yards to reduce picking operational costs. We also show that techniques from the container storage optimisation in LL1 can be reused to optimise pallet storage in LL4 within warehouses with some adjustments and alterations.

IBM also explores how both machine learning and standard algorithmic approaches can be adapted into API based web services in a modular manner that results in a set of synergetic services that will address optimisation problems as part of a PI Service stack.

Further work will now focus on integrating these capabilities with the existing PI stack. In the literature it has been found that existing techniques do not actually factor inclusion of optimisation services as part of the PI Stack. In this regard IBM has also proposed a new approach integrating optimisation systems within the PI node into the PI service stack and into a new version of the PI Sequence Flow that has been shown previously in deliverables **D2.4 – PI Networking, Routing, Shipping & Encapsulation Layer Algorithms and Services V2.**

The research avenues explored in this document will be further explored in terms of research and implementation in D2.14 (V3).

2 Introduction

IBM explored a number of new approaches in the field of PI-Node Optimisation. The main approach implemented includes embedding of machine learning techniques within the Physical Internet as reported in this deliverable. The usage of historical data of activities within a PI-Node has been investigated, in order to improve the efficiency of the operations therein. In the PI concepts presented by ICONET, a PI-Node continuously receives PI-Containers and manipulates them in order to send them to the final destination. In the scenarios defined in D2.19, a PI-Node can only see the demand of the nodes in the neighbourhood or local network. As a result, the node has limited visibility of the demand of the terminal users. However, information concerning the requirements of the final user can be derived via the flow of containers through the node itself. In the same manner as the digital internet, the Physical Internet (PI) has no central management or optimisation system. As a consequence, a single node can optimize its service only by looking to the demands of its neighbour nodes. ICONET has identified a particular use for this information - clustering for correct storage of the PI-Containers on arrival. This is a new area of research in optimisation within a logistics hub, and this topic is covered in more detail in the **PI-Node Optimization** section.

In particular, this document provides details on a number of PI-Hub operations and how machine learning (ML) techniques can be adopted to optimize these operations. This is within the accordance of feedback received from ICONET mid-term review with regards to deliverable D2.13 For navigational purposes, we provide a table below that maps the review feedback to sections of the document where the relevant feedback is addressed.

2.1 Summary of previous versions

In D2.12, version 1 of this deliverable, we investigated a number of approaches with regards to optimization within a PI-Hub or a PI-Node. Initial steps were focused on understanding the internal structure of PI-Node and the operations that took place within and from there understanding the optimization challenges that exist within a PI-Node. One of the main conclusions drawn from version 1 was that there was no overarching “PI Optimization Solution” that could be broadly applied to any and all PI-Nodes as they would each have specific optimization use cases that would each require specific approaches. Instead, an overall optimization framework was put forward which could assist with management and track individual optimizations (or local optimizations) while still tracking optimization at a higher (or global) level. Some initial algorithms were also proposed to address some of the specific use cases within PI-Nodes and some basic data structures and functional modules were set up to be used as a basis for APIs to be used in the web services for this deliverable.

3 PI-Node Optimization

In this chapter, we first present a general optimization framework that can be used for heterogenous use cases for PI. After that, we present details for some PI optimization problems with links to real living lab use cases arising in the project.

3.1 Optimisation Context in relation to PI and ICONET Project

Although Optimisation is not specifically part of the OLI or NOLI (for more detail see **D1.11 - PI Protocol Stack and Enabling Networking Technologies)**, reference models, optimisation remains a key element to bringing the PI from concept to reality and this deliverable proposes a way forward in terms of integrating the PI Optimisation Service into the PI Service Stack. For the PI to become a reality, there will need to be a shift towards automation from a physical and digital perspective. Human interaction with items on the supply chain will drop and the volume of shipment traffic will increase. To accommodate this shift in approach and parallel strain on supply chain resources from additional throughput, optimising current processes and systems will be a critical requirement. In particular, even minor hubs will see an increase in activity as every hub now becomes a potential point of modal change. Such hubs will have to improve efficiency to meet this requirement and ensure high throughput and low wait times are achieved. In a PI network (as opposed to a traditional transport and logistics network) each node or hub now has the potential to be a dependency for every other. If backlogs and queues start to form, the entire network can become constricted. Transparency of the current status within a node and neighbour nodes can help avoid such backlogs and such transparency must be data driven.

3.2 Data Driven PI-Node Optimization

In deliverable D2.12, we introduced an optimization framework and provided details on how this framework is used for LL1 use-cases (see D2.12 for more details on optimization framework). In this version of the deliverable, we provide details of our on-going work of investigating and applying various data-driven techniques to logistics operations in the context of a PI-Hub. In particular, we first present our approach of applying ML or data-driven techniques to PI-Hub operations followed by the details on operations that are considered for optimization along with specific ML algorithms considered for the selected operations.

3.2.1 Our Approach: Ensemble ML for PI Hub Operations Optimization

Ensemble approach is a technique where the predictions of multiple models are aggregated (by some way) into a single prediction (Metzger, 2019) particularly with respect to misclassified instances. Specifically, models need to be *diverse* such that their decision boundaries are different from each other. One way to achieve diversity in a set of models is to use different training data instances obtained through resampling methods such as *bootstrapping* or *bagging*, *boosting* of the overall training data. For the multilayer perceptron (MLP) or more advanced variants of neural network-based models (e.g. RNNs, LSTMs, etc.) diversity is achieved by training with different combinations of weight initializations, number of network layers, error goals, etc. Adjusting parameters allows one to control the instability of the individual model and hence contribute to diversity.

For example, let's assume we have a classification task in which we wish to train ML models that are able to accommodate training data and also able to predict incoming data. Here, the ML system architect

might rely on just the different architectures of the neural network-based models in an ensemble setting, due to their ability to control instability. Alternatively, they may combine entirely different types of classifiers, such as decision trees, nearest neighbour classifier, and support vector machines for added diversity. However, combining different models or different architectures of the same model has to be specifically designed for the task in hand. This leads us to conclusion that designing of an ensemble ML model involves two stages (Polikar, R; 2006):Generating individual models for the training data

1. Generating individual models for the training data
2. Determining how models differ from each other for the same task and combining them

In an ensemble approach, the second stage is critical where model's diversity contributes to final prediction accuracy. However, question may arise, how to quantify model's diversity? There are several measures that can be used for quantitative assessment of diversity. One simple way is determining *correlation* between two models by observing the correctly predicted instances by both models. There are also non pair-wise measures such as entropy which assumes diversity is highest if half of the models are correct and remaining ones are incorrect (Polikar, R; 2006).

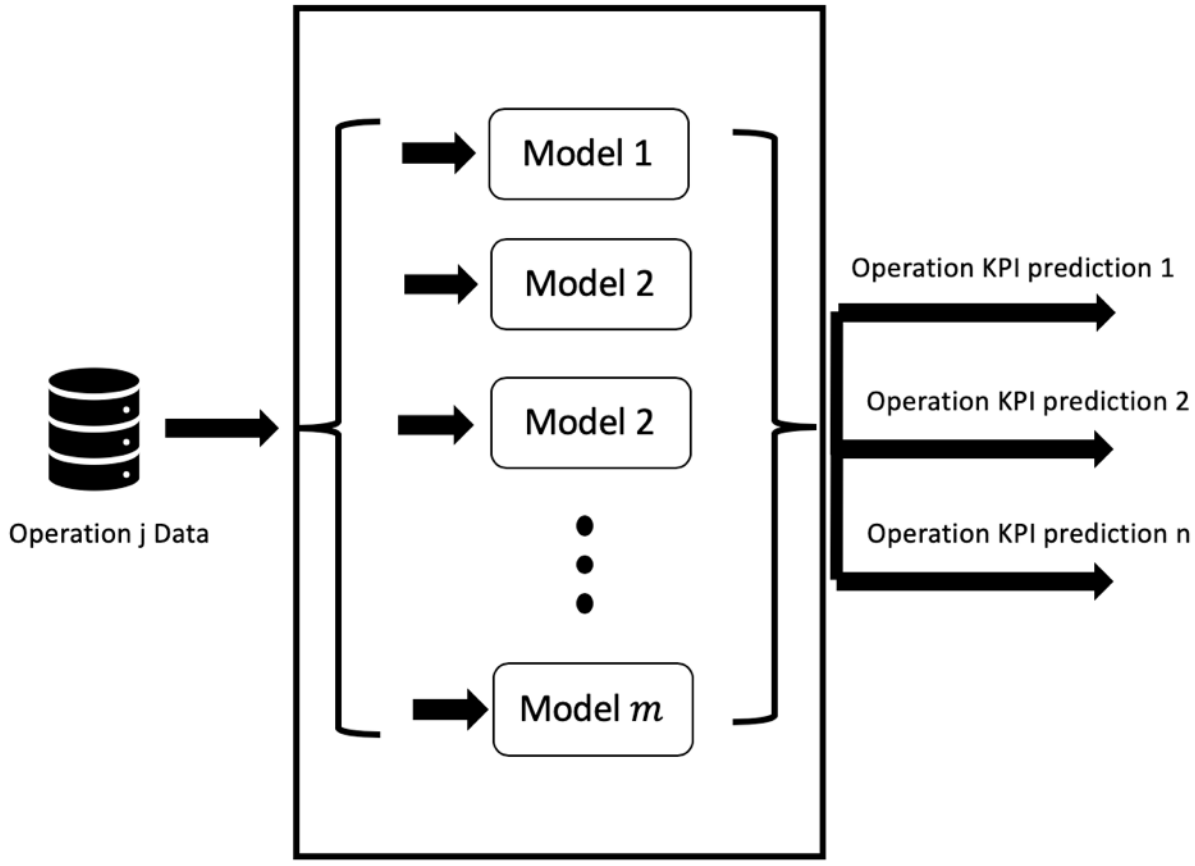


Figure 1. Ensemble approach for PI Hub operations predictive analytics

For ICONET, our ML based optimization approach leverages ensemble technique as shown in Figure 1. We assume that data related to PI-Hub operation j as input to our ensemble, a set of *m* models are trained

and combined to fit over the operation's data and provide predictions. The model parameters are optimized for the business KPIs of LLs in the project.

For example, assume we want to predict the departure time of trains after loading containers on to wagons in LL1. Assume our task is to determine if a train is going to be delayed or not. In such case we can leverage the deep neural network architectures such as recurrent neural networks Long- Short-Term Memory (LSTM) over the timestamp-based sequence data. The architecture could include a shared layer and three hidden layers to predict process activity, timestamp and binary class whether train is late or not. Such an architecture has already been implemented by authors in (Evermann, J., Rehse, J. R., & Fettke, P., 2017) for the process monitoring purpose. Multiple neural network architectures are ensembled using bootstrap aggregation (bagging) technique (Dietterich, T.G, 2000). Bagging generates m new training data sets from the whole training set by sampling from the whole training data set uniformly and with replacement. For each of the m new training data sets an individual deep learning model is trained and prediction is made on test data. Each individual RNN-LSTM model in ensemble architecture will deliver a prediction time of departure at a checkpoint or operation step j. These individual predictions are combined some way to give a final predicted departure time T_j. One simple approach to compute the predicted departure time T_j using ensemble ML modelling is compute the mean value of individual predictions $T_{i,j}$ from each prediction model.

$$T_j = \frac{1}{m}\sum T_{i,j}$$

From this prediction reliability estimate for a data instance d_i is computed as the fraction of predictions that predicted that instance correctly. Let's say in the original test data, $d_i = true$ for operation step j then prediction reliability is:

$$\rho_i = \frac{1}{m}.|i : d_{i,j} = true|$$

In the **Application of ML for Container Management** section, we give details of a number of machine learning techniques that we believe are relevant to PI operations for LL1 and LL4 of ICONET and will be explored within this ensemble framework.

3.3 PI-Node Operations from ML perspective

In order to reach an optimization goal within a PI-Node, finding the optimal location for storage is not sufficient: we need to optimize also the transport of goods to and from this location within the PI-Node. While each PI-Node will have a different configuration (type, size, throughput), the general topology remains the same – move goods in, store, and move them out. In a warehouse, distribution centre or even retail location, a major optimisation challenge is in the storage of pallets or boxes in a three-dimensional space, utilizing floor to ceiling shelving, routes and lanes between shelves and systems for placing and removing items from said shelves. While operations in a port work in a different physical situation, the issue remains the same – containers are stored in a three-dimensional space with containers stacked, with

lanes and accessways between them. A potential optimisation is in the placing and retrieval of these containers. ICONET considers PI-Node optimization in five phases, in keeping consistent with the OLI model:

1. Clustering Phase: independently from the algorithm chosen, this service will learn, periodically in time, the clusters of containers. This topic is covered in more detail in the next sections.

2. Storage Phase: Given the distance between the clusters, the system will identify the correct positioning of the goods.

Note: The Clustering and Storage phases outlined above will optimize the storage of the goods. However, once a set of container transportation requests (with load sizes and origin-destination information) is received, a typical logistics operator is typically required to consolidate these requests and build internal routes with existing transport assets (trucks, forklifts, pallet movers).

3. Packing Phase: This phase happens just prior to departure; the system will provide the operator in the PI-Node the right loading schema of containers in PI-Movers in order to optimize the space and the allocation of movers. This in turn would minimize the number of movers which will allow other PI Nodes to leverage these resources as the Physical internet concept dictates.

4. Routing Phase: Routing reads the result of packing operations and finds a series of node sequences to determine the order in which to transport containers so that all the containers reach their final destination. The sequence of nodes visited by a transport carrying different containers is called a route. Routes are created such that first and last node visited by a transport coincide. In other words, each route created by the routing phase is a loop.

5. Shipping Phase: The shipping model reads the list of the routes created during the routing phase and associates each route as a unique job that should be assigned to a transport. The scheduling model works similar to an assignment problem to find the minimized cost of performing all the routes with the given set of transports in a given delivery time span.

The following sections focus on extending the research and the state of the art in Clustering, Storage and Packing phases while leveraging the work carried out with respect to the Routing and Shipping phases in deliverable **D2.3 – PI Networking, Routing, Shipping and Encapsulation Layer Algorithms and Services V2.**

3.3.1 Clustering of Containers (relevant to LL1 and LL4)

After several meetings with ICONET partners, IBM noted that usually there is a common process in all Living Labs: a transported unit is collected, stored for some time and then delivered together with others. In the ICONET Living Labs or PI-Networks in general, this transported unit could be a container, a pallet or the goods item itself, but we will generally refer to them as PI-Containers. A PI-Node can learn the relationship between the goods in arrival by the historical pattern of the delivery operations computed. For example – "Containers arriving on X day at X time by different means often go to X destination". There could be several reasons why some PI-Containers have a high probability of traveling together – for example materials travelling to a manufacturing plant with different origins but common destinations once in country. In some cases, like for Warehouses, a PI-Node will see the direct realization of the relationship between goods, in other cases it will see the relationship between destinations. In any case both these relationships will be generated by the demands of the users while efficiently allocating its

resources for cost minimization and as per PI environmental benefits. The clustering technique can enable a PI-Node to adapt its operations to the demand of the users. Through continued usage and changes in approach based on past events, the delivery time of PI-Containers can be decreased. Clustering is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense) to each other than to those in other groups (clusters) (Bekkerman, 2011). This is a main task of data mining, and a common technique for statistical data analysis. This process can be extremely powerful in order to improve the efficiency of operations in a PI-Node. In fact, a PI-Node can use this technique in order to identify clusters of PI-Containers that usually leave together from the PI-Node. This information can then be reused in order to store the containers closer in order to save time when 'picking' the containers. In the following sections we outline the possible clustering use cases in order to enable the system to learn the most optimal allocation of the Containers.

3.3.2 Storage of Containers (LL4)

In this section we present the operations within a Warehouse PI-Node (LL4). It should be noted that operations within different PI-Nodes vary significantly and have many elements adding layers of complexity. With the goal of finding a set of common operations between different nodes of a logistic network, we will consider a simplified model of operations usually conducted in ports, warehouse and shops. In warehousing management, operations consist of six activities: receiving, transferring and loading, order picking/selection, accumulation/sorting, cross docking and transportation. Studies have shown that order picking costs account for more than 55% of the total operating costs (Pang et al., 2016). Therefore, it is important to optimize the order picking operation. With this in mind, we must first consider the storing of goods in a suitable location of the warehouse as this will have a significant impact on the efficiency of order picking operations in terms of travel distance required to retrieve the requested goods (Roodbergen, 2001). There are four ways to reduce the travel distance of order picking operations:

1) Well-planned order picking path.
2) Warehouse area division.
3) Allocate products to the proper storage locations.
4) Picking up orders in batches.

While points 1, 2 and 4 are heavily related to the type of business and the type of infrastructure used by the logistic operator, (i.e. right path for picking warehouse divisions and the type of sorting of the orders depending on the storage area and on the type of commerce conducted by the operator), they are somewhat addressed in existing research and logistics operations. Problem in Point (3) above – product allocation – is an under-researched area; furthermore, it can be considered universal across logistic operators. In fact, for ICONET's LLs, the right storage location could be generalized given the nature of the data collected (records of movement of goods and orders) and the nature of the storage areas, that in most of the cases is a large rectangular area. Therefore, the work on this deliverable began by investigating the third problem, improving order picking efficiency by optimizing the storage location of Containers. These Containers are stored on shelves in the warehouse (or placed in stacks in a yard), and only one Container can be stored in a single position.

3.3.3 Packing of Containers

Packing of containers involves picking the containers and loading them onto PI-Movers. In the Living Lab use cases, we are addressing in this document the container-to-mover mapping can be 1-to-1 or 1-to-many. For example, a forklift removing a single pallet from a warehouse shelf or a bin wheeled around a warehouse during picking collecting multiple items in a single journey.

3.3.4 Picking Containers (Applicable to LL1 & LL4)

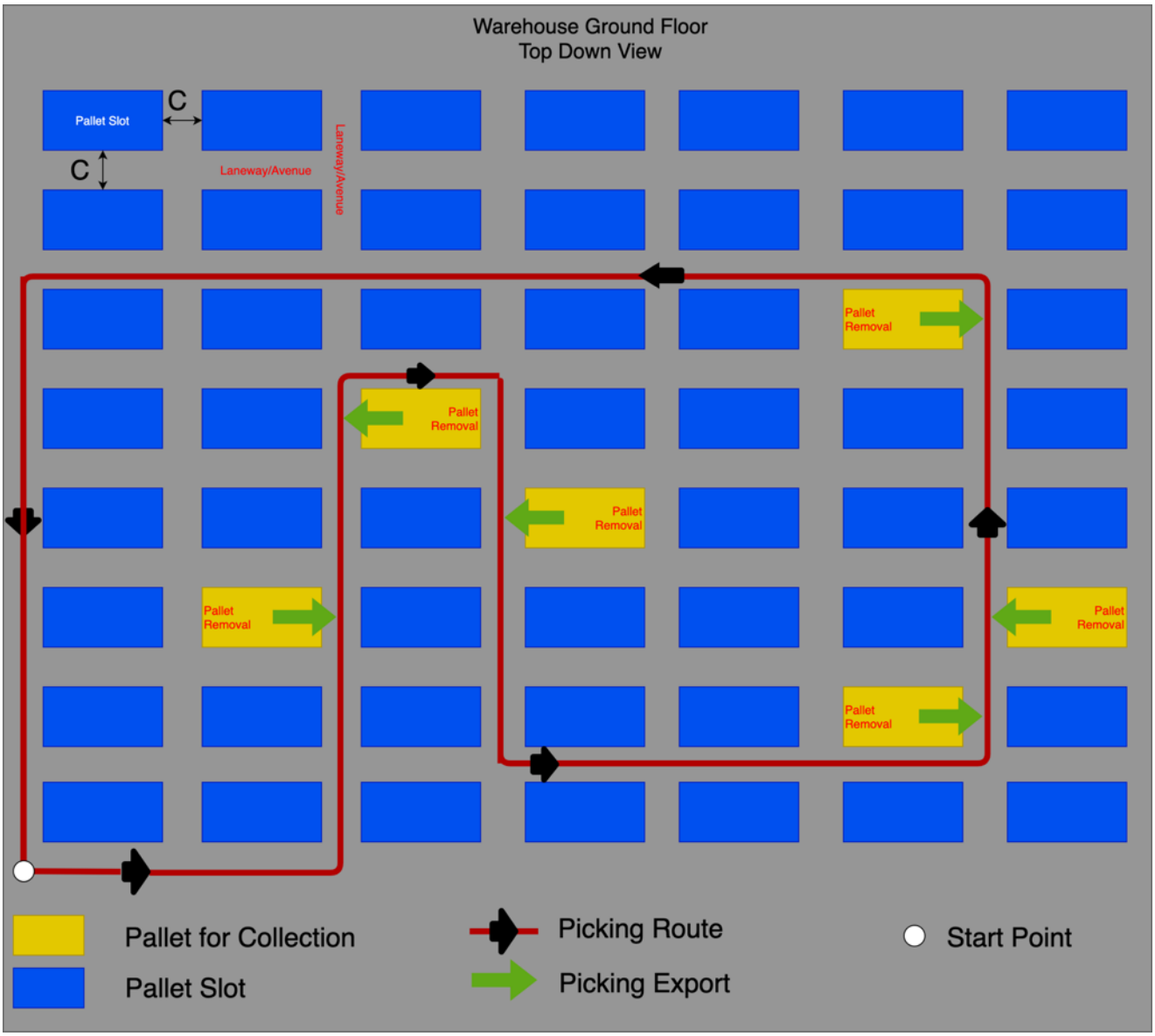


Figure 2 - Warehouse Layout Map

To understand how best to approach and address Point (3) we must gain an overview of the existing domain as well as the current approaches taken (i.e. route optimisation). This will help demonstrate why the optimised storage of PI-Containers can lead to a reduction of costs and time in the picking phase. Our simplified warehouse is a bi-dimensional projection on the ground of a real warehouse where containers are stored in shelves. The position of shelves in the warehouse can be viewed as a matrix of m rows and n columns as shown *Figure 2 - Warehouse Layout Map*. Therefore, the position of each shelf can be expressed in binary array *(x, y)*, in which $x \in [1, n]$ and $y \in [1, m]$. The distance between every two rows is denoted by r, and the distance between every two columns is denoted by c, where $r/c = 1/2$. Furthermore, $[n, m]$ can be used to represent the size of the warehouse. The meaning of this equation is that Containers are stored on the shelf of y rows and x columns in the warehouse, which is expressed in the following equation: $(r, c) = (y, x)$. In this example, we specify the storage location of the picking origin or destination as (1,1) approximately (this is a loading zone or conveyor belt for picked packages).

When an order arrives, the picker needs to pick up the PI- Containers starting at the origin and moving to the specified location of each Container in turn according to the list of Containers appearing in the order. After all of the Containers in the order is obtained, the picker returns to the origin and an order picking process is completed. In this process, the distance picker travelled is the total cost of the order picking operation. Rules for the picking route as follows:

1. Containers can be picked on both sides of a shelf, so the picker can pick two columns of Containers on one lane (as shown by the green arrow in Figure 2). If both column J and column $J + 1$ have containers in the order, the road to the right of column J will be selected. If column J has containers in the order but column $J + 1$ does not, the road to the left of column J will be selected. The coordinate of Containers we need to be on the selected road is $\{(y_1, x), (y_2, x), \cdots, (y_k, x)\}$. Let $Y_{min} = min\{y_1, y_2, \cdots, y_k\}$ and $Y_{max} = max\{y_1, y_2, \cdots, y_k\}$. If ($Y_{min}$,x) is closer to the picker's position than (Y_{max}, x), then Y = Y_{min}. Otherwise, Y = Y_{max}. Then the start-point (as shown by the white circle in Figure 2) is defined as the intersection of straight-line y = Y and the selected road. The picker walks to the start-point, then gets the Containers we need on the selected road in turn.
2. Do the same for the remaining columns as for step1. When all the containers in the order have been picked, the picker returns to the start point.

In a warehouse where containers are stored randomly or according to some local, the distance covered by the picker is longer. In reality, warehouses are not bi-dimensional and additional rules are applied in order to reduce the picker distance, but in practice very few of warehouses take into consideration the pattern of the orders of items that they move (Ballestin, 2013).

3.3.5 Loading Containers (Applicable to LL1)

In this scenario, our optimization objective is to find the best wagon slot*s*, on the wagon *S*, for a container *i*, with its length *h,* width *w* and weigh *p*, so that the loading factor of the train can be maximized. In other words, the number of unloaded containers on the subset of the container-stack is minimized. We consider the following constraints:

- The capacity and the length of the wagon
- The priority of the product contained
- The dangerous nature of the product

As understood from the LL D3.3, in the real world there are no more than 2 or 4 slots for each wagon, but building the optimization assuming this remains always true could be wrong. In the future development of Physical internet an increase in the number of different containers sizes can occur. For this reason, we would like the optimization to take into account this possibility. Another PI constraint has to be added: Guillotine constraints. Guillotine cutting constraints derives from the limit of orthogonal cutting of several machine in industrial operations (zhang, 2016). The objective is to minimize the waste or height of material or maximize space utilization in a bin.

3.3.6 Wagon Bundling Problem (Applicable to LL1)

In this scenario, our partners from PoA would like to analyse how optimization techniques can be applied in bundling wagons designated for the same terminal of destination on the same train. By doing this optimally, it will help different operators save time, costs, and make the overall system more efficient. A schematic diagram of the problem is depicted in *Figure 3* , where wagons of the incoming train need to

be sorted and bundled to a local train with the same destination terminal. As shown in Figure 3, the core idea is that instead of having 16 direct trains traveling from each hinterland terminal to each corresponding port terminal, now we only need to have 4 direct (bundled) trains.

Some detailed problem descriptions are given as follows:

- Containers from an inland terminal need to go to different maritime port terminals, and vice versa.
- Volumes are often too small to operate different direct trains from inland terminal to individual maritime terminals with an acceptable frequency, this is why a lower cost bundling system is required.
- The optimization problem needs to consider the following parameters/factors:
    - Optimization scope is for those trains with less than 80% of maximal length/weight. In other words, for trains with real length/weight of more than 80% of the maximal length/weight are not in the scope of our optimization, which is due to the fact that these trains have already achieved decent loading factors, and thus the benefit for such trains involved in the bundling process is limited compared to the complexity of the bundling process and operational costs involved.
    - Number of available locomotives in the port is limited.
    - The rail capacity in the PoA is also limited.
    - Terminal capacity for rail at the terminal of destination
    - Handling and transport costs for the railway undertaking and for the terminal. For example, if it is more expensive to re-arrange wagons in another train, the train can go directly to the terminal of destination.
    - Existing scheduled terminal slots for trains going to the same terminal of destination.

At this stage of reporting, we have built the basic data structure for analysing the wagon bundling problems.

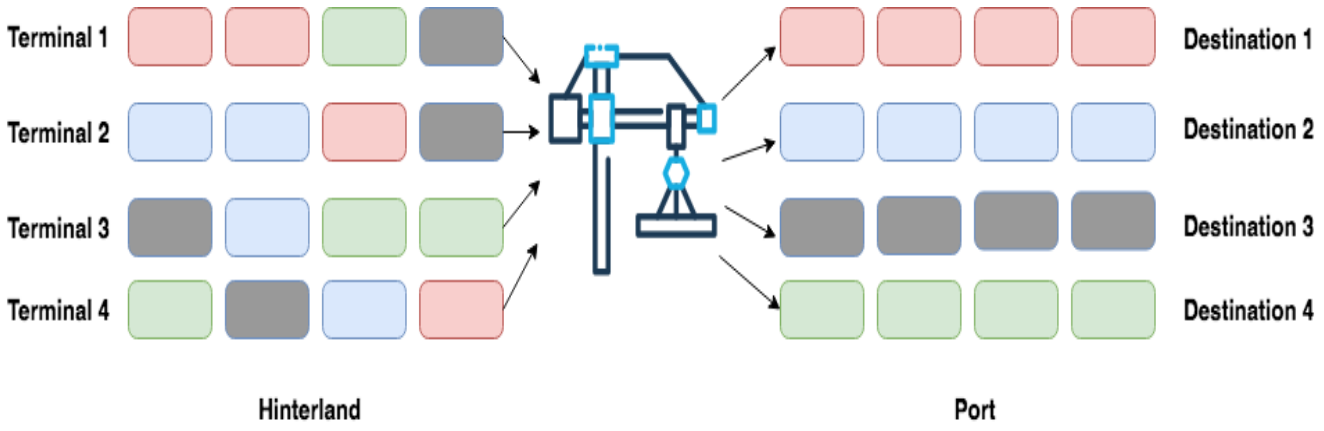


Figure 3 – Wagon Bundling Problem

4 Bin Packing Solution – Packing Optimization

There are several approaches for optimally packing containers in PI-Movers. The right allocation of the containers will impact the number of PI-Movers so an increase of these movers can produce delays and over utilization of the network which leads to an increase in working hours and pollution. The author in (Fazili, M., 2016) has addressed this problem and a comprehensive solution has been provided and tested over a simulation. One of the objectives of this deliverable is to extend the research already completed in this area. In (Fazili, M., 2016) the authors present the application of a bin packing algorithm as a heuristic solution to the packing optimization required as first step for optimizing the PI-Container packing process.

Packing problems are related to multiple fields of operations research as they have different industrial applications such as material cutting, multi-task scheduling and cargo loading. These operations can be seen as packing problems with specific constraints and objectives. Possible constraints include guillotine cutting and fixed orientation packing. Guillotine cutting constraints derives from the limit of orthogonal cutting of several machine in industrial operations. The objective is to minimize the waste or height of material or maximize space utilization in the bin. In a possible PI–Hub scenario, several containers of rectangular shape should be loaded on a fixed wagon of the same shape – the relevance of the guillotine cutting constraint is evident. According to the typology of packing problems as presented by by (Zhang, 2016), the guillotine rectangular packing problems (GRPP) include two types, with each type involving two alternatives:

• Strip packing problem (SPP): Given an open bin of width W and unlimited height, and a set of n rectangular items with sizes (h_i, w_i), $i = 1,...,n$, the objective is to place each item in the bin without overlapping, such that the bin's required height is minimized. This problem includes two variants: OG and RG, where O denotes the case where items are placed with a fixed orientation, G denotes guillotine constraints are required, and R denotes items may be rotated by 90 degrees.

• Single bin packing problem (SBPP): Given a rectangular bin of width W and height H, and a set of n rectangular items, the objective is to maximize space utilization (or "filling rate") of the rectangular bin. Similarly, this problem includes two-variants: OG and RG.

GRPP is NP-hard and is difficult to solve. Some researches for GRPP have shown exact algorithms only solve small-scale problems (Belov,2003); heuristic algorithms are for real-world operations because of stability and performance. Constructive heuristic algorithms cannot guarantee a solution of good quality, but they can find a feasible solution in relatively short time [16].

4.1 Container Loading as SBPP Problem

According to the above classification, the Container loading problem is a SBPP type with rotation only possible on the plane parallel to the ground. We propose a variation of the original solution proposed by (Zhang, 2016), which proposes to create a priority list based on the geometrical properties of the item to be packed. We present a solution which considers commercial and physical constraints in addition to geometrical properties.

A recursive technique is useful for GRPP as it may be used to restrict containers' locations such that they can satisfy the guillotine constraint. The concept is simple as a rectangular space (Wagon) may be divided into several smaller rectangular spaces (slot), and each smaller space can be divided recursively. In Figure 4 (a), we observe that the wagon loading plane S is determined by its position (x, y), and its width w and its length h. The core purpose of the proposed algorithm is to fill S efficiently.

4.1.1 Geometrical Priority

Each unloaded container can be placed into *S* resulting in five scenarios, as shown *in Figure 4 - Geometrical* Priority (Zhang, 2016):

(a) is the best because the item fills up the whole space.

(b) the container has the same width of the wagon but not length.

(c) opposite of (b)

(d) the container is smaller than the wagon in both dimensions and requires careful partitioning

(e) represents that no container can be loaded into *S* and *S* is wasted

Cases (b) and (c) are better than case (d) because the remaining space in cases (b) and (c) is of rectangular shape, while the remaining space in case (d) requires careful partitioning. Whether case (b) or (c) is better depends on the practical problems. Assume that the containers are sorted by non-increasing length (which is a basic requirement of every operational research solution), then containers that match cases (a), (b), (c), (d) and (e) are assigned priority 1, 2, 3, 4 and 5, respectively. In reality, in a port like PoA, the only possible choices will be case (a), (b) or (e), but to not lose generality and to keep the door open to possible evolution of PI-Hub scenario we think it is good to keep these features.

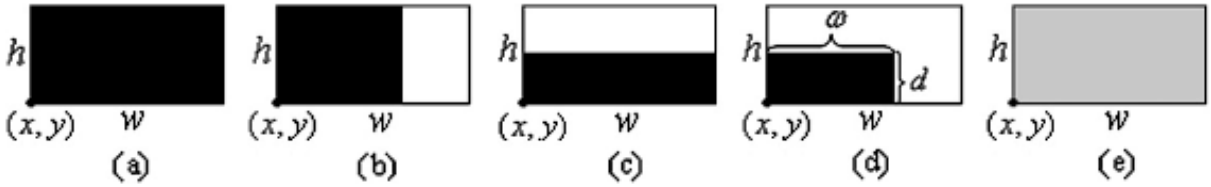


Figure 4 - Geometrical Priority (Zhang, 2016)

4.1.2 Commercial and Physical Priority

The above solution is not different from the standard SBPP problem (Zhang, 2016). But, as presented in D2.12, the reality of a PI-Hub could be drastically more complex with respect to the standard operational research solution. In order to capture this complexity, we present here a variation to the priority list created above: we will move a container in case (e) (i.e. no space on the wagon due to weight limits) if the weight capacity of a wagon is filled even if theoretical there is space. In the case of premium or dangerous goods inside a container we will revert to case (a) from any other case (except case (e)).

4.1.3 Bin Packing Algorithm

In order to simplify the explanation of the algorithm, let us clarify that digital operations discussed below are not related to any physical activity; this is because we assume that the PI objects present in the PI Hub are all IOT devices. In this scenario, our optimization objective is to find the best wagon slot*s*, on the wagon *S*, for a container *i*, with its length *h,* width *w,* priority *p,* destination *d* and weight *q*, so that the loading factor of the train can be maximized. In other words, the number of unloaded containers on

the subset of the container-stack is minimized. Given that all containers in a wagon should go to the same destination, we will apply the solution to subgroups of container sharing the same destination *d*. This is represented in the first step of the algorithm flow chart presented in Figure 6. To satisfy the guillotine constraint we apply a recursive solution. The concept is simple, we need to split the wagon surface *S* into smaller rectangles.

Among the unloaded Containers (all of them sharing the same destination), those with the highest priority (1 is the highest) are chosen for placement first. We stop packing *S* when case (e) occurs because all unloaded containers cannot be packed into *S*. If two or more containers have the same priority, then the first container hit in the sorted list is packed first. For case (b) and (c) a container can be loaded without worrying about the division of the surface: with only one call of the recursive function on the remaining space. For case (d), the division of *S* is important, and is done as follows: Let min *w* and min *h* be the two parameters related to all unplaced containers where, for fixed orientation packing, min *w* is the minimum width of all unplaced containers and min *h* is the minimum length of all unplaced containers. Let ω denote loaded width, and *d* – loaded length.

According *to Figure 5 - Case(d) in more detail* (Zhang, 2016) below, specifically case (b), if the value $w - \omega$ (loaded width) is less than min *w*, *S* is divided into $S1$ and $S2$, as in (a) instead of as in (c), Both in case (a) and (c) $S2$ will be wasted; however, wasted space $S2$ in (c) is larger than $S2$ in (a). Therefore, this partition can make $S1$ larger, so that it can be used by other unplaced containers. Similarly, if the value $h - d$ *(loaded length)* is less than min *h*, then *S* is divided into $S1$ and $S2$, as in (c), implying that $S1$ will be wasted. Otherwise, there are two ways to divide *S*, as in (a) and (c). One partition is as in (a), another partition is shown in (c). Which partition is selected depends on if ω is less than min *w*. If ω is less than min *w*, then the former is selected because condition $\omega <$ min *w* leads to waste of bin $S1$ as in (c).

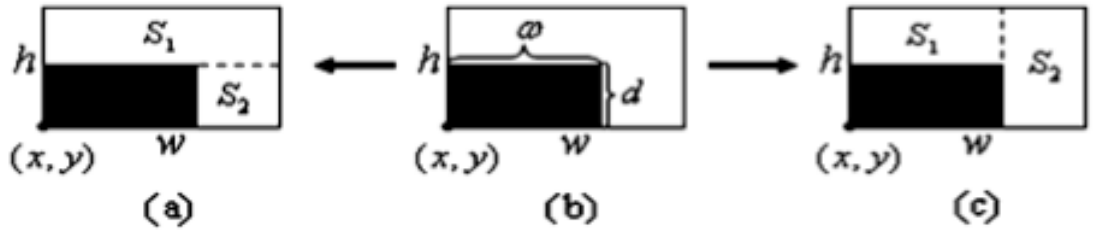
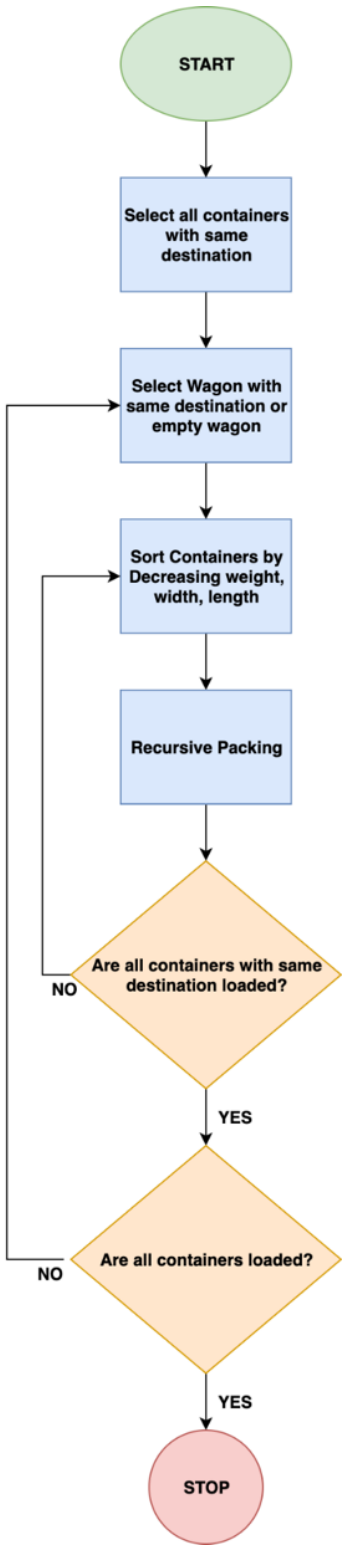


Figure 5 - Case(d) in more detail (Zhang, 2016)

In fact, the orientation of the partition is the determining factor and depends on the way the containers are sorted. The aim of the strip packing problem is to minimize the height of the bin, such that the partition in (c) is more efficient because items with large heights have greater chance to be placed. For the single bin packing problem, the orientation of the partition is mainly determined by the way the items are sorted. For this reason and to take into account the role of the weight capacity we sort the containers according first to their commercial priority and then their weight and finally the width as described in the flow chart below.




```
RecursivePacking(x, y, w, h)
    if no item can be placed into S or all the items are placed into
S, then
        return;
    else
        for each unplaced item i do
            for j = 0 to D do
                determine the priority of item i;
        select an item r with highest priority from unplaced items,
        record its orientation j;
        if the highest priority is less than 5 then
            place item r into S by the orientation j;
            switch (priority)
            case 1: break;
            case 2: RecursivePacking(x + ω, y, w − ω, h); break;
            case 3: RecursivePacking(x, y + d, w , h − d); break;
            case 4: if w−ω < min w then
                    divide S (as in Figure 5 (a)) into S1 and S2 (S2
is wasted);
                    RecursivePacking(x, y + d, w , h − d);
                elseif h−d < min h then
                    divide S (as Figure 5(c)) into S1 and S2 (S1 is
wasted);
                    RecursivePacking(x + ω, y, w − ω, h);
                else if ω < min w then
                        divide S into S1 and S2 ;
                    else
                        divide S into S1 and S2 ;
                        Recursively pack the larger space among S1 and
S2;
                        Recursively pack the smaller space among S1
and S2;
                break;
```

Figure 6 - State Diagram Representation & Pseudocode for Bin Packing Algorithm

4.2 Wagon Bundling Problem (LL1)

In this case we can consider the Wagon Bundling as an SBPP problem. Differently from the above case, we can ignore the geometrical properties of the wagon, given the extreme similarity between them. The major quantities considered are weight, the overall length of the train and the priority of the wagons. Based on these considerations we created the pseudo code presented *Figure 7 - Pseudocode*. In this scenario, the Local optimizer gets the number of available locomotives in his position and starts adding wagons with the same destination. Following these constraints, we have to consider the limits in rail capacity. For this reason, we compute the minimum between the number of locomotives and the number of trains that can run on the rail system.

After that, as the pseudo code shows, we first select all wagons going towards destination d and then we sort them according to their priority and weight. After that, a process of matching wagons to maximize the overall train weight is applied and then the selected wagon is assigned to the train. If no matching is possible this means all trains are loaded and they are cleared to go, the remaining wagons will be the first to be added when new locomotives are available.

```
K=Read all locomotive in the yard (this is set by the global optimizer)
For all destination d= 1, D do
  T=read number of train on the line with destination d (provide by the global optimizer)
  F=maximum number of train running on that route
  M=min(K,F-T)
  Wagons=all wagons with destination d
  Wagons= sort wagons according priority, weight
  for All wagons i = 1,2,...,n do
   for All train(d) j = 1,2,...M do
     if wagon i fits in train j and train j < 80% then
       Calculate remaining capacity after the wagon has been added.
     end if
    end for
                add wagon i in train j, where j is the train with minimum remaining capacity after adding the
    If no such train exists than
      Let trains go
      Change priority of wagon to 1 and put in queue for the next iteration
      break
  end for
end for
```

Figure 7 - Pseudocode

Figure 7 - Pseudocode above is the pseudocode for the recursive packing process for *S*, where *D* denotes whether the problem considers orientation constraints or not, $D = 0$ denotes the items are placed only with the fixed orientation and $D = 1$ denotes that rotation is permitted. The highest priority is an integer from 1 to 4 which corresponds to four cases from (a) to (d) in *Figure 4 - Geometrical Priority* (Zhang, 2016) respectively, so the priority in the switch operator is between 1 and 4.

4.3 Extending Existing Bin Packing Approaches - Reinforcement Learning

Bin packing algorithms are in use currently within the logistics domain. In particular in (Fazili, 2016)

algorithm uses a one-dimensional bin packing solution to calculate the minimum number of the containers needed to be transported from sources to destinations. Load sizes chosen in (Fazili, 2016) made a one-dimensional bin packing sufficient to calculate the optimal number of bins. However, if smaller load sizes are introduced, two- or three-dimensional bin packing will be required to calculate the minimum number of containers required. In order to make such an improvement we have researched a

2-dimensional bin packing algorithm. In the final version of this deliverable we will increase this complexity by adding a 3rd dimension. Details on this can be found in the **Port of Antwerp** section.

Due to the challenges of obtaining optimal solutions of Bin Packing Problems, researchers have proposed various approximation or heuristic algorithms (Coffman et al., 1980; Martello et al. 2000; Martello et al., 2003). To achieve relevant results, heuristic algorithms have to be designed specifically for different types of problems or use cases, so heuristic algorithms have a limitation in generality. In recent years, artificial intelligence, especially deep reinforcement learning (DRL), has been heavily researched and positive results have been achieved in many fields. In addition, DRL method has shown huge potential to solve combinatorial optimization problems ([Vinyals *et al.*, 2015], [Bello *et al.*, 2016]). Reinforcement learning (RL) is an area of machine learning concerned with how software agents ought to take actions in an environment in order to maximize the notion of cumulative reward. Recently its fusion with neural networks of deep learning techniques has shown to yield extremely powerful scalable solutions. For this reason, ICONET is researching the concept of adding a deep reinforcement learning layer to its heuristic solution. This is covered in more detail in the **Reinforcement Learning - Train Formation section.**

5 Application of ML for Container Management

The activities of warehouses or cross-docking yards according to ICONET partners are recorded in time series of containers at dock. From this particular type of data structure there are several techniques that one can use to associate or cluster goods or Containers. Before entering in the details of the algorithms, we define here the standard input of the ICONET optimization service.

Table 1 Typical features associated with Pi-Containers

Feature	**Description**
Transaction date or time	Time label of the transaction
Container ID	An identifier of the container
Weight	The total weight of the container
Dimension 1	Spatial dimension
Dimension 2	Spatial dimension
Dimension 3	Spatial dimension
Product	Content of the container
Type of movement	Reason for the movement within the pi node
Destination	Destination of the journey

For each type of PI-Node the actual content of the table can change. Different types of PI-Nodes can have different types of operations.

The general approach for utilizing machine learning algorithms is as follows:

1. Acquire Data – While we have obtained a number of datasets from Living Lab partners, ongoing data acquisition would be required for better ML model accuracy.
2. Pre-Process & Prepare Data – The data sources and data formats will differ significantly between different stakeholders and entities. Specific techniques will be required to bring the data in line with the chosen algorithm.
3. Select an appropriate Machine Learning Algorithm – For this deliverable three algorithms have been selected as described in the following paragraph.
4. Train Model – This is the most computationally intensive, expensive and time-consuming part. of the process and where the bulk of the datasets are initially used.
5. Test Model – Parts of the dataset not used for training are used to validate the accuracy of the model.

The following three ML techniques have been identified as relevant to the container sorting scenario:

1. Association rules-based clustering (Jafarzadeh, H., 2015) - The departure day of a PI-Container and its content or destination can be used to learn the association between PI-Containers. The model can count the number of times a co-occurrence of two containers appear and then identify this has a rule. This would work well with real orders.
2. Distance based clustering (Wagstaff et al., 2001)- This is the family of models like k-means or hierarchical clustering. These models aggregate the instances according to their distances in Euclidian space. The instances are formed by the PI-Containers. To each PI-Containers a list of features could be computed.
3. Self-Organizing Map or Self Organising Feature Map (CJ Davis, 2017) - SOM or SOFM is a type of artificial neural network (ANN) that is trained using unsupervised learning to produce a

low-dimensional (typically two-dimensional), discretized representation of the input space of the training samples, called a map, and is therefore a method of performing dimensionality reduction.

While the first two techniques require a subsequent function to derive the storage location, SOM will not require this because it can derive the unique location from the data. Therefore, we have done our initial research on the SOM model and will evaluate the remaining two when additional data is acquired or synthesised and then select the strongest performers to pursue with respect to the Living Lab use cases. This will be reported in the final version of this deliverable.

5.1.1 Self-Organized Maps – Container Storage

An alternative approach to existing ones within the logistics domain can be found in the utilization of Self Organized Maps (SOMs). Details on how these are applied for pallet storage for LL4 can be found the **Stockbooking** section.

The reason SOMs are an attractive method to utilize in clustering is the ability to determine the best overall solution given the number of partitions for massive datasets. Neural networks are an extremely robust technique. Their use is well suited to big datasets, which leads to better solutions; neural networks perform much better with large quantities of data which is ideal for warehouses that have many different products with lots of order information with differing requirements.

Considerations for SOMs are utilization and workload balance in which clusters could be weighted disproportionately. Workload balance and travel distances have to be kept in mind in order to determine proper zones. Zones refer to the picker boundaries within a forward area. One advantage of SOMs is the visual nature of the output data. It is easy to see which Containers were ordered together and which Containers are not. Through the visible output lines can be drawn based upon the clusters that the SOM finds. This is seen *in Figure 8 - SOM Cluster Map.*

Figure 8 - SOM Cluster Map represents a SOM analysis example, with each square being a node. A processing element, neuron, or "node" is determined by the topographical size of the SOM. If a 10x10 matrix is used, then 100 processing elements, neurons, or nodes are present. Lighter more dense nodes contain items ordered together more frequently, and those that are less frequently ordered, are darker. This figure makes it easy to see some clusters within the data.

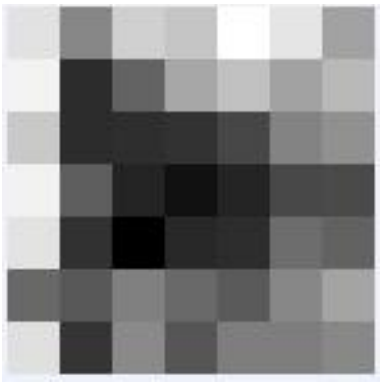


Figure 8 - SOM Cluster Map

5.1.2 Association Rule-Based Model - Container Storage Location Assignment

The association rule-based storage location assignment policy optimizes warehouse storage location allocation by analysing the relationships between different products in customer orders (Pang, 2016).

Association analysis is the discovery of interesting associations and related relationships between item sets from large amounts of data. Association analysis process can be expressed as follows:

1. Define the set of all items as $I = \{I_i, i = 1,2,\cdots,m\}$.
2. Given a set of orders defined as $D = \{D_i,\ i = 1,2,\cdots,n\}$, in which each order D_i is a set of items. $D_i \subseteq$ I.
3. Given sets A and B, they satisfy that $A \subseteq D_i, B \subseteq D_i$.
4. We can say that A appears once in D, if $A \subseteq D_i$. Then the confidence of A is the ratio of the number of appearances of A to n. The expression form of an association rule is $A \Rightarrow B, where\ A \subseteq I, B \subseteq I and A \cap B = \emptyset$.

This rule means that if a customer purchases itemset A, they will also purchase itemset B at the same time. Association analysis is the generation of frequent itemsets and association rules with all support and confidence greater than a specified threshold. The confidence of an association rule $A \Rightarrow B$ is the percentage of the number of times $A \cap B$ appearing in D and the number of times A appearing in D. Support reflects the frequency with which associations occur, and confidence reflects the strength of rules in the data (Agarwal, 1993). In the case of strong correlation among order items, the association rule-based storage location assignment policy is a good choice for storage location allocation.

5.1.3 K-Means and Hierarchical Clustering – Container Storage

K-means and hierarchical clustering are well known techniques for clustering data (Agarwal, 1993). The key problem of this approach is to identify a set of relevant features to separate items in clusters that represent storage time of items and the orders in which they appear in the PI-Node's records of activities. We believe a key strength of this approach could be the identification of a measure of distance between clusters. It is in fact the selected formula of the distance that determines the shape of the clusters. This metric can be extremely helpful in mapping the items in a later phase in their positions. But it remains unclear which is the best set of features to use for each living lab and in general in PI. We plan to use the following features for clustering:

a. storage time in days
b. number of departures from the node
c. quantities in departure from the node
d. mean quantities in departure per each day
e. trend in the last 3 weeks of quantities in departure

Other quantities could be generated using information about items in arrival. It is important to notice that the number of clusters to be identified will be decided by the Node according to its warehouse structure. It is important to remark that this technique will identify the clusters, not their mapping into the node's storage area, so if this technique is applied another function has to be created and executed to map the clusters into the storage area.

Every clustering technique uses a measurement of distance between the items in order to identify the clusters. Once a model (like the ones cited above) has been trained and properly validated, it is possible

to use metric of distance for our purpose of grouping containers. after we have identified the clusters and a function for a distance, we can use the same function to compute the distance between the clusters. A solution to the problem is to rank the clusters according to their mean period of storage and start adding them to the warehouse from those having less period of storage. Once the first to store is selected the others will be added following the distances identified by the clustering methods. The process of deciding which cluster *uk* is stored in the *ith*(*i* starts at 1) row as follows:

1. Calculate *D* (distance)for each *uk*.
2. Store *uk* with the min *D* in *ith* row.
3. If $i = n$ (n represents the total number of rows in the warehouse), end the algorithm; else $i = i + 1$ and repeat step 1 and step 2.
4. Reshape the shape of the result to [*n, m*] (m represents a fixed total number of columns).

6 Living Labs Implementation

Above we have presented a list of operations that can be seen as a focused study of operations conducted in port yards, warehouse and outlets. We have also indicated a list of techniques that can be used to learn useful pattern in the records of the activities of the PI-Node. In the following sections we will describe the work done so far to implement the presented ideas. Note that in the current business and data management of the living labs, some information like destination of PI-Containers, their dimensions, their content or the nature of the good transported can be known only by a specific actor in the logistic network and are usually not shared both for legal or technical reasons. One of the goals of ICONET is to resolve these issues by using the power of PI and the use of an IOT system. The description of this system is out of the scope of this deliverable but let us say that with the use of the technological platform provided by ICONET all the below operations will be possible.

6.1 Port of Antwerp (LL1) – Container Loading

As defined in D3.3, the railway operations (which are the key aspect of LL1) are planned independently from all other elements of the logistics chain. Moreover, the available capacity (tracks / terminal / rolling stock / personnel) determines when a train can leave, which is often not aligned with customers, needs, resulting in a very inefficient use of the infrastructure. The primary business goal here is to shift to a more efficient model for railway transportation through better capacity/slot management, improved tracking and tracing management, transhipment operations, collective load planning, transport management and information management with all railway stakeholders.

For LL1, we investigated the possibilities to optimize rail traffic operations/planning in a hub environment (such as a seaport) based upon the principles of PI.

In the context of LL1 a possible real deployment of the phases presented above would be:

1. A set of PI-Containers arrive in the port and they are stored according to the position derived from a clustering algorithm that puts containers with common routes together. In addition to the clustering algorithm, containers that have similar weights would be stored together.
2. After this the train formation starts, the wagons are first loaded according to a loading schema derived by a 2-dimensional bin packing algorithm and secondly, they are assigned to all locomotives that are ready to leave the PI-Node of the port. The routing service has already defined the destination for each wagon so a neural network model can arrange trains fully loaded that can reach their second node.

In this process the right allocation of the containers and correct formation of the train will provide an increase in the number of containers moved but with minimal utilization of movers.

Living Lab 1 partners have provided data related to the movement of wagons in the Belgium railway system and related to the train composition in PoA. Such a dataset can be traditionally used for routing and packing optimization, however for the creation of a clustering model we have to take an alternative approach. LL1 partners have indicated that it is very difficult for logistics operators to share data regarding the content of containers, so the mapping of the users' demand has to change in the context of port operations. The historical dataset has to be derived from the composition of the trains, so that port authorities will know how much space has to be given to containers moving towards a usual direction. Not only information on the storage sequence of the containers has to be derived, but also the routes usually taken by the trains. This means that we cannot derive from the current dataset the right allocation

of containers but only after several runs of the application of packing and routing optimization. For this reason, we focus here more in the creation of the reinforcement learning aspect of packing and then we will present the possible solution for the clustering optimization.

6.1.1 Reinforcement Learning - Train Formation

In this Living Lab two reinforcement learning based use cases have been identified, one related to containers loading operations and the other is the formation of trains. While for the first use case we have already presented a heuristic solution in the **Bin Packing Solution – Packing Optimization** section, with the help of reinforcement learning we will try to improve those algorithms. To improve the efficiency of LL1, we decided to use as objective function the remaining surface of each wagon loaded, and weight and number of trains loaded.

In this case we will apply the sequence to sequence neural network proposed by Bello (Bello, 2016) and Hu (Hu, 2017). In particular Hu applied the sequence to sequence model to a vector formed by 3 spatial dimensions of the bin. In our case the 3 spatial dimensions are replaced with the weight, the priority and the presence of dangerous element in the wagon.

The neural network architecture in our research is shown in Figure 9 – RNN Architecture . The input to this network is a sequence of size data (weight, priority and danger) of PI-Wagons to be bundled, and the output of this network is another sequence which represents the order in which to bundle those wagons. The network consists of two Recurrent Neural Networks (RNNs): an encoder network and a decoder network. At each step of encoder network, the size data of one wagon is embedded and given as input to the Long Short-Term Memory cell and the cell output is collected. After the final step of the encoder network, the cell state and outputs are given to the decoder network. At each step of the decoder network, one of the outputs of the encoder network is selected as the input of the next step. Furthermore, the attention mechanism and glimpse mechanism proposed in (Bello *et al.*, 2016) are also used to integrate the output information of the decoder cell and the outputs of the encoder network to predict which wagon will be selected in each step.

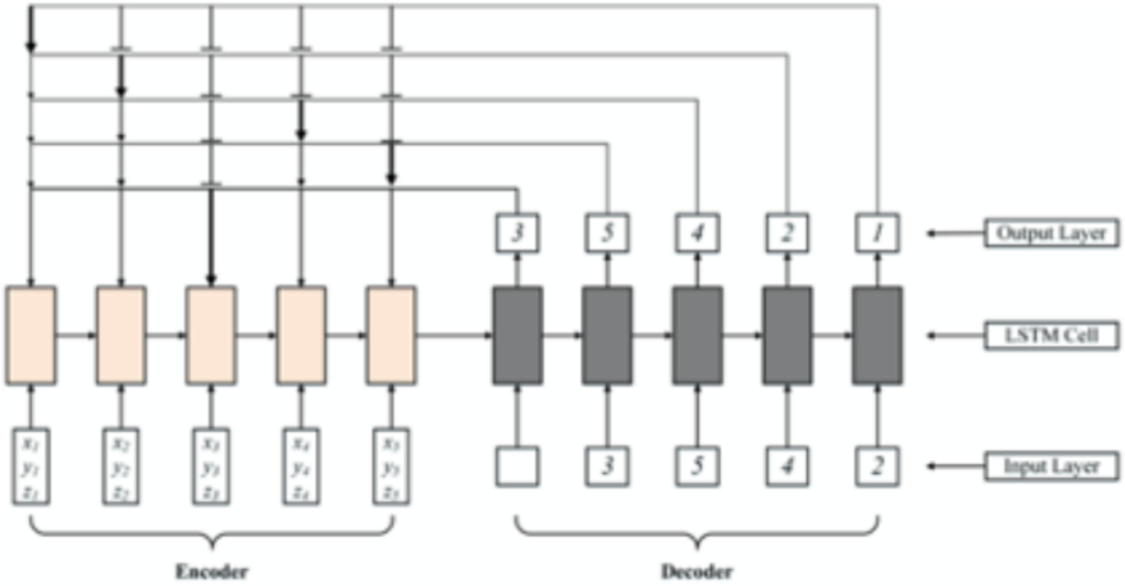


Figure 9 – RNN Architecture

6.1.2 Port of Antwerp (LL1) - Container Storage

In this section we present an implementation using synthetic data. Similar analysis will be applied to valid datasets as they become available to IBM and reported on in the final version of this deliverable.

PI-Containers were ranked by the number of days in which they appeared, see table below ("1" represents a container going to a specific destination, "0" represents no order to that destination). Note that in the context of Living lab 1, containers will be labelled according to their outgoing travel route. Let us say that in our synthetic PI-Node there are 50 destinations. A sample input for our model is as follows:

	Munich	Paris	Berlin	Vienna	Ljubljana
2015-01-01	0	1	0	0	0
2015-01-04	1	0	0	1	1
2015-02-02	0	0	1	1	0
….	……	…..		…..	…..

Table 2 - Sample Input

Existing research (Davis C. J, 2017) shows that Self organizing Maps is a good solution for LL1 so let us present a possible scenario of its usage. In this scenario only the departure day is used but one can test also adding quantities like weight or adding variations to the containers labelling like destination and dangerous goods and so on.

Each day that may or may not have PI-Containers on that day will be used by the SOM to find clusters of days that contain similar PI-Containers. The SOM will create a heat map for the warehouse.

To show what this looks like, an example shown from testing is shown in figures with 4 clusters, 100(10x10) neurons and 1699 days. The days that are assigned to nodes 0- 99 can be seen in *the Figure 10 - Heat Map*. This is the result of the SOM test. Notice that some nodes contained no days as their neighbouring nodes had a better match with the data.

	0	1	2	3	4	5	6	7	8	9
0	78	1	44	5	44	5	49	11	2	38
10	1	11	0	0	12	0	7	0	3	14
20	193	24	51	18	25	4	40	3	21	9
30	60	7	18	0	7	0	2	1	5	7
40	42	0	42	0	38	1	41	9	5	20
50	41	23	2	0	3	0	6	10	6	37
60	6	0	38	4	38	5	6	6	0	5
70	56	8	2	2	18	4	10	1	39	25
80	6	3	4	4	0	1	8	0	3	16
90	57	10	22	33	16	14	15	49	2	17

Figure 10 - Heat Map of Clusters in a Warehouse

Winning nodes can be read on heat maps by taking row headers and adding them to the column headers. Thus, node 99 is row 90 + column 9 and is in the far-right bottom corner on *Figure 10 - Heat Map* with a value of 17 winning days. An initial assignment of PI-Containers to zones can be achieved by splitting the number of days fairly evenly throughout the four zones. This method was chosen to possibly maintain some balance amongst the number of days each zone would potentially have without looking at PI-Containers density in those zones.

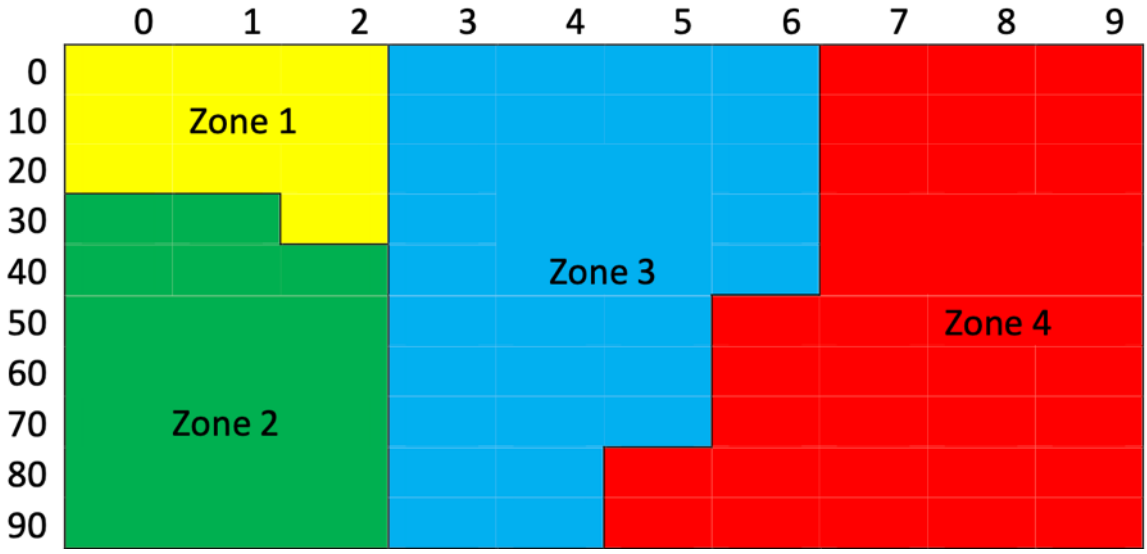


Figure 11 - Zone Map

In Figure 11 - Zone Map, shows visualisation of zones. The algorithm to select the destinations and compare how well the clustering performs is as follows: The number of days in which each container's destination appears in each zone is based on the SOM output calculations. The most popular destination is selected if the maximum number of allowable destinations has not been reached for that zone. In this example, 13 destinations are assigned to each zone, given that we have 4 zones and 50 destinations. A clustering that will assign all destination to a zone is a bad clustering.

In order to measure if the position is optimal one can compute the total distance computed by the picker to fill the loading order. By knowing this information, we can now plan and arrange containers by zones to reduce picking time and distance.

Deliverable D3.3 defined the following KPIs to measure the value for the real-world implementation of the ICONET solutions:

a. Number of transports per day (on the PI network)
b. Number of consolidated loading units (at PI node)
c. Use of railway capacity (in %) (on PI network)
d. Number of operations (at PI Node)
e. Number of pre-reserved slots (at PI node)

A direct consequence of using a high accurate packing algorithm will be an increase of the number of containers loaded (b) in a single train because the algorithm will allow to use all the available space in the train. Furthermore, we believe the above solutions will lead, to a direct decrease of the number of operations (d) required to load the train because the containers will be found closer in the yard area than

in the past. In fact, the packing algorithm will enable a better usage of the train loading space and reduce the number of movers used for the operations, leaving more free slots for rail movements.

6.2 Sonae (LL3) – Products Storage

As previously mentioned, the datasets from Sonae are pending but existing clustering analytics techniques applied to LL1 and LL4 will be reused here when datasets become available. KPIs' specific descriptions for LL3, described in D3.9, are as follows:

a) Decrease total e-commerce stock holdings
b) Increase in value of sales
c) Improve chain reliability, by decreasing the required product changes (with other equivalent products) in order fulfilments
d) Less stock-outs or product substitutions
e) Decrease the average delivery lead time
f) Decrease total cost of order fulfilment

It is evident that a) and b) are directly impacted by the demand of the user, so it is extremely difficult to act against these KPIs. A reduction of the stock and an increase in sales is related to customer behaviour and other influences outside the supply chain and logistics domain. Our solutions will design the shape of the warehouses or dark stores by indicating where to store the goods and cluster them according the usual associations made by customers, so it will be possible for Sonae staff to identify what are the most frequently requested products and restock only them. In particular, one of the algorithms belonging to the association rule-based technique, called Apriori (see the description in the following section), computes the association rule only between products having a high number of orders. So, the final result of our solution could design specific storage areas only with products frequently ordered by customers. This fact will reduce stock outs by having shops always containing products most used by customers. We also envisage that the clustering technique and the packing technique will decrease the cost and time required for filling an order. In fact, the packing service will increase the space utilization of trucks for delivery and the clustering will create a shop where the picker has to walk every time a short distance in order to fill an order

6.3 Stockbooking (LL4) – Container/Pallet Picking and Storage

In the ICONET vision the warehouse operations can be associated to a PI-Node optimization and this optimization can follow the phases indicated in previous sections.

In the context of LL4 a real deployment of the phases presented would be:

1. A set of PI-Containers (in this case palettes) arrive in a warehouse and they are stored according to the position derived from a clustering algorithm that put closer palettes having common routes. The clustering algorithm is applied in combination with group pallets of similar weights.

2. After this the truck loading starts, the trucks are loaded according to a loading schema derived by a bin packing algorithm. The routing service will then indicate the route to the truck using existing operational systems (PDA or smartphone of the driver, GPS on the truck), with data supplied by the PI Service Stack for that node.

The initial approach was to collect historical data on activities of a warehouse in order to see if it would be possible to identify patterns in the delivery process of the warehouse.

Stockbooking has collected data on the movements of products in a warehouse from March 2015 to January 2020. In particular we were interested on the quantities and products leaving the warehouse. In the below it is possible to see a description the dataset collected.

Table 3 - LL4 Dataset Description

Field	description
maj	Indicates the date of the movement
tmv	Indicates the type of movement. There are 5 types: arrival (EE), departure (SS), free the space (LS), internal putting (DE), internal pushing (DS). In some cases, there are no records so not applicable (NA)
pbru	Indicates the weight moved
allee	Line of the warehouse
prof	Depth of the warehouse
niv	Level of the warehouse
ref	Unique id of the products
npalfm	Unique id of a palette
qte	Number of units of a palette moved

A summary table is given **in Appendix 3: LL4 Data Summary**. It is possible to see that the dataset represents a series of records of movement of various quantities of products between palettes and outside of the warehouse. We focused our attention only on the movements of goods leaving the warehouse because we are interested in the presence of patterns in the demand of goods.

There are significant similarities in the approaches taken for both LL1 and LL4. In fact, even here palettes will be stored in specific location derived from the clustering algorithm and then they will be packed into trucks when they leave the warehouse. Thanks to the help provided by our partners in LL4 we are able to show some results of the analysis of data in a single warehouse as a means to demonstrate the validity of the approach. In **Appendix 2 –** it is possible to see association rules identified in the dataset. At this moment of the development we can only show some results related to the application of one of the techniques presented in the above sections. We have applied association rules-based models to the collected dataset. This fact can lead us to understand if there is the presence of pattern in this dataset that can be leveraged to optimize the service. Given the nature of the dataset (i.e. a sequence of transactions) we decided to apply an Apriori algorithm (Jafarzadeh, H., Torkashvand, R. R., Asgari, C., & Amiry ,A., 2015). Apriori algorithm are well known association models that after several computations, identify an element that is consequent to the presence of ta set of others. The identification is usually done by considering a set of specific values that indicate an evident pattern. In other words, if a set of antecedents is present in a percentage of records, how many times do you see a consequent (this value is represented by the confidence). We have cleaned the dataset by only considering 250 products delivered frequently and in high number of quantities. Next, the Apriori algorithm was applied to the cleaned dataset to identify the association rules between products. We have selected only rules having 10% in support and 70% as confidence. This means that we have selected only rules that appears 10% of the times in the records. Given the length of time period analysed this corresponds to almost 1 day per week. The 70%

confidence represents the number of times in which the consequent product appears. In the results table presented in Appendix 2, it is possible to see that in testing the percentage of the support and confidence remains constant, indicating that pattern in the data exists. In the next set of experiments, we plan to apply self-organizing map to this dataset in order to identify the right allocation schema. Unfortunately, the anonymization of the dataset makes it impossible to see which product name is related to the other, but we can see the relation between the codes. From this we can argue that a pattern is present. Similar to LL1 above, once these patterns are identified, better plans for loading and picking can be created.

In deliverable D3.12 Stock booking has defined the following KPIs as a measure of the improvement given by ICONET to its business:

a. Increase Warehouse provider participating
b. Increase volume of e-warehousing booking
c. Increase warehouse Quality of Service
d. Improve warehouse floor space
e. Make workshop & increase audience

Even in this case we have direct effect and undirect effect of the usage of our solution. KPI (c) is potentially a direct effect of the usage of our optimization. In fact, a warehouse is a place where a client can put their palette or multiple of them according to their location preferences. A warehouse that is capable of providing fast delivery will always be preferred to those in which such a thing it is not possible to be implemented. The clustering techniques we presented can increase the quality of the service provided by making faster order picking. As already highlighted, by putting together products that typically leave together, the warehouse will increase the chance that the new order will contain the goods of the same already processed order. It is also evident to us that for natural reasons a warehouse will start storing goods from the ground level, this process is chosen because at the moment of storage the worker does not know when and how the palette will be reused and puts it where it is more convenient at the that time. A clustering technique instead will identify specific areas for the storage of the goods, giving to the worker the opportunity of forecasting the future usage of the palette. For example, knowing that goods are likely to leave together the warehouse, one can put the one with more weight in the ground and the less heavy on upper level. This will lead to the increase of free space on the floor, as specified in KPI (d).

7 Algorithms as Services & Integration

7.1 Optimisation Service PI Integration

The PI Service stack as proposed by ICONET is based on the Open Logistics Interconnection model as discussed in (Ballou & Montreuil, 2012). This model lays out specific layers or services that play a distinct role in the movement of goods in a PI scenario. The general approach is that each layer solves a problem or issue in the shipment process with the info available. The physical layer is then passed all the info, insights and instructions which are carried out as physical operations (loading, driving etc.). Notably, this OLI model does not have an Optimisation Layer. This is because the OLI model was designed for creating the top-level operations for a shipment, such as: *Take container A to Warehouse Z at 10am on this truck*. There is no stake in how these operations will be performed by the logistics operator within the node. In the example above, no instructions are given to warehouse Z that the container should be stored in a particular slot on arrival, as that is beyond the context of the PI Service Stack.

This deliverable proposes an alteration in this approach. While logistics operators will continue to make their own decisions regarding internal node operations, relevant data can be leveraged (as shown in the **Data Driven PI-Node Optimization** section)to increase the efficiency in a given node. In the PI Services sequence diagram **Appendix 1 – Sequence Diagram** we see that the shipping service is the first contact point for node operations. This can be either an order created at or by that node or it can be an incoming order in progress from a neighbouring node. In each case the shipping service receives the PI order details digitally from either a logistics web service or the neighbouring node directly. This allows the PI Node to prepare, or reject the order if it is past capacity, has an issue etc. However, this information can now also be used for PI Node Optimisation operations.

While there are, as previously stated, commonalities between the different optimisation use cases within a node, the realities are, each operation to be optimised will have its own stakeholders, participants, IT systems and machinery. Therefore, the most realistic model for an optimisation service is to create a core service that can employ a number of techniques or algorithms to solve optimisation problems across some key use cases but for this service to be extended and/or customised to the specific use case scenario being addressed. This is an adoption approach previously put forward in the SELIS project and has been met with positive feedback by logistics industry stakeholders in the project.

By ensuring the core optimisation service leverages the PI service stack and follows the PI Data Model (as developed by CLMS as part of their architecture design work in **D2.1 – PI reference Architecture V1**), the optimisation service is guaranteed to be plug-and-play compatible with existing PI frameworks, ensuring that any extension or customisation is compatible with the specific requirements of an existing IT system within a PI node or the specific circumstances of the business model.

Part of the customisation process will require data streams to be connected where appropriate – an optimisation service for optimising storage in a warehouse needs access to real time and historic data to function efficiently. Therefore, as new incoming PI orders are declared via the shipping service from a neighbour node, this info will also be forwarded to the optimisation service where it can be made part of the optimisation strategy. Similarly, any internal systems keeping historical data on all past PI orders will be made available in the same manner.

If we use the diagram in **Appendix 1 – Sequence Diagram** as a basis but focus and expand on the elements relating to internal node operations, we can see some details of the data exchange between actors.

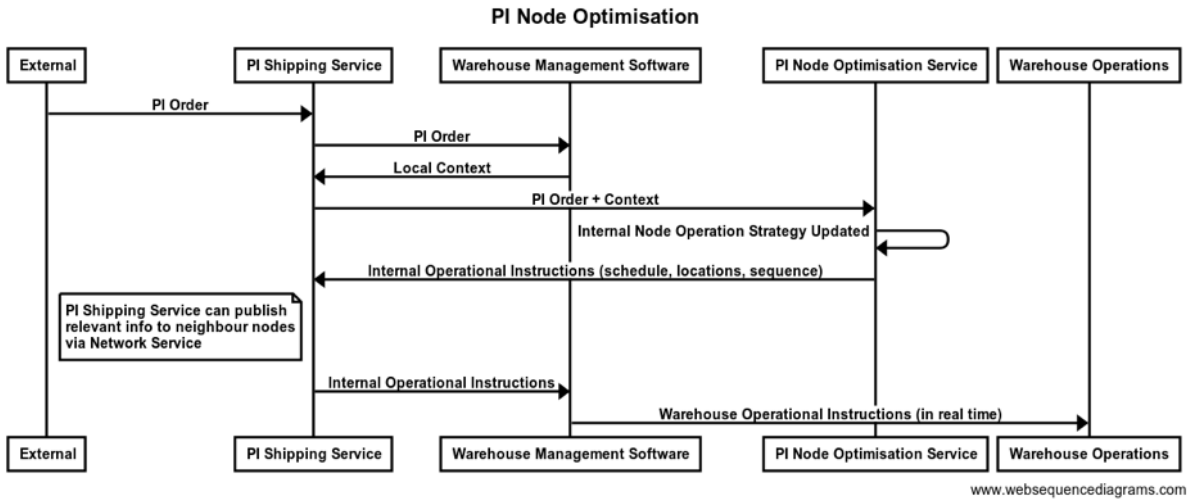


Figure 12 - PI Optimisation Service Sequence Diagram

In Figure 12 - PI Optimisation Service Sequence Diagram we see a basic PI Node in the form of a warehouse in which there is:

- A PI Shipping Service, acting as the PI Interface to the warehouse
- A PI Optimisation Service to optimise operations within the warehouse
- Warehouse Management Software (WMS) which governs the warehouse operations
- Warehouse Operation which represents actual actors (staff, trucks, forklifts etc.)

What we note here is that the PI Shipping Service continues to be the interface to the warehouse systems. Information passed by the WMS to the optimisation service goes via the Shipping service as that interface is already formalised, published and established. The Shipping Service coordinates the exchange of information between the Optimisation Service, the WMS and external entities. The WMS finally interacts with Warehouse Operations as this is a non-standardised interaction that will be specific to the node, and this is the stage where actual actions to be taken are published to the actors to carry them out (i.e. a forklift instructed to collect a specific pallet or container to be brought to a specific location).

7.2 Web Service Implementation

This deliverable has proposed a number of different algorithms and machine learning approaches for tackling optimisation goals within a PI Node. In order for these algorithms and approaches to be used in a PI Service Stack deployment they need to be deployed as API compatible web services. Initially, each distinct algorithm or machine learning model will require its own dedicated service. These can be then further grouped together at a later point as a service offering through an API gateway service.

Any of the optimisation web services can implemented as a Representational State Transfer (ReST) API. This ReST API is made accessible with the use of a web server and portability is achieved through the use of Docker images, allowing the service to be run in a wide variety of systems and offerings by major cloud vendors. The following sections describe each of the components of a web service implementation.

7.2.1 Optimisation ReST API

The example optimisation API implemented here concerns the loading of containers onto train wagons implemented in a standard algorithm. The API accepts two input parameters, a list of available wagons to be loaded and a list of containers to be despatched. The API calculates an optimised loading plan for the inputs and returns this as a JSON string to the caller. The API is implemented in the Python[1] programming language and uses the Flask-ReSTPlus[2] Python library to create the ReST API functionality.

ReST APIs use the HTTP request methods (DELETE, GET, POST, PUT) to perform a logically related action. In this case, we are creating an optimised wagon loading plan, so the implementation is associated with the PUT HTTP method.

7.2.2 Python Flask Web Server

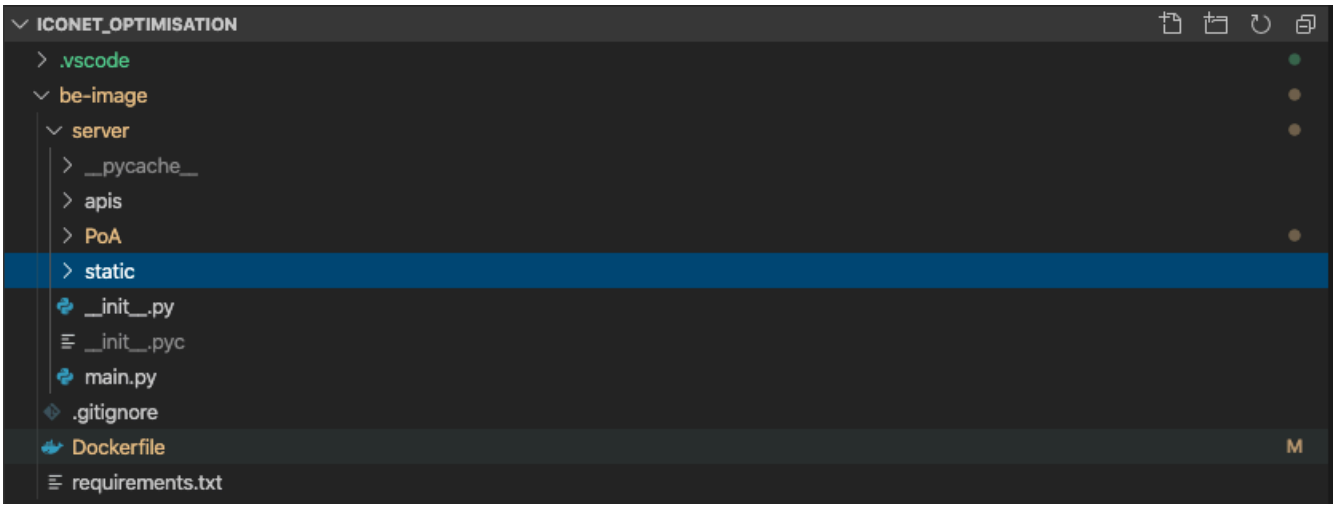


Figure 13: Python Flask Web Server

The Python Flaskweb[3] server is used to make the Wagon Loading ReST API available as a cloud-based web service. *Figure 13: Python Flask Web Server* shows the structure of the web server in a source code editor. The components of the web server are:

- `main.py`: the entry point for the server Python code
- apis: a subdirectory used to organise the ReST API code. The Flask-ReSTplus based classes are located in this directory.
- PoA: a subdirectory used to separate the core optimisation library Python code from the server code
- static: a subdirectory containing static resources e.g. images, UI code

[1] https://www.python.org/

[2] https://flask-restplus.readthedocs.io/

[3] https://flask.palletsprojects.com/

7.2.3 Machine Learning Models as API Services

The approach for leveraging a machine learning model as an API service is similar to the approach described above for a traditional algorithm. As above there is a folder structure that contains the required elements for a Python Flask web server such as the `main.py` Python file and the APIs and static subdirectories. In the case of a service powered by a machine learning model, the model itself is also included as a Python file (or files) and is in turn used by a Python program that can take requests coming from the API and run them against the model.

An extra element that must be addressed is training the model. This is done manually in our initial deployments but in V3 of this deliverable and indeed in a production level deployment this process would need to be automated. In ICONET, datasets are provided once off in a static manner by Living Lab partners, chosen for their relevance but also by how safe the data is to share in terms of privacy and confidentiality. In an implementation where the service owner is also the data owner, the data is much more readily available. Machine learning models are only as good as the data they are trained with so there is a significant benefit to retraining models with the newest and most up to date data, especially if data mining techniques that acquire this data have also been improved. To avail of this an automated pipeline that captures the latest datasets and trains new models will be used. This new model will in turn be passed to a pipeline that builds a new API service docker image with the newly trained model within. This can then be tested and verified before being deployed, replacing the existing service API. This process will be discussed in more detail in **D2.21 – PoC Integration Environment V3.**

7.2.4 Docker Image

To make the Python Flask web server and the Wagon Loading ReST API portable, they have been packaged as a Docker[4] image. Docker is a containerisation technology that encourages applications to be built using microservice architecture and packages everything an application requires to execute in a portable, binary format. Docker images can be composed from multiple images, thereby separating functionality and promoting re-usable components. The Docker image for the Wagon Loading API uses a popular, base Flask web server image called tiangolo/uwsgi-nginx-flask. Docker images are defined in a simple text-based file format, called a Dockerfile, and are then built into a binary format that can be executed. The Dockerfile for the Wagon Loading ReST API imports the base Flask web server image, sets the hosting configuration and installs the Python runtime requirements.

7.3 Simulation Integration

The initial manner in which the Optimisation Service described above can be validated is via the simulation services and models provided by Itainnova using the Anylogic Simulation System. This is deployed within the Proof of Concept Environment, and the Optimisation Service is able to freely interact with it and vice-versa. In the simulation model shown in Figure 14 - Simulation Model, we can see the creation of trains from individual wagons based on those wagons ultimate destination (branch). Subsequent iterations of this model will include the creation of those wagons by the adding of individual containers. We can see a number of trains (grey rectangles) in the process of being created with the coloured boxes representing wagons of specific destinations. There are also graphs indicating which branches are currently filling fastest, and most required overall.

[4] https://www.docker.com/

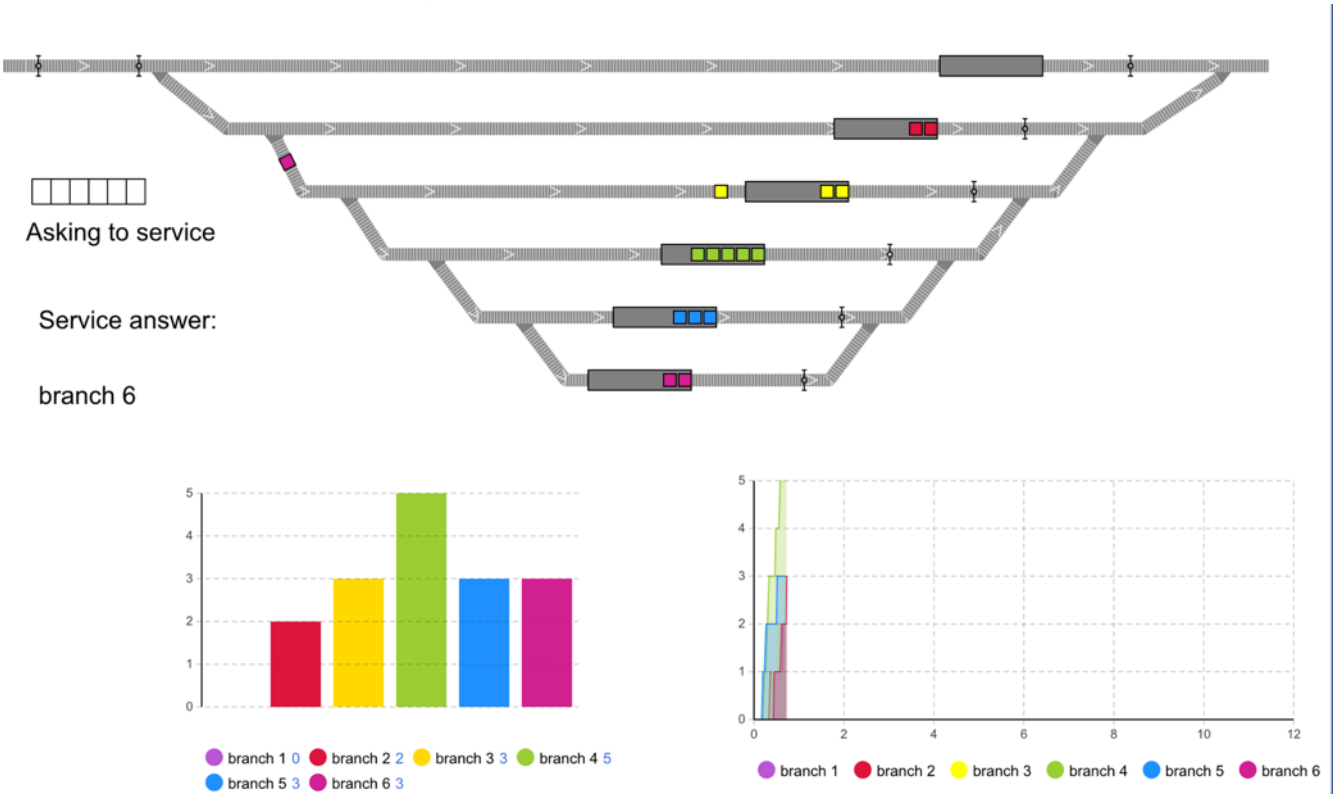


Figure 14 - Simulation Model

Currently the focus is on the integration between the simulation service running the model and the services themselves. As discussed above, the optimisation is running a REST API available at <IP_Address>/ENDPOINT.
The simulation will generate a list of containers and their associated attributes. Most critical is the destination, which will define what train the container will be added to. Next most critical are size and weight as this is how the loading into wagons will be evaluated. Also provided is a train schedule. The port of Antwerp has a pre-set schedule of track availability (booked in advance by port operators) so the times and destinations of whole trains are pre-determined. The simulation will pass these pieces of info to the Optimisation Service. The optimisation Service calculates the optimal approach and returns a "loading plan" which is a list of simple ordered instructions such as

- "Load Container A onto Wagon 2"
- "Add Wagon 2 to train on track 5"

The simulation will accept this plan and follow the instructions within the simulated model loading containers and moving wagons as the optimisation service suggests.

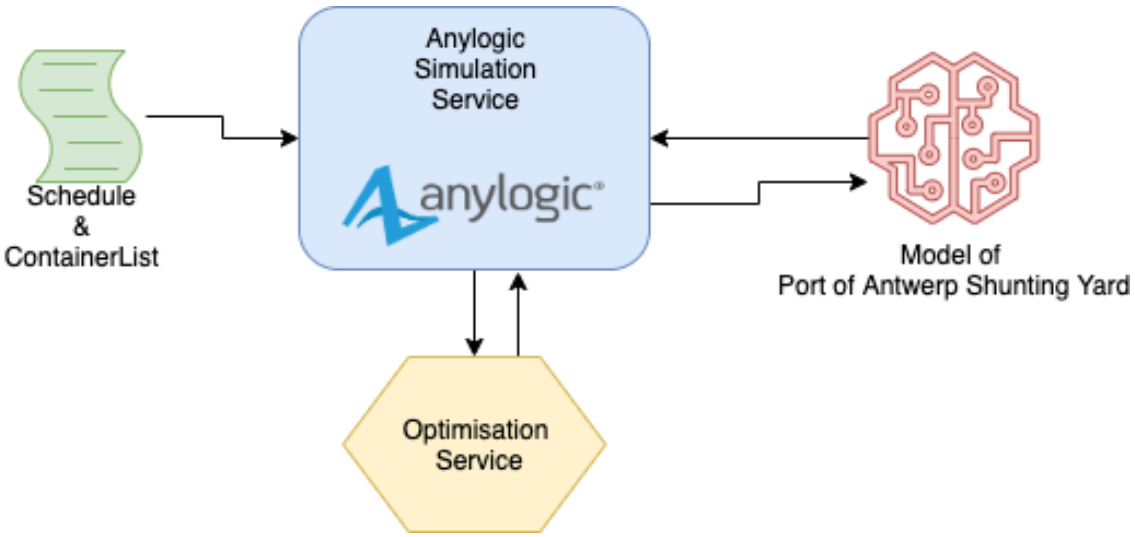


Figure 15 - Simulation Integrations

In the next version of this deliverable the simulation and service will allow for an "on the fly" preplanning. So, if mid-way through the loading plan something has changed (a new container arrives to be loaded for example), the simulation can ask the optimisation service for a new loading plan based on the current state and new information.

In this scenario the loading requirements are static. This is to allow for us to compare two simulations running the same requirements – one running the optimisation service, and one running on a random or simple first come first serve approach. This latter approach would be similar to the one employed in the Port of Antwerp shunting yards currently. Once the results are available from the two simulations, it is possible to compare via a number of KPIs:

- Throughput – How many containers can be moved through the yard in a given time
- Idle Time – How long are trains sitting idle before they are able to move
- Average Container Wait – How long is the container sitting in the shunting yard before it leaves
- Loading Efficiency – How many wagons were used to transport how many containers.

8 Conclusions

The main conclusion drawn from the work done in this deliverable is that data is a key factor in driving optimisation within a PI Node or Hub. Any PI Node may have a number of distinct optimisation challenges within and while there are often significant overlaps of these use cases across different PI Nodes, each use case has its own edge cases and challenges unique to that node. For this reason, it's not possible to design a single optimisation service, but rather better to design a number of services that broadly intersect the majority of use cases and allow for minor customisations that close the gap. Once services are customised, data access is a key factor to their success. This deliverable concludes that data sharing and access for internal node optimisation purposes must become a goal of the PI Service Stack and PI model in general. The PI concept is built around data sharing and formalised exchange of information through the PI protocol. This deliverable concludes an addition should be made to facilitate internal node optimisation. The initial steps of this integration are discussed as part of the PI Service implementation, specifically how the PI Node Optimisation Service would become integrated into the PI Service stack.

Closely related to the need for data exchange is the application of machine learning to the PI-Node optimisation challenge. Our research indicates that the most effective manner in which machine learning can be applied is through an ensemble approach. By combining the strengths of multiple models, a greater chance of accuracy and success can be achieved. While a given model may be able to successfully address a given use case, not all models can be applied to all use cases. While there are some commonalities, it was decided that multiple algorithm types should be investigated to cover the widest possible set of use cases. A number of approaches and models were trialled against living lab data with reasonable success but in version 3 a greater cross section of models and data will be applied to the living labs to expand on these successes.

Again, noting data exchange as a critical element, this deliverable concludes that alternative approaches would need to be developed in the event of incomplete, low quality, or curated data. In Living Lab 1 the datasets available granted only a partial picture of the current operations within the port. For this reason, a shift to reinforcement learning through neural networks was adopted, which allowed insights to be gathered from the available datasets. We also observed a significant crossover in the optimisation techniques used in LL1 and LL4, showcasing the adaptability of the chosen approaches and identified the prospect of further extending into LL3 as data becomes available. Finally, we saw that when optimisations were applied in LL1, LL2, and LL3 there was an opportunity for node optimisation to address the different KPIs across the Living Labs. This demonstrates the important role optimisation must play in any future PI frameworks as internal operations can affect performance in just as meaningful a manner as the PI network itself.

9 References

Pang, K. W., & Chan, H. L. (2017). Data mining-based algorithm for storage location assignment in a randomised warehouse. *International Journal of Production Research*, *55*(14), 4035-4052.

Roodbergen, K. J., & De Koster, R. (2001). Routing order pickers in a warehouse with a middle aisle. European Journal of Operational Research, 133(1), 32-43.

Jafarzadeh, H., Torkashvand, R. R., Asgari, C., & Amiry, A. (2015). Provide a new approach for mining fuzzy association rules using apriori algorithm. Indian Journal of Science and Technology, 8(8), 707-714.

Wagstaff, K., Cardie, C., Rogers, S., & Schrödl, S. (2001, June). Constrained k-means clustering with background knowledge. In Icml (Vol. 1, pp. 577-584).

Davis, C. J. (2017). Using Self-Organizing Maps to Cluster Products for Storage Assignment in a Distribution Center (Doctoral dissertation, Ohio University).

Agrawal, R., Imieliński, T., & Swami, A. (1993, June). Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD international conference on Management of data (pp. 207-216).

M. Fazili, ""Physical internet, conventional, and hybrid logistic systems: An optimization based comparison."," 2016.

Coffman, Jr, E. G., Garey, M. R., Johnson, D. S., & Tarjan, R. E. (1980). Performance bounds for level-oriented two-dimensional packing algorithms. SIAM Journal on Computing, 9(4), 808-826.

Martello, S., Monaci, M., & Vigo, D. (2003). An exact approach to the strip-packing problem. INFORMS journal on Computing, 15(3), 310-319.

Martello, S., Pisinger, D., & Vigo, D. (2000). The three-dimensional bin packing problem. Operations research, 48(2), 256-267.

Zhang, Defu, et al. "A priority heuristic for the guillotine rectangular packing problem." *Information Processing Letters*116.1 (2016): 15-21.

Fazili, Mehran. "Physical internet, conventional, and hybrid logistic systems: An optimization based comparison." (2016).

Vinyals O., Fortunato M. , and Jaitly N. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

Bello J., Pham. H , Le Q. , Norouzi M ., and Bengio S.. Neural combi- natorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*, 2016.

Hu, Haoyuan, et al. "Solving a new 3d bin packing problem with deep reinforcement learning method." *arXiv preprint arXiv:1708.05930* (2017).

Pang K W , Chan H L . Data mining-based algorithm for storage location assignment in a randomised warehouse[J]. International Journal of Production Research, 2016, 55(14):1-18.

Ballestín, Francisco, et al. "Static and dynamic policies with RFID for the scheduling of retrieval and storage warehouse operations." *Computers & Industrial Engineering* 66.4 (2013): 696-709.

Belov, Problems, models and algorithms in one- and two- dimensional cutting, Ph.D. Thesis, Technischen Universität, Dresden, 2003.

Coffman, Garey, Tarjan, Performance bounds for level oriented two-dimensional packing algorithms, SIAM J. Comput. 9 (4) (1980) 808–826.

Metzger, A., Neubauer, A., Bohn, P., & Pohl, K. (2019, June). Proactive process adaptation using deep learning ensembles. In International Conference on Advanced Information Systems Engineering (pp. 547-562). Springer, Cham.

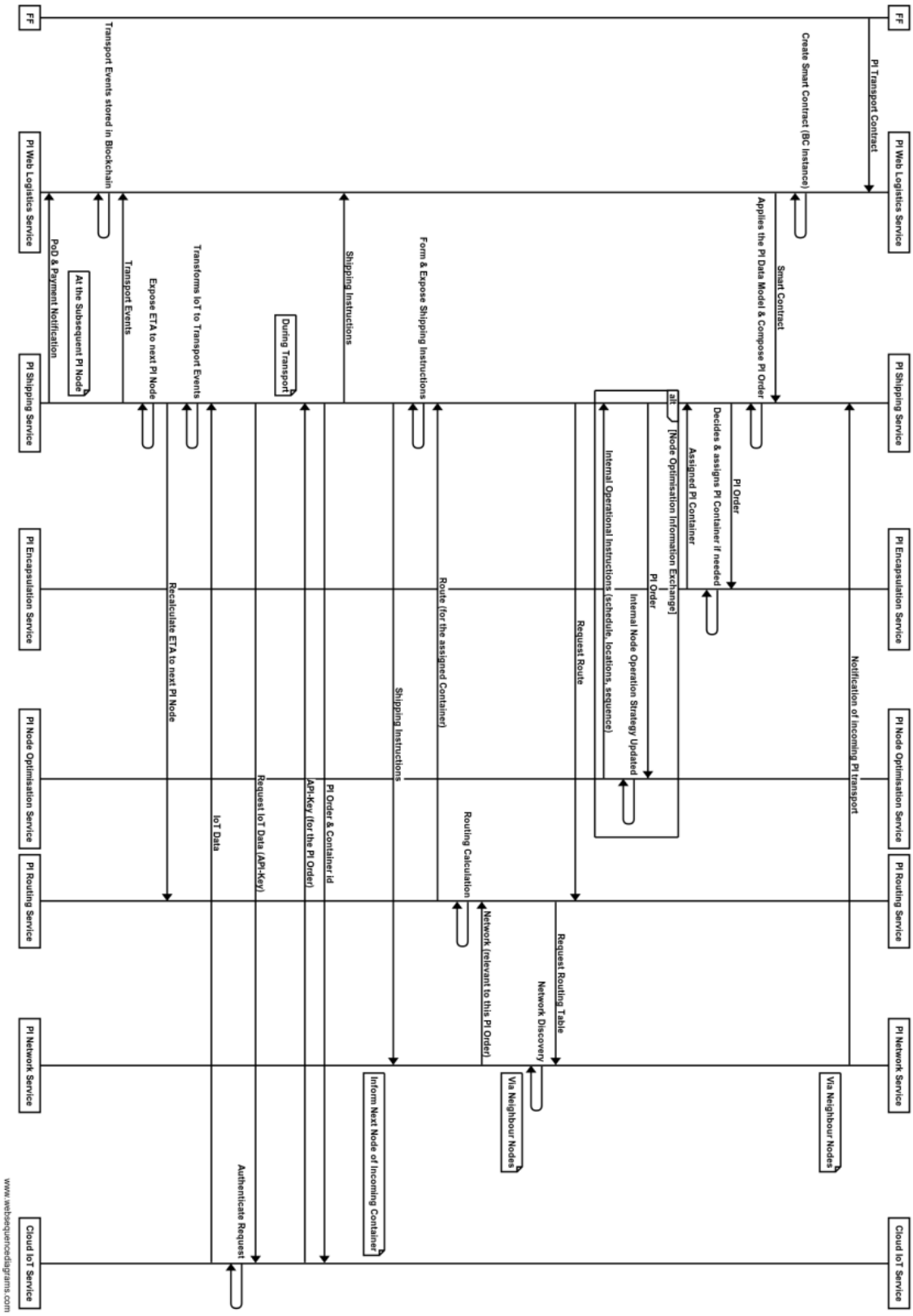
Bekkerman, R., Bilenko, M., & Langford, J. (Eds.). (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.

Montreuil, B., Meller, R. D., & Ballot, E. (2012). Physical internet foundations. IFAC Proceedings Volumes, 45(6), 26-30.

Dietterich, T. G. (2000, June). Ensemble methods in machine learning. In *International workshop on multiple classifier systems* (pp. 1-15). Springer, Berlin, Heidelberg.

Evermann, J., Rehse, J. R., & Fettke, P. (2017). Predicting process behaviour using deep learning. *Decision Support Systems*, *100*, 129-140.

10 Appendix 1 – Sequence Diagram



11 Appendix 2 – Results of Apriori Algorithm Applied to Stockbooking Dataset

Consequent	Antecedent	Support %	Confidence %
12107445	12282287 and 12066190 and 12217669	10.272	73.529
12237459	11896127 and 12221448	10.977	72.477
12107445	12172021 and 12107618 and 12237471	10.574	72.381
12107445	12231620 and 12066190 and 12273970	11.883	72.034
12237459	12019901 and 12177526 and 12107445	10.07	72
12200193	12265849 and 12128308 and 12251709	10.775	71.963
12019902	12019904 and 12204136 and 12260193	10.373	71.845
12107445	12282287 and 12066190 and 12273970	10.272	71.569
12107445	12282287 and 12217669 and 12218664	10.272	71.569
12107445	12183394 and 12217669 and 12272871	10.272	71.569
12107445	12124851 and 12273970 and 12272871	10.171	71.287
12207032	1622590 and 12116712 and 12143041	10.473	71.154
12107445	12172021 and 12191407 and 12107618	10.473	71.154
12107618	12265430 and 12265859	10.775	71.028
12207032	12291031 and 12243545 and 12044161	10.07	71
12237472	11337186 and 12282891 and 12204136	10.07	71
12107445	12183394 and 12217669 and 12266404	10.07	71
12207032	12102570 and 12062867 and 12237471	10.07	71
12251709	12273509 and 12200193 and 12207032	10.07	71
12258581	12266400 and 12273730 and 12066190	10.07	71
12237459	12258540 and 12124851 and 12258221	10.675	70.755
12237459	12275928 and 12098836	10.977	70.642
12066190	12282287 and 12264025 and 12107445	10.272	70.588
12207032	12120336 and 12208858 and 12258221	10.272	70.588
12218664	12164847 and 12208858 and 12207032	10.272	70.588
12258582	12124851 and 12256678 and 12219016	10.272	70.588
12218664	12208858 and 12155005 and 12256678	10.272	70.588
12219010	12209677 and 12265849 and 12191407	10.272	70.588
12107445	12141756 and 12273970 and 12272871	11.279	70.536
12208858	12167638 and 12155005 and 12218664	11.279	70.536
12107445	12208858 and 12107618 and 12258582	10.574	70.476
12019902	12303316 and 12098780	10.876	70.37
12251709	12120336 and 12272870 and 12241747	10.473	70.192
12019902	12019903 and 12258582 and 12251709	10.473	70.192
12107445	12141756 and 12219015 and 12237471	10.473	70.192
12107445	12208858 and 12066190 and 12191407	10.473	70.192
12256678	12092337 and 12265859	10.775	70.093
12207032	12221448 and 12116712 and 12272871	10.775	70.093

12258221	12120336 and 12273802 and 12207032	10.775	70.093
12107445	12303294 and 12273509	10.07	70
12062867	12275913 and 12270450	11.078	70
12107445	12102570 and 12272870 and 12273970	10.07	70
12177528	12230324 and 12177526 and 12258582	10.07	70
12251709	12266400 and 12265849 and 12116714	10.07	70
12107445	12141756 and 12272870 and 12273970	10.07	70
12251709	12161316 and 12265849 and 12116714	10.07	70
12208858	12167638 and 12218664 and 12207032	11.078	70
12218664	12207210 and 12241747 and 12177528	10.07	70

12 Appendix 3: LL4 Data Summary

Field	Min	Max	Mean	Std. Dev	Median	Mode	Unique	Valid
maj	2015-01-06	2020-04-01	--	--	2017-08-28	2018-10-12	--	7191271
tmv	--	--	--	--	--	EE	6	7191271
pbru	0.000	1029.600	197.390	157.762	178.360	220.320	--	7191271
allee	1.000	999.000	253.922	243.926	176.000	160.000	--	7191271
prof	1.000	999.000	135.987	151.787	72.000	201.000	--	7191271
niv	1.000	10101.000	8.773	22.758	3.000	1.000	--	7191271
ref	140039.000	43932990.000	12120466.273	2184508.043	12263964.000	12310574.000	--	7191271
npalfm	--	--	--	--	--	--	--	7191271
qte	0.000	448.000	50.656	43.296	42.000	63.000	--	7191271

Field	Outliers	Extreme	Action	Impute Missing	% Complete	Valid Records	Null Value	Empty String	White Space	Blank Value
maj	0	0	None	Never	99.999	7191271	58	0	0	0
tmv	--	--	--	Never	99.999	7191271	0	58	58	0
pbru	133796	420	None	Never	99.999	7191271	58	0	0	0
allee	1	0	None	Never	99.999	7191271	58	0	0	0
prof	58059	3503	None	Never	99.999	7191271	58	0	0	0
niv	1284	29	None	Never	99.999	7191271	58	0	0	0
ref	99657	54090	None	Never	99.999	7191271	58	0	0	0
npalfm	--	--	--	Never	99.999	7191271	0	58	58	0
qte	89855	26708	None	Never	99.999	7191271	58	0	0	0