Contract No 777594

Collaborative project GA-777594

OptiYard - Optimized Real-time Yard and Network Management

Deliverable 5.2

Yard Optimization Algorithm, Network Decision-Support Tool and Integration Framework

Due date of deliverable: 30/09/2019

Actual submission date: 12/11/2019

Leader of this Deliverable: IFSTTAR

Reviewed: Yes

Project funded from the European Union's Horizon 2020 research and innovation programme		
Dissemination Level		
PU	Public	✓
CO	Confidential, restricted under conditions set out in Model Grant Agreement	
CI	Classified, information as referred to in Commission Decision 2001/844/EC	

DOCUMENT STATUS

Document status		
Revision	Date	Description
1	31.07.2019	Description of the state of the art on yard optimization
2	23.08.2019	Description of the simulation and optimization framework
3	23.09.2019	Description of the state of the art in network integration
4	23.09.2019	Description of the strategies for network integration
5	29.09.2019	Description of the optimization algorithm
6	15.10.2019	Introduction, Conclusion and Executive summary

REPORT CONTRIBUTORS

Name	Company	Details of Contribution
Chapter 1	IFSTTAR	Description of the state of the art on yard optimization
Chapter 2	UNIVLEEDS	Description of the state of the art in network integration
Chapter 3	IFSTTAR	Description of the optimization algorithm
Chapter 4	IFSTTAR	Description of the simulation and optimization integrated framework
Chapter 5	DICEA and UNIVLEEDS	Description of the strategies for network integration
Introduction, Conclusion and Executive summary	IFSTTAR	Introduction, Conclusion and Executive summary

EXECUTIVE SUMMARY

This report is the second deliverable in Work package (WP) 5.2 "Optimization of information and communications methods" of the Optimized Real-time Yard and Network Management (Optiyard).

We present here the OptiYard optimization and simulation integrated framework, together with strategies to include it in an overall railway network management.

We start by describing the state of the art on the optimization of yard management processes, considered in their own or in connection with the network. This leads to the presentation of the algorithm designed in OptiYard and to the description of the integrated framework. Finally, strategies for the inclusion in the network are proposed.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ACRONYMS

BTAP = Block to Track Assignment Problem
DP = Dynamic Programming
DSS = Decision Support System
ETA = Estimated Time of Arrival
ETD = Estimated Time of Departure
FEP = Feasible Earliest Path
HSP = Hump Sequence Problem
HYBA = Hump Yard Block-to-Track Assignment
IM = Infrastructure Manager
KPI = Key Performance Indicator
IP = Integer Programming
PAP = Pullout Allocation Problem
RAS = Railway Applications Sections
RU = Railway Undertaking
SSSWT = Single Stage Sequencing Problem with Weighted Tardiness
TAF-TSI = Telematics Applications for Freight Technical Specification for Interoperability
TIS = Train Information System

1. INTRODUCTION

Managing a freight yard is a very complicated task that is currently performed mostly manually by yard operators. In the optimization literature, this management is typically decomposed in subproblems which are solved separately, considering only a subset of yard resources as constraining. Typically, only tracks are considered as bottlenecks in the system, hence only their utilization is an object of study. Indeed, this is a limited view of reality, in which shunting locomotives and crews are also necessary to carry out operations on trains. They are typically available in limited quantities and their movements and schedules need to be precisely organized. Moreover, the literature on such isolated subproblems does not try to answer a fundamental question in yard management: how can this management be done consistently with the overall network? In Chapter 2, we report an analysis of the current state of the art, in which we highlight the reasons why it needs to be extended.

In Chapter 3, we propose an optimization algorithm that may represent such an extension. In its current form, it considers interactions with the network represented by regular updates on trains expected arrival times. It fills the main gaps previously identified, namely: it optimizes the schedule of all yard resources, considering their movements on a microscopically modeled infrastructure, as it does for trains. Moreover, it is a generally applicable algorithm, since it makes no assumption on the structure of the yard and of its operations. This algorithm is designed to be used in the operational phase, while train services are being carried out and when it may be necessary to quickly respond to perturbations coming from the yard itself or from the network.

In Chapter 4, we present the OptiYard optimization and simulation integrated framework. It brings together the designed algorithm and the Villon microscopic simulator described in deliverable D4.2 “Yard simulation software for WP6” [1]. It aims to reproduce in laboratory a realistic situation in which a yard is automatically managed in an optimized way. In the framework, the simulator plays the role of a real yard and constantly communicates with the optimization module in charge of making decisions. These decisions concern all resource assignments and all wagon treatments necessary to effectively carry on yard operations. The optimization algorithm updates its decisions periodically, to make sure it always considers the actual state of the yard and the latest forecasts on arriving trains. It communicates constantly these decisions to the simulator which implements them. This type of framework is named “closed-loop” framework.

In Chapter 5, we discuss how such a framework, and more in general how the OptiYard Decision Support System designed in WP4 – Modelling, can be integrated in the overall network management. In particular, we report a proposal matured in collaboration with members of the FR8HUB consortium, FR8HUB being a strictly related ongoing Shift2Rail project.

Finally, in Chapter 6, we draw conclusions.

2. STATE OF THE ART ON YARD OPTIMIZATION AND NETWORK INTEGRATION

The literature on railway system optimization is vast. Some of the main problems are called line planning, timetabling, platforming, rolling stock and crew scheduling [2]. These problems typically consider passenger railway systems. However many studies on less known problems exist, as the ones treated in this paper, in which we propose a literature review specific to the freight railway system.

Rail freight traffic has been in steady decline for about 25 years [3]. However, political authorities as the European Union continue to support research on rail freight transport, mainly for social and environmental aspects. For example, new technologies are to be introduced in shunting freight yards (named simply *yards* in the rest of the document), which are areas where operations are performed to create and recombine freight trains. Today, these yards automate some of their operations (e.g., through automatic switches and automatic brakes) but the integration of optimization tools is still lacking.

In this chapter, we report the review of the state of the art on the optimization of yard management. Then, we summarize the contributions considering the yard in connection with the network.

2.1 YARD OPTIMIZATION

In rail freight transport, there are two types of services: the Full Train Load and the Car Load [4]. In the former, trains keep their structure throughout the whole journey since all cars have the same origin and the same destination. In the latter, trains must be split, and cars recombined to form new trains since they do not share the same origin and/or destination. This literature review deals with the management of yards which are related to the Carload services. Indeed, when Full Train Load services need to stop in yards, the operations to which they are submitted can be seen as a subset of those performed on Car Load ones.

Although many variants exist, as we will discuss in the rest of the chapter, the general sequence of operations in a yard can be described as follows. Once a train arrives, it is stored on a first set of tracks where its cars are inspected. Then, the locomotive is stored to be coupled later with a new train and the cars are detached from each other. A shunting locomotive arriving behind the cars pushes them over an artificial hill named *hump*. Rolling down from the hump with the aid of gravity, the cars are classified on tracks dedicated to the new trains they will leave the yard on. The route setting is typically automated through a car identification system. Once a new train is formed, it is stored on a last set of tracks to wait the right time to enter in the rail network.

The efficiency of yards has an impact on the fluidity of the global rail system and on trains travel time. In [5], it is stated that the percentage of yard time compared to total travel time of a train may be from 10 to 50%. Efficiently managing yards may allow the decrease of this percentage. Today, this efficiency is sought mostly manually by operators, and practically no intelligent decision-support tool is used in real cases.

Our analysis follows two relatively recent literature reviews [6], [7] on yard management. Many of the algorithms presented there are interesting mostly from a theoretical perspective but are not necessarily practical in real-world yards. This is due to the fact that only very specific sub-problems are typically tackled, neglecting the complexity of the overall operations [8]. Since 2012, some studies consider more comprehensive models (i.e., models covering all or most of the operations performed in yards) moving a step closer to reality.

2.1.1 Layouts, Rolling Stock and Operations in Yards

In addition to the way operations are organized, the efficiency of yard management is directly linked to the particular layout of the infrastructure [9]. Even if the layout considered in the literature is often the same, there is a great variety of yards in reality. Figure 1 represents a typical yard layout where the tracks can be divided into four different parts: *receiving tracks*, hump, classification *tracks* and departure *tracks*.

Moreover, yards can have a return track which connects the classification to the receiving tracks. The presence and characteristics of these parts differentiate yard layouts.

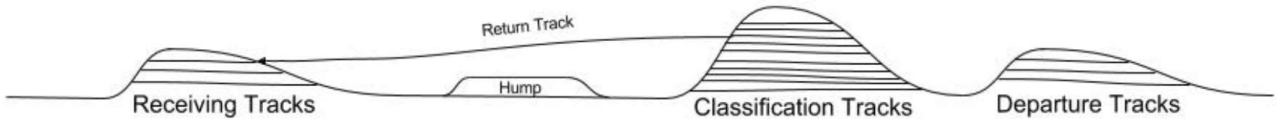


Figure 1. Schematic representation of a typical yard

Trains entering a yard are called *inbound trains*. They are stored in the receiving tracks, and once the cars have been uncoupled from their locomotive and are detached from each other, the series of cars on a receiving track is called a *cut*. Train locomotives are mainly used to enter and to leave the yard, while *shunting locomotives* are used to push and pull cars along yard tracks. The new trains which are built on the classification tracks and which will leave the yard through the departure tracks are called *outbound trains*.

Layout

The layout can differ from one yard to another. Its particular characteristics can make the system and the management processes more or less complex. We partition these characteristics depending on the four specified track groups: receiving tracks, hump, classification tracks and departure tracks. For each of these classes, we list the main aspects impacting complexity in the following:

- **Receiving tracks.** In yards with long receiving tracks, several inbound trains can be sent to one track in order to form a cut. Once all the locomotives have been detached and moved, the full cut is pushed over the hump. In this case, the combination of the different cuts on the receiving tracks can be strategic for the effectiveness of yard management.
  Some yards do not have any receiving track. If there are no receiving tracks, inbound trains are stored directly on the classification tracks.
- **Hump.** There is not necessarily one and exactly one hump even if this is the most common case. Depending on the presence and the characteristics of humps, there are 3 categories of yard:
  - flat-shunted yard,
  - gravity yard,
  - Hump yard.

  The flat-shunted yard has neither hump nor hill: the shunting locomotive pushes the cars until the classification tracks. In this case, more energy is in general consumed by locomotives than in the case of classic humps, and this can have an impact on the strategy employed by operators. The gravity yard does not have any hump as well but trains are going downhill from receiving to departure tracks. It is considered as the most efficient and is very often used on large systems [10] [11].
- **Classification Tracks.** The length of classification tracks can have a remarkable impact on the shunting process. For example, if these tracks are too short to build a full outbound train, the train can be divided into sub-trains on several tracks, sub-trains which are then merged on a longer departure track.
  The number of classification tracks is also important. If there are fewer tracks than the number of outbound trains to build starting from a set of inbound trains, the shunting process must be adapted. In this case, some classification tracks called *mixing tracks* can be used to store cars while waiting for

an available track for building the outbound train to which they are aimed. The typical classification tracks are then called *formation tracks* and are used to build outbound trains.

- **Departure Tracks.** Some yards do not have departure tracks. In parallel to the case in which no receiving tracks are present, if there are no departure tracks, outbound trains which are ready to leave the yard stay in the classification tracks until their departure. The number of available classification tracks is then more critical.

In addition to the presence and characteristics of the specified tracks, other peculiarities can differentiate yard layouts. In some yards, trains enter and leave from the same side (they are called “yards with one end”). Here, tracks can have several functions concurrently (receiving tracks, classification tracks, etc.). This case appears generally in yards which deal with a small number of trains since capacity is significantly reduced. Moreover, in some cases there are specific tracks, like transit tracks used to bypass the classification area, or tracks used to store the locomotives decoupled from inbound trains. If present, the latter typically start between the receiving tracks and the hump and end between the classification and the departure tracks. A storage area can then be accessible in order to store locomotives. In this area, locomotives can wait the outbound trains they will be attached to.

Operations

Operations differ somehow from yard to yard. A main difference is linked, first, to the consideration of one or several classification stages. A *stage* is a sequence of classification operations which concretize in a series of cars being moved once from a receiving to a classification track. If this movement does not bring the cars in a suitable configuration for building an outbound train, for example if they are stored in a mixing track as mentioned above, a second classification stage is necessary: the cars need to be brought back to the receiving tracks. To realize this operation, which is called a *pullback*, a shunting locomotive pulls the cars through the return track.

Depending on the number of stages which characterize operations, we can classify the types of classification of a yard into 3 categories:

- single-stage classification,
- multi-stage classification with mixing tracks,
- multi-stage classification with car ordering.

In single-stage classification yards, cars are moved only once from receiving to classification tracks. Once they reach the latter, they start composing the outbound train to which they are aimed. Figure 2 shows an example of operations in a single-stage classification yard. Here, a single inbound train is stored on a receiving track. There are two expected outbound trains which must be built from the 4 cars of the inbound train. Two pairs of cars share the same destination: (1,2) and (3,4). No order is required for the cars of each outbound train.

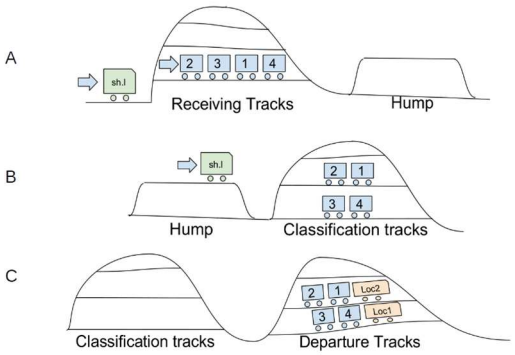


Figure 2. Example: Single stage classification

In Figure 2.A, the inbound train becomes a cut when its locomotive and cars have been uncoupled. The cut is pushed by the shunting locomotive over the hump and the cars go downhill to the selected classification of each car. Then, in Figure 2.B, (2,1) and (3,4) are coupled. Finally, in Figure 2.C, once the outbound trains have been pulled to be stored on the departure tracks and attached to their respective locomotive, they wait the right moment to leave the yard.

In multi-stage classification with mixing tracks, there are not enough classification tracks to start building all outbound trains as soon as their first car is received. Hence, cars are stored in one or more specific mixing tracks waiting for the suitable moment to start building the corresponding outbound trains. These cars need to be pulled back to the receiving tracks at least once.

Finally, in multi-stage classification with car ordering, cars can be immediately used for composing the outbound train they are aimed to, but they must be placed in a specific order. Hence, it may be necessary to pull them back to the receiving tracks once or several times.

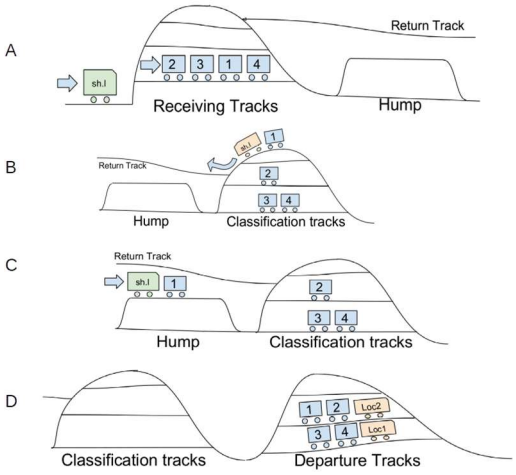


Figure 3. Example: Multi stage classification

In Figure 3, we propose an example of a multi-stage classification with car ordering. As in the single-stage example, the outbound trains to build are (2,1) and (4,3) but this time this specific car order is imposed. In Figure 3.A the cut is pushed to the hump, but differently from the single-stage case car 1 is not routed to the same classification track as 2 since they would not be in the correct order. In Figure 3.B, car 1 is pulled back through the return track to be re-rolled-in in Figure 3.C, to end up on the same track as car 2. Then, the cars

of both outbound trains are in the correct order: (2-1) and (4-3). Finally, the trains are pulled to the departure tracks, waiting to leave once their locomotive is attached as we can see in Figure 3.D.

Figure 4 shows the different actions of pushing, pulling and exploiting gravity force which are used in a yard with a multi-stage classification. We consider here the general case of a hump yard.

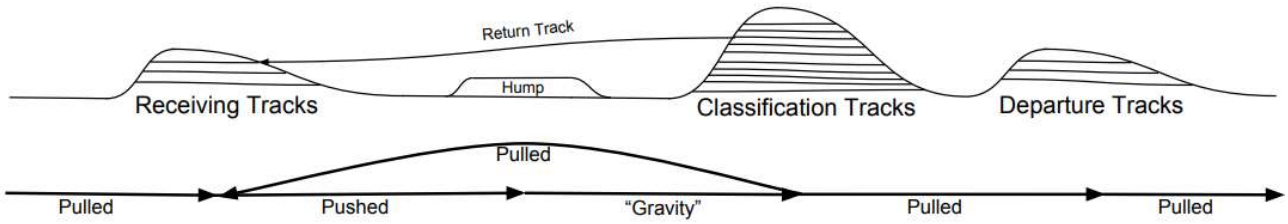


Figure 4. Schematic representation of the actions/forces moving cars in the different part of the layout

Disregarding the number of stages characterizing the operations, the main activities carried out in a yard can be grouped depending on their object: inbound train, cut and cars or outbound train.

- **Inbound train operations.** These operations start with the selection of the receiving track where the train is stored. Then, specific crews working on the yard inspect the train, uncouple the locomotive and the cars go to different destinations.
- **Cut and cars operations.** A cut is considered by the system once the cars and the train locomotive are detached. A shunting locomotive is then brought behind the cut (i.e., on the left side of the receiving track in the typical representation of a yard in Figure 1) to push it. The cut is pushed over the hump and the cars roll down to the classification tracks according to the route setting sequence defined. This operation is called *roll-in*. This sequence is typically automatic, i.e., the system recognizes the wagons and sets automatically the right route sequence to the selected classification track. In a multi-stage classification with mixing tracks, if some cars are not classified at the current stage, they are routed on the mixing tracks and pulled back to a receiving track at the end of the stage. Then, for the cars which have been pulled back, the activity starts again for a new stage. Remark that if the process is a multi-stage classification with car ordering, we consider the operations aiming at the sorting within the Outbound train operations.
- **Outbound train operations.** The system considers an outbound train being built once a classification track is chosen and the first car has reached it. During classification, the cars are accumulated on the classification track. They can be pulled back, once or more than once, from the classification to the receiving tracks if we are in a case of multi-stage classification with car ordering. Once the classification in done, cars are coupled. Then, either a shunting locomotive or the outbound train locomotive pulls the outbound train to the selected departure track. If a shunting locomotive is used, the outbound train locomotive will be attached to the train on the departure track. After standard checks, the outbound train waits until its departure time to leave the yard.

Different operations can be performed simultaneously on different cars and trains, or of course outbound train operations on a train can precede inbound train operations on another train.

Remark that we mention in this description only operations directly pertaining to trains and cars shunting. Indeed, yard crews may have to perform some other operations, such as cars registration and maintenance,

that we do not explicitly consider in this paper. They can be included in the test and check activities if necessary.

In the same way, we do not focus on the movement of the shunting locomotive. We made this choice to simplify the report, and it is not restrictive for our literature review since no existing approach deals with these specific operations and movements.

Problems tackled in the literature

The operations described in the previous section result in several decisions that operators have to make for guaranteeing the functioning of a yard. In turn, they can be formalized in optimization problems, which are schematized in Figure 5.

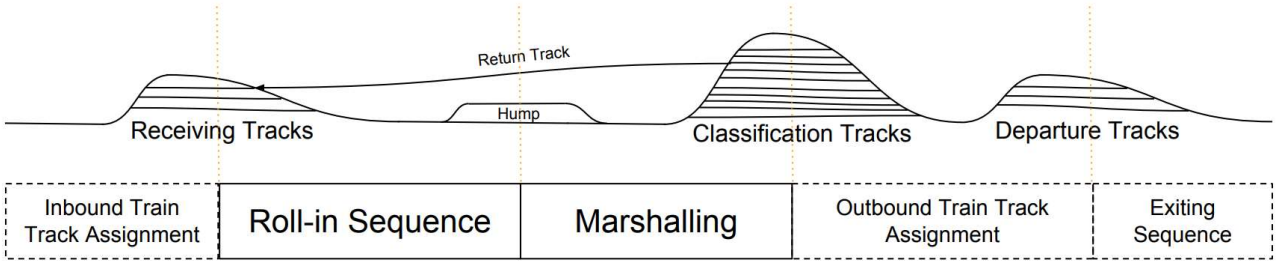


Figure 5. Optimization problems emerging from yard operations

The Inbound Train Track Assignment problem consists in deciding on which receiving tracks inbound trains are stored. This may be particularly relevant if receiving tracks are long enough to host more than one train. In parallel, in some yards, departure tracks are very long, and several trains can be placed on the same track: this gives rise to the Outbound Train Track Assignment problem. In both problems, the solution chosen may have an impact on the efficiency of the yard. In some cases, it is the network infrastructure manager (IM) who decides these assignments. Still in Figure 5, the Roll-in Sequence problem considers the order of trains/cuts pushed over the hump and the Marshalling Problem corresponds to the shunting process, solved through single-stage or multi-stage classification.

Of all these problems, the literature almost exclusively focuses on marshalling one. Some works consider that managing the arrival of trains in receiving tracks or the departures of new trains from the departure tracks is not relevant.

Whatever the problem considered, however, input and output definitions are quite constant. Specifically, most problems consider (at least) the following input:

- Estimated Time of Arrival (ETA) of inbound trains,
- Estimated Time of Departure (ETD) of outbound trains,
- Composition (i.e., sequence of cars) of inbound and outbound trains,
- Layout and rolling stock characteristics (e.g., number of tracks, tracks length, type of cars, cars length etc.),
- Duration of shunting operations for different types of cars.

The solution returned typically includes (at least), the following output:

- Schedule of the operations (time to leave the receiving track, time to enter the classification track, time to operate a pullback, etc.),
- Schedule of locomotives,
- Planned departure time of all outbound trains.

The current schedule of operations and locomotives can also be part of the input in case the optimization problem is not starting from scratch but is trying to improve a perturbed state of the yard.

For optimizing yard operations, from a given input (defining the problem instance), the models aim to obtain an output (feasible solution of the instance) which respects a set of constraints. The goal is to find the best output (optimal solution) following the direction of the optimization, which is given by an objective function to maximize or minimize. The most commonly used objective function, for both single-stage and multi-stage classifications, is the minimization of the number of classification tracks used. For the multi-stage case, minimizing the number of pullbacks is also frequent. Another typical objective function is the minimization of delay of outbound trains. According to the type of yards and to the models evolution in the literature over time, several other objective functions have been considered. For example, some papers focus on the minimization of locomotives energy consumption. There are also multi-criteria objective functions characterizing some problems, where the criteria are often contradictory (e.g., maximizing the efficiency of the system while reducing the use of resources).

Even if yard management problems can often be modeled as well-known optimization problems, their complexity was not deeply studied in the literature until recently. Specifically, [12], [13] and [14] propose an overview on some problems complexity, and, more recently, [9] propose an elaborate study of complexities for the general single-stage and multi-stage classifications. All these problems are sorted by complexity from polynomial to NP-Hard.

Marshalling problems

In this section, we report a review of the state of the art on Marshalling problems. We start with single-stage and continue with multi-stage classification, with mixing tracks and car ordering.

Methods for the Single-Stage classification

Following the definition of single-stage classification discussed in the section on yard operations, the main optimization problem tackled most often in the literature consists in selecting the sequence of trains and cuts to be sent over the hump, and to determine the classification track for each car.

An early contribution for this problem is the one by the authors in [15], who propose an algorithm called HSS (Hump Sequencing System) to build the sequences of cuts ready to be rolled-in. They consider a system where inbound trains can enter the yard with some delay, and their cars are to be coupled to a later outbound train. HSS optimizes the average yard throughput costs, represented by the cars' idle time. The main method behind HSS is based on dynamic programming which provides efficient solutions but requires quite a long computation time (a computer from early 80's required one calculation day for an instance with 20 trains). If the number of trains is too high, some trains to be rolled-in will be filtered (delayed) by a screening procedure according to a specific priority factor given by the authors.

The work of [16] proposes an event-based model for the same problem. The model runs on a rolling horizon basis: at the occurrence of each event the system is modified and optimized based on the new status.

A very similar problem, although applied to passenger trains, is the one treated by [12]. The authors consider two objectives: first, the minimization of the number of tracks used and, second, the delays of cars. The authors consider a case where the composition of outbound trains is modifiable. The model considers possible late inbound trains whose cars originally expected in an outbound train can be delayed and be part of the next one with the same destination. Delays are weighted according to the priority of cars. To find the best assignment of trains to classification tracks the problem is transformed in the so called chromatic number problem, which consists in finding the smallest number of colors needed to color the vertices of a graph such that no edge has two vertices of the same color. The authors consider here a permutation graph.

Recently, [17], [18] propose an exact algorithm and two heuristics for another variant of the problem. Here, multiple outbound trains have the same destination and the selection of the outbound train to which cars are to be assigned is a further decision to make. An optimal solution of the problem minimizes the weighted tardiness of all outbound trains. The authors introduce the Single Stage Sequencing Problem with Weighted Tardiness (SSSWT), where the decisions concern the roll-in sequence of inbound trains. The problem, which is proven to be NP-hard, is modeled mathematically as an integer program (IP). Two heuristics are also proposed, in addition to two branch-and-bound procedures tackling the IP. The best of these heuristics is a Tabu Search (for more details on the Tabu Search algorithm, we refer the interested reader to [19]).

Inspired by the shunting problem emerging in Chinese yards, [20] deals with what they call the Train Marshaling Problem. To optimize the cut and cars operations, the authors aim to minimize the number of tracks necessary to shape the outbound trains. All cars are sorted by defining a partition, in which each set includes the cars assigned to a track for a future outbound train. Using the properties of the Identity Permutation matrices, the authors establish the roll-in sequence.

Focusing on another modeling of single-stage classification, [21] studies the assignment of each car from an inbound train to an outbound train. The authors assume that the inbound trains are pre-allocated to receiving tracks. The problem does not detail the sequence of cars over the hump nor the working time and the routing of the shunting locomotive. The authors propose an exact and a heuristic method, both based on dynamic programming (DP) and both maximizing the priority values given according to the urgency and importance of the cars loads. For example, they handle empty cars with a very low priority. The time needed to optimally solve the problem grows exponentially with the size of the instances, even with the use of the sophisticated upper bound proposed in the paper. The heuristic is based on the same DP algorithm as the exact one, but it does not explore the whole tree structure. It returns feasible solutions in a short time for small and medium size instances.

[22] tackles the Hump Yard Block-to-Track Assignment (HYBA) problem. This problem was proposed as a challenge in Railway Applications Section (RAS) of the Institute for Operations Research and the Management Sciences in 2014. Given a planning horizon, the aim is to determine the schedule (time to be rolled-in) and the routing of cars. The objective function is the minimization of the outbound trains' delays. The authors split the problem into 3 sub-problems: the Hump Sequence Problem (HSP), the Block to Track Assignment Problem (BTAP) and the Pullout Allocation Problem (PAP), where a *pullout* operation consists of pulling a series of cars from the classification to the departure tracks using a specific locomotive. The HSP consists in deciding the sequence according to which the cars are to be pushed to the hump. The quality of a sequence depends on the departure day of the final car to be processed. The BTAP seeks the best assignment of blocks

to classification tracks, where a block represents the same car destinations. Finally in the PAP, the decision on which pullouts to perform from the classification tracks to the departure track is made. This problem concerns the outbound train operations. Two of these problems, the HSP and the PAP, are solved by IP while the BTAP is solved by a heuristic. The overall algorithm starts solving the IP of the HSP. This can be done once since the roll-in sequence remains fixed for the whole process. Then the authors use a greedy algorithm to solve the BTAP and obtain the schedule of the cars from the hump to the classification tracks. The IP of the PAP is solved once all the cuts went down the hump or the classification tracks are full.

Somehow similar to the PAP, [23] formulates as an IP the problem of optimizing the sequence to be followed when assembling outbound trains. The objective is to find a good balance between the minimization of the dwell time of each car and the minimization of the outbound trains' delays at departure. The time horizon of interest is split in intervals of several hours treated sequentially. In the paper these intervals are called stages, although they have nothing to do with pullbacks. We include this work in the review of single-stage classification methods for its connection to the aforementioned work by [22].

The algorithms discussed in this section focus on only one stage but they can be used as a sub-routine to compute a multi-stage classification schedule as underlined by [9].

Methods for the multi-Stage classification with mixing tracks

As for the single-stage classification, the typical objective of the problem is minimizing outbound train delays, with additional complexity added by the small number of tracks available for composing them.

[24], [25] propose an IP model for the cars classification problem. These works have been classified within the single-stage methods in some review papers in the literature. However, we think they more suitably fit in the multi-stage methods since they consider at least one pullback from the classification tracks to the hump. Indeed, the operations studied include three pullbacks per day (one every eight hours). The main inputs of the model are the arrival times of inbound trains, the expected departure times of outbound trains and a feasible sequencing plan for cars. As output, the solution of the IP returns a roll-in sequence. The model minimizes an exponential function based on the delays of outbound trains. As in [15], which is discussed above, a distinctive feature of this model is the possibility for a car to be delayed and inserted in a latter outbound train with the same original destination of the car. In [24], the author runs an IP solver for a limited amount of time. The process stops the Branch-and-Bound algorithm and returns the best-found solution (which could be an unproven optimal solution). This technique is called a truncated Branch-and-Bound.

In the variant of the problem tackled by [4] the classification tracks are reserved for building specific outbound trains, and where the schedules of inbound and outbound trains are known. The classification tracks have a limited and non-homogeneous length. A mixing track is used to store cars and to wait the reserved classification tracks to be available. All cars in the mixing track are pulled back at each stage even if, in real life, the operator may decide not to. The authors propose two IPs to tackle the problem with a single mixing track. The first one has an exponential number of variables and is solved with a Branch-and-Price method. The second one is a compact model, i.e., involving a polynomial number of variables and constraints in the problem size parameters. The latter obtains the best results.

Methods for the multi-stage classification with car ordering

When a specific car ordering must be achieved in outbound trains, several stages may be necessary. Simple sorting algorithms are typically used, such as sorting-by-block, sorting-by-train, triangular sorting or geometric sorting [26], [7], [9].

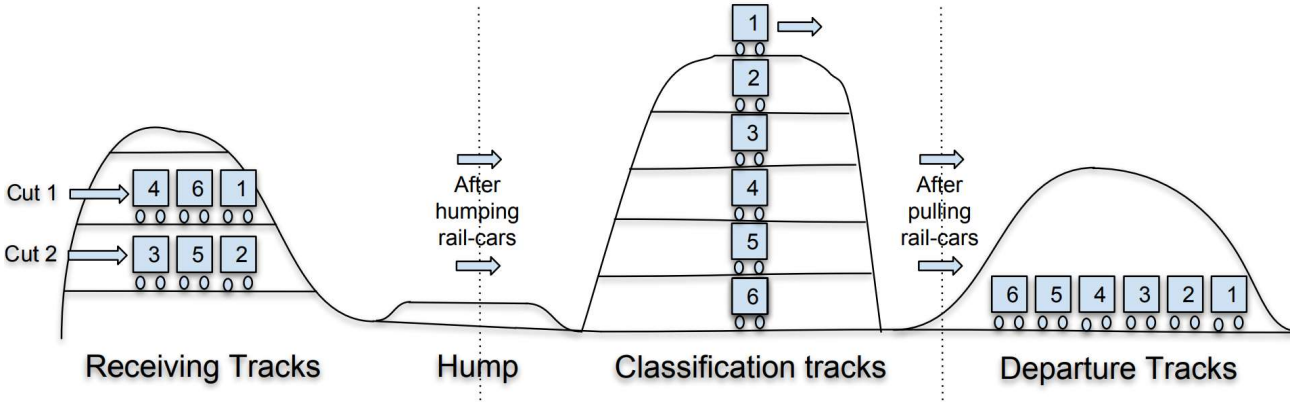


Figure 6. Three main steps of the sorting-by-block algorithm

Figure 6 represents the main steps of the sorting-by-block algorithm. The idea is to use one classification track per car of a given outbound train. Although the algorithm does not specify what to do with the cars aimed at other trains, they are usually stored on a mixing track. However, if the number of classification tracks is higher than or equal to the total number of cars, the classification can be single stage as in Figure 6. Once the cars are stored in the classification tracks, they are pulled one by one to the chosen departure track in the correct order. This algorithm can be efficient when outbound trains are formed by a low number of cars and yards have many classification tracks. Indeed, the orders of cars are quite easy to obtain with this algorithm, although the number of tracks used is potentially high.

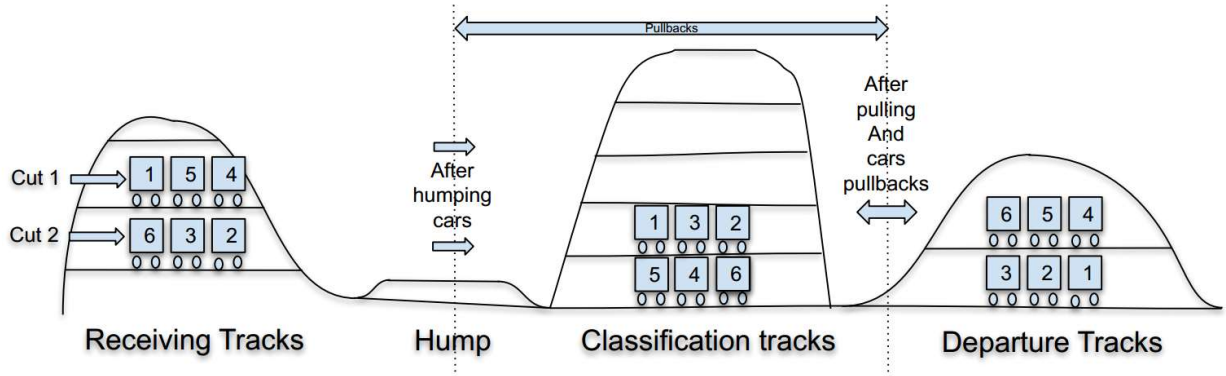


Figure 7. Three main steps of the sorting-by-train algorithm

The sorting-by-train algorithm is represented in Figure 7. The main idea is to dedicate each classification track to an outbound train. While the cuts are rolled-in, the cars are sent to the classification tracks according to the aimed outbound train. In the example, the operators have to build two trains, sorted as (4-5-6) and (1-2-3). Then, the sorting-by-train algorithm uses two classification tracks, one for each outbound train. After the first stage of roll-in, they obtain trains (6-4-5) and (2-3-1). In another stage, car (1) is pulled back to be sent to another classification track. The series of cars (2-3) is also pulled back to be finally sent to the same track as (1), ending up on its left. Similar operations are performed on the second train: when the two trains have the right order, they are sent to the departure tracks.

The triangular sorting algorithm indexes the cars according to a sophisticated calculation based on the length of the trains. The obtained indexes lead to an assignment to the classification tracks. This algorithm uses fewer tracks than the two previous ones but is very costly in terms of pullbacks. An illustrated example is given in [7]. The geometric sorting algorithm is an evolution of the triangular sorting algorithm and considers some sorting-by-train algorithm principles [27].

On more general grounds, the multi-stage classification with car ordering be a sorting problem with *n* stacks, *n* being the number of tracks corresponding in the analogy to *n*-1 “classification stacks” and 1 “hump stack”. [28] introduces this sorting problem. Since this work, the literature does not include papers proposing new algorithms for such a fundamental problem.

To present, analyze, and develop classification methods, [14] proposes a powerful encoding of classification schedules. This encoding can be used to analyze the efficiency of commonly used multistage methods, as shown in the paper for the simple methods introduced above. Through the encoding, the authors prove the optimality of a variant of the geometric sorting in terms of sorting steps, considering presorted input.

More recently, [29] proposes an IP model for this problem. The authors consider a rather high level of detail to describe complex yard operations. For instance, they take into account the different types of locomotives (roll-in and pullback). The mathematical model is based on a flow model and it is solved with the help of lot-sizing problem's valid inequalities.

Another IP model is presented by the authors of [30], who minimize the number of sorting steps first, then the number of cars rolled-in during the classification process. In their bi-objective algorithm, the authors solve a sequence of IP. Moreover, they consider the case of a yard with multiple humps. In this case, two classification activities, one per hump, can be run in parallel. In the modeling proposed, the activities deal with independent cars partitions: there is one shunting engine operating on each hump, and each available classification track is accessed from exactly one hump; furthermore, every outbound train is composed using only cars from exactly one partition. The assignment of trains to partitions is part of the optimization process, and it is hence included in the IP model. An additional contribution of [30] is the application of a microscopic yard simulator for assessing the performance of the algorithm. The simulator is named Villon and is developed by Simcon, partner of the OptiYard project [31]. Such an application allows the evaluation of the performance independently from simplifying modeling hypothesis which are necessarily present in optimization algorithms.

Considering a quite specific utilization of classification tracks, [32] proposes and IP model and a Tabu Search algorithm for the multi-stage problem with car ordering. Classification tracks are grouped into empty, clean and mixed tracks. On clean tracks, outbound trains are built. Mixed tracks are divided into temporary and dirty tracks: the former host cars ordered as in their aimed outbound trains; the latter are occupied by cars in scattered order.

Overall yard management problem

To the best of our knowledge, only one paper in the literature deals with a problem modeling more than one-yard operation. Specifically, [8] proposes a comprehensive IP model for track allocation and roll-in operations timing. The authors consider a specific yard layout with mixing track, and no departure tracks. For inbound trains, the model takes the arrival tracks capacity into account. For the classification track allocation, the model is a variant of the one by [4], including a few differences due to the layout of the yard. Classification tracks are allocated as groups of tracks (clusters) with the same length. Shunting operation start times are

variables of the model while arrival and departure times of inbound and outbound trains are given as input. The main part of the work focuses on scheduling pullbacks, differently from [4] who looks for the best sequence on each track without any time consideration.

Scheduling constraints consider that roll-ins and pullbacks must not overlap on the hump (in the considered yard layout, both roll-ins and pullbacks use the hump). Every possible combination of activities is considered and modeled into constraints. For example, constraints formulate how a pullback can fit between a roll-in and a departure, between a departure and a roll-in, or even between two departures. Many precedence constraints are also developed. For example, if the classification tracks are all busy, one of them must be freed before new cars can be rolled-in to create a new outbound train on it. In the end, all these constraints must be activated depending on the case.

The work of [8] optimizes a multi-stage yard model with two objective functions. The first one minimizes the work effort which is in fact the minimization of the number of pullbacks from the classification tracks to the hump. It actually minimizes the number of shunting operations like the number of couplings, decouplings, checks and shunting locomotive movements. The second objective is the minimization of the track costs, that is, of the weighted sum of the number of receiving tracks and the number of classification tracks used.

The resulting model has many big-M parameters which, together with the large number of constraints, make the resolution of the model very complex. The results presented show how difficult it is to find the optimal solution to relevant instances. However, the model is shown to achieve interesting performances in several instances.

Other yard management problems

The literature also proposes algorithms for other, less commonly considered, yard management problems. Among them, [33] proposes an IP model aiming at dynamic empty cars assignment to outbound trains. The hypothesis is that all cars, full and empty, are planned to leave a yard on a specific outbound train. While for full cars this is considered a constraint, the assignment of empty ones is taken as modifiable, provided that the planned number of empty cars of each type (box cars, flat cars, gondolas, etc.) departs on each outbound train. Indeed, in case of delayed arrival of inbound trains, the planned assignment may be infeasible. Specifically, the authors aim to minimize the empty cars time in yard. The algorithm is based on a sliding time window principle, in which only empty cars in not yet rolled-in cuts can be re-assigned.

[34] deals with the problem of container loading in yards in which train compositions remain unaltered. Here, wagons transport containers that must be moved from one train to another to reach their destination. Containers are typically first accumulated in specific sites before being moved to outbound trains. Yards where this type of activity takes place are named rapid rail-rail transshipment shunting yards. The problem considered in the paper consists in determining the initial loading site of containers and their reloading place on outbound trains to minimize their transfers within the yard and therefore the use of handling equipment. Consequences of this optimization is the reduction of train processing time in the yard and thus total time for correspondence. The authors propose heuristics and an IP model for different variants of the problem.

2.2 NETWORK INTEGRATION

Little literature exists on the integration of yard and network management.

Moving toward this integration, the problem of scheduling rail freight node operations through a slot allocation approach is studied in the PhD thesis of Schönemann [35]. In this thesis, the performance of

complex freight nodes with a focus on the transshipment from and to rail is examined. The goal is the increase of the productivity and thus the throughput capacity of goods, wagons and trains through complex freight nodes as an overall system. A robust hub scheduling approach, which synchronizes the work processes in freight yards with the preceding and succeeding links in the transport chain, is used for solving the existing problems found in practice. By replacing the timetable-free operations in the railway yard with a process-oriented coordination mechanism based on a time windows (slot management) strategy, the railway-specific requirements shall be integrated with customer's and logistics demands. At the beginning, the railway-specific production process from freight nodes is analyzed. This analysis examines the properties of various terminal types, the cargo handling techniques, and the process sequences that freight trains which pass through the hubs are involved in. The process analysis finds that as a result of the large number of entities involved in complex hubs and due to the lack of holistic coordination, a significant potential for increasing the efficiency of the nodes is expected thanks to the newly developed dispatching strategies.

A two-step simulation and optimization model is proposed for investigating the productivity of the cargo transshipment to and from rail. The performance of freight nodes is analyzed in a combined queueing and simulation environment at the first stage of the modelling method. The model allows to examine the efficiency of individual system components and to identify bottlenecks. It is shown how the interrelationships between the various processes can have a negative influence on the terminal performance when process coordination is insufficient. Especially, how the turnover rate of the hub declines when facing overloading is demonstrated. The computation of distribution functions from observed data is a central aspect of the model in order to determine stochastic process durations in the different areas of freight hubs. They form the backbone for the development of the slot management system in the thesis.

The second step is the optimization phase, where slot plan sequences for the processing of trains through the freight node are developed by means of a linear programming approach. Allocation of railway infrastructure in the formation tracks of the marshalling yard and of the loading tracks in the transshipment terminals are controlled by these slot plans. The slot system is tested for robustness under realistic conditions by gradually populating stochastic data from real-life observations. How slot plans get influenced by various optimization rules and external factors is examined through a stability analysis in several scenarios. Unpunctual incoming freight trains have been identified as having a significant impact on the slot stability. It is further discovered that the stability of the slot plans can be ensured even with heavy arrival time deviations with the development of appropriate rescheduling mechanisms. The hub production system is kept in a steady state by a powerful tool obtained by extending the slot management with the rescheduling approach, which provides the necessary transparency to other involved stakeholders, particularly to railway undertakings. It allows reducing the dwell times of trains with arrival delays and thus reduces the likelihood that these delays affect the train's subsequent trips.

Transshipment nodes were identified as the weakest link in the transport chain according to the current production system. It is shown that the position of transshipment nodes can be significantly strengthened by the proposed slot management scheme in conjunction with a powerful rescheduling approach. Not only can the processes of the hub itself be optimized by the presented slot-based hub production approach, but also can the processes that will benefit other stakeholders in the chain of transport, such as forwarders, shippers or railway undertakings.

Finally, Figure 8 from the thesis illustrates the relations between different fields in transport when the interactions between yards ('freight terminal planning and operations') and networks ('railway operations') are considered.

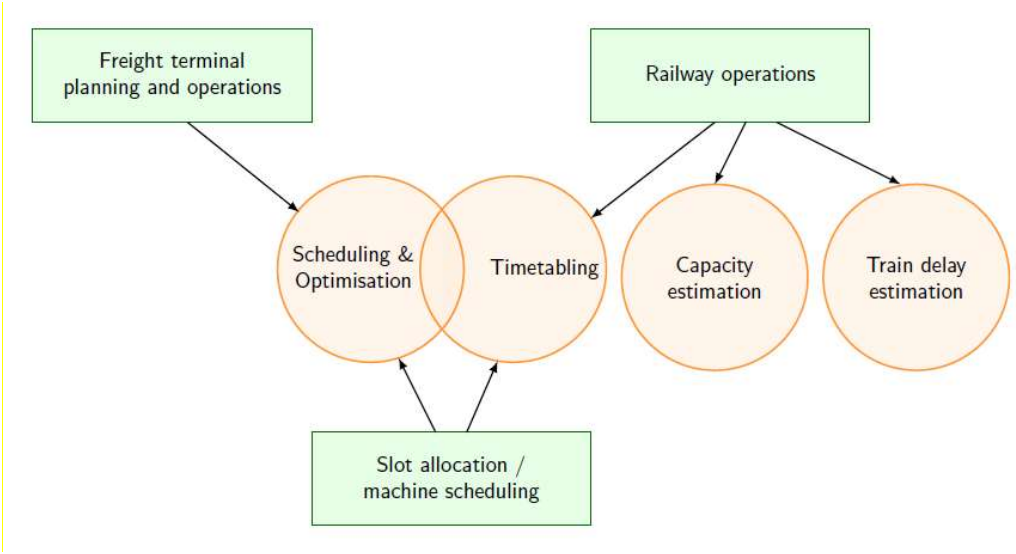


Figure 8. Classification of the research topic in transport areas [38]

Apart from the academic literature, two Shift2Rail IP5 projects deal with the yard and network integration for freight traffic. These are the ARCC and FR8HUB projects discussed in the following.

Automated Rail Cargo Consortium (ARCC)

ARCC - "Automated Rail Cargo Consortium: Rail freight automation research activities to boost levels of quality, efficiency and cost effectiveness in all areas of rail freight operations" is a Shift2Rail funded project focusing on Yard management but it also handles the network. The relevant part for yard-network integration is WP2, which started in October 2016 and ended in September 2018.

According to WP2, there is a need to make improvements on the interaction among the infrastructure manager (IM), yard manager (YM) and railway undertakings (RU) by digitalization and process enhancement. Therefore, the project seeks the development of a real-time yard management system interacting with a real-time network management system that will improve the quality of traffic, improve punctuality, system efficiency and the competiveness of freight rail transport.

There are several deliverables in WP2 of ARCC, among which the most relevant ones are surveyed as the follows:

1. **Interaction among yards, terminals and network management [37]**
   In real-world freight railway operations, it is common to see freight trains depart from marshalling yards in times that are not in accordance with the planned timetables, which causes various problems such as the re-planning of many aspects of the relevant transport in the operational setting. Therefore, the risk that original plans cannot be followed is obvious, bringing inefficient resource utilization on the infrastructure, rolling stock and staff. For instance, when a freight train approaches

the destination (a marshalling yard or a terminal), the yard may not be able to handle the arriving train because of its limited capacity. As a result, the arrival train must temporarily stay at some sidetrack along the line, which inevitably reduces the rail network's capacity and efficiency. In addition, the yard congestion problems will also appear, with the impossibility to process any additional arriving trains. This again will make the network less efficient.

One remedy for the above conundrum is to improve the preparedness of the trains that are very likely to run outside the timetable slots that are re-planned, which forms the cornerstone of the proposed development scenario in the project. This particular scenario for improving the coordination between lines and yards includes the following facets:

a. Once the RU knows that there is a need to operate a train outside the planned timetable slot, he/she will report this to the IM;
b. Before the departure of the trains outside the slots originally planned, a new and conflict-free timetable slot will be created which will ensure the train does not create and is not exposed to any unpredictable problems along its way to the destination;
c. The newly created timetable slot should be as good as possible, subject to the operational condition of the specific day and the allowed adjustments to the timetable slots of the other trains;
d. Ahead of the departure, the yard's capacity of receiving trains should be guaranteed, such that trains do not have unplanned waiting time along the line, due to the yard's limited arrival capacity.

The following changes are needed to ensure the above scenario:

a. Earlier and clearer communications between RU and IM on the predictable deviations in departure times – with respect to departures that are moved forward and backward in time;
b. Improved coordinated decisions with respect to the new departure time and its possible consequences, involving both RU and IM, so that it is more than simply an agreement between the driver of the train and the local train dispatcher;
c. The consequence of changing the departure time of a train on the line all the way from the origin to the destination should be analyzed and understood;
e. Departure time changed at the arrival yard should also be better analyzed and understood;
f. The risk of unplanned waiting along the line caused by limited arrival capacity at the arrival yard should be reduced;
g. Improved coordination and prioritization of the RU's trains;
h. Improved coordination of the dispatching process from the IM's side;

In conclusion, there is a great potential for improving coordination and making more informed decisions with respect to operational departure times for freight rail from yards/terminals. The scenarios for development include both automation (i.e., information processing and communication) and optimization/simulation (primarily about timetable circulation, yard/terminal capacity calculation and the estimation of the operation time). It is believed that the benefits from the improved scenarios will be shared by the freight operating companies, the infrastructure manager and the railway system in a holistic view.

2. **Consideration of yard operation requirements in timetable planning**

Specific movements and operations on yards are not planned in detail during timetable planning. Instead, rules of thumb and high-level estimations are the common practice, which often results in timetables with unsatisfactory robustness.

On the other hand, daily variations and shunting operations may also advance the departure time of some trains and thus increase the timetabled buffer time during rush hours. Moreover, an

experienced good dispatcher could handle a tight timetable better than a less experienced one. Potential solution methods are guidelines for high-level planning, improved support systems and improved cooperation. For instance, the patterns of arrival and departure trains will affect the marshalling work. The yard staff may concentrate on either the arrival or departure trains, if they are not mixed, which could result in more resource effective solutions. In addition, the marshalling shunting work required could be affected by the number of concurrent departure trains at the yard.

FR8HUB: Real time information applications and energy efficient solutions for rail freight

The FR8HUB project [37] is a Shift2Rail funded project which is focused on increasing the efficiency in the nodes, hubs and terminals in the railway system for freight and to continue the development in freight locomotives of the future. One of the objectives of its WP3 is "develop improved methods in connecting yards/terminals and network", such that a high-level model on freight capacity in the yard/terminal and network system will be created.

According to the descriptions of WP3 [37], methods for improved interaction between network and yard management will be developed, such as the specification of integration layers for real-time yard management and real-time network management applications, and the evaluation of how a technological upgrade of one hub will affect other hubs and nodes in the network. WP3 consists of seven tasks and the following two are relevant to yard-network integration:

Task 3.1: Scanning of innovations and best practice in interaction between network management and yard management (P10)

> Ensure a state-of-art and best practice in interaction between network management and yard management. In this sense, it will specify benefit and need for real-time yard management and real-time network management applications.

Task 3.3: Design of demonstrator functionality and high-level methods for network management (P11)

> The task will define demonstrator functionality and methods for interaction between network management and yard management.

Generally, more operational changes are requested by freight operators compared with passenger train operators. The major innovations in real-time management derived from FR8HUB are related to decision support for strategic, tactical and operational capacity planning to support these requests. These innovations can be summarized in the following aspects.

1. **Best practice of timetable planning and operational traffic control**
   Considerations on minor capacity constraints (e.g., minor construction sites) and adjustments to vehicle and crew management should be considered during short-term planning. Daily adjustments are needed for crew and vehicle management, as well as operational traffic regarding resource dispatching and disturbance handling. From a perspective of freight operations, changes should be also made with respect to demands, since demands will affect both the length and weight of the trains, and the possible allocation for additional train resources.
   FR8HUB will consider how to improve the planning activities in railway systems, especially, in microsimulation and optimization for re-planning timetables to increase overall freight speed and to manage disturbances during the tactical and operational phases.

2. **Specification of innovations in network management**
   The innovations in network integration are closely related to the innovations from the area of data-driven future operative and strategic decision support systems. For instance, as depicted in WP3 of FR8HUB, the core new data flow will be the automated reporting of car and train movements with technologies from the ongoing digitalization effort, e.g., the Intelligence Video Gate (IVG) as developed in WP4 of FR8HUB.
   As an initial plan, an information model will be developed where data from a set of IVGs comes in a time-stamped series of the IDs of trains and load carriers. IVGs located at the entry and exit points of rail yards will be considered. In particular, the following kinds of data will be considered as important input demonstrator:
   a. A train arrives/departs early
   b. A train arrives/departs late
   c. A train arrives/departs with dangerous goods
   d. Moreover, the following user input scenarios will also be included:
      i. Cancellation requests for trains
      ii. New slot requests for trains

   What will also be constructed are realistic scenarios where dynamic re-planning is performed as a sequence of cancellations and slot requests. For instance, time and space related re-planning within a slot can be considered as a cancellation followed by a new slot of request. Space related re-planning includes the use of alternative tracks at double lines as well as rerouting in other parts of the network. In order to make the approach more realistic, some sequences of requests will be more likely to happen than other series, since rolling stock and train crew may share some slots. Therefore, staff schedules and rolling stock circulation will have a significant impact on flexibility regarding re-planning.
3. **Tactical data driven timetable planning**
   Based on a two-step method by [38] for tactical timetabling, FR8HUB has proposed its approach that can be summarized in four steps:
   a. Create a core timetable model as in the hybrid micro-macro model developed by [35], and extend it with parametric hub/yard delay models, using an activity;
   b. Calibrate the parametric yard delay model based on historic train data;
   c. Simulate IVG indication of how increased hub processing time leads to an increase in predicted yard delay (with simulated tailing-off);
   d. Based on the newly developed approach, perform re-planning train timetables.
4. **Operative traffic control adjustment for a single train**
   The timetable will be regarded as a fix, and tiny modifications should be made under operational traffic control. Especially, adjustments on single trains can be considered. To this aim, the algorithm in [36], based on a given timetable, searches for a new train path between two yards or terminals, subject to certain side constraints on departure, travel and arrival time, and other logical and technical constraints derived from train traffic control. This very method by [39] can be extended to searching for train paths in alternative geographic areas, in contrast to most other proposed similar approaches. This is important when major disturbances and stretch closures appear. In addition, finding train paths through complex station areas is also crucial in this problem.

Other capabilities and goals are also included in WP3 of the FR8HUB project, such as robustness, travel time, constraints for departure, arrival and intermediate stations as a result of staff schedules.

2.3 CONCLUSION

The review of the state of the art reported in this chapter shows that some major gaps exist in the field of yard optimization. First, only specific sub-problems of the overall yard management problem have been studied. Second, yard resources are in general neglected or considered to be infinitely available. Third, only limited and very recent proposals exist on how yards may be integrated to the railway network. In the next chapters we will present the OptiYard approach aiming to fill these gaps.

3. YARD OPTIMIZATION ALGORITHM

In this chapter, we propose an algorithm tackling the overall yard management problem. As mentioned in the conclusion of Chapter 2, this algorithm is particularly original with respect to the state of the art for several aspects. First, it considers the whole service operated by trains: they are managed from their arrival at the yard up to their departure. Second, all resources are modeled in detail: not only tracks, as often done in the literature, but also shunting locomotives and crews. The algorithm takes into account their number, their capacity and their time availability. Third, trains and locomotives movements are modeled microscopically, to detect the occurrence of conflicts (situations in which an unplanned braking is necessary due to traffic) and take into account realistic running times. Finally, the algorithm is general in the sense that it can be applied to virtually any yard, independently from its layout and from the type of operations which are carried out in it.

3.1 MAIN PRINCIPLES

The complexity of the problem leads to the implementation of a heuristic algorithm. Specifically, we designed an iterative algorithm which repeatedly performs a randomized exploration of the feasible solution space. A randomized greedy heuristic makes all decisions, returning a solution describing the yard management plan considering all trains whose arrival is forecasted when the optimization starts. After the construction of this solution, the algorithm compares it to the best solution found so far (if it is not the first iteration) and if the objective function has a better value, the new solution is stored as the new best one. The objective function measures the total delay of departing trains and the total time in yard of wagons. The two objectives are considered in a weighted sum in which a second of delay weighs 100 times more than a second of time in yard.

3.2 DETAILED DESCRIPTION

The service to be carried out on each train is represented in the form of a flow chart. Flow charts are named technologies, following the naming used in the Villon simulator [1]. Hence, a technology is a series of activities, some of which demanding the assignment of a track, a shunting locomotive or a crew. For each of these resources, a specific group of possibilities can be associated to each activity.

During the randomized greedy heuristic, we perform successive insertions of resources in the current schedule of the yard, as soon as an activity requiring an assignment is encountered. For each assignment, the duration of all the activities carried out before the resource release is estimated. Thanks to this estimation, the assignment is performed as soon as possible, given the already planned utilization of the chosen resource. The resource is immediately reserved for the whole estimated duration, so that it is immediately possible to assess the feasibility of a following assignment for performing an activity of another train. For crew and locomotives, the selection greedy in the first iteration and in the ones between the sixth and the tenth: the resource that can be used to complete the concerned train's operations at the earliest is assigned. Otherwise, it is based on a uniform probability distribution. For tracks, the selection returns the track which can host the train the sooner. This latter type of selection represents a greedy component of the algorithm.

A cycle (or deadlock) avoidance procedure is implemented in resource assignment. For example, it is possible that a train requires a track and a crew one after the other, and it can only release the track after the crew has completed its task and has been released itself. Concurrently, another train requires the same types of resource in the same sequence. Indeed, we must avoid the assignment of the same track and the same crew

to the two trains in overlapping intervals, so that the first train is the first to use the track and the second train is the first to be assigned the crew. If such an assignment is not explicitly avoided, the first train will never receive the crew and hence never release the track. In turn, the second train will never have the access to the track and will never be able to release the crew. Similar reasoning leads to cycle avoidance when resources of different types are involved (tracks, locomotives and crews) and when the order of the assignments required by each train is different.

In the algorithm *RandomizedGreedyHeuristic* below, this structure appears in four blocks:

- The activities management of transit trains,
- The first activities of inbound trains management,
- The last activities of inbound trains management,
- The activities management of the outbound trains.

ALGORITHM RANDOMIZEDGREEDYHEURISTIC

```
while Not all transit trains completed do
        ttt <— selectTransitTrain();
        NextActivityManagement(ttt);
        if last activity performed then
          Set transit train completed;
        end if
end while
while Not all inbound trains completed do
        tit <— selectInboundTrain ();
        NextActivityManagement(tit);
        if first route traversed then
          Set inbound train completed;
        end if
end while
Set all inbound trains not completed;
while Not all inbound trains completed do
        tit <— selectInboundTrain();
        NextActivityManagement(tit);
        if last activity performed then
          Set inbound train completed;
        end if
end while
Select outbound trains depending on wagon marshalling;
while Not all outbound trains completed do
        tot <— selectOutboundTrain();
        NextActivityManagement(tot);
        if last activity performed then
          Set outbound train completed;
        end if
end while
```

The sets of transit, inbound and outbound trains are sorted. The first two sets are sorted according to the ETA of the transit and inbound trains while the last one is sorted according to the ETD of the outbound trains. Remark that outbound trains may include different groups of wagons which are originally built on different classification tracks and are then joined. This may be done with the objective of obtaining a specific order of wagons. In this case, a leading train is identified, and its technology includes the regrouping activities.

The methods *selectTransitTrain(), selectInboundTrain(),* and *selectOutboundTrain()* select the train to be dealt with at every step. In the first six iterations, the choice is greedy: the trains are considered in order of arrival at the yard, or of expected departure if they are outbound trains. In the subsequent iterations, the train is randomly drawn among the best choices, i.e., among the first elements of the sorted sets. Indeed, only non-complete trains are considered at each step. Combined with the resource selection mode, this gives an algorithm which is completely greedy in the first iteration, greedy as for train selection and randomized as for resource assignment for five iterations, randomized as for train selection and greedy for resource assignment for five more, and then completely randomized.

Transit trains typically only require track resources. The same holds for inbound trains until their first route is traversed. By completing transit trains and inbound trains until their first route execution, we ensure that these trains run without being disturbed by shunting locomotive movements. Indeed, in reality if a train and a locomotive must cross (if a conflict occurs) the train always has priority. This is what happens in our case, in which the locomotive which will have its routes affected later, will only be able to run in the interval not previously occupied by trains. If necessary, the locomotive will then stop to give priority.

After completing transit and inbound trains, a selection of active outbound trains is performed. Specifically, only outbound trains to which a sufficient number of wagons have been assigned are created. Indeed, for outbound trains, the minimum and the maximum length, weight and number of wagons is specified in input. If an outbound train is not created because one of these minimum thresholds has not been reached, then the wagons are re-assigned to a future outbound train with the same destination, which will have to be built on the same classification track.

When departure tracks are a bottleneck of the system due to their limited number, we impose a constraint stating that trains cannot move out to one of these tracks too much in advance with respect to their ETD. The definition of the acceptable advance is a parameter of the algorithm. In case some resource assignments overlap the occupation of the departure track, also their schedule is constrained. For example, if we allow trains to move to the departure tracks no more than three hours before their ETD, and if a train requires a locomotive before asking for the track and releases the locomotive after moving there, then we also impose the locomotive assignment to occur no more than three hours before the train's ETD. In this way, we keep resources blocked as short a time as possible.

In technologies, some decisions involve route selections for trains, their locomotives and the yard shunting locomotives. It is the role of algorithm *RouteSelection* given below. If the considered rolling stock has at least one route already planned for a first move, the first step of the algorithm will filter the next route selection in order to have the last track of the previous route matching with the first track of the next route.

The remaining routes are then evaluated according to specific criteria such as the number of conflicts which would emerge if the route was used and the length of the route. All the costs of these routes are computed and saved in a set of pairs where the first element is the index of the route and the second the cost (see pairsVector in the algorithm below). Once all routes are evaluated, the set is sorted according to the costs. A

route is then selected among the *l* best routes (if *l* is bigger than the number of routes, it can be reduced to that number).

ALGORITHM ROUTESELECTION

Input*: Set of Routes SR for the rolling stock rs, l an integer;*
lastTrack <— lastTrack(rs);
If *lastTrack* exists **then**
    For all routes *r* of *SR* **do**
        If lastTrack ≠ lastTrack(r) **do**
            SR <— SR \ {r};
        End If
    End For
End If
For *i* from 1 to |*SR*| **do**
    r <— SR(i);
    cost <— cost(r);
    pairsVector(i).first() = *r.index*();
    pairsVector(i).second() = *cost;*
End For
sort pairsVector on the basis of the second element from the lowest cost to the highest;
If *l* > |*RS*| **then**
    l <— |RS|;
End If
SelectedIndex is randomly chosen between 1 and *l;*
routeIndex <— pairsVector(*SelectedIndex*).first();
Return (*routeIndex);*

Remark that whatever the type of operations performed in the yard, they can be represented as complex technologies, which can hence be dealt with by the algorithm. If the yard operates single wagon loads, all sets of trains (transit, inbound and outbound) will be populated. Instead, if only block-train operations are carried out, outbound trains will not exist, and this simply translates in the elimination of the last loop in the *RandomizedGreedyHeuristic* algorithm.

3.3 SOFTWARE IMPLEMENTATION

The program have been developed in C++. Thus, we were able to follow the object-oriented paradigm. The choice of data structure such as the containers respect the principle of elementary algorithms [40]. The Unified Modeling Language (UML) [41] class Diagram in Figure 9 represents an overview of different classes of our program. Relations of composition (filled diamond shape with a single line), aggregation (hollow diamond shape with a single line) and inheritance (hollow simple arrow with a single line) between couples of classes are represented.

In this diagram, we see easily that a mobile resource can be of two types, either a crew or a yard locomotive, while a locomotive is either a train locomotive or a shunting locomotive. Tracks represent a third type of non-mobile resource handled in the same manner as crews and yard locomotives. The relations of Inheritance bring to the specified classes all the properties and behaviours from the general classes. There are other

specifications: i) a locomotive can be either a yard locomotive or a train locomotive; ii) a train is either a transit train, an inbound train or an outbound train.

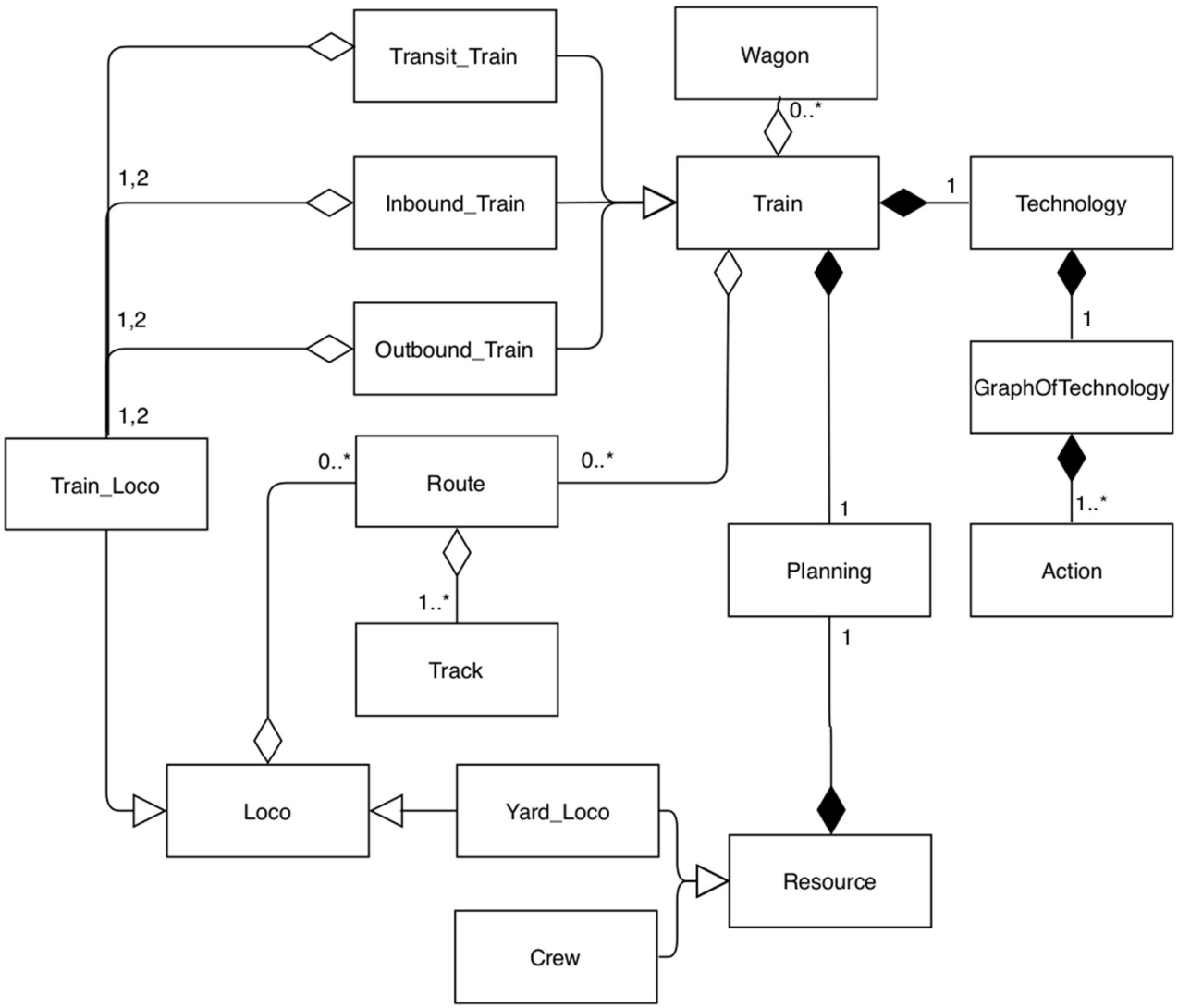


Figure 9. UML Class Diagram for mobile resources

After reading the static XML files, the optimization module creates a model for each type of technology, named GraphOfTechnology. During the creation of each train in the optimization module, we instantiate the right technology to compose the train from these models, possibly including grouping operations for trains. Each of them is composed of a GraphOfTechnology itself composed of several Actions. The class diagram in Figure 9 shows two other compositions: a train and a resource have a planning directly included in the composition of the two classes. The Route class represents here the selected routes in the train or locomotive planning. They are characterized by a series of tracks (here, this is not a composition but an aggregation since a track can be linked to several routes). Trains and locomotives can have several available routes. Finally, a train can have either zero or several wagons.

3.4 CONCLUSION

In this chapter, we described the main features of the optimization algorithm designed for managing a whole yard, considering simultaneously all the sub-problems typically challenged separately in the literature. In Chapter 4, we will describe how it is combined with the Villon simulator in the optimization and simulation framework. In Chapter 5, we discuss how this algorithm may be integrated in the management of railway traffic in the network.

4. YARD OPTIMIZATION AND SIMULATION FRAMEWORK

The aim of the yard optimization and simulation framework is reproducing in laboratory a realistic situation in which a yard is automatically managed. Hence, in the framework, the simulator plays the role of a real yard and constantly communicates with the optimization module in charge of making decisions. These decisions concern all resource assignments and all wagon treatments necessary to effectively carry on yard operations. In this chapter, we first describe how we set up the framework to achieve this reproduction of reality. Then we report the details of the communication and data sharing architecture that we developed.

4.1 COMMUNICATION AND DATA SHARING

A two-way communication is established between optimization module and simulator. Specifically, the optimization module receives information by the simulator through XML files. Two types of XML files are produced: those including static data and those including dynamic ones. Static data are sent to the optimization module only once at the beginning of the simulation. This data represents anything that will not change all along the simulation such as:

- Static data related to infrastructure,
- Static data related to outbound trains and their destinations,
- Static data related to resources.

On the contrary, new dynamic data are sent periodically all along the simulation. This data represents anything that may change such as:

- Dynamic data related to inbound trains arrivals,
- Dynamic data on the current state of wagons and trains,
- Dynamic data on the current state of resources (rolling stocks and crews).

Figure 10 schematically shows the evolution in time of the system. At time t0, the simulation starts and the optimization module receives an XML file including static data. Then, at time t1, t2 etc…, the simulator sends a request for optimization and, concurrently, an XML file including all dynamic data.

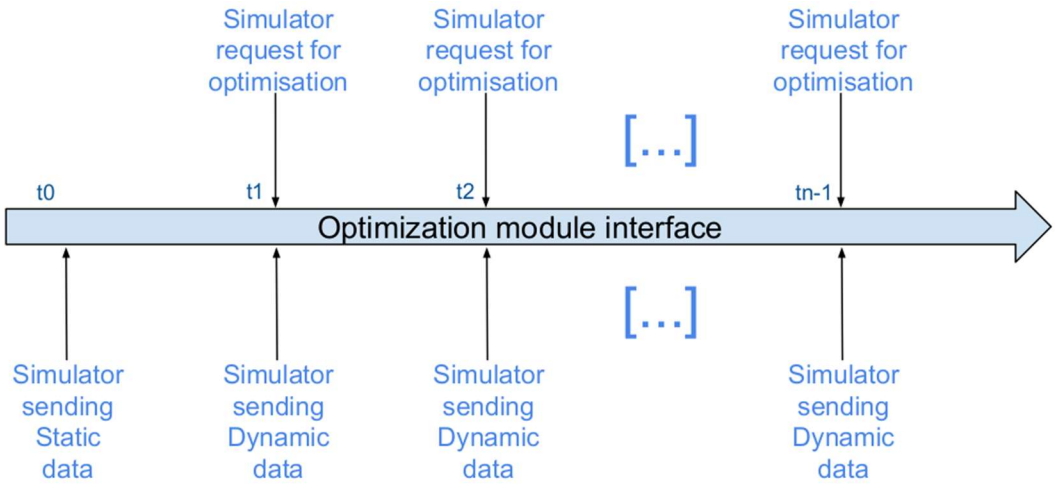


Figure 10. Schematic representation of optimization trigger and data sharing from the simulator

Furthermore, a continuous communication system is designed for decision sharing from the optimization module to the simulator.

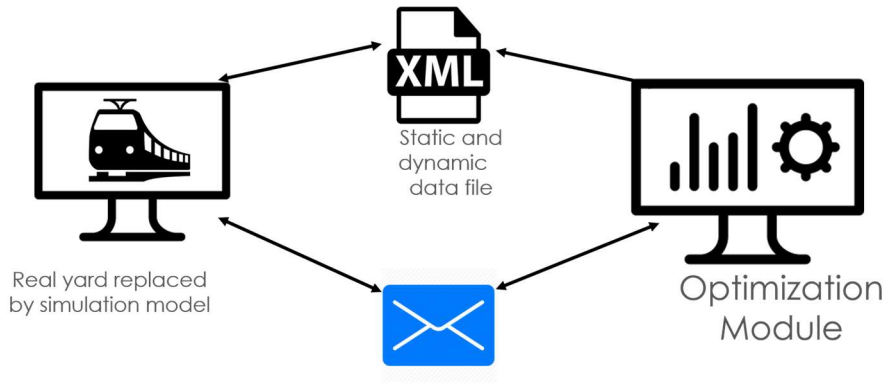


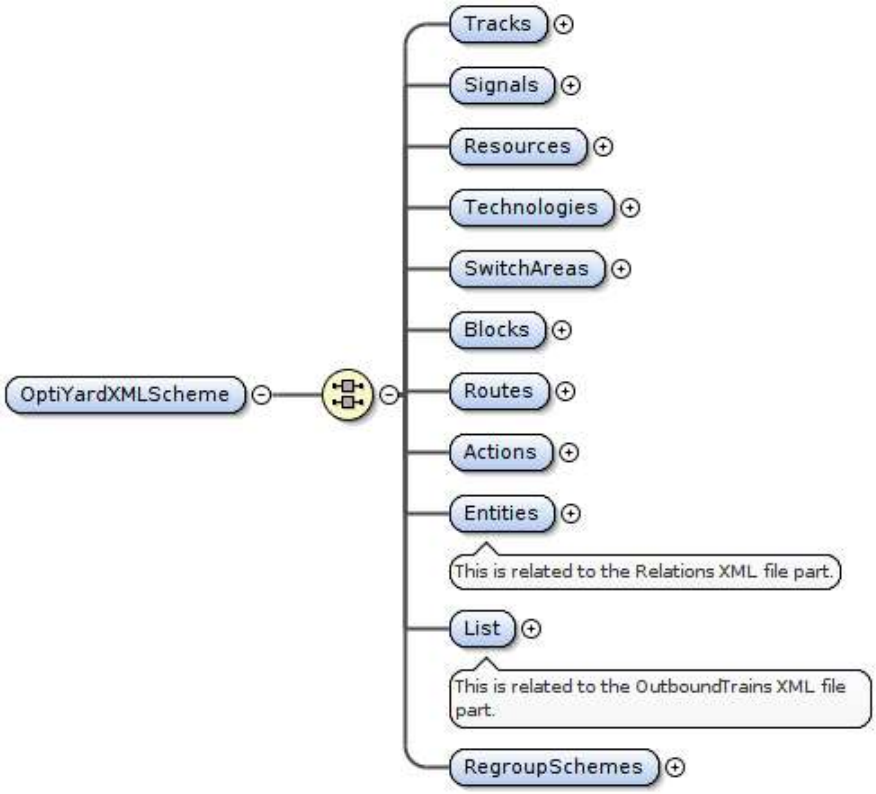
Figure 11. Representation of the communication architecture

Figure 11 depicts the structure of the communication architecture implemented, including XML static and dynamic data files and continuous messages.

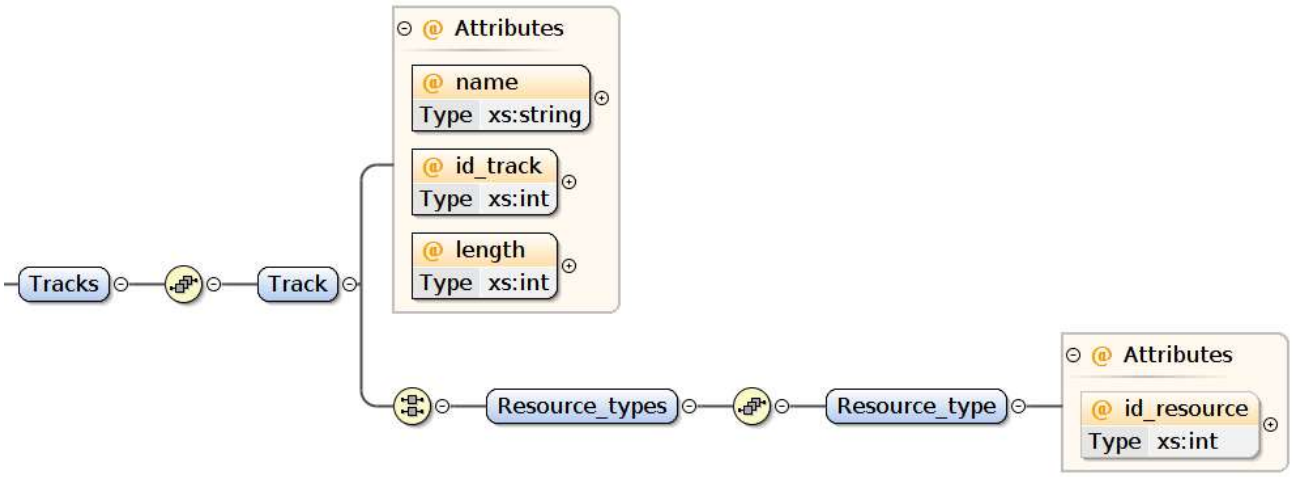
In the following, we describe static and dynamic data structures in the XML files, as well as the format of the messages for the continuous decision sharing.

4.1.1 Static Data

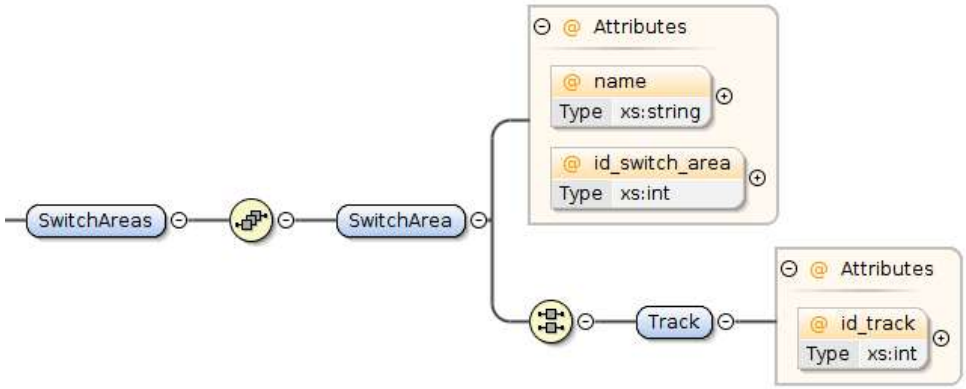
The static data have 11 parts. “Tracks”, “Signals”, “SwitchAreas” and “Blocks” XML data give information on the infrastructure of the yard. The crew and the yard locomotives are defined in “Resources”, and “Routes” is the list of the paths that the rolling stocks can take. “Technologies” express the graph of “Actions” linked to inbound and outbound trains. Indeed, for a given train, each step, such as the wagon uncoupling process, is described in an “Action”. Finally, “Entities” and “List” give us information on the planned outbound trains and their destination.



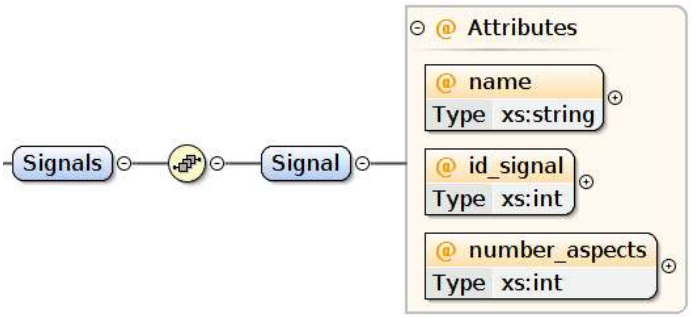
The “Tracks” have a “name”, an “id_track” and a “length”. They are considered as resources and get an id as such (“id_resource”).



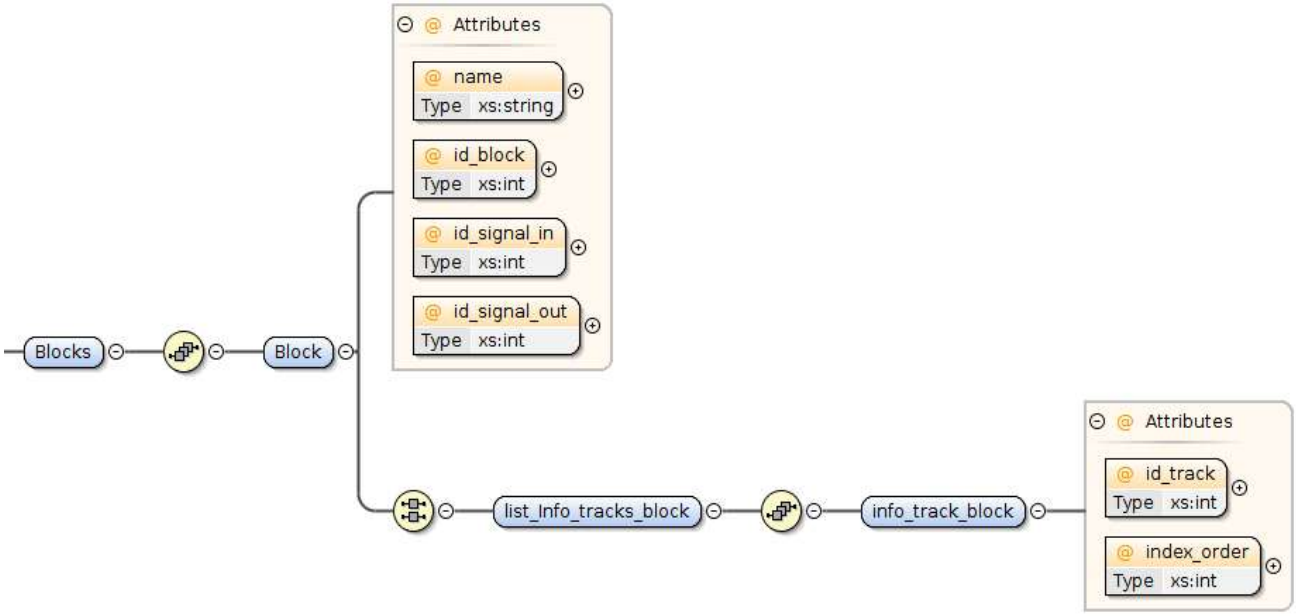
Dense locations of cross points are gathered in “SwitchAreas”. One “SwitchArea” is identifiable by a “name” and an “id_switch_area”. It has also a list of tracks recognizable with “id_track”.



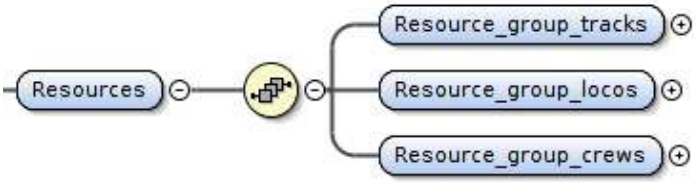
The “Signals” are only defined with a “name”, an “id_signal” and a number_aspects where it is effective in a yard.



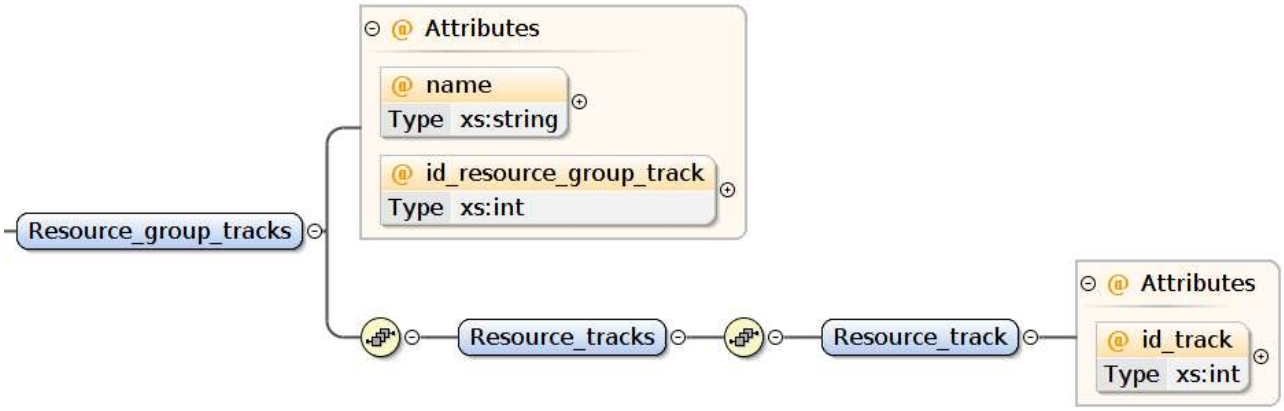
All the blocs are listed in “Blocks”, and each one is recognizable with a “name” and an “id_block”. Moreover, the first signal and the last signal are given (with their ids: resp., “id_signal_in” and “id_signal_out”). The series of tracks and their order are given through “id_track” and “index_order”.



The “Resources” are of three types: “Resource_group_tracks”, “Resource_group_locos” and “Resource_group_crews”.

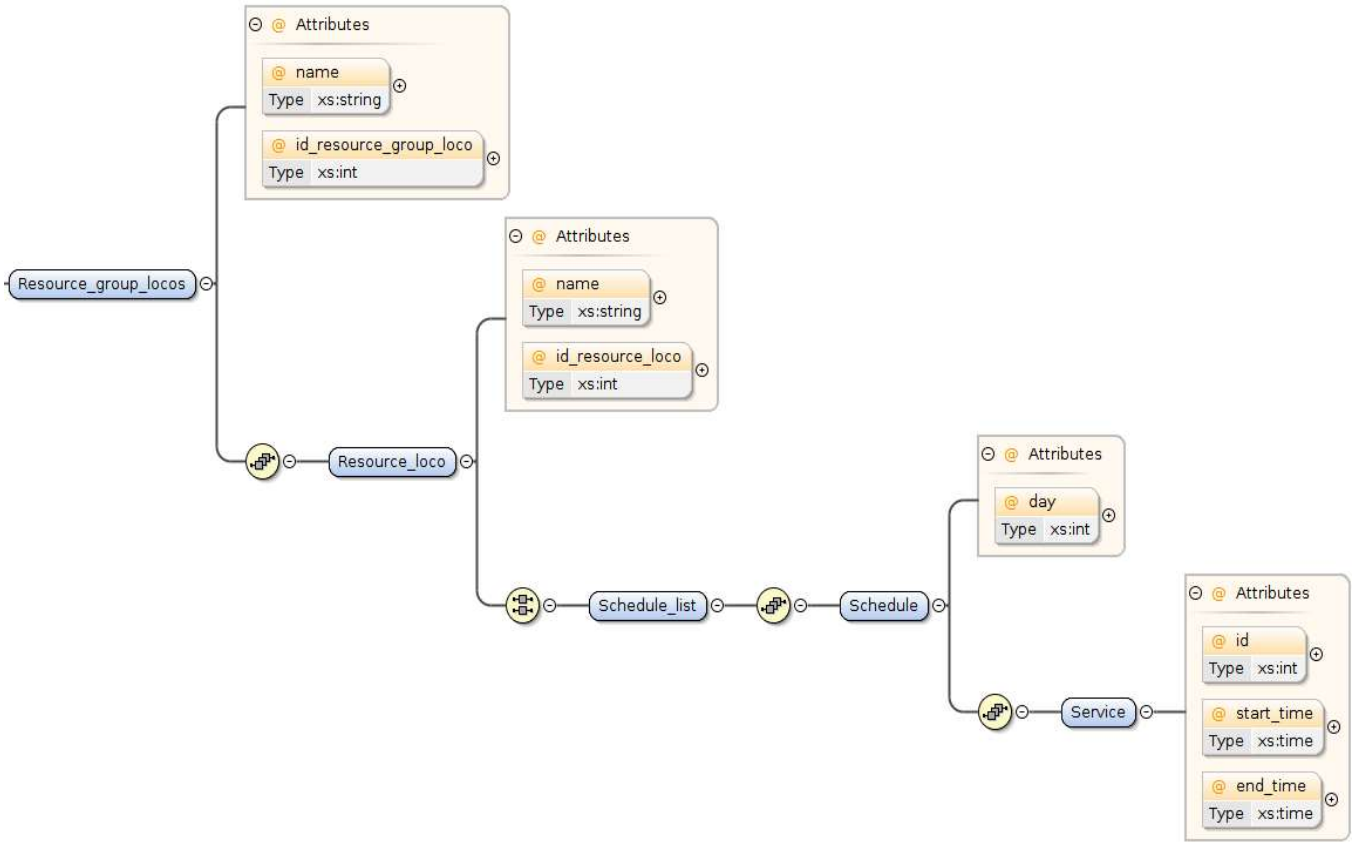


“Resource_group_tracks” groups the tracks by their type. It is identifiable with its “name” and its “id_resource_group_track” while the tracks are identifiable with their “id_track”.

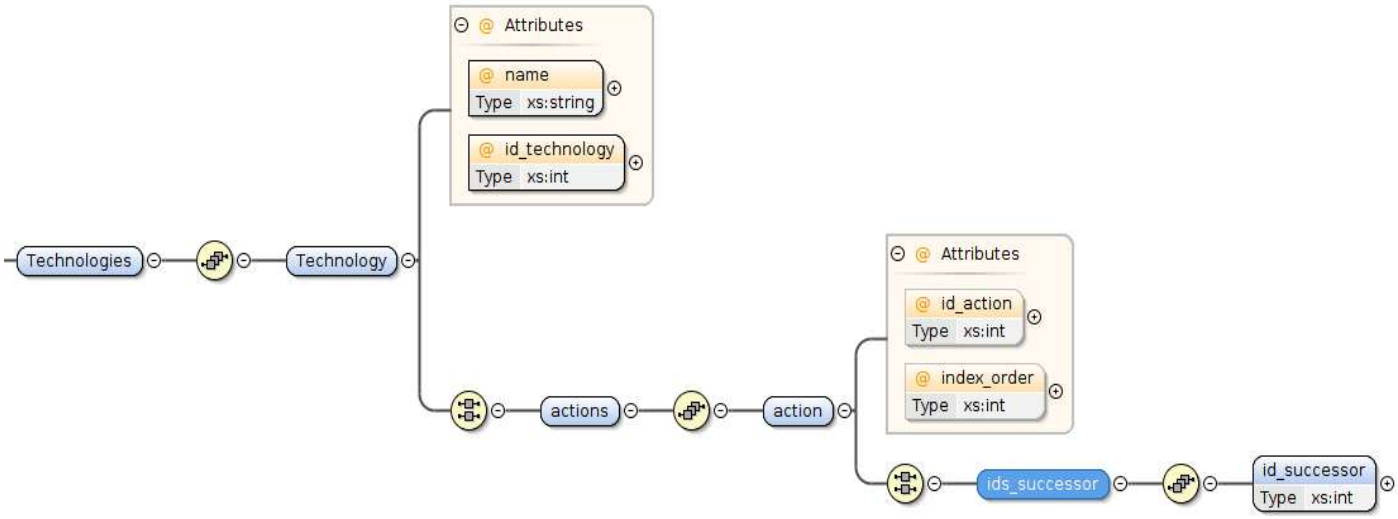


“Resource_group_locos” and “Resource_group_crews”, related respectively to the yard locomotives and the different crews, are similar: these resources are grouped by types and their schedule is given for each day of operations. We take “Resource_group_locos” to describe both.

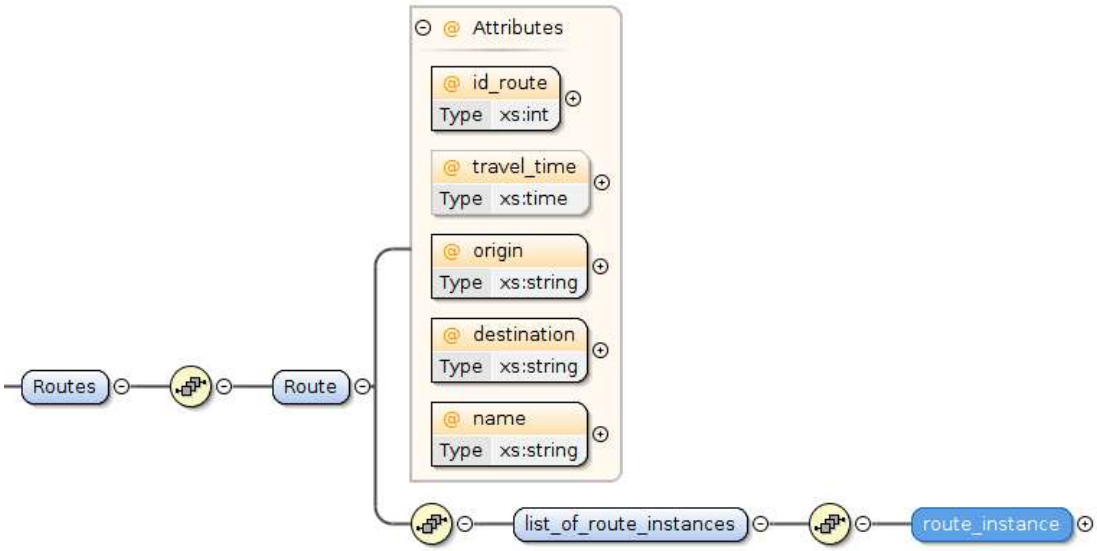
A type of resources related to the yard locomotives is defined with “name” and “id_resource_group_loco” while each resource of said type is defined with “name” and “id_resource_loco”. For each such resource, a schedule is given for each day (“day”) with a series of time windows [“start_time”, “end_time”]. Each time window is a couple of moments between which the resource is available.



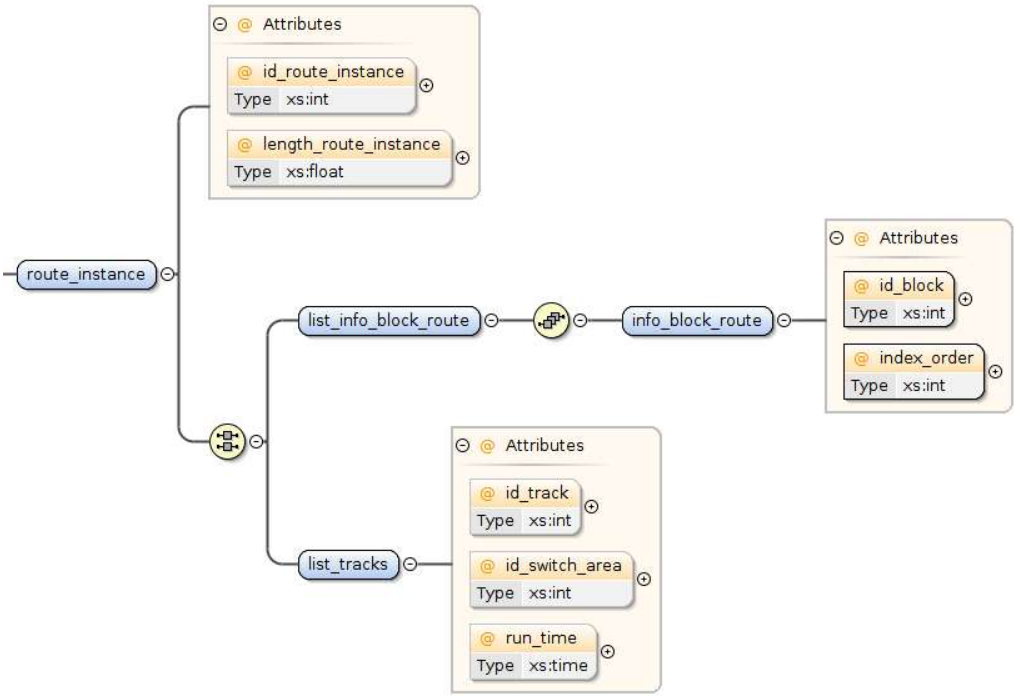
The sequence of operations to be performed on a train is named a technology, as explained in Section 3.2 [1]. A technology is identifiable with a “name” and an “id_technology”. It is a series of “Actions” identifiable with “id_action”. Some “Actions” operations can be executed in parallel, then each action can have one or several successors (“ids_successor”) in a “Technology”. However, since some “Actions” can appear several times in one technology, the “ids_successor” is not a sufficient information to rebuild completely the graph of “Actions”. The data provide a unique “index_order” to each “Action” of a “Technology”, in order to correctly build the series of succeeding “Actions” in the different parallel branches of the graph of the “Technology”.



Each rolling stock which must move is linked to a series of potential “Routes” identifiable with their “id_route” and “name”. Each “Route” has a “travel_time” required to go from the “origin” to the “destination”. All the possible paths from this origin to this destination is given in the list “list_of_route_instance”.



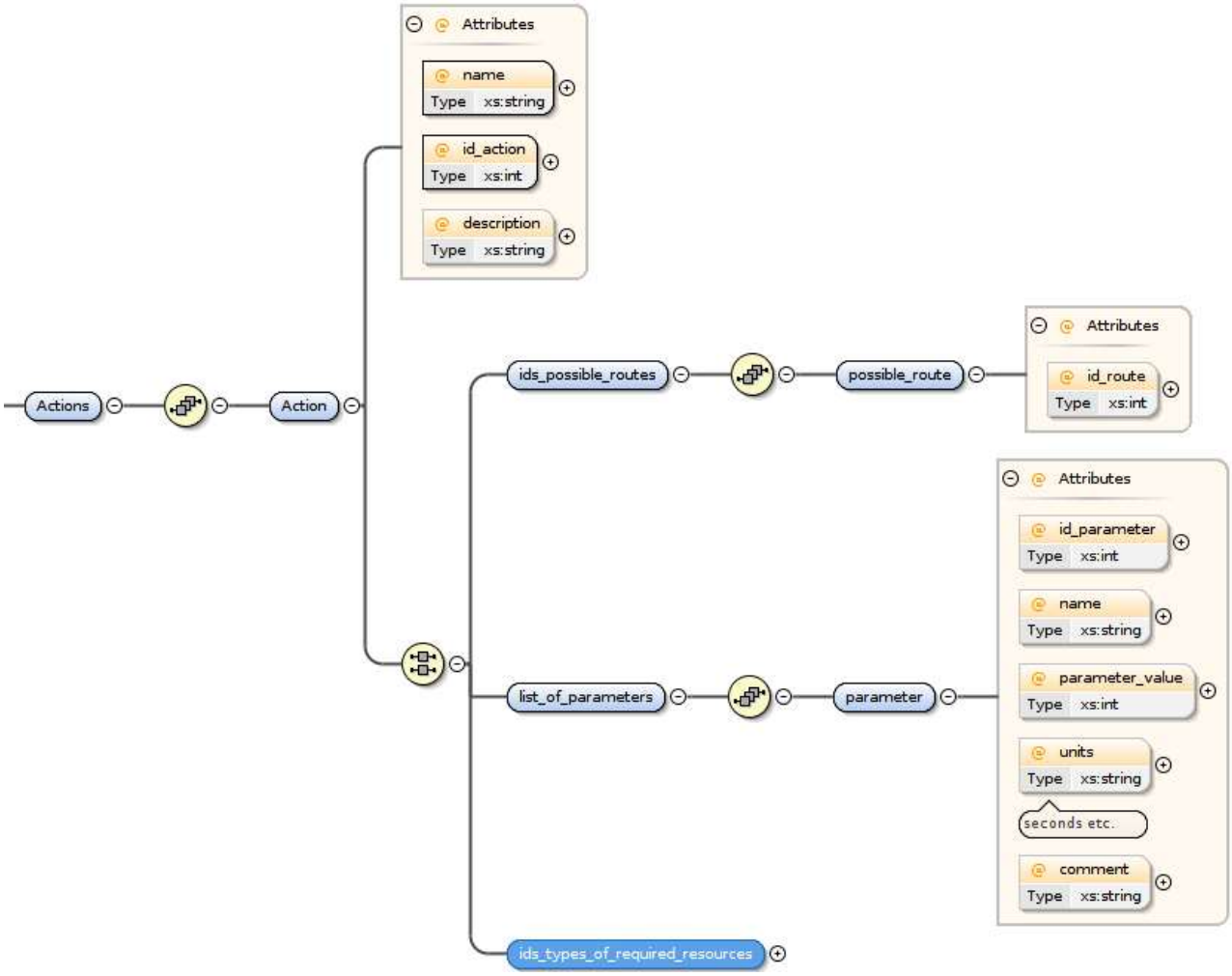
A “route_instance” is identified with “id_route_instance” and has a length called “length_route_instance”. Each “route_instance” has a series of tracks given with their “id_track”, their “id_switch_area” (if it exists) and a “run_time” corresponding to the running time to cross the full track. The blocks along a route are also given with a series of couples (“id_block”,”index_order”).



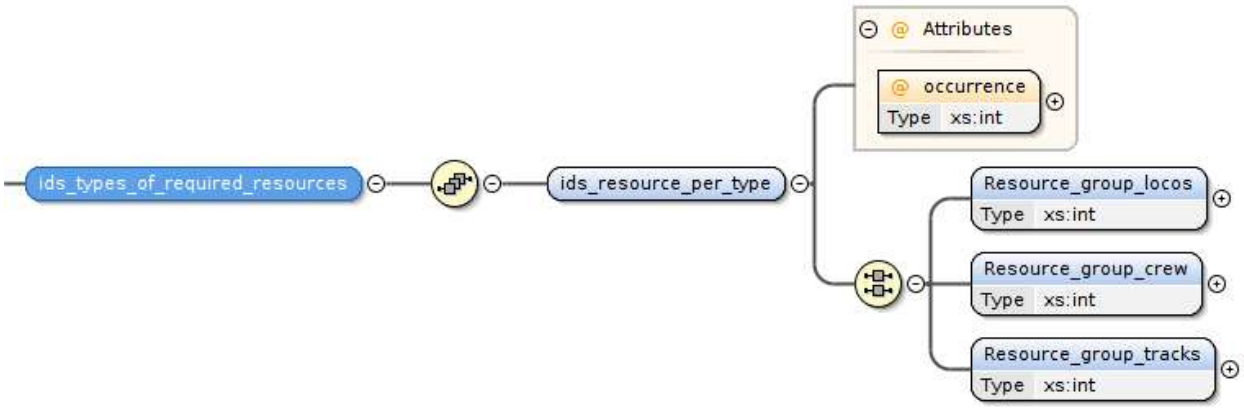
An “Action” is first composed of a “name”, an “id_action” and a “description”. Moreover, it has a “list_of_parameters” which gives all the information to compute the duration of the action:

- “name” and “Id_parameter”. One of these two data can be used to recognize which parameter to take in the duration calculation. For instance, “name=Length Parameter” indicates that the duration is calculated according to the length while “name=Wagon Count Parameter” makes the number of wagons a central focus;
- “Parameter_value”. This is the value to take into account in the duration computation. For instance, if “name=Wagon Count Parameter”, the total duration of the action will be the product of the number of wagons and “Parameter_value”;
- “units”. This is the unit of the important data of the action (meter, speed etc.);
- “comment”. The comment may give some further indication on the calculation.

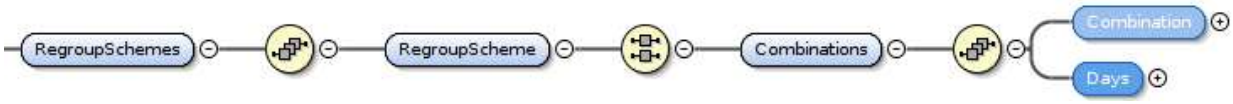
In case of a train or of a yard locomotive movement, the action proposes a series of possible routes ids in “ids_possible_routes”.



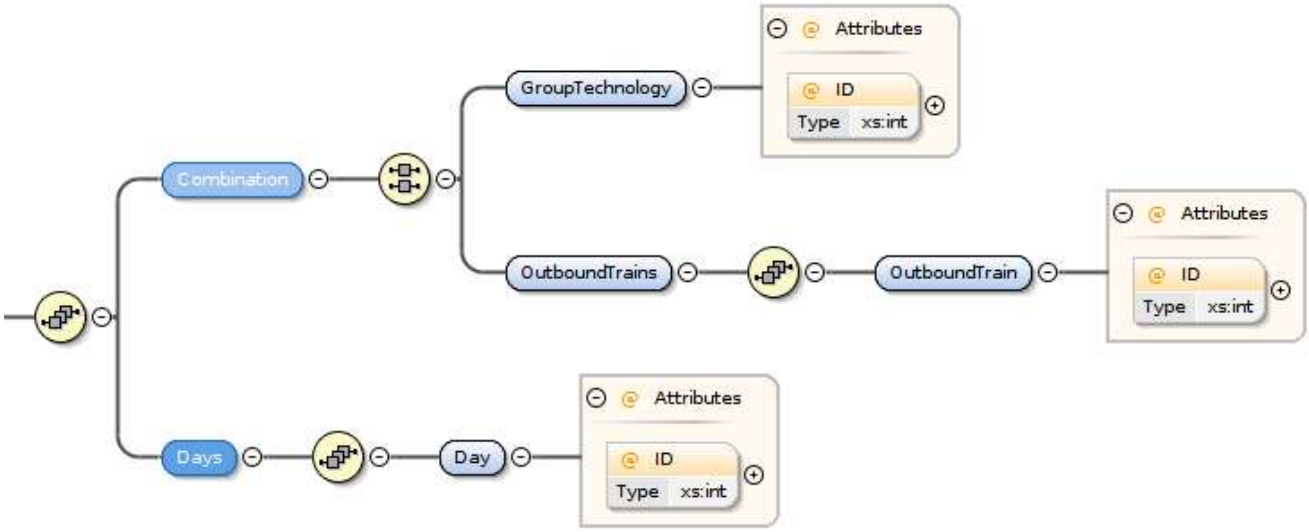
“ids_types_of_required_resources” gives the ids of the types of resource required for this “Action”. “occurrence” is the number of times the related resource is needed. “Resource_group_locos”, “Resource_group_crew” and “Resource_group_tracks” indicate which type of resource is required (resp. yard locomotive, crew and track).



Some outbound trains require to be regrouped before leaving the yard. In Villon, each combination of these "pre"-outbound trains starts their own technology before launching a common technology for regroupingall trains. To make this process understandable for the optimization module, a "super" technology is built for all possible "Combinations" of trains for any "Days".

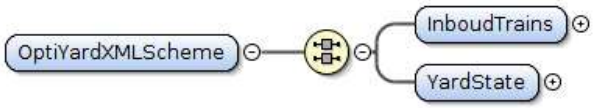


A "Combination" is made of a "GroupTechnology" recognizable with its "ID". The "pre"-outbound trains combination is given with "ID" in "OutboundTrain". Each combination can be executed only in certain "Days".

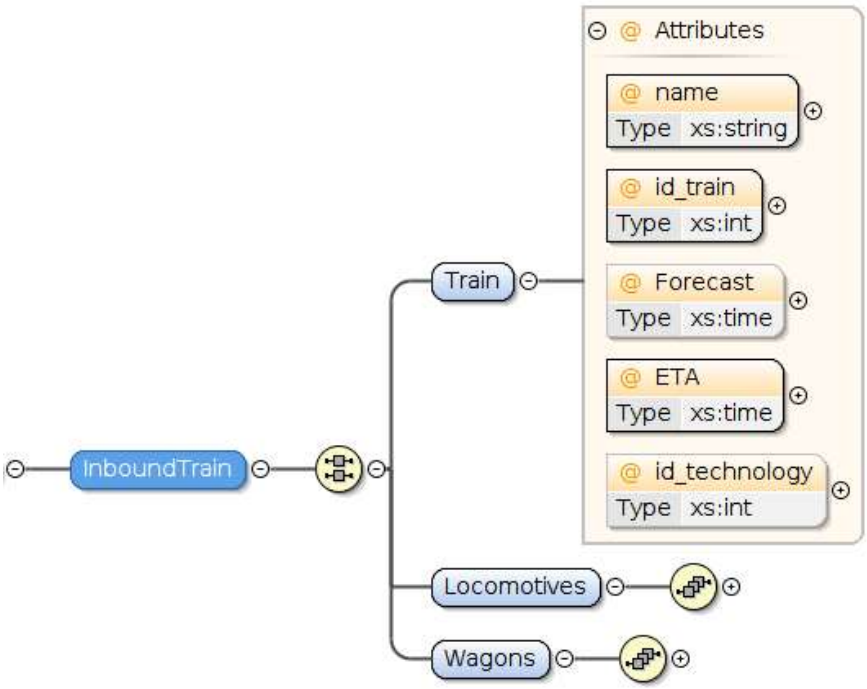


4.1.2 Dynamic Data

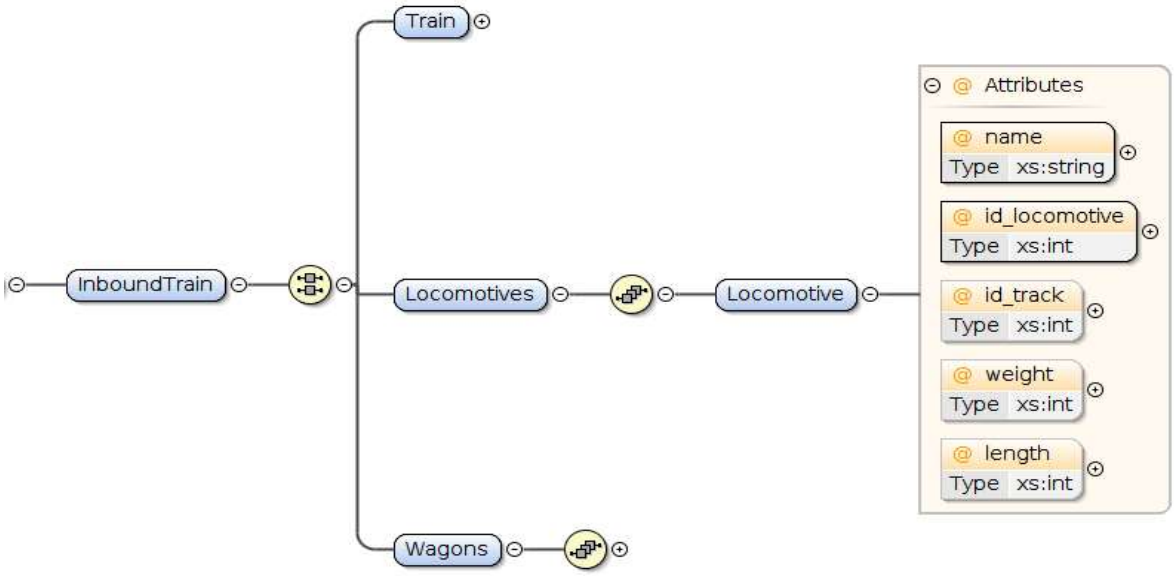
The dynamic data file has two parts: the data related to the inbound trains (which includes the transit trains) and the data related to the current state of the yard.



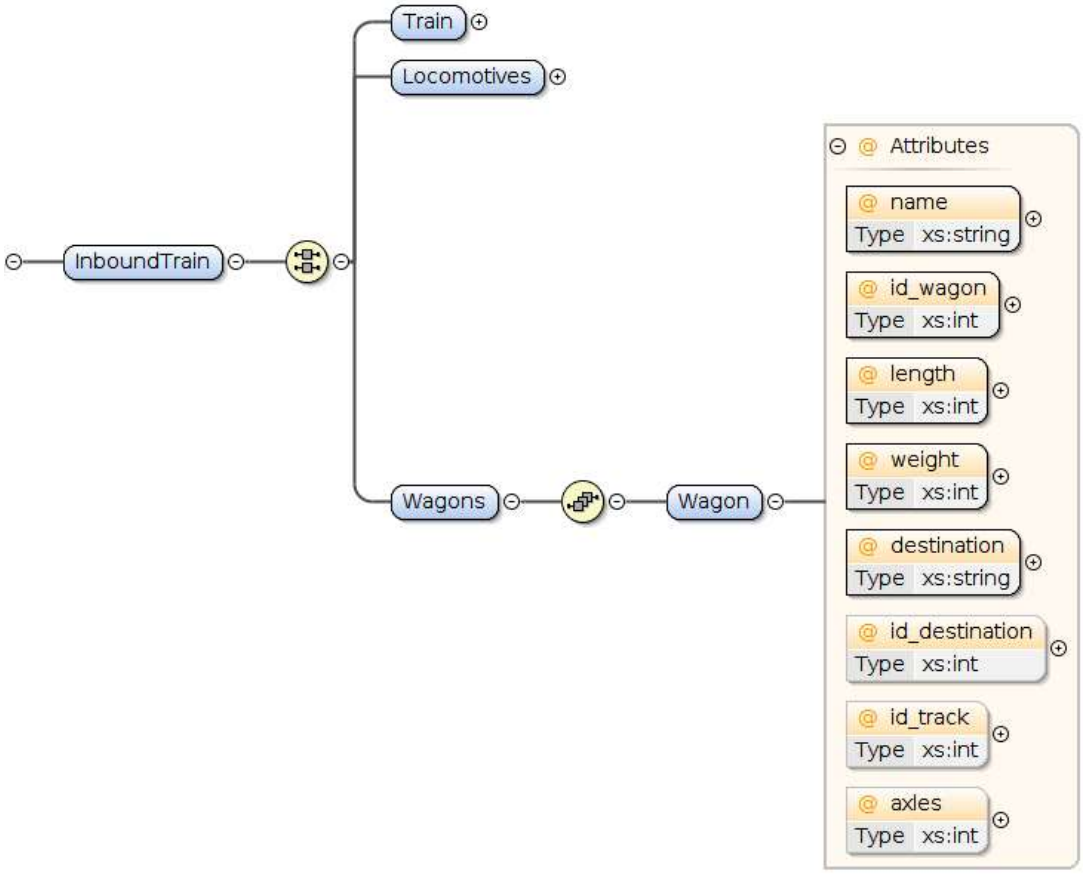
As for inbound trains, the system is warned at the time "Forecast" of their arrival. The "ETA", the "name" and the "id_train" identify the train while "id_technology" determines the technology linked to the train.



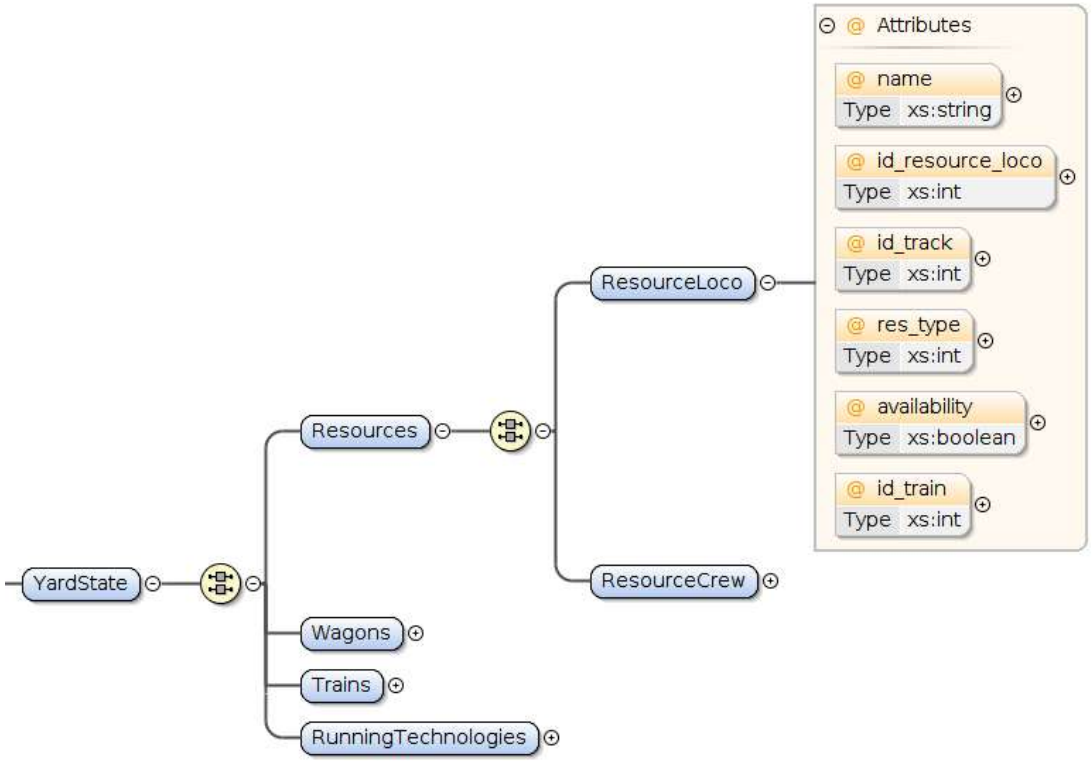
Each locomotive of each train has an “id_locomotive” and a “name”. Its characteristics reported in the XML file are the “length” (meters) and the “weight” (kilograms).



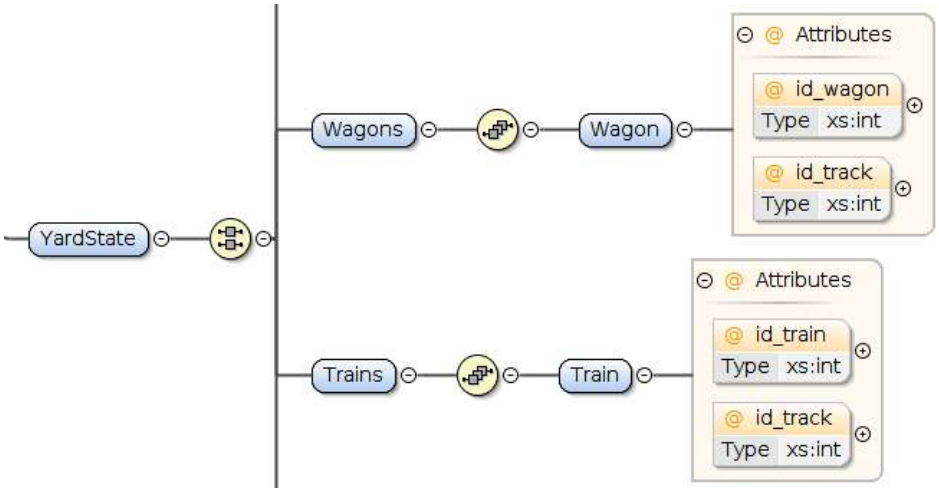
The “length” and the “weight” are also given through the data describing wagons. These data are supplemented with the number of “axles”. In addition to their names and ids (resp. “name” and “id_wagon”), the destinations (“id_destination” and “destination”) are given to select the right outbound train. Moreover, an “id_track” represents the arrival track of the train.



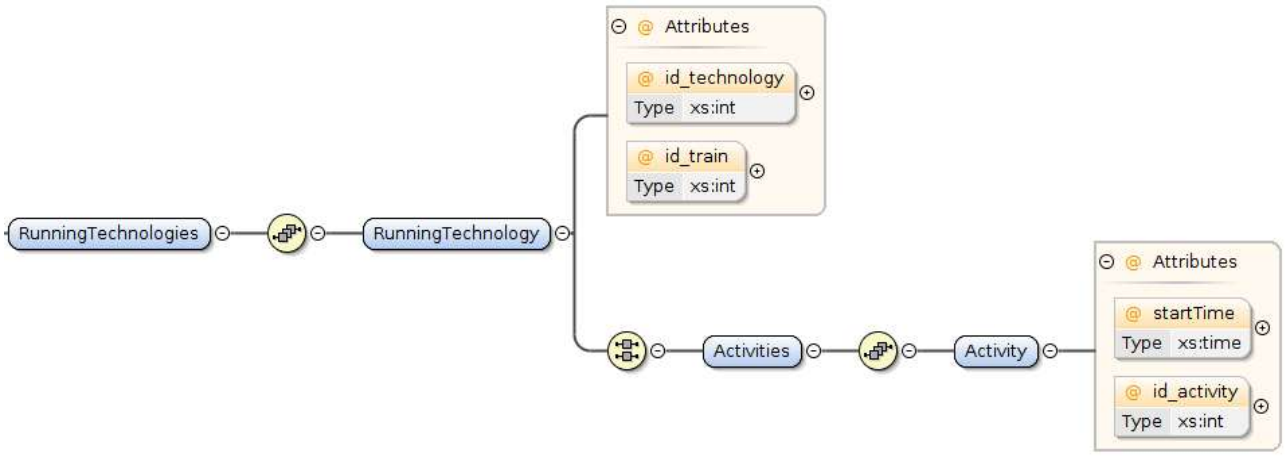
As for the current state of the yard, the XML dynamic files give information on the current state of the resources, the wagons/cuts, the trains and the running activities/actions in technologies. The “Resources” are of two types: the rolling stocks (“ResourceLoco”) such as the shunting locomotive and the crews (“ResourceCrew”) such as shunters or examiners. The attributes of “ResourceLoco” and the attributes of “ResourceCrew” are essentially the same: “name” and “id_resource” used for identification, “res_type” defining the type of resource, “id_track” giving its location, “availaibility” indicating if the resource is currently able to perform a new task and “id_train”, which is different from -1 only if the resource is used on a specific train.



The state of a “Wagon” or a “Train” is simply given by a rolling stock id (resp. “id_wagon” and “id_train”) and an “id_track” which is the track where the wagon or the train is located.



The last element defining the state of the system in the dynamic XML data concerns technologies. This part provides the series of technologies which have been started and not finished (“RunningTechnologies”). These technologies are identified with the couple (“id_technology”, “id_train”). Finally, the current running activities/actions are given and identified with a starting time (“startTime”) and an id (“id_activity”).



4.1.3 Continuous communication

The continuous communication, thanks to which the optimization module shares with the simulator its decisions, is implemented through message exchange. It is based on a client-server architecture and it uses socket communication. There are two clients, one querying and one answering, both communicating only with the server. The communication server provides connectivity, security and data validation. All forwarded messages are in standard JSON format. Thanks to this system, as soon as a decision is needed on an imminent operation to be simulated, it is requested to the optimization module that provides it immediately. For example, if it is the moment to perform a push operation on a cut, then the simulator will request which locomotive to use, and, supposing more than one is available, the optimization module will indicate the most appropriate one given the current state in the yard and the overall plan.

In addition to messages for testing communication and detecting errors, messages with a common structure have been defined for sharing decision. For sake of brevity, we report only a few examples as the other message types have a very similar structure:

- Track request from simulation with id 302:

```
{
    "id_message_type": 302,
    "id_message": 10100,
    "sim_time": "2019-04-11T07:00:00",
    "id_technology": 86,
    "id_act": 101023069,
    "id_train": 9,
    "id_loco": -1
}
```

  Here, at 7am of April 11th 2019 (“sim_time”), message (“id_message”) 10100 is sent for asking which locomotive (“id_loco”) shall be affected to train (“id_train”) 9, to perform activity (“id_act”) 101023069 in technology (“id_technology”) 86.

- Resource request from simulation with id 300:

```
{
   "id_message_type": 300,
    "id_message": 9871,
    "sim_time": "2019-04-11 09:00:00",
    "id_technology": 84,
    "id_act": 101023067,
    "id_train": 7,
    "id_resource": -1
}
```

Here, at 9am of April 11th 2019 (“sim_time”), message (“id_message”) 9871 is sent for asking which resource (“id_resource”) shall be allocated to activity (“id_act”) 101023067 in technology (“id_technology”) 84 for train (“id_train”) 7. The resource here is a crew.

- Track request from simulation with id 301:

```
{
    "id_message_type": 301,
    "id_message": 9633,
    "sim_time": "2018-04-11 08:00:00",
    "id_technology": 85,
    "id_act": 101023068,
    "id_train": 8,
    "id_track": -1
}
```

Here, at 8am of April 11th 2019 (“sim_time”), message (“id_message”) 9633 is sent for asking to which track (“id_track”) train (“it_train”) 8 should be sent during activity (“id_act”) 101023068 in technology (“id_technology”) 85.

All these messages are received by the optimization module which sends back the same message replacing the “-1” concerned by the request by its decision, which will then be implemented by the simulator.

4.2 FRAMEWORK SET-UP

As mentioned in the introduction of this chapter, the framework is based on a closed-loop integration of simulator and optimization module. Specifically, in this closed loop, the simulator periodically sends information on the yard state to the optimization module. In turn, the optimization module continuously sends indications on how and when operations shall be performed. These indications are based on the solution found by the optimization algorithm described in Chapter 3 for the latest available yard state.

For the initialization of the procedure, before starting any operation, the simulator produces the XML file containing all static data on the yard infrastructure, resources, technologies that can be operated and planned outbound trains. In an actual deployment, this file should be generated only when a change in one of these elements occurs.

Concurrently, a first dynamic data XML file is produced reporting the state of the yard and the forecasted inbound trains. At this stage, the optimization starts its first execution and runs for *s* seconds searching for the best feasible set of decisions *D*.

After this first optimization, yard operations start. In particular, the simulator sends messages asking for specific assignments of crew, track, locomotive, etc. to trains. At each message, the optimization module answers immediately the assignment defined in the set of decisions it made (*D*). The simulator then carries on the planned operations with the resource assigned, respecting all execution times and constraints that characterize its realistic model.

For closing the loop, every *m* minutes the simulator produces a new XML file containing updated dynamic data, based on which the optimization module computes a new set of decisions. More precisely, to run the algorithm described in Chapter 3, the optimization module first computes a short-term yard state prediction. Specifically, it calculates what the new state of the yard will be in *p* seconds if the set of decisions *D* keeps being implemented, given the current state. This predicted state is then considered as the starting point of the optimization: no decisions can be changed with respect to *D* between the time at which the optimization starts and this time plus *p* seconds. As soon as the optimization ends, set *D* of decisions being implemented is replaced with the set of decisions just made. Remark that in the current implementation of the OptiYard framework, p is set equal to 0: the simulator pauses as soon as it sends an XML file, so there is no need to carry out a yard state prediction. This choice is due to the fact that, being Villon an event-based simulator, time does not flow linearly: if nothing happens for some seconds, the simulator simply skips this time lapse.

Hence, a synchronization is not really possible. In any case, the framework produced can be trivially extended to include a strictly positive short-term prediction horizon.

The algorithm running time *s*, the number of minutes passing between two creations of two dynamic data XML files *m* and the duration of the yard state prediction *p* are parameters of the framework. Their value must be such that the algorithm has enough time to find a good solution, and the optimization is executed often enough to be able to promptly react to unexpected events. Clearly, it must hold that $s<p$ and $s<m$. In other words, the algorithm must provide a set of decisions consistent with the previous one when it starts being implemented ($s<p$ implies that when the run is over we are still in the period in which no decisions could be changed). Moreover, the algorithm must have the time to return a set of decisions before being executed again on a new yard state ($s<m$).

This closed loop allows to manage the yard considering always optimized decisions, made on the latest available forecast and state of trains and resources.

We observe that this closed loop can be perfectly reproduced in a real yard. The only issue that may emerge concerns the yard functioning before the first optimization execution. If the yard is empty or if operations can be stopped while waiting to produce the first set of decisions *D*, then no issue arises. Otherwise, a default plan must be available to play the role of *D* while the optimization algorithm searches its first solution.

Furthermore, note that this framework allows completely automatic operations. However, it is definitely possible to include human intervention, to validate the optimization decisions or to take the lead in specific situations. How this intervention may be set up and what kind of interface may be provided are open questions. They are out of the scope of the OptiYard project, and they will deserve specific ergonomic studies.

4.3 CONCLUSION

In this chapter, we reported the description of the integrated optimization and simulation framework. To the best of our knowledge, this is the first attempt to design and develop a closed loop between two such modules in the context of yard management. As it emerges, some parameters need to be fine-tuned to allow the achievement of the best possible results. In Chapter **Erreur ! Source du renvoi introuvable.** we will see the relevance of the parameters of the framework through a sensitivity analysis. This will allow the definition of the configuration used in the experiments carried out in WP6.

5. STRATEGIES FOR NETWORK INTEGRATION

This chapter describes the results of conceptual developments towards the provision of improved real-time interaction between yards and network. The work described here has benefitted greatly from collaboration with the FR8HUB project, introduced in Section 182.2, through the joint FR8HUB-OptiYard working group. We conceptualize that a freight train moves through a network between two yards using the OptiYard real-time yard management DSS. Yards and network form part of a network managed by IT systems based on the real-time network management tools developed in FR8HUB and described in FR8HUB Deliverable 3.2 “Demonstration of FR8HUB Network management concept” [42]. One yard is considered as the sending yard and the other is the receiving yard.

5.1 VISUALIZATION OF THE OPTIYARD DECISION SUPPORT SYSTEM

The visual representation of the structure of the overall OptiYard Decision Support System (DSS) is shown in Figure 12, together with a high-level representation of its interactions with the IT systems for the relevant parts of the surrounding railway infrastructure. There are three interacting models: two microscopic models (one for the yards and one for the surrounding network) and one model implementing the optimization algorithm. They are developed based on data obtained from the yard managers and from public sources of network information.

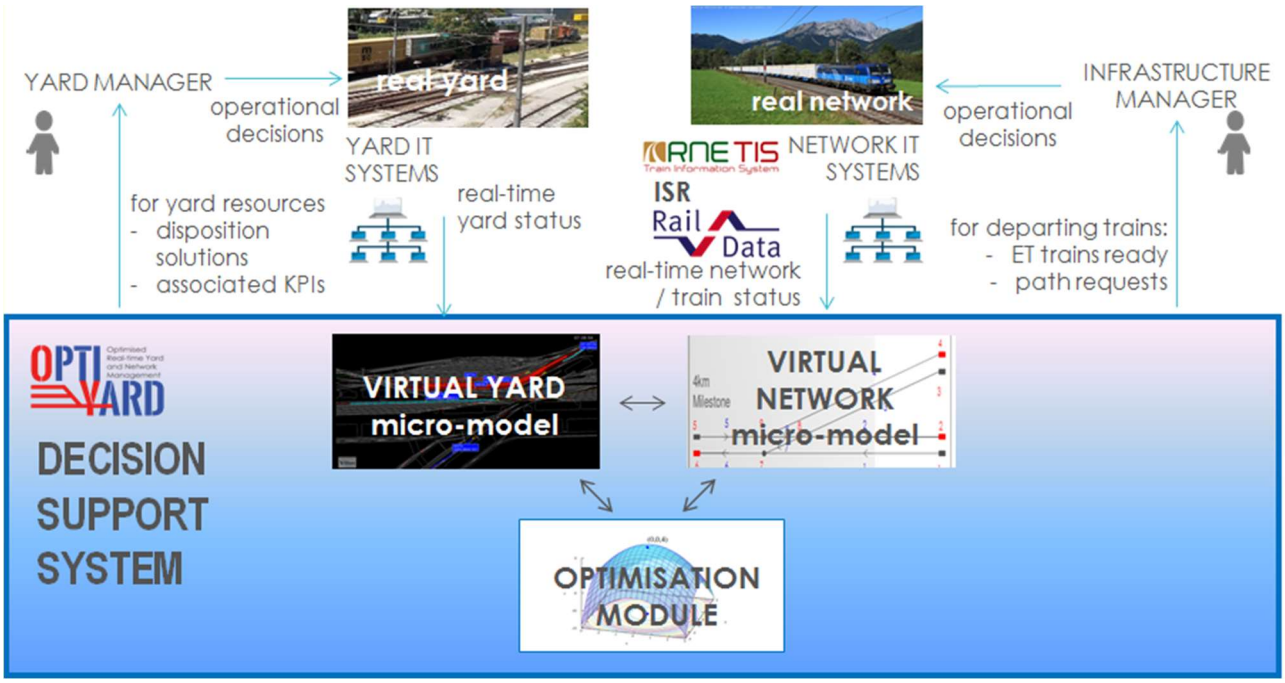


Figure 12. The OptiYard Decision Support System in its interaction with the yard and the surrounding network.

The outputs of the DSS are directed mainly to the yard manager. This allows for more informed decisions regarding the disposition of yard resources - locomotives, staff, wagons etc. However, in order to optimize yard operations, it is also important to gather and transfer information from the network infrastructure manager and operators (railway undertakings, terminal operators). A key reason for this is that the yard managers need to act on information about, for example, expected late arrivals of freight trains into the yard. Key outputs for the network infrastructure manager, on the other hand, are the expected times for trains to be ready for dispatch and associated train path requests. Through these interactions, the DSS can contribute to the optimization of operations on the wider network as well as the optimization of resource use and operations within the yards.

The OptiYard DSS complements and receives input from the increasingly developed IT systems that the yard manager may use to monitor the state of the yard via, e.g., a sensor network, portals (see, e.g., FR8HUB's Intelligent Video Gate IVG), or manual data input. It is also designed in such a way as to benefit from further improvements to such IT systems as they continue to evolve. The OptiYard DSS is also conceived to draw input from the IT system which represents railway network status for timing data and train composition data. The target here is full compatibility with RailNetEurope's Train Information System (TIS), along with the legacy IT systems of the infrastructure manager, as well as with the Telematics Applications for Freight Technical Specification for Interoperability (TAF TSI). Moreover, data from wagon tracking & tracing systems, such as RailData's ISR, may be used as input.

Outputs are produced by the optimization model, with the optimization results being validated through micro-simulation and used for the calculation of the set of Key Performance Indicators (KPI) required for subsequent impact appraisal. For the purposes of optimization, the network micro-simulation model delivers Expected Times of Arrival (ETAs) of trains at the yard's home signals with higher accuracy than available from the network's IT systems. This is done via real-time network status information. For departing trains, the network micro-simulation model provides foreseeably available train paths, again based on real-time information. Through yard optimization, the DSS is also capable of providing expected times at which any given train will be ready for departure and hence suggest when path requests are necessary.

5.2 CORRESPONDENCE WITH FR8HUB RESULTS

Here we describe the work on how we conceive the interaction of the network with two OptiYard DSS-equipped connected yards.

This relates to the "Single Train Insertion" use case described in FR8HUB deliverable D3.2 [42], where a new algorithm for insertion of an additional train path in an existing railway timetable is described. The Single Train Insertion algorithm searches for a Feasible Earliest Path (FEP). It does so by searching for available resources (i.e., time gaps between two consecutive trains) at the marshalling yard as early as possible. It then identifies the circumstances that allow for a train path to be selected between a sending and a receiving yard, taking into account any intermediate time constraints, such as changes of crew along the route, and a required minimum robustness (minimal temporal gap between the proposed path and existing trains, e.g., 100 seconds, see Figure 13).

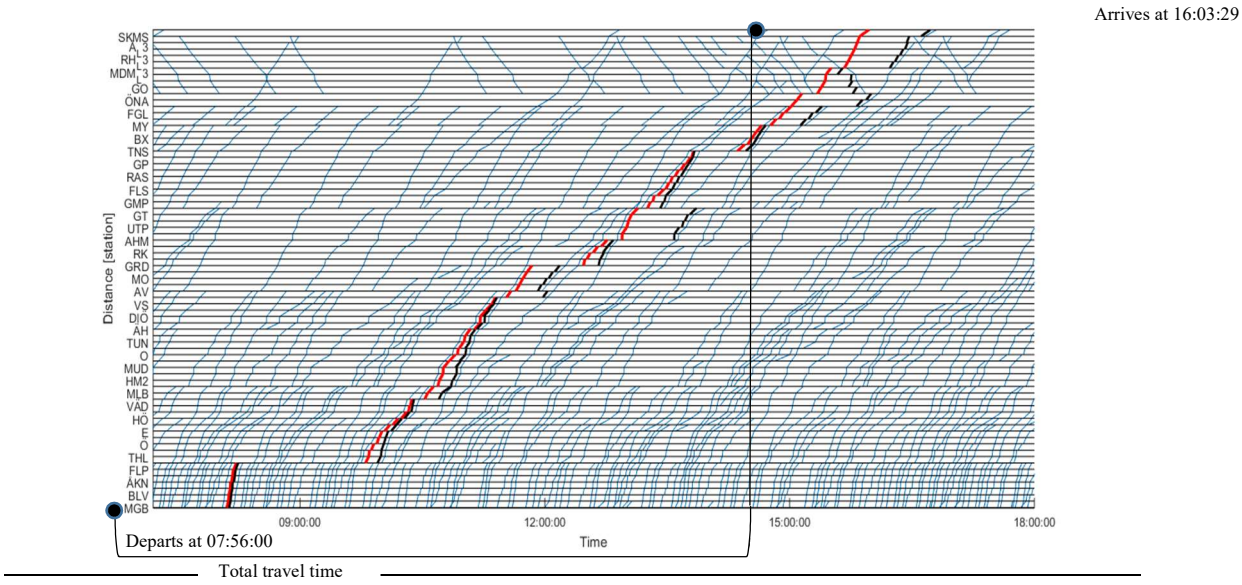


Figure 13. Time distance diagram of inserted freight train obtained with the FEP algorithm, minimum robustness requirement = 100 s (source: FR8HUB Deliverable D3.2 [42]).

The D3.2 deliverable [42] states the following in its conclusions.

"*The algorithm gives a satisfying result in a reasonable time (in a real time) for operational use in congested networks. In our test set of experiments, we focused on the first eligible path targeting to minimize the dwelling time at intermediate station and the total travel time. We compared the proposed algorithm to the current state of the art that is the maximum bottleneck path algorithm. Experimental results show that the proposed first eligible path algorithm significantly reduces the dwelling time at intermediate stations and reduce the total travel time. However, comparing to the maximum bottleneck path algorithm, the proposed algorithm finds the path with smaller robustness value. In order to enhance the algorithm, we added minimum robustness requirement to keep the robustness above a suitability threshold and while finding the first eligible path. Moreover, pre-processing may be applied to omit train paths with unwanted properties such as a travel time exceeding the train driver's maximum allowed working time. In order to have multiple trains inserted consecutively into the existing timetable the FEP algorithm can be called iteratively. It is expected that the algorithm can be further developed to solve the same problem once real-time perturbations of the timetable are considered.*"

The diagram shown in Figure 14 outlines the way the interaction between the real-time yard and network systems has been conceptualised.

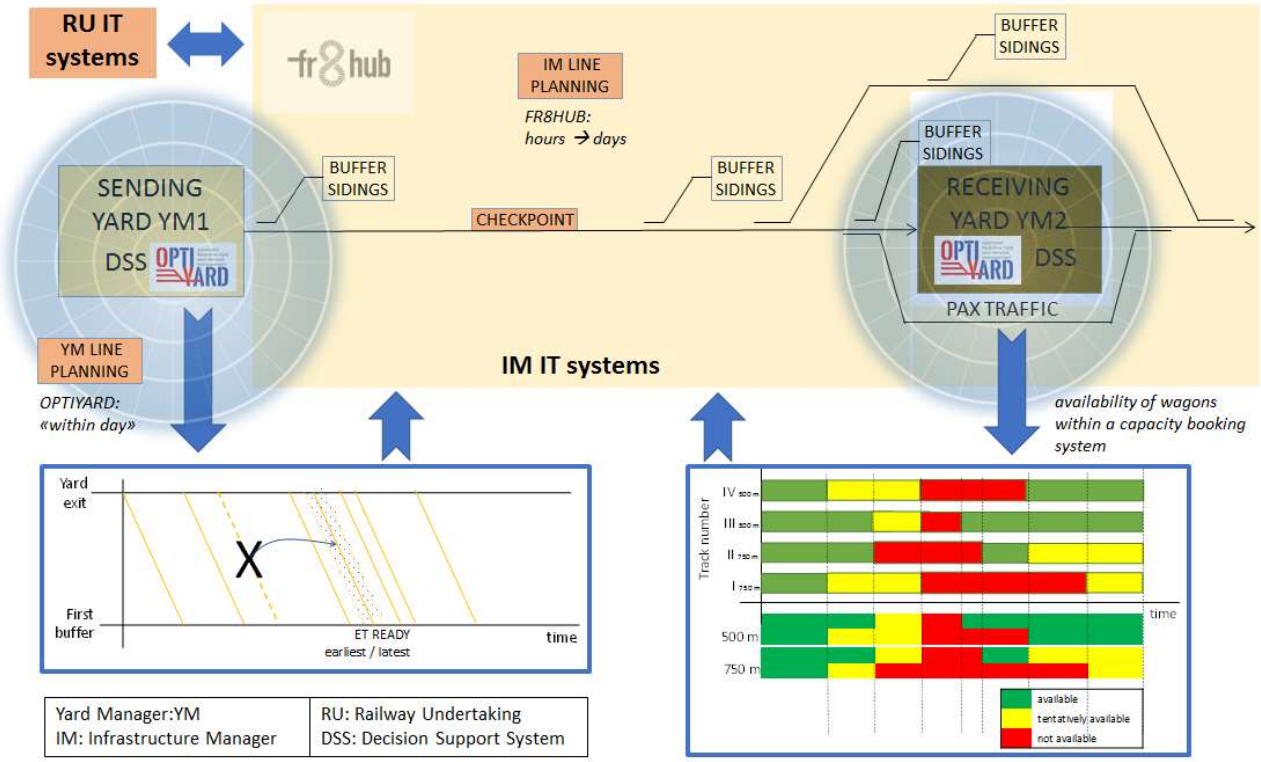


Figure 14. Inputs and outputs exchanged between the OptiYard Decision Support System and the Infrastructure Manager's IT systems.

Two rail yards (of any category such as marshalling yards, yards for intermodal traffic etc.) are connected by means of a railway line. The yard managed by Yard Manager YM1 is the sending yard, whilst YM2 manages the receiving yard. Both these yards are equipped with a real-time OptiYard DSS. Of course, these two roles can be exchanged depending on the direction of travel, leading the outputs from each yard's system to differ from each other in terms of their relationships with the Infrastructure Manager's IT systems, which contains real-time network management capabilities based on the Single Train Insertion algorithm of FR8HUB described above. The FR8HUB project work employs a real-world case study example around the Hallsberg – Malmö connection in Sweden but clearly has much wider generality.

We anticipate that both the network and the yards will aim to operate as closely as possible to predefined planning. However, with rail freight operations it is difficult to maintain punctuality to within minutes, even on days when there are no major disruptions. One such day free of major disruption is considered. One difference between OptiYard and FR8HUB relates to the time-horizon, where the focus of the OptiYard DSS is essentially "within day" (i.e., down to hours), whereas FR8HUB is more concerned with the range from hours to days, the latter time is being the one required, e.g., to complete an ad-hoc path request.

To give an example of how actual operations might develop in such a context, consider the case where at a given point in time, the sending yard's DSS calculates a few hours before a scheduled departure that a certain departing train is expected to suffer a significant delay. This may be for example due to sudden unexpected lack of yard personnel of which the DSS was informed a few minutes earlier via manual input. The IM systems are automatically informed via the SHIFT2RAIL integration layer whose requirements are set out in IN2RAIL, XRAIL2 and IMPACT 2 deliverables, probably in RailML format. If the expected delay is above a given threshold for the IM (to be determined, e.g., 15 minutes) this will trigger the need for (short-term) rescheduling on the

network. The IM systems thus immediately require the Yard 1 dispatcher to send confirmation of the train delay. Upon receipt of such confirmation, the IM systems notify Yard 2 so that their DSS can plan accordingly and reschedule to make use of the capacity that is now free. The receiving yard's DSS recalculates a prediction of track availability time windows considering the major train delay now confirmed. This sets a requirement for the time horizon of the DSS based on the travel time between sending and receiving yard. For the Hallsberg - Malmö example, the yard DSS should be looking at least about 12 hours into the future – 4 hours for advance notice of expected delay to the IM + 8 hours' travel time. The predicted track availability is provided in the form "track available, tentatively available or not available" based on given criteria (e.g., considering number of available tracks for the required train characteristics, degree of yard saturation). The Single Train Insertion algorithm receives this availability input[1] and uses the updated time windows for Yard 2, along with those coming from the intermediate service points along the track including buffer yards, to generate an initial set of path calculations (e.g., ETA-Robust, high-robustness, long-travel-time path, and ETA-Fast, low robustness, quick path) for the delayed train. The IM dispatcher selects a preference (e.g., ETA-Fast, in order to free up network capacity as soon as possible). A request is issued to the dispatcher of Yard 2, who examines actual availability as proposed by the OptiYard DSS (particularly if the result is "tentative") and either accepts and commits or else refuses. In case of acceptance the train path is confirmed by the IM. In case of refusal by the receiving yard, the next-fastest path is proposed, and so on, until Yard 2 accepts a solution.

5.3 OTHER CURRENT AND POTENTIAL FUTURE CAPABILITIES

The Optiyard system allows for consideration of various complex elements of rail operations, including:

- Presence of mixed passenger and freight traffic. For example, Figure 14 shows the case where a passenger line is contained within the "relevant network" of the OptiYard DSS for Yard 2, as well as a "through-line" for freight traffic. The OptiYard DSS is conceptually capable of re-estimating the ETAs to the yard by simulating all train trajectories in the vicinity of the yard, taking the real-time information coming from IM systems into consideration. (On corridors with the Train Information System TIS of RailNetEurope, otherwise legacy systems, integrated with information on on-going disruptions via the asset management information through the integration layer). However, how to integrate this with the Single Train Insertion algorithm is an open issue to be solved once the network model of OptiYard is further defined.
- Use of network capacity to make up for lack of yard capacity. This is represented through the availability of "buffer sidings" on the network but relatively close to the yard, where freight trains can be held waiting for yard capacity to become available. The nearest such location to the yard can be usefully considered as the boundary of the yard DSS scope. The STI considers these along with their availability time windows.
- Presence of network "checkpoints" along the line where time window constraints are possible (e.g., where freight trains might wait for activities such as driver change, loco change, dropping or collecting wagons etc.).

[1] Possibly in the form of a time window, consistently with the algorithm's approach. Currently the OptiYard software is not configured this way but could be with further study.

- Interactions with and between different railway undertakings (RUs), where several could be operating in the yards. For the moment the methodology is developed on the basis that the RUs agree with all decisions by IMs and YMs.
- Optimizing flow versus capacity booking (integration) and group trains. Capacity booking could be a key element in the re-launch of SWL traffic as envisaged by the EU. Here, wagons would be booked onto specific outbound trains for each rail terminal encountered during the journey, with the purpose of ensuring their arrival at the final terminal (or the customer's siding) on an exactly specified train. It is thus extremely important that trains leaving a yard or terminal do so with high punctuality even if not all expected wagons are included, otherwise those wagons that are actually in the train may be delayed and miss their next connection, which would jeopardize the very purpose of capacity booking.
- Another key element for SWL traffic is the possibility of "dropping" groups of wagons at intermediate stops between two terminals. From such locations the wagons can then be shunted to the customers' sidings or be unloaded in dedicated facilities. For this type of traffic, it is important that in the marshalling yard the specified wagon order for departing trains is achieved regardless of any perturbations affecting incoming trains. Given the importance of achieving the exact planned wagon order in departing trains for a large majority of situations, it is important to try to prevent deviations from timetable for incoming trains from breaking the sequence of operations in the yard as far as possible.

As discussed earlier, the Optiyard system focuses on shorter-term within-day operations. In principle, however, longer timescales can be accommodated. An example would be an ad-hoc path timetabling request perhaps made a few days before the actual train is to be run, e.g., upon request of a customer. Although the OptiYard time horizon is within day, the annual timetable can be incorporated as an input with the yard manager's planned track occupations for such timetabled trains available in advance. This would allow the FEP algorithm to search for an available path a few days in advance, thus supporting ad-hoc timetabling through its interaction with the yard's DSS. Such a search could be updated if actual operations do not actually follow the timetable, as the interaction evolves towards full real-time capabilities.

Even though the OptiYard DSS envisages more accurate ETA and ETD predictions than is currently the case, it would also be advantageous to quantify the accuracy of time (window) predictions. Each time prediction should be made within an agreed probability parameter, e.g., 'earliest time' means less than 5% probability that it could actually be even earlier, 'latest time' means less than 5% probability that it could actually be even later. This reflects the highly non-linear nature of the reliability problem. Methods for this should be developed, e.g., in future Shift2Rail projects.

5.4 CONCLUSION

In this chapter, we proposed strategies that may be put in place for integrating the OptiYard system and the overall network management. In addition to considering the current structure of the OptiYard DSS, some possible evolutions are also presented. The new ideas reported show how the integration with the network is possible in coherence with other Shift2Rail projects, and namely with the very strictly connected FR8HUB project.

6. CONCLUSIONS

In this deliverable we describe the OptiYard proposal for optimizing yard management operations.

We started with an analysis of the state of the art, showing the main gaps that needed to be filled by our work. Then, we proposed an optimization algorithm filling these gaps and we presented the OptiYard optimization and simulation integrated framework. Finally, we described strategies for including the obtained optimized yard management in the overall picture covering the whole railway network.

Through the optimization and simulation integrated framework, realistic situation in which a yard is automatically managed in an optimized way can be reproduced in laboratory. Hence, the OptiYard project managed to produce a functioning TRL4 demonstrator. Here, in the closed-loop framework, the simulator plays the role of a real yard and constantly communicates with the optimization module in charge of making decisions. The optimization algorithm updates its decisions based on the most recently communicated state of the yard and forecast on arriving trains.

In addition to providing a functioning demonstrator, OptiYard also brings significant contributions to the practice and the state of the art. First, it proposes an optimization algorithm capable of dealing with all processes and constraints appearing in a yard, without limitations in terms of yard structure. Second, it pushes forward the reflection on how yard and network can be managed together, to maximize the efficiency of the whole system. Third, it carries into effect a closed-loop framework. Such a framework has never been put in place of a freight yard, and it has very seldom been achieved for the network. A previous example of closed-loop is represented by the one designed in the ONTIME FP7 project for the network, from which we took inspiration in OptiYard. The design and implementation of the closed-loop framework for yard management is a major contribution of OptiYard to the state of the art, showing how an optimized system could be deployed in practice. This design paves the way for the optimization of yard processes on the one hand, and the integration of yard and network management on the other.

7. BIBLIOGRAPHY

[1] A. Caprara, L. Kroon and P. Toth, "Optimization Problems in Passenger Railway Systems," *Wiley Encyclopedia of Operations Research and Management Science,* pp. 3896-3905, 2010.

[2] N. Boysen, M. Fliedner, F. Jaehn and E. Pesch, "A Survey on Container Processing in Railway Yards," *Transportation Science,* vol. 47, no. 3, pp. 312-329, 2013.

[3] M. Bohlin, S. Gestrelius, F. Dahms, M. Mihalák and H. Flier, "Optimization methods for multistage freight train formation," *Transportation Science,* vol. 50, no. 3, pp. 823-840, 2016.

[4] Y. Bontekoning and H. Priemus, "Breakthrough innovations in intermodal freight transport," *Transportation Planning and Technology,* vol. 27, no. 5, pp. 335-345, 2004.

[5] M. Gatto, J. Maue, M. Mihalák and P. Widmayer, "Shunting for Dummies: An Introductory Algorithmic Survey.," *Robust and online large-scale optimization,* vol. 5868, pp. 310-337, 2009.

[6] N. Boysen, M. Fliedner, F. Jaehn and E. Pesch, "Shunting yard operations: Theoretical aspects and applications," *European Journal of Operational Research,* vol. 220, no. 1, pp. 1-14, 2012.

[7] S. Gestrelius, M. Aronsson, M. Joborn and M. Bohlin, "Toward a comprehensive model for track allocation and roll-time scheduling at marshalling yards," *Journal of Rail Transport Planning and Management,* 2017.

[8] M. Bohlin, R. Hansmann and U. T. Zimmermann, "Optimization of Railway Freight Shunting," in *Handbook of Optimization in the Railway Industry*, Springer, 2018, pp. 181-212.

[9] E. Petersen, "Railyard modeling: Part I. Prediction of put-through time," *Transportation Science,* vol. 11, no. 1, pp. 37-49, 1977.

[10] M. Marinov and J. Viegas, "A simulation modelling methodology for evaluating flat-shunted yard operations," *Simulation Modelling Practice and Theory,* vol. 17, no. 6, pp. 1106-1129, 2009.

[11] G. Di Stefano and M. L. Koči, "A graph theoretical approach to the shunting problem," *Electronic Notes in Theoretical Computer Science,* vol. 92, pp. 16-33, 2004.

[12] R. S. Hansmann and U. T. Zimmermann, "Optimal Sorting of Rolling Stock at Hump Yards," in *Mathematics - Key Technology for the Future: Joint Projects Between Universities and Industry 2004-2007*, H. Krebs and W. Jäger, Eds., Berlin, Heidelberg, Springer Berlin Heidelberg, 2008, pp. 189-203.

[13] R. Jacob, P. Márton, J. Maue and M. Nunkesser, "Multistage methods for freight train classification," *Networks,* vol. 57, no. 1, pp. 87-105, 2011.

[14] S. Yagar, F. Saccomanno and Q. Shi, "An efficient sequencing model for humping in a rail yard," *Transportation Research Part A: General,* vol. 17, no. 4, pp. 251-262, 1983.

[15] M. Saeednia, D. Bruckmann and U. Weidmann, "Event-Based Model for Optimizing Shunting Yard Operations," *Transportation Research Record,* vol. 2475, no. 1, pp. 90-94, 2015.

[16] F. Jaehn, J. Rieder and A. Wiehl, "Minimizing delays in a shunting yard," *OR Spectrum,* vol. 37, no. 2, pp. 407-429, 2015.

[17] F. Jaehn, J. Rieder and A. Wiehl, "Single-stage shunting minimizing weighted departure times," *Omega,* vol. 52, pp. 133-141, 2015.

[18] F. Glover and M. Laguna, "Tabu search," in *Handbook of combinatorial optimization*, Springer, 1998, pp. 2093-2229.

[19] E. Dahlhaus, P. Horak, M. Miller and J. F. Ryan, "The train marshalling problem," *Discrete Applied Mathematics,* vol. 103, no. 1-3, pp. 41-54, 2000.

[20] N. Boysen, S. Emde and M. Fliedner, "The basic train makeup problem in shunting yards," *OR spectrum,* vol. 38, no. 1, pp. 207-233, 2016.

[21] J. T. Haahr and R. M. Lusby, "A matheuristic approach to integrate humping and pullout sequencing operations at railroad hump yards," *Networks,* vol. 67, no. 2, pp. 126-138, 2016.

[22] S. He, R. Song and S. S. Chaudhry, "An integrated dispatching model for rail yards operations," *Computers & operations research,* vol. 30, no. 7, pp. 939-966, 2003.

[23] E. Kraft, "A hump sequencing algorithm for real time management of train connection reliability," in *Journal of the Transportation Research Forum*, 2000.

[24] E. Kraft, *Priority car sorting in railroad classification yards using a continuous multi-stage method,* Google Patents, 2002.

[25] C. F. Daganzo, R. G. Dowling and R. W. Hall, "Railroad classification yard throughput: The case of multistage triangular sorting," *Transportation Research Part A: General,* vol. 17, no. 2, pp. 95-106, 1983.

[26] C. F. Daganzo, "Static blocking at railyards: Sorting implications and track requirements," *Transportation Science,* vol. 20, no. 3, pp. 189-199, 1986.

[27] D. E. Knuth, *Fundamental Algorithms, volume 1 of the Art of Computer Programming,* Addison-Wesley, 1973.

[28] T. Shi and X. Zhou, "A mixed integer programming model for optimizing multi-level operations process in railroad yards," *Transportation Research Part B: Methodological,* vol. 80, pp. 19-39, 2015.

[29] P. Márton, J. Maue and M. Nunkesser, "An improved train classification procedure for the hump Yard Lausanne Triage," in *OASIcs-OpenAccess Series in Informatics*, 2009.

[30] N. Adamko and V. Klima, "Optimisation of railway terminal design and operations using villon generic simulation model," *Transport,* vol. 23, no. 4, pp. 335-340, 2008.

[31] Y. Zhang, R. Song, S. He, H. Li and X. Guo, "Optimization of Classification Track Assignment Considering Block Sequence at Train Marshaling Yard," *Journal of Advanced Transportation,* 2018.

[32] T. Bektas, T. G. Crainic and V. Morency, "Improving the performance of rail yards through dynamic reassignments of empty cars," *Transportation Research Part C: Emerging Technologies,* vol. 17, no. 3, pp. 259-273, 2009.

[33] N. Bostel and P. Dejax, "Models and algorithms for container allocation problems on trains in a rapid transshipment shunting yard," *Transportation science,* vol. 32, no. 4, pp. 370-379, 1998.