

# Empty Railcar Repositioning Subject to Travel Time Uncertainty

Romain Wlodarczyk

Thesis submitted to the Faculty of the  
Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

Master of Science  
in  
Industrial and Systems Engineering

Pierre Lemaire, Co-Chair  
Joel A. Nachlas, Co-Chair  
Hanif D. Sherali

July 20, 2009  
Blacksburg, Virginia

Keywords: Rail, Optimization, Freight, Repositioning, Empty Railcars

Copyright 2009, Romain Wlodarczyk

# Empty Railcar Repositioning Subject to Travel Time Uncertainty

Romain Włodarczyk

(ABSTRACT)

The empty railcar repositioning strategy generates no income but is crucial for a good service quality. It should satisfy two main objectives: fulfilling the customer demand and generating as little expense as possible. Moreover, because of breakdown or heavy traffic, variation on travel times happens to be the main cause of uncertainty in railroad scheduling and must be taken into account to suggest a robust plan.

This thesis presents the linear program used in a prototype tool for the optimization of empty railcar repositioning strategy designed for the SNCF<sup>1</sup>. The resulting schedule is computed with CPLEX and minimizes moving cost, delay and unfulfillment penalties. Substitutions of railcar categories are also permitted and eventually penalized. In addition, uncertainty on travel times is handled by considering the expected cost of a move (regarding delay probability and possible penalties) and by adding slack periods at the end of moves. The robustness can be modulated through the use of a cursor. Finally, the model enforces a decision making process previously defined by the SNCF to ensure that the suggested plan can be easily grasped and trusted by users.

Schedules have been generated based on randomly generated data and simulated. Results show a potential saving of 10% on considered costs and a good range of use of the robustness cursor is suggested.

Finally, paths for improvement of this prototype are proposed to meet the eventual schedulers' further needs in order to move forward the production of this tool at the company scale.

## GRANT INFORMATION

This work received support from the SNCF<sup>1</sup>

---

<sup>1</sup>Société Nationale des Chemins de fer Français (*French National Railways*)

# Acknowledgments

I would like to thank all the members of my thesis advisory committee for their support and assistance throughout this project : Dr. Pierre Lemaire, Dr. Joel A. Nachlas and Dr. Hanif D. Sherali. All have offered their time and guided me during all the process. They handled with professionalism the problems involved by the geographical distance. Dr. Pierre Lemaire deserves special thanks for the time he takes to double check my works, for his encouragements, advice and suggestions. I would like to express my sincere gratitude to my SNCF tutor, Dr. David De Almeida, for his sympathy, guidance and support. His knowledge and expertise helped me tackling several problems during all the project. I would like to thank Christian Weber for having given me the opportunity to carry out my internship at the Innovation and Research Department at SNCF.

I would like to thank my officemates Ouissam, Stéphane, Christian and my colleagues for their help, their sympathy and their contribution to the good ambience which prevails during my internship.

This thesis gives me also the opportunity to thank all the VT staff, faculty and students for having made my semester at Blacksburg a stay full of memories and a tremendous experience. I really appreciated their warm welcome and their help to make me accomodate to a new culture; I am particularly thinking of my professors and the Cranwell Center staff, they have done an incredible work to make me feel at home in Blacksburg. I extend my thanks to Dr. Joel A. Nachlas, Dr. Olivier Peton, Dr. Bruno Castanier and John Miller Jones for their support and guidance during my application to the dual degree program. I thank also my classmate, Matthieu Courmont, who showed me the way to Virginia Tech.

Finally, I would like to express my gratitude to my family for their support and help from the very beginning to the end of the VT experience. I include also my host family in Blacksburg: Jerry and Joan. Last but not least, I would like to give a special friendly thank to Stéphane for his dedicated support during all my research and, generally speaking, during all my graduate studies.

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Background . . . . .	1
1.2	Objectives and Motivation . . . . .	2
1.2.1	Empty Railcar Repositioning . . . . .	2
1.2.2	SNCF's Request . . . . .	2
1.3	Problem Statement . . . . .	3
1.4	Assumptions . . . . .	4
1.4.1	Planning Level . . . . .	4
1.4.2	SNCF's Decision Making Process . . . . .	4
1.4.3	Time Horizon . . . . .	5
1.5	Outline of Thesis . . . . .	7
<b>2</b>	<b>Literature Review</b>	<b>8</b>
2.1	Flow Representation . . . . .	8
2.2	Uncertainty on Data . . . . .	11
2.3	Approaches to the Empty Railcar Repositioning Problem . . . . .	12
2.4	General Approaches to Stochastic Model . . . . .	14
2.4.1	Proactive Scheduling . . . . .	15
2.4.2	Reactive Scheduling . . . . .	15
2.4.3	Proactive and Reactive Scheduling . . . . .	16
2.4.4	Rescheduling . . . . .	16

2.5	Model Implementation . . . . .	17
2.6	Motivation for Further Research . . . . .	17
<b>3</b>	<b>Empty Railcar Management</b>	<b>19</b>
3.1	Terminology . . . . .	19
3.2	Notation . . . . .	20
3.2.1	Sets . . . . .	20
3.2.2	Orders . . . . .	21
3.2.3	Information Extracted from Information Systems . . . . .	21
3.2.4	Costs . . . . .	23
3.2.5	Parameters . . . . .	24
3.3	Linear Program . . . . .	24
3.3.1	Decision Variables . . . . .	25
3.3.2	Secondary Decision Variables . . . . .	25
3.3.3	Objective Function . . . . .	26
3.3.4	Constraints . . . . .	27
3.3.5	Network Structure and Integrality Constraints . . . . .	30
3.4	Decision Making Process . . . . .	33
3.4.1	Description . . . . .	33
3.4.2	About the Myopic Structure of the Model . . . . .	38
3.5	Feasibility . . . . .	38
<b>4</b>	<b>Coping with Uncertainty on Travel Times</b>	<b>40</b>
4.1	Anticipating Delay On Travel Times . . . . .	40
4.1.1	Overview of the Proposed Approach . . . . .	41
4.1.2	Notations . . . . .	41
4.1.3	Anticipated Delay . . . . .	43
4.1.4	Linear Program With Travel Time Uncertainty . . . . .	50
4.2	Robustness Level Cursor . . . . .	53

4.3	About the Control of Equity . . . . .	54
4.4	About the Pre-Processing . . . . .	54
<b>5</b>	<b>Model Evaluation</b>	<b>55</b>
5.1	Data Generator . . . . .	55
5.2	Size and Computing Time: 3-step LP and Global LP Comparison . . . . .	57
5.3	Simulator . . . . .	59
5.3.1	Description . . . . .	59
5.3.2	Transience . . . . .	60
5.4	$\gamma_{rob}$ Cursor Test . . . . .	64
5.5	Contribution of Stochastic Approach Tests . . . . .	69
5.6	Anticipated Delay Pertinence Tests . . . . .	69
5.7	Conclusions . . . . .	70
<b>6</b>	<b>Conclusions and Future Research</b>	<b>77</b>
6.1	Summary . . . . .	77
6.2	Perspectives for Future Research . . . . .	78
	<b>Bibliography</b>	<b>80</b>
	<b>Appendix A : Deterministic Linear Program <math>LP_{glob}</math></b>	<b>82</b>
	<b>Appendix B : Stochastic Linear Program <math>LP_{stoc}</math></b>	<b>85</b>
	<b>Appendix C : Simulator Source</b>	<b>88</b>

# List of Figures

1.1	Execution Process of Optimization Tool . . . . .	6
2.1	White and Bomberault's Network . . . . .	10
3.1	Network Structure . . . . .	31
3.2	Layers Structure . . . . .	32
4.1	Different Anticipated Moves . . . . .	44
4.2	Scenarios $s(\varphi, \sigma_r)$ and Risk measure . . . . .	45
4.3	Time Decomposition of a Forbidden Anticipated Delay . . . . .	48
4.4	Time Decomposition of a Corrected Move . . . . .	49
5.1	Simulator Flow Chart (Part 1) . . . . .	61
5.2	Simulator Flow Chart (Part 2) . . . . .	62
5.3	Mean Stabilization . . . . .	63
5.4	Delay Cost Variation Depending on $\gamma_{rob}$ and Ratio OD . . . . .	66
5.5	Holding Cost Variation Depending on $\gamma_{rob}$ and Ratio OD . . . . .	67
5.6	Sum of Holding and Delay Cost Variation Depending on $\gamma_{rob}$ and Ratio OD . . . . .	68
5.7	Delay Cost Variation Depending on $\gamma_{rob}$ and Distribution . . . . .	71
5.8	Holding Cost Variation Depending on $\gamma_{rob}$ and Distribution . . . . .	72
5.9	Sum for Holding and Delay Cost Variation Depending on $\gamma_{rob}$ and Distribution . . . . .	73
5.10	Moving Cost Variation Depending on $\gamma_{rob}$ , ratio OD and Distribution . . . . .	73
5.11	Unfulfillment Cost Variation Depending on $\gamma_{rob}$ , ratio OD and Distribution . . . . .	74
5.12	Cost Decomposition Input C1 . . . . .	74

5.13 Cost Decomposition Input C2 . . . . .	75
5.14 Cost Decomposition Input C3 . . . . .	75
5.15 Sum of Costs for 7 Solving Methods . . . . .	76

# List of Tables

1.1	Steps of Decision-Making Process . . . . .	5
3.1	Subsets of Decision-Making Process . . . . .	34
4.1	Computing Modifying costs: $\sigma_a = \sigma_{lim}(\varphi)$ . . . . .	46
4.2	Computing Modifying costs: $\sigma_a = 2 > \sigma_{lim}(\varphi) = 1$ . . . . .	47
5.1	LP-sequence Comparison . . . . .	59

# Chapter 1

## Introduction

### 1.1 Background

SNCF (Société Nationale des Chemins de fer Français<sup>1</sup>) is the French public-service corporation for railway transportation. SNCF was created in 1934 when all railroad companies were merged in order to gather all networks in a nation-wide one. In 2005, the group employed 208,000 people with a turnover of €21 billion and transported 74 billion passenger-kilometers and 43.4 billion ton-kilometers of goods. Its network consists now of 4,715 stations linked by 31,000 km of tracks.

SNCF is divided into 4 branches:

- Long-distance and high-speed passenger trains
- Intercity and suburban passenger trains
- Freight trains

---

<sup>1</sup>French National Railways

- Maintenance and management of railway infrastructure, engineering, etc.

The work described in this thesis is a part of a research project intended for the freight division. Up to March 2006, SNCF had a monopoly on both passenger and freight transportation. Since this date, freight industry has been opened to competition. This has involved many changes in the SNCF's strategy and its freight branch has now to face a new competitive environment.

## 1.2 Objectives and Motivation

### 1.2.1 Empty Railcar Repositioning

In order to be competitive SNCF has to answer quickly and efficiently to customers requesting for empty railcars while being able to fulfill orders submitted previously. In addition, the transportation flows are not symmetrical and location of demands and offers of empty railcars are different. Consequently, empty railcars must be repositioned frequently. SNCF has not yet developed an optimization tool; railcars shortage at terminals and delivery delays have been reported.

### 1.2.2 SNCF's Request

As an intern at the Innovation and Research department, I was proposed to design the prototype of a software optimizing the empty railcar repositioning and working with the CPLEX engine. This tool should extract data from currently used Information Systems (IS) and handle the uncertainty on travel times. This software should also fit the decision making process exploited at SNCF.

### 1.3 Problem Statement

The empty railcar management problem consists in allocating empty railcars to orders. If railcars are not available at the delivery terminal, a move request is submitted to another terminal, missing railcars are then transferred with a given *moving cost* and they are set aside for the underlying order.

Travel times are known; however, deviations to theoretical times often occur and must be taken into account in order to make production plans more robust and reliable.

In addition, we suppose that an allocated railcar cannot be unallocated or reallocated to another order even when facing unpredicted events. Thus railcars should not be set aside too early and we assume that a railcar is allocated to an order at the latest possible time period prior to delivery. That is, the date of allocation is assumed to be equal to the order due-date or greater if the railcar is late.

Moreover, when facing shortage and if customers allow it, a category of railcars can be substituted by another. Substitution is usually authorized when the customer's facilities allow the loading and the unloading of different types of railcars. Since the ordered category is sometimes more suitable than its substitutes, a penalty called *substitution cost* is applied when categories are substituted.

When railcars are not available and no railcars can be sent on time to a terminal to fulfill an order a *delay penalty* is applied. On the same hand, if the order cannot be fulfilled by the end of the horizon time, an *unfulfillment penalty* applies.

The stakes of empty railcar management lie therefore in the minimization of the total cost, the on-time delivery of all orders and the ability to face unpredicted delays.

Finally, in order to increase the effectiveness and applicability of our work, data will also

be extracted from existing information technologies. Two of them are currently used for operational empty railcar management:

- **BIG** provides information about inventory of empty railcars parked in terminals,
- **TIF** provides information about current allocation of railcars.

## 1.4 Assumptions

### 1.4.1 Planning Level

The empty railcar allocation problem can be studied at different degrees of accuracy. However, SNCF is willing to support agents by suggesting them an optimal solution. It is not wished to identify every single railcar allocated to each order; the size of the fleet is much too big to be able to identify every railcar and locate it at anytime. It is only requested to model allocations as quantities. Moreover, the current network cannot be modified and railcars must not be allocated to trains. It is neither asked to allocate railcars to an automotive unit. Therefore, the requested planning level can be qualified as a tactical plan. This in contrast with strategic and operational planning, where the latter would have actually involved allocating individual railcars to trains and the former to schedule the trains.

### 1.4.2 SNCF's Decision Making Process

SNCF requires our work to be consistent with the decision making process to avoid users' reluctance in using the strategy suggested by the software and the associated extra outcomes (see section 2.5). Orders are currently divided into two categories: committing orders and not-committing ones. An order is committing if SNCF has pledged to deliver railcars on

time. Generally speaking, only orders submitted lately are not committing. The solving process is then divided into three steps characterized by the due-date of orders and their priority (committing or not). The sequence of orders considered is depicted in table 1.1. The suggested process of execution is depicted in Figure 1.1

Table 1.1: Steps of Decision-Making Process

<b>step</b>	<b>Orders considered</b>
step 1	Committing orders due on first day of time horizon
step 2	Other committing orders and step 1 unfulfilled orders
step 3	Not-committing orders and step 2 unfulfilled orders

This decision-making process clearly degrades the solution with regard to a global optimization approach, making it sub-optimal. Indeed, it can be qualified as “myopic”, especially in the first step: orders considered in this step will be processed without taking into account subsequent ones. The consequences of such a process on the solutions are identified by SNCF; however, priority is given to the tool implementation and its user friendliness. The process could be modified thereafter though.

Nevertheless, the chosen process has the advantage of decreasing the size of the model. As orders are divided and processed separately, the number of variables decreases and the solving process will probably speed up.

### 1.4.3 Time Horizon

The size of the time horizon is chosen to enforce the following assumptions:

- Travel times are smaller than the time horizon;
- Due-dates can not be out of the time horizon.

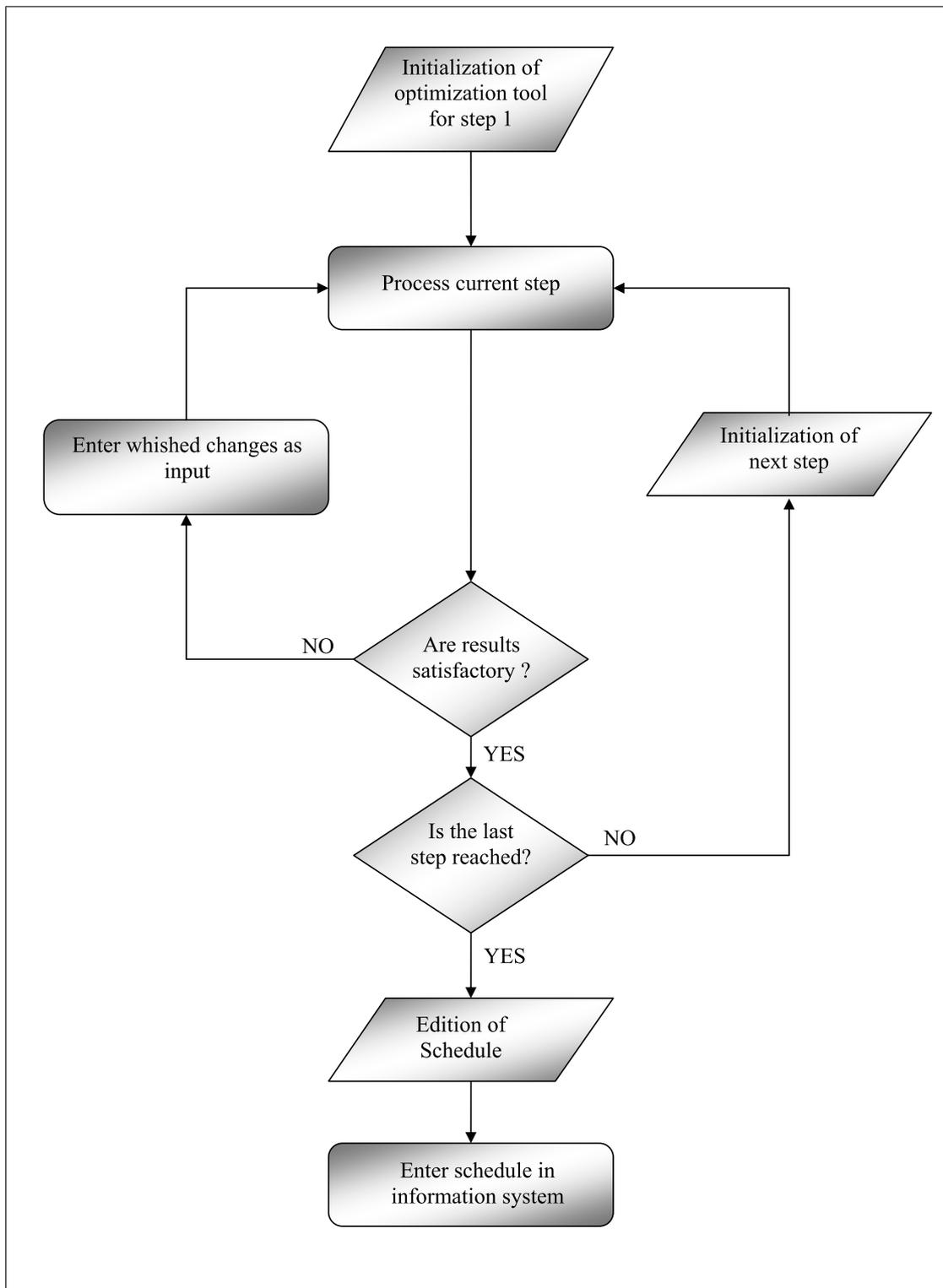


Figure 1.1: Execution Process of Optimization Tool

Remark: railcar moves expected to arrive after the last day of the time horizon will not be considered.

## 1.5 Outline of Thesis

This thesis is organized as follows. In chapter two, a literature review presents articles and existing works that already address empty railcar management and scheduling problems. In Chapter three, notations are introduced and an integer linear program is proposed for the deterministic case. Chapter four shows how to extend the previous model in order to capture travel time uncertainties. Chapter five deals with the tests and the evaluation of the model. Finally, the last Chapter presents conclusions and suggestions for future research.

# Chapter 2

## Literature Review

Much research has been carried out on the problem of empty freight cars distribution and a lot of articles have been published in the literature. As illustrated by Crainic and Laporte [2], this is mainly due to the various possible planning levels (strategic, tactical or operational planning) adopted by authors, and the size of the railroad networks considered. In addition, the problem is sometimes extended to container management or generalized to any transportation system. As previously explained in section 1.4.1, the model requested by SNCF can be positioned on a tactical planning level; almost all articles addressed in this review are therefore limited to this level.

### 2.1 Flow Representation

Before trying to solve any problem, the first concern lies in the representation of the rail network through time. White and Bomberault [23] proposed a network (showed in Figure 2.1) depicting explicitly the time horizon. The nodes of this network consist in replicating each terminal at each time period over the time horizon. Four kinds of arcs are represented

on this network:

- **Populating arcs:** they link the main source node to the terminal nodes at the initial period in order to set-up the railcars inventories.
- **Depopulating arcs:** similar to the populating arcs, they conduct flows from nodes at the last period to the main sink node.
- **Holding arcs:** they link a terminal to the same terminal at the next time period. A flow circulating on such an arc represents railcars held at the terminal. The cost associated to these arcs is the holding cost of a railcar at this terminal.
- **Traveling arcs:** they link a terminal to another terminal. A flow circulating on a traveling arc represents a move of cars from the first terminal to the second. These arcs should reach the second terminal at the theoretical arrival period. Costs associated to these arcs are the cost of moving one railcar from the first terminal to the second.

The empty railcar management problem can be modeled as a flow problem on this network. The objective function is therefore the minimization of the sum of all flow costs. This network was initially intended to a homogeneous fleet, it can be extended to a heterogeneous fleet by adding a commodity for each railcar category. However, the problem becomes then a more complex multi-commodity flow problem. Unfortunately, classical flows optimization techniques, especially in the multi-commodity flow problem, may be too slow and it may turn out to be difficult to implement rules and practical constraints without biasing the generic flows model. However, this network representation is very popular and very convenient to display results, this is actually the one which will be used in this thesis.

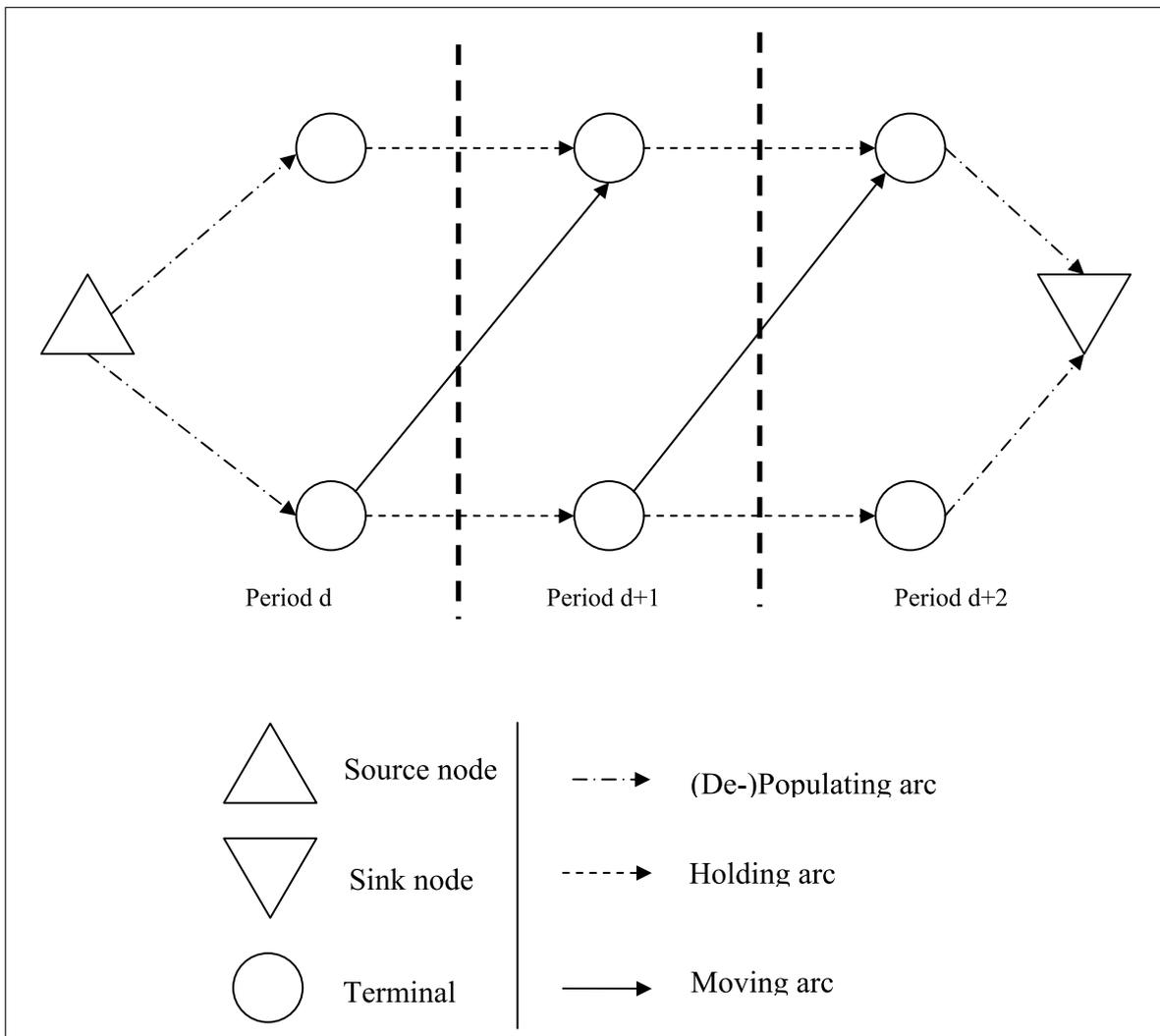


Figure 2.1: White and Bomberault's Network Flow Representation (Travel time: 1 period)

## 2.2 Uncertainty on Data

The main stochastic concept addressed in published works deals with the uncertainty on available data. The most relevant approaches to our problem address up to three sources of uncertainty:

- The uncertainty on demand: it is due to the unpredictable behavior of customers who cancel orders, modify them or submit them lately.
- The uncertainty on supply: this is mainly due to human errors in entering data but can also be the result of breakdowns.
- The uncertainty on travel times: because of breakdowns or heavy traffic, deviations are usual between theoretical travel times and factual ones.

At SNCF, orders have to be submitted 6 days before their delivery times, late orders are treated as low priority ones with residual resources and without obligation. Thus, the uncertainty aspect concerns rather travel times than demand or supply. Few stochastic models include uncertainty on travel times. Jordan and Turnquist [8] rely on variances and expected values for supply, demand and travel times. It is assumed all three follow a given distribution. The objective consists then in maximizing the sum of expected profits using nonlinear programming techniques. However, available statistics at SNCF are not precise enough to compute such a distribution. Moreover, in order to deal with uncertainty, yard managers can substitute a railcar category to another (if allowed by the customer); this low-cost substitution is not taken into account in Jordan and Turnquist's model.

## 2.3 Approaches to the Empty Railcar Repositioning Problem

A first approach consists in addressing the problem as an Inventory Management model, as presented in Philip and Sussman [13]. The objective consists then in computing the optimal inventory of empty freight car in a terminal regarding the holding costs and the shortage costs. However, this approach does not provide an optimal solution over a set of terminals nor details about moves of railcars which is the SNCF's main concern.

From the end of the 70's, thanks to the development of information technologies, the increasing popularity of optimization techniques led a large number of scientists to publish researches about empty railcars management. Dejax and Crainic [4] made a comprehensive state-of-the-art recording all models relevant to the problem of empty vehicles allocation.

Misra [12] was one of the first to use Linear Programming to deal with empty railcar management. Misra suggested minimizing the cost of countering the unbalance between loaded and empty railcar flows at the nodes of a railroad network. Further refinements of this deterministic model (Herren [5]) showed a lack of robustness. The uncertainty on data prevents solution to be applied on real-life large problems with the reliability requested by SNCF.

Many models have been published further to Misra's one, the most relevant articles are presented in the survey of Cordeau et al. [1]. All of them are linear flows problems.

Kwon et al. [9] proposed a model considering the train schedule and the traffic density to maximize the rate of utilization of trains while fulfilling order on-time. The problem is then solved with a column generation method. Simultaneously, Holmberg et al. [6] wrote a similar model where the aim is to complete all incomplete freight trains with empty railcars. The

authors consider that the real cost is the one of under-using the maximum capacity of a train rather than the cost of moving a railcar (as the train will circulate anyway), their model is solved with CPLEX. Both approaches require to enter the train schedule and the capacity of trains as an input which is not wished by SNCF.

Later Joborn et al. [7] have designed a specific Tabu heuristic to solve large car assignment problems. In addition, the aim is not only to maximize the use of train capacity any more but also to minimize the technical interventions on railcars (separation or connection). To do so, they compute bigger blocks (a block is group of railcars not separated at intermediate yards) to make an economy-of-scale. However, suggesting a blocking strategy has not been identified as an objective by SNCF and time allocated to this research do not make the study of this path possible.

Despite all are deterministic, these models call attention to interesting ideas of empty railcar management.

Sherali and Suharko [20] have designed a stochastic model where uncertainty is considered differently than in Powell's publications: the pre-emptive priority multi-objective function is modified as an equivalent linear objective function and all objective criterions are included in the network arcs costs. The problem is then solved with a specifically designed heuristic. Although uncertainty on travel times is considered, the studied fleet was homogeneous. This model is in present-day usage by railroad companies in the USA.

Between 1998 and 2003, W.B. Powell produced many papers about the inclusion of uncertainty and substitution in models. First of all, Powell and Shapiro [15] suggested a representational paradigm to gather similar problems, that they called: "Dynamic Resource Transformation Problem" (DRiP), which includes the freight car management problem and all problems consisting in allocating resources to tasks. Powell and Topaloglu [17] applied

these notations to a generic DRiP considering substitutions and variability of travel times. Then, Powell, Shapiro and Simão [16] designed a heuristic method able to solve very large problems, then Topaloglu and Powell [22] fastened the resolution by approximating non-linear functions. However, all models assume that data are updated regularly and is available immediately to yard managers. Thus, previous decisions can be modified when facing unexpected events (shortage, lateness, etc). This is not a realistic assumption for common disruptions like delay. The frequent re-execution of the software and the correct use of its output would involve a training of users which is not desired by SNCF.

Finally, current optimization softwares available at SNCF provides efficient LP solvers enabling us to design Linear Program which can be deployed at any company's facility. This way of modeling the problem is therefore highly favored. Strobbe [21] a former intern at the SNCF, has done an interesting preliminary work on this issue. Strobbe designed a model where uncertainty on travel times is taken into consideration by minimizing the highest probability of delivering an order lately for each customer priority category. Nevertheless, some aspects, as substitution, were not yet identified and therefore not yet considered. Moreover, computing time is long on large problems.

## 2.4 General Approaches to Stochastic Model

Many articles dealing with scheduling issues assume that the environment is stable and perfectly known before hand. However, this static situation happens rarely in real-life. Data (such as demand, inventory and travel time) can be subject to uncertainty or unpredictable events can occur making the schedule impossible. In order to build models able to face uncertainty, many techniques have been developed. They are mainly applied to the job shop scheduling problem, but some of them can be definitely adapted to our problem. Davenport

and Beck [3] have surveyed and classified them as follows.

### 2.4.1 Proactive Scheduling

Proactive scheduling handles uncertainty a priori. Available information about uncertainty enables the schedule to be more robust. “[A robust model] is likely to remain valid under a wide variety of disturbances” [10]. Standard proactive techniques are mainly based on the concept of slack times. Assuming a task will disrupt with a given probability, idle time periods are inserted between tasks or at the end of the schedule. In railroad scheduling, proactive techniques can be easily applied to uncertain disjunctions which affect moderately the schedule (as delay on a travel times). They can be adapted to our problem using slack times to cover delays.

### 2.4.2 Reactive Scheduling

Contrary to proactive scheduling, reactive scheduling techniques manage uncertainty a posteriori. First, a schedule is generated without uncertainty considerations. This schedule can be either approximated (i.e. some decisions are to be made later) or comprehensive. An approximated schedule will further be refined in real time, while a comprehensive schedule will be modified as unpredicted disruptions occur. When a new schedule is generated, it is critical not to flood the users with changes. McKay et al. [11] show that changing erratically and too often a schedule involves an increase in cost due to the “shop floor nervousness”. The main drawback of reactive scheduling is its myopic behavior. Only the current failure is considered and not the potential next ones. Moreover, the computing time needed to regenerate a schedule can be prohibitive, preventing scheduler from suggesting other schedules on time. Finally, if disruptions occur frequently, it is difficult to generate many new schedules

without falling in the “shop floor nervousness” pitfall. This last aspect prevent us to apply this approach to our problem.

### 2.4.3 Proactive and Reactive Scheduling

This approach is a combination of both previous approaches. Uncertain but predictable events (e.g. uncertain job time) are managed thanks to proactive (off-line) scheduling while unpredictable events (e.g. breakdowns) involve the regeneration of a schedule (on-line scheduling). Generally speaking, the off-line step is based on a short-term schedule. It is assumed that an unpredictable event will occur so it is useless to generate a comprehensive schedule before getting certain data. In other cases, a long-term schedule is created, but with fewer details in order to be easily modified as information become available. Both methods fall in the scope of “Least Commitment and Delayed Commitment Scheduling”. However, when talking about freight transportation, commitments have to be scheduled early. This makes Proactive and Reactive techniques difficult to apply to the current SNCF’s organization.

### 2.4.4 Rescheduling

Rescheduling consists in generating a first feasible, comprehensive and optimal schedule. If it is disrupted, it will be repaired using fast and simple rules to minimize the perturbation. Contrary to reactive scheduling, this new schedule is not necessarily optimal. Iterative repairs are controversial; on one hand Zweben et al. [24] state proposed solutions are straightforward and easily understandable and applicable. On the other hand, Sadeh et al. [18] oppose that iterative repairs is a myopic approach and affirm that reparations of conflicts lead to other conflicts. Finally, Sakkout et al. [19] propose to reschedule by minimizing the perturbation

using Linear Programming and Constraint Programming. Very few detailed and applied works have been published about rescheduling techniques; the explanation can lie in the fact that they are an undoubtedly competitive advantage. In addition, they are mainly based on “job-constraints” very specific to the problem they are made for. Published literature provides interesting general approaches; they need to be completed to be applied to SNCF’s request though. In addition, the time allowed to carry out the project is not long enough to identify the repairing rules which are not straightforward.

## 2.5 Model Implementation

Powell et al. [14] emphasize the importance of taking into consideration the job constraints when modeling an optimization software. To be truly able to propose an optimal solution, a model has to be quick enough compared to the pace of the environment in which it will be used. Moreover, solutions proposed and changes on solutions should be easily understood by users. As a result, solutions proposed should be relatively stable after the process of an unpredicted event, even if this involves proposing an under-optimal solution. An optimal tool revolutionizing work habits will tend not to be used correctly and be a source of extra outcomes while a sub-optimal tool involving slighter changes will be more successfully integrated.

## 2.6 Motivation for Further Research

None of the works described above fits perfectly SNCF needs. SNCF specificities lie in the size of its fleet and the structure of its decision making process. The SNCF is a nationwide industry, where it takes time to communicate decisions to their addressee. Moreover, a

minimal coordination of agents is required before changing a decision that affects the whole traffic. Thus, SNCF can hardly implement an optimization tool which changes completely the strategy every single day or which assumes all decisions can be made instantly. Moreover, a potential optimization software should rather fit the current decision-making process in order to be easily implemented. This last aspect is obviously not taken in consideration in the published literature as each railroad company has its specific process.

This thesis consists then in designing an optimization model able to manage the uncertainty on travel times in order to suggest to schedulers the more profitable decisions while partitioning the set of orders in the priority categories showed in table 1.1.

Deterministic works cannot be applied to the SNCF's empty railcar management. Indeed, the random variation of travel times is currently identified as the major disturbance and cause of delay on delivery. On the other hand, stochastic models described in section 2.2 are not directly applicable to the SNCF's organization. Other models as [20] address homogeneous fleet and therefore solve a flow problem. However, these works can hardly be extended to problems related to heterogeneous fleets. Indeed, such problems could require multi-commodity flow models which are far more complex to solve. In addition, when heterogeneous fleets are considered, substitution possibilities are rarely identified as a mean of countering uncertainty (and consequently shortage) and are therefore not considered. In Powell's publications [17] and [15], substitution policy is implemented, however his dynamic models involve accessing all available data and making operational decisions. Such a degree of accuracy is not requested and may be beyond the capability of existing information systems. Similarly [6], [7] and [9] assume too much precise input regarding the decisions granularity wished by SNCF. Finally, the division of the set of orders used at SNCF is never addressed and makes this problem unique even if similarities can be found with other models.

# Chapter 3

## Empty Railcar Management

The empty railcar management consists in allocating railcars to unfulfilled orders. Available railcars at a delivery terminal are allocated to orders every period and missing railcars are then moved from other terminals to be allocated. The model we intend will have to supply user with this output. In addition, some allocations can exist prior to the model execution since current information systems allow user to allocate railcars that are not yet available at a terminal. This must be considered because it can lead to a negative inventory which must be filled.

### 3.1 Terminology

Railcars are characterized by different status. A railcar is qualified as:

- **in-transit** if it is moving between two terminals (even if it is stopped and waiting at an intermediate terminal)
- **present** if it is physically at a terminal **and** not in-transit

- **allocated** if it is present at a terminal and booked to fulfill an order
- **available** if it is present and not yet allocated to an order

It is reminded that an order is:

- **committing** if an agreement has been signed between SNCF and the customer pledging that railcars will be delivered on time. Usually, such orders are those submitted more than 6 days before the delivery date
- **not-committing** if there is not such an agreement (*e.g.* late orders)

## 3.2 Notation

### 3.2.1 Sets

Let:

- $\mathcal{H}$  be the time horizon composed of several time periods:  $\mathcal{H} = \{d^0, d^1, \dots, d^f\}$
- $\mathcal{I}$  be the set of empty railcar orders
- $\mathcal{T}$  be the set of terminals
- $\mathcal{C}$  be the set of railcar categories
- $\mathcal{U}$  be the set of customers
- $\forall u \in \mathcal{U}, \mathcal{R}_{sub}(u) \subset \mathcal{C} \times \mathcal{C}$  be the substitution referential of customer  $u$ . It is a set of category couples each couple  $(c_1, c_2) \in \mathcal{C} \times \mathcal{C}$  indicating that railcars of category  $c_1$  can be substituted by railcars of category  $c_2$  for customer  $u$ .

### 3.2.2 Orders

$\forall i \in \mathcal{I}$ , a railcar order is defined by the following attributes:

- a due-date  $d_i \in \mathcal{H}$  for empty railcars delivery
- a customer  $u_i \in \mathcal{U}$
- a delivery terminal  $t_i \in \mathcal{T}$
- a requested category  $c_i \in \mathcal{C}$
- an amount of ordered railcars  $n_i \in \mathbb{N}$
- a cost of delay  $C_{delay}(i)$  which the penalty per missing railcar and per period applied between the due-date  $d_i$  and the actual delivery date
- an unfulfillment cost  $C_{uff}(i)$  per missing railcar for the fulfillment of order  $i$  at  $d^f$

We also consider the following simplifying notations:

- $\mathcal{I}(t) \subset \mathcal{I}$  is the set of orders that must be delivered at terminal  $t$ :  

$$\mathcal{I}(t) = \{i \in \mathcal{I} | t_i = t\}$$
- $\mathcal{I}(t, c) \subset \mathcal{I}$  is the set of orders of railcars of category  $c \in \mathcal{C}$  that must be delivered at terminal  $t$ :  $\mathcal{I}(t, c) = \{i \in \mathcal{I} | t_i = t, c_i = c\}$
- $\mathcal{R}_{sub}(u, c)$  is defined as  $\mathcal{R}_{sub}(u, c) = \{c' \in \mathcal{C} | (c, c') \in \mathcal{R}_{sub}(u)\}$ ,  $\forall u \in \mathcal{U}$  and for  $c \in \mathcal{C}$ .

### 3.2.3 Information Extracted from Information Systems

Input data are extracted from several Information Systems (IS),:

- TIF which records allocations of railcars made at every terminals,
- BIG which supplies the quantity of empty railcars present at a terminal
- ATHEOS which supplies data about traveling times.

These data are then noted as follows:

- $Q^{TIF}(i, d)$  is the amount of railcars of category  $c_i$  recorded in TIF as allocated to the order  $i \in \mathcal{I}$  on day  $d \in \mathcal{H}$  at terminal  $t_i$ .
- $Q_{sub}^{TIF}(i, c, d)$  is the amount of railcars of category  $c \in \mathcal{R}_{sub}(u_i, c_i)$  recorded in TIF as allocated to the order  $i \in \mathcal{I}$  on day  $d \in \mathcal{H}$  at terminal  $t_i$  as a substitute to railcars of category  $c_i$
- $N^{BIG}(t, c)$  is the amount of cars of category  $c \in \mathcal{C}$  **present** at terminal  $t \in \mathcal{T}$  on period  $d^0$  according to BIG
- $N_{arr}^{BIG}(t, c, d)$  is the amount of cars of category  $c \in \mathcal{C}$  **in-transit** expected to arrive at terminal  $t \in \mathcal{T}$  at period  $d \in \mathcal{H}$  according to BIG
- $N_{ret}^{BIG}(t, c, d)$  is the amount of cars of category  $c \in \mathcal{C}$  returned by customers at terminal  $t \in \mathcal{T}$  on day  $d \in \mathcal{H}$  according to BIG
- $\Delta T(t, t') \in \mathbb{N}^*$  is the traveling time (in periods) from terminal  $t \in \mathcal{T}$  to  $t' \in \mathcal{T}$  extracted from ATHEOS.

Following inputs are computed from IS data:

- $inventory(t, c, d)$  is the balance of the category  $c \in \mathcal{C}$  railcars at terminal  $t \in \mathcal{T}$  on day

$d \in \mathcal{H}$  due to decisions made before the execution of the model.

$$\begin{aligned} inventory(t, c, d^0) = & N_{arr}^{BIG}(t, c, d^0) + N_{ret}^{BIG}(t, c, d^0) + N^{BIG}(t, c) \\ & - \sum_{i \in \mathcal{I}(t, c)} Q^{TIF}(i, d^0) - \sum_{i \in \mathcal{I}(t)} Q_{sub}^{TIF}(i, c, d^0) \end{aligned} \quad (3.1)$$

and  $\forall d > d^0$ ,

$$\begin{aligned} inventory(t, c, d) = & N_{arr}^{BIG}(t, c, d) + N_{ret}^{BIG}(t, c, d) \\ & - \sum_{i \in \mathcal{I}(t, c)} Q^{TIF}(i, d) - \sum_{\substack{i \in \mathcal{I}(t) \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}^{TIF}(i, c, d) \end{aligned} \quad (3.2)$$

- $prevAllocations(i)$  (stands for *previous allocations*) is the amount of railcars allocated to order  $i \in \mathcal{I}$  previously to the model execution,

$$prevAllocations(i) = \sum_{d \in \mathcal{H}} \left( Q^{TIF}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}^{TIF}(i, c, d) \right) \quad (3.3)$$

### 3.2.4 Costs

In addition to  $C_{uff}$  and  $C_{delay}$  (see section 3.2.2), following costs are also considered in the model:

- $C_{move}(t, t', c)$  is the cost for moving one category  $c \in \mathcal{C}$  railcar from terminal  $t \in \mathcal{T}$  to terminal  $t' \in \mathcal{T}$
- $C_{sub}(c, c', u_i)$  is the cost for substituting one category  $c \in \mathcal{C}$  railcar by one category  $c' \in \mathcal{R}_{sub}(u_i, c_i)$  railcar to fulfill an order  $i \in \mathcal{I}$  emitted by customer  $u_i \in \mathcal{U}$

The goals of the optimization model consist in providing a strategy to fulfill orders and to reposition railcars. Holding costs are therefore not relevant to our problem and will not be considered in the deterministic case.

### 3.2.5 Parameters

$W_{move}$ ,  $W_{uff}$ ,  $W_{sub}$ ,  $W_{delay}$  are the weights of the four criteria of the objective function; respectively, the total cost of moves, the sum of unfulfillment costs, the sum of substitution costs and the sum of delay costs. These weights are initially set to 1. Nevertheless, they can be modified by the user if a criteria must be reinforced with regards to the others.

## 3.3 Linear Program

This problem is represented by a linear programming model named  $LP_{glop}$ . It will be first introduced without considering the industrial decision-making process. Moreover, in this part, only the deterministic linear program will be addressed; stochastic refinements will be presented in chapter 4.

Remark : Some indexes are therefore useless in the deterministic model, they only stand to prepare the stochastic case. For example, for  $N_{tr}(t, i, c, d, d')$  the fifth index  $d'$  indicates the arrival date of a move though it depends only on the other parameters in the deterministic case ( $d' = d + \Delta T(t, t_i)$ ). Moreover, allocation variables  $Q_{aff}$  and  $Q_{sub}$  are indexed with the day of allocation even if only  $d \geq d_i$  is currently permitted.

### 3.3.1 Decision Variables

Decision variables can be split into two categories; the “primary” ones which reflect the optimal decisions and the “secondary” ones which are computed as a function of primary ones.

#### Primary Decision Variables

It is recalled that the allocation date  $d$  of railcars to an order  $i$  is assumed not to be lower than the due-date  $d_i$  of this order (see section 1.3).

- $Q_{all}(i, d)$ ,  $d^0 \leq d \leq d^f$  is the amount of category  $c_i$  railcars allocated to order  $i \in \mathcal{I}$  (at terminal  $t_i$ ) on day  $d \in \mathcal{H}$ ,  $d \geq d_i$ ; these railcars are clearly expected to be available at  $t_i$  on day  $d$ .
- $Q_{sub}(i, c, d)$ , is the amount of category  $c \in \mathcal{R}_{sub}(u_i, c_i)$  railcars allocated to order  $i \in \mathcal{I}$  on day  $d \in \mathcal{H}$ ,  $d \geq d_i$  in substitution of category  $c_i$  railcars.
- $N_{tr}(t, i, c, d, d')$  is the amount of category  $c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\}$  railcars leaving from terminal  $t \in \mathcal{T}$  on day  $d \in \mathcal{H}$  to arrive at terminal  $t_i \in \mathcal{T}$  to deliver order  $i \in \mathcal{I}$  on day  $d' \in \mathcal{H}$  with  $d \neq d'$ ,  $d' \geq d_i$  and  $t \neq t_i$ . The second index refers to  $i \in \mathcal{I}$  because this information will be useful to estimate the delay penalty in the stochastic model (described in section 4.1).

### 3.3.2 Secondary Decision Variables

- $N_{av}(t, c, d)$  is the quantity of category  $c \in \mathcal{C}$  railcars available at terminal  $t \in \mathcal{T}$  at the end of period  $d \in \mathcal{H}$

- $allocation(i)$  is the total amount of railcars allocated to the order  $i \in \mathcal{I}$  including substitutions and IS data

### 3.3.3 Objective Function

The objective function is made of four parts:

#### Total Cost of Moves

$$TCM = \sum_{t \in \mathcal{T}} \sum_{\substack{i \in \mathcal{I} \\ t_i \neq t}} \sum_{c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\}} \sum_{\substack{d \in \mathcal{H} \\ d + \Delta T(t, t_i) \geq d_i}} N_{tr}(t, t_i, c, d, d + \Delta T(t, t_i)) \times C_{move}(t, t_i, c) \quad (3.4)$$

#### Total Unfulfillment Cost

$$TUC = \sum_{i \in \mathcal{I}} (n_i - allocation(i)) \times C_{uff}(i) \quad (3.5)$$

#### Total Substitution Cost

$$TSC = \sum_{i \in \mathcal{I}} \sum_{\substack{d \in \mathcal{H} \\ d > d_i}} \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \times C_{sub}(c_i, c, u_i) \quad (3.6)$$

#### Total Cost of Delay

$$TCD = \sum_{i \in \mathcal{I}} C_{delay}(i) \times \sum_{\substack{d \in \mathcal{H} \\ d > d_i}} \left( \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right) \times (d - d_i) \right) \quad (3.7)$$

The objective function is then:

$$\min z = W_{move} \cdot TCM + W_{uff} \cdot TUC + W_{sub} \cdot TSC + W_{delay} \cdot TCD \quad (3.8)$$

### 3.3.4 Constraints

#### State Equation at $d^0$

The first constraint corresponds to the balance of the flows of:

- allocations and starting inventory collected in  $inventory(t, c, d^0)$ ,
- leaving railcars represented by  $N_{tr}$ ,  $Q_{all}$ ,  $Q_{sub}$ .

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C},$$

$$\begin{aligned} N_{av}(t, c, d^0) = & inventory(t, c, d^0) \\ & - \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t}} N_{tr}(t, i, c, d^0, d^0 + \Delta T(t, t_i)) \\ & - \sum_{\substack{i \in \mathcal{I}(t, c) \\ d^0 \geq d_i}} Q_{all}(i, d^0) - \sum_{\substack{i \in \mathcal{I}(t) \\ d^0 \geq d_i \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}(i, c, d^0) \end{aligned} \quad (3.9)$$

As traveling times are not equal to zero, no incoming moves are admitted at  $d^0$ .

**State equation at  $d^0 < d \leq d^f$** 

The second constraint is similar to the first one but concerns the next periods. Incoming moves are allowed (see third term in the left hand-side)

$$\begin{aligned}
& \forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H} \setminus \{d^0\}, \\
& N_{av}(t, c, d) = N_{av}(t, c, d - 1) + \text{inventory}(t, c, d) \\
& \quad + \sum_{\substack{i \in \mathcal{I}(t) \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t \neq t' \\ d - \Delta T(t', t_i) \geq d^0}} N_{tr}(t', i, c, d - \Delta T(t', t_i), d) \\
& \quad - \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t \\ d + \Delta T(t, t_i) \leq d^f}} N_{tr}(t, i, c, d, d + \Delta T(t, t_i)) \\
& \quad - \sum_{\substack{i \in \mathcal{I}(t, c) \\ d \geq d_i}} Q_{all}(i, d) - \sum_{\substack{i \in \mathcal{I}(t) \\ d \geq d_i \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}(i, c, d)
\end{aligned} \tag{3.10}$$

**Demand Fulfillment**

The variable *allocation* is computed as follows:

$$\begin{aligned}
& \forall i \in \mathcal{I}, \\
& \text{allocation}(i) = \text{prevAllocations}(i) + \sum_{d \in \mathcal{H}} \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right)
\end{aligned} \tag{3.11}$$

Then the quantity of railcars allocated cannot be greater than the demand

$$\begin{aligned} \forall i \in \mathcal{I}, \\ n_i &\geq \text{allocation}(i) \end{aligned} \quad (3.12)$$

### Allocation After Moving

A move of railcars is made from a departure terminal to the delivery terminal of an order so that **this** order can be fulfilled. The following constraints ensure that moved railcars are effectively allocated to the underlying order. This constraint stands with “ $\geq$ ” sign because allocations of already present railcars can be added to moved railcars. Moreover, as no holding costs are considered and allocations are delayed as much as possible, moves do not have to arrive before  $d_i$ .

$$\begin{aligned} \forall i \in \mathcal{I}, d \geq d_i \\ Q_{all}(i, d) &\geq \sum_{\substack{t \in \mathcal{T} \\ d - \Delta T(t, t_i) \geq d^0}} N_{tr}(t, i, c_i, d - \Delta T(t, t_i), d) \end{aligned} \quad (3.13)$$

$$\begin{aligned} \forall i \in \mathcal{I}, d \geq d_i, \forall c \in \mathcal{R}_{sub}(u_i, c_i), \\ Q_{sub}(i, c, d) &\geq \sum_{\substack{t \in \mathcal{T} \\ d - \Delta T(t, t_i) \geq d^0}} N_{tr}(t, i, c, d - \Delta T(t, t_i), d) \end{aligned} \quad (3.14)$$

### 3.3.5 Network Structure and Integrality Constraints

The network structure of the problem enable us not to require variable to be explicitly set as integer. Indeed, the graph showed on figure 3.1 (see page 31) depicts the network structure of the one-category problem. To solve the multi-category problem, it can be reproduced for each category and the set of graphs can be organized in layers (see figure 3.2,p. 32 ). Substitution cases are then handled drawing an arc from a layer to another one; two kinds of arcs can be represented:

- From a terminal-node to an order-node to handle classical substitution,
- From the main source node to an order node to handle TIF substitution.

The main source and sink nodes depicted in figure 3.1 are then unique and the same for all layers. The problem is therefore modeled as a mono-commodity flow problem subject to capacity and state constraints. The Integer Program is therefore equivalent to the Linear Program.

All decision variables are therefore set as non-negative real numbers:

$$\begin{aligned} \forall c \in \mathcal{C}, \forall d \in \mathcal{H}, \forall t \in \mathcal{T}, \forall i \in \mathcal{I} \text{ with } t \neq t_i \text{ and } d + \Delta T(t, t_i) \leq d^f \\ N_{tr}(t, i, c, d, d + \Delta T(t, t_i)) \in \mathbb{R}^+ \end{aligned} \quad (3.15)$$

$$\forall i \in \mathcal{I}, d \in \mathcal{H}, d \geq d_i, Q_{all}(i, d) \in \mathbb{R}^+ \quad (3.16)$$

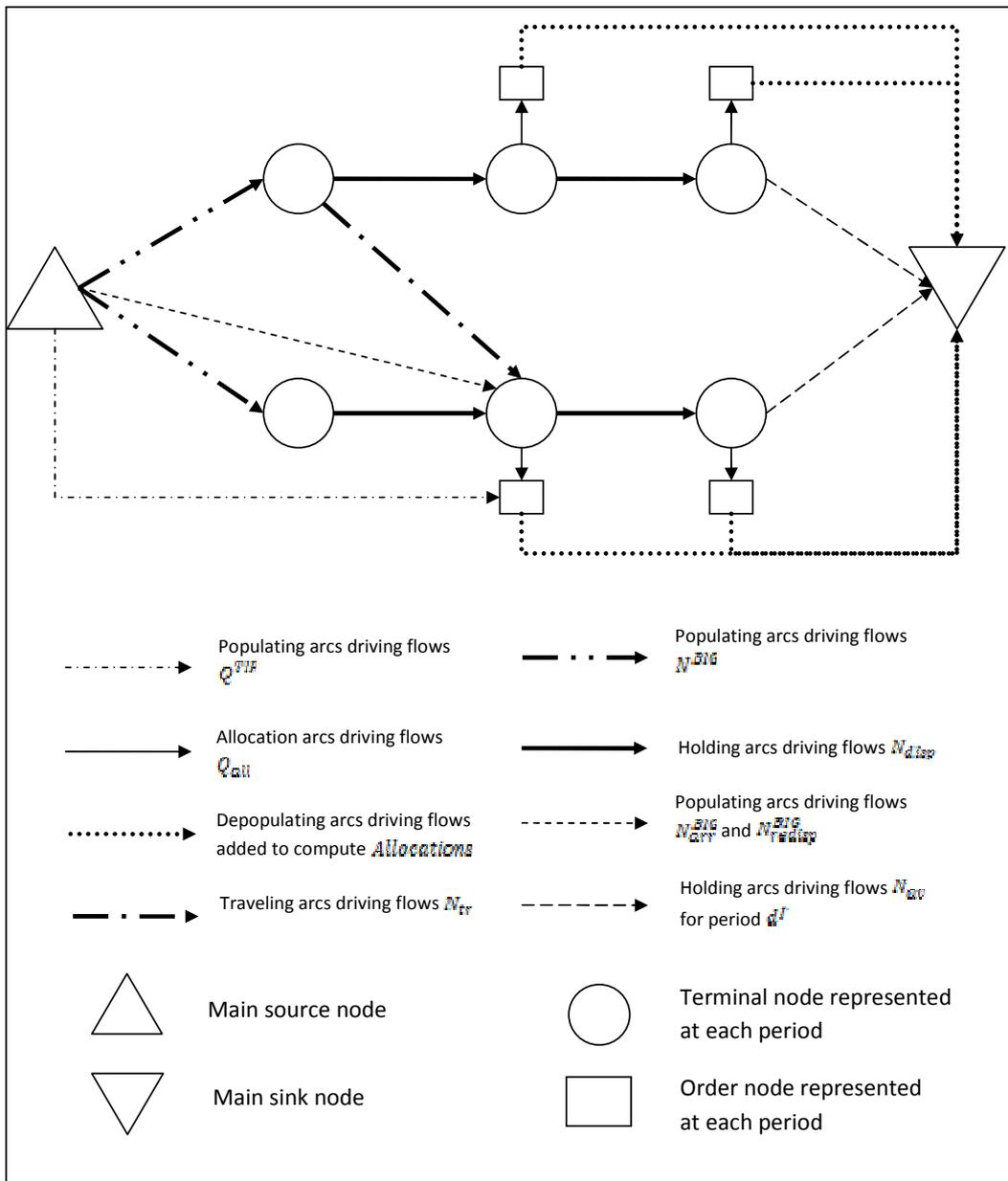


Figure 3.1: Network Structure

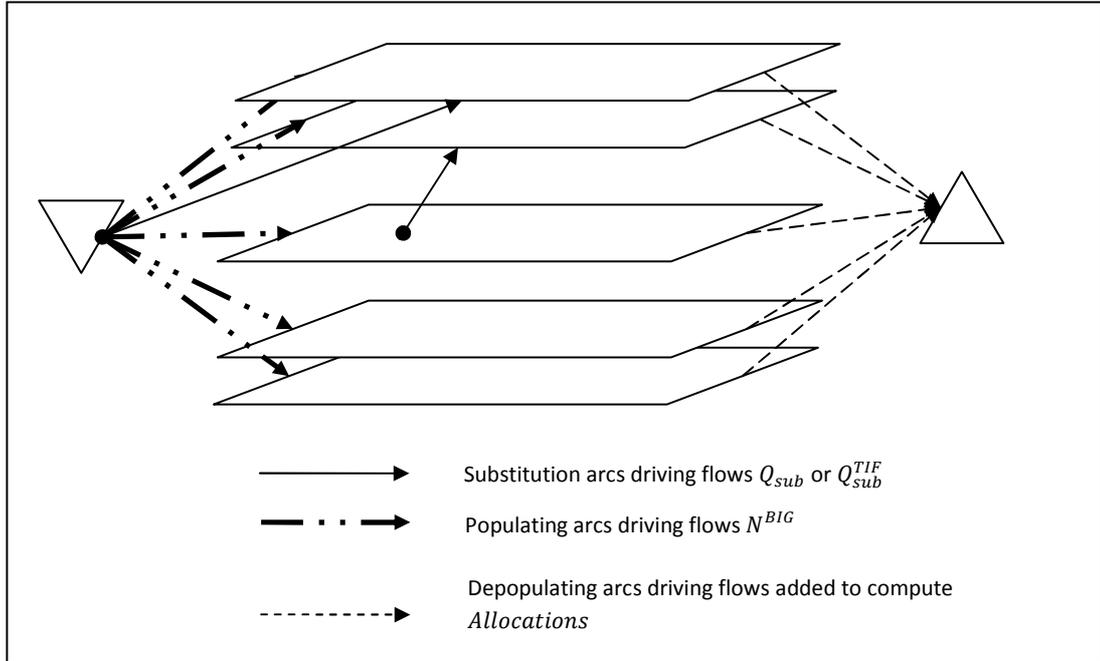


Figure 3.2: Layers Structure

$$\forall i \in \mathcal{I}, \forall c \in \mathcal{R}_{sub}(u_i, c_i), d \in \mathcal{H}, d \geq d_i, Q_{sub}(i, c, d) \in \mathbb{R}^+ \quad (3.17)$$

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H}, N_{av}(t, c, d) \in \mathbb{R}^+ \quad (3.18)$$

$$\forall i \in \mathcal{I}, allocation(i) \in \mathbb{R}^+ \quad (3.19)$$

## 3.4 Decision Making Process

### 3.4.1 Description

In order to capture some organizational constraints that are targeted by the SNCF freight activity, the decision making process is sequenced in 3 steps (see also table 1.1 and figure 1.1).

Let therefore :

- $I_E(d) \subset \mathcal{I}$  be the subset of committing orders due on day  $d \in \mathcal{H}$
- $I_{\bar{E}}(d) \subset \mathcal{I}$  be the subset of not-committing orders due on day  $d \in \mathcal{H}$
- $I_E = \bigcup_{d \in \mathcal{H}} I_E(d)$
- $I_{\bar{E}} = \bigcup_{d \in \mathcal{H}} I_{\bar{E}}(d)$

We note that

$$\bigcap_{d \in \mathcal{H}} I_E(d) = \bigcap_{d \in \mathcal{H}} I_{\bar{E}}(d) = I_E \cap I_{\bar{E}} = \emptyset$$

and

$$I_E \cup I_{\bar{E}} = \mathcal{I}$$

Finally,  $I_I, I_{II}$  and  $I_{III}$  are the orders considered respectively at step 1, 2 and 3.  $I_1, I_2$  and  $I_3$  are the residual order demand after processing of steps 1, 2 and 3 are processed. Thus the description of each step can be refined, as shown in table 3.1.

$IP_{glob}$  can be executed three times independently through these three steps. This can be done by transmitting decisions made from one step to the next one using the same pattern as with IS data. The variable *inventory* can therefore be defined as follows.

- $\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H}$  *inventory*( $t, c, d$ ) is the total balance of category  $c$  railcars incoming, leaving and allocated to an order at terminal  $t$  on day  $d$ .

Then let rename  $IP_{glob}$  by  $IP_{glob}(\alpha, \beta)$  where parameter  $\alpha$  is the set of orders processed and parameter  $\beta$  is the horizon time considered.

The complete procedure to solve the empty railcar management problem according to the 3-step decision making process is presented below.

#### Initialization

1. Extract initial demands  $n_i$  and TIF and BIG data
2. Initialize *inventory* according to equation (3.1) and (3.2)
3. Initialize *prevAllocation* with equation (3.3)

Table 3.1: Subsets of Decision-Making Process

step	Description	Orders considered	Residual Orders
step 1	committing orders due on day $d^0$	$I_I = I_E(d^0)$	$I_1$
step 2	committing orders due on day $d > d^0$	$I_{II} = I_E(\mathcal{H} \setminus \{d^0\}) \cup I_1$	$I_2$
step 3	not-committing orders	$I_{III} = I_{\bar{E}} \cup I_2$	$I_3$

**Step 1: Committing Orders Due on Day  $d^0$** 

1. Optimize  $IP_{glob}(I_I, \{d^0\})$

2. Update of *inventory* variables

$$\forall c \in \mathcal{C}, \forall t \in \mathcal{T},$$

$$inventory(t, c, d) \leftarrow - \sum_{i \in \mathcal{I}(t, c) \cap I_I} \left( Q_{all}(i, d^0) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d^0) \right)$$

3. Update of previous Allocations variables

$$\forall i \in I_I$$

$$prevAllocations(i) \leftarrow prevAllocation(i) + Q_{all}(i, d^0) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d^0)$$

**Step 2: Committing Orders Due on Day  $d > d^0$** 

1. Optimize  $IP_{glob}(I_{II}, \mathcal{H})$
2. Update of *inventory* variables

$$\begin{aligned}
& \forall c \in \mathcal{C}, \forall t \in \mathcal{T}, \forall d \in \mathcal{H}, \\
& \text{inventory}(t, c, d) \leftarrow \text{inventory}(t, c, d) \\
& \quad - \sum_{i \in \mathcal{I}(t, c) \cap I_{II}} \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right) \\
& \quad - \sum_{d' \in \mathcal{H}} \sum_{\substack{i \in \mathcal{I} \cap I_{II} \\ t_i \neq t}} N_{tr}(t, i, c, d, d') \\
& \quad + \sum_{d' \in \mathcal{H}} \sum_{i \in \mathcal{I}(t) \cap I_{II}} \sum_{\substack{t' \in \mathcal{T} \\ t' \neq t_i}} N_{tr}(t', i, c, d', d)
\end{aligned} \tag{3.20}$$

3. Update of previous Allocations variables

$$\begin{aligned}
& \forall i \in I_{II}, \forall d \in \mathcal{H} \\
& \text{prevAllocations}(i) \leftarrow \text{prevAllocations}(i) + Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d)
\end{aligned}$$

**Step 3: Not-Committing Orders**

1. Optimize  $IP_{glob}(I_{III}, \mathcal{H})$
2. Update of *inventory* variables

$$\forall c \in \mathcal{C}, \forall t \in \mathcal{T}, \forall d \in \mathcal{H},$$

$$inventory(t, c, d) \leftarrow inventory(t, c, d)$$

$$\begin{aligned} & - \sum_{i \in \mathcal{I}(t, c) \cap I_{III}} \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right) \\ & - \sum_{d' \in \mathcal{H}} \sum_{\substack{i \in \mathcal{I} \cap I_{III} \\ t_i \neq t}} N_{tr}(t, i, c, d, d') \\ & + \sum_{d' \in \mathcal{H}} \sum_{\substack{i \in \mathcal{I} \cap I_{III} \\ t_i = t}} \sum_{\substack{t' \in \mathcal{T} \\ t' \neq t_i}} N_{tr}(t', i, c, d', d) \end{aligned} \quad (3.21)$$

3. Update of previous Allocations variables

$$\forall i \in I_{III}, \forall d \in \mathcal{H} \quad (3.22)$$

$$prevAllocations(i) \leftarrow prevAllocations(i) - Q_{all}(i, d) - \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \quad (3.23)$$

**Final Step**

1. Edit allocations schedule
2. Edit unfulfilled orders

### 3.4.2 About the Myopic Structure of the Model

Once the optimization tool is executed, the scheduler enters the allocations and moves in the information system making the necessary changes (see figure 1.1 on page 6). At the next execution of the tool, which probably occurs one week later or sooner in case of major changes in processed orders, the optimization tool extracts the up-to-date data from the information system. These new data are the new input of the tool, they contain the real course of events (which can differ from the expected one because of order cancellations, delay in the processing of orders, etc.) and the decisions previously entered in the information system. This behavior underlies a myopic strategy; indeed, future orders cannot have a consequence on already scheduled decisions. The main shortcoming of such a strategy occurs when there is no more resource to process a last-minute high priority order; in this case, the tool cannot suggest a correction to the already entered strategy.

Such last minute changes involve considering many variables as the reactivity of services affected, the technical possibility of making the changes, the costs of these changes, etc. These data are not explicitly available and only known through the experience of the scheduler. That is why, it has been accepted that the opportunity to change or not the strategy and the identification of such changes has to be made by the scheduler himself. He should then enter opportune decisions in the information system and make the necessary changes manually.

## 3.5 Feasibility

The problem described previously allows delays and order unfulfillments. It is therefore always feasible, except if the allocation made by TIF cannot be processed. For example, assuming the number of available railcars at a terminal is lower than the number of railcars

allocated by TIF, railcars have then to be moved to this terminal. If there is no terminal close enough to bring railcars before the date of the TIF allocation, the problem is infeasible. At the prototype step, this case should not happen ; however, if it does, the relaxation tool of CPLEX enables identifying the infeasible constraint and to correct the problem.

# Chapter 4

## Coping with Uncertainty on Travel Times

In order to make the optimized production plan more robust and reliable, the uncertainty on travel times is considered and modelled in a first phase. A *robustness cursor* is then introduced to enable the user to choose the level of robustness.

### 4.1 Anticipating Delay On Travel Times

A natural approach to anticipate deviation between theoretical and actual travel times consists in adding slack time to the theoretical time (see 2.4.1). As a consequence, if an arrival is delayed, this slack will act as a buffer preventing as much as possible the allocation from being delayed. The real issue is then to choose slacks achieving a trade off between the risk of delay (slacks too short), and the holding costs (slacks too long).

### 4.1.1 Overview of the Proposed Approach

To model the different possible slacks, each moving arc of the deterministic model is replicated in several instances (moving arcs) and each instance is associated to a slack time called *anticipated delay* and denoted by  $\sigma_a$ . Figure 4.1 shows the four replications of a moving arc when the maximum slack time is equal to 3 periods. The decision making model is then expected to choose a single instance for each move, corresponding to an anticipated delay (slack) for this move. However, the quality of this choice should be guided by a measure of the risk associated to each instance. A cost associated to each replicated arc is then computed as the expected value of the cost variation between the case corresponding to the anticipated delay and the cases corresponding to other possible actual delays (denoted by  $\sigma_r$  — see figure 4.2).

### 4.1.2 Notations

Additional notations used in this section are noted as follows:

- $\sigma_a \geq 0$  is the delay anticipated for a move  $\varphi$
- $\sigma_r \geq 0$  is the real delay, in periods, for a move  $\varphi$
- $D_{max} \in \mathbb{N}$  is the maximum admissible delay (in periods) and, therefore, the upper bound of  $\sigma_a$  and  $\sigma_r$ . Previous work by Strobbe [21] and Sherali and Suharko [20] shows that  $D_{max}$  can be chosen as equal to 3 days.

To simplify further notations, let us define the set of possible anticipated moves by:

$$\Phi = \left\{ \varphi = (t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \in \mathcal{T} \times \mathcal{I} \times \mathcal{C} \times \mathcal{H} \times \mathcal{H} \right. \\ \left. | t \neq t_i, c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\}, 0 \leq \sigma_a \leq D_{max}, d^f \geq d + \Delta T(t, t_i) + \sigma_a \geq d_i \right\} \quad (4.1)$$

Each element of the set  $\Phi$  corresponds to a moving arc and  $\sigma_a$  is the *anticipated delay* for this moving arc. The optimization software will select the move (among those in  $\{(t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \in \Phi\}_{0 \leq \sigma_a \leq D_{max}}$ ) corresponding to the best  $\sigma_a$  value so that the anticipated delay is included *a priori* when making the schedule. Note that for orders due on  $d^f$  no delay is allowed, this “boundary effect” is corrected when the model is re-executed at a later period (and therefore with a different time horizon).

Let us then define:

- $\sigma_{lim}(\varphi) \in \mathbb{Z}$  as the maximum delay available in move  $\varphi$  in order to keep an on-time delivery for order  $i$ . A greater delay means that railcars will reach the arrival terminal after the due-date of the targeted order.

$$\forall \varphi = (t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \in \Phi, \sigma_{lim}(\varphi) = d_i - d - \Delta T(t, t_i) \quad (4.2)$$

- $p(\sigma_r, t, t')$  as the probability of having a delay of  $\sigma_r \in [0, D_{max}]$  for a move from  $t \in \mathcal{T}$  to  $t' \in \mathcal{T} \setminus \{t\}$
- $s(\varphi, \sigma_r)$  as the scenario where, given a move  $\varphi = (t, i, c, d, d + \Delta T(t, t_i) + \sigma_a)$ , a delay of  $\sigma_r$  periods occurs while a delay of  $\sigma_a$  periods had been anticipated.
- $C_\Delta(s(\varphi, \sigma_r))$  as the extra cost involved by scenario  $s(\varphi, \sigma_r)$  compared to the deterministic case (where  $\sigma_r = \sigma_a$ )
- $C_a(\varphi)$  as the expected cost to be added to the moving cost of the deterministic case

No holding costs were considered in the deterministic model (see section 3.2.4); however, when arrival dates are subject to uncertainty, it becomes necessary to penalize early arrival dates. Indeed, without such a penalty, the decision model would introduce a large delay in

every moves of railcars. Railcars that arrived earlier than expected (*i.e.* on-time deliveries or deliveries arriving with a delay lower than the maximum delay allowed in the model) would be set aside. This would result in a decrease of the turnover rate of empty railcars, which is opposite to the goals of the model. That is why an unexpected holding cost is introduced:

- $C_{hold}(t, c)$  is the cost generated when holding one category  $c \in \mathcal{C}$  railcar at terminal  $t \in \mathcal{T}$  without scheduling it.

### 4.1.3 Anticipated Delay

Let us review the example depicted in figure 4.1. Railcars of category  $c$  have to be transported from terminal  $t$  to fulfill order  $i$ ; the due-date is  $d_i = 6$  and the theoretical travel time is  $\Delta T(t, t_i) = 2$ . Depending on the anticipated delay, different anticipated moves exist:  $\varphi = (t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \in \Phi$ , for  $\sigma_a \in \{0, \dots, D_{max} = 3\}$ .

Figure 4.2 continues the example depicted in figure 4.1. It illustrates the different possible scenarios  $s(\varphi, \sigma_r)$  when the value of  $\sigma_a$  is equal to 2. If the real delay is  $\sigma_r = \sigma_a = 2$ , then there is no extra cost. However, an additional cost  $C_\Delta$  has to be considered in every other case:

- $2 \times C_{hold}$  if  $\sigma_r = 0$  (railcars are hold for 2 time periods).
- $1 \times C_{hold}$  if  $\sigma_r = 1$  (railcars are hold for 1 time period).
- $1 \times C_{delay}$  if  $\sigma_r = 3$  (railcars are 1 time period late).

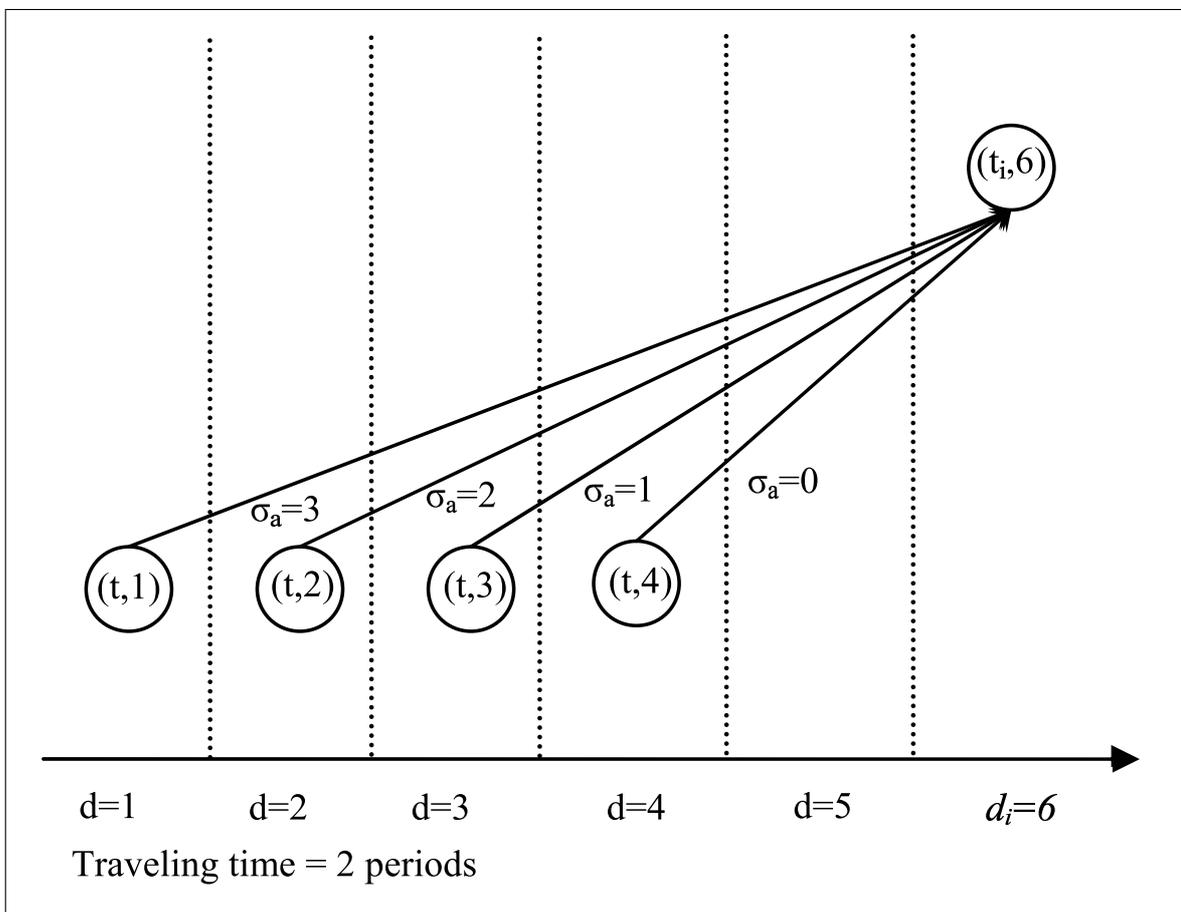


Figure 4.1: Different Anticipated Moves

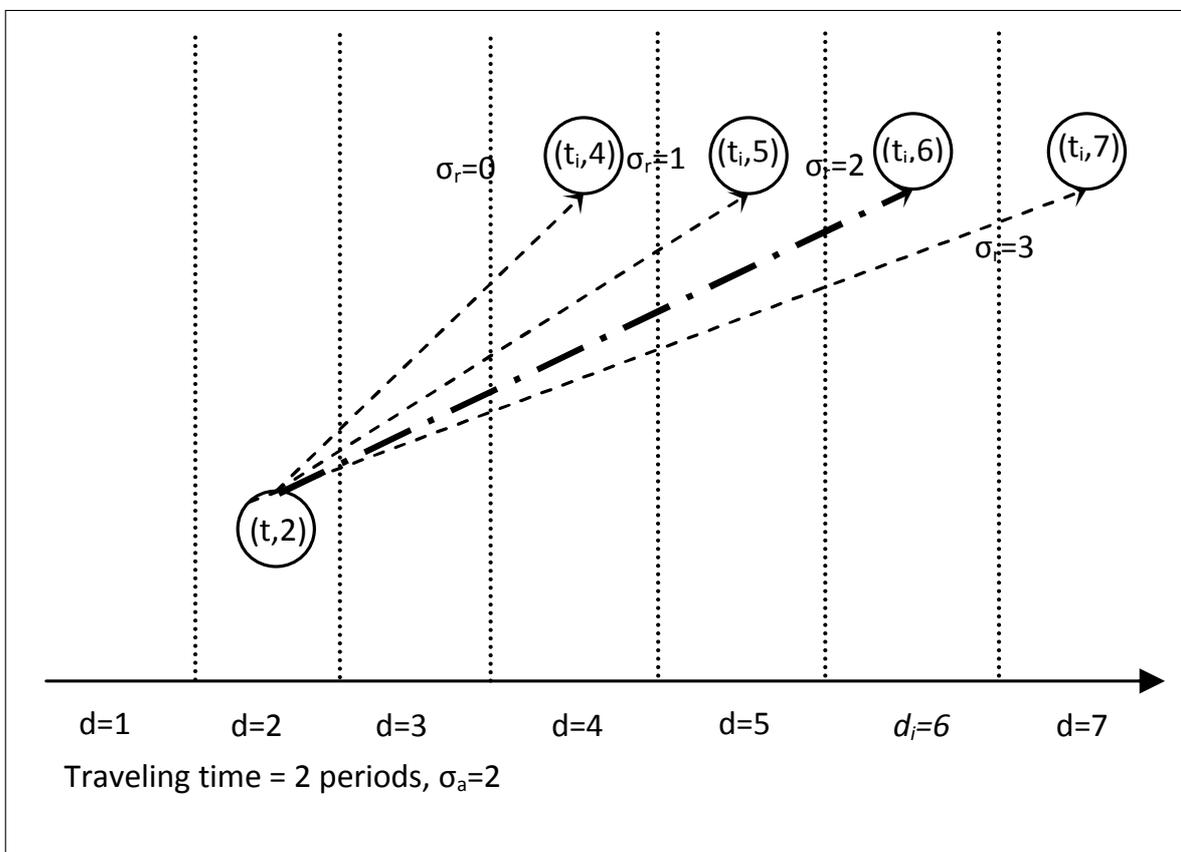


Figure 4.2: Scenarios  $s(\varphi, \sigma_r)$  for  $\varphi = (t, i, c, d = 2, d + \Delta T(t, t_i) + \sigma_a = 6)$  and risk measure for instance  $\varphi$

The cost of an anticipated move  $\varphi = (t, i, c, d, d + \sigma_a + \Delta T(t, t_i))$  from a terminal  $t$  for a customer  $i$  can then be evaluated as its expected value over the potential scenarios  $\{s(\varphi, \sigma_r)\}_{0 \leq \sigma_r \leq D_{max}}$ , that is:

$$C_a(\varphi) = \sum_{\sigma_r=0}^{D_{max}} C_{\Delta}(\varphi, \sigma_r) \times p(\sigma_r, t, t_i) \quad (4.3)$$

(where, as defined before,  $p(\sigma_r, t, t_i)$  is the probability of having a delay of  $\sigma_r$  for a move from  $t$  to  $t_i$ ).

Table 4.1 illustrates the values of the cost  $C_{\Delta}(\varphi, \sigma_r)$  when  $\sigma_a = \sigma_{lim}(\varphi)$  (*i.e.* the anticipated delivery date is the due-date of the targeted order).

Table 4.1: Computing  $C_{\Delta}(s(\varphi, \sigma_r))$  when  $\sigma_a = \sigma_{lim}(\varphi)$

scenario	$\sigma_r$	$p(\sigma_r)$	$\sigma_a = 0$	$\sigma_a = 1$	$\sigma_a = 2$	$\sigma_a = 3$
on-time	0	$p_0$	0	$1 \cdot C_{hold}$	$2 \cdot C_{hold}$	$3 \cdot C_{hold}$
1-day late	1	$p_1$	$1 \cdot C_{delay}$	0	$1 \cdot C_{hold}$	$2 \cdot C_{hold}$
2-day late	2	$p_2$	$2 \cdot C_{delay}$	$1 \cdot C_{delay}$	0	$1 \cdot C_{hold}$
3-day late	3	$p_3$	$3 \cdot C_{delay}$	$2 \cdot C_{delay}$	$1 \cdot C_{delay}$	0

remark:  $C_{hold} \equiv C_{hold}(i, t_i)$  and  $C_{delay} \equiv C_{delay}(i)$

A move  $\varphi$  can also be planned to arrive after the due-date of the targeted order ( $\sigma_a > \sigma_{lim}(\varphi)$ ). In this specific case, if  $\sigma_r < \sigma_a$  (*i.e.* railcars arrive earlier than anticipated) and if the real arrival date is greater or equal to the due-date of the order, railcars are not hold at terminal but immediately delivered. Thus they are delivered “less late” than anticipated and delay costs must be subtracted in  $C_{\Delta}$ . Table 4.2 illustrates this case for a move  $\varphi = (t, i, c, d, d + \Delta T(t, t_i) + \sigma_a)$  where  $d_i = d + 2$ ,  $\Delta T(t, t_i) = 1$  ( $\sigma_{lim} = 1$ ) and  $\sigma_a = 2$ . However, this could provide the scheduler with an optimal solution consisting in scheduling an anticipated delay that makes the order late while it was possible to arrive on time (without the anticipated delay, see Figure 4.3). This can be justified from an economic point of view,

for example if the delay is highly probable. However, this situation is not desired from a job point of view. Indeed, scheduler could not admit scheduling a late delivery when it is theoretically possible to arrive on time. For the same reasons which have made necessary the implementation of the decision process, this situation is forbidden. To do so, the following constraint is added.

$$\forall t \in \mathcal{T}, \forall i \in \mathcal{I}, \forall c \in \mathcal{C}, \forall d_{dep} \in \mathcal{H} \setminus \{d^0\}, \forall d_{arr} > d_{dep} + \Delta T(t, t_i), d_{arr} > d_i$$

$$N_{tr}(t, i, c, d_{dep}, d_{arr}) = 0 \quad (4.4)$$

It forces the flow of moving railcars to be null, when the arrival date is later than the theoretical arrival date and the deadline. In other words, the anticipated delay on a move will never create or worsen a delivery delay. Figure 4.4 shows how the move illustrated by Figure 4.3 will be corrected. This constraint could be deactivated when schedulers will have experimented and approved the tool.

Table 4.2: Computing  $C_{\Delta}(s(\varphi, \sigma_r))$  when  $\sigma_a = 2 > \sigma_{lim}(\varphi) = 1$

scenario	$\sigma_r$	$p(\sigma_r)$	$\sigma_a = 2$
on-time	0	$p_0$	$-1 \cdot C_{delay} + 1 \cdot C_{hold}$
1-day late	1	$p_1$	$-1 \cdot C_{delay}$
2-day late	2	$p_2$	0
3-day late	3	$p_3$	$+1 \cdot C_{delay}$

remark:  $C_{hold} \equiv C_{hold}(i, t_i)$  and  $C_{delay} \equiv C_{delay}(i)$

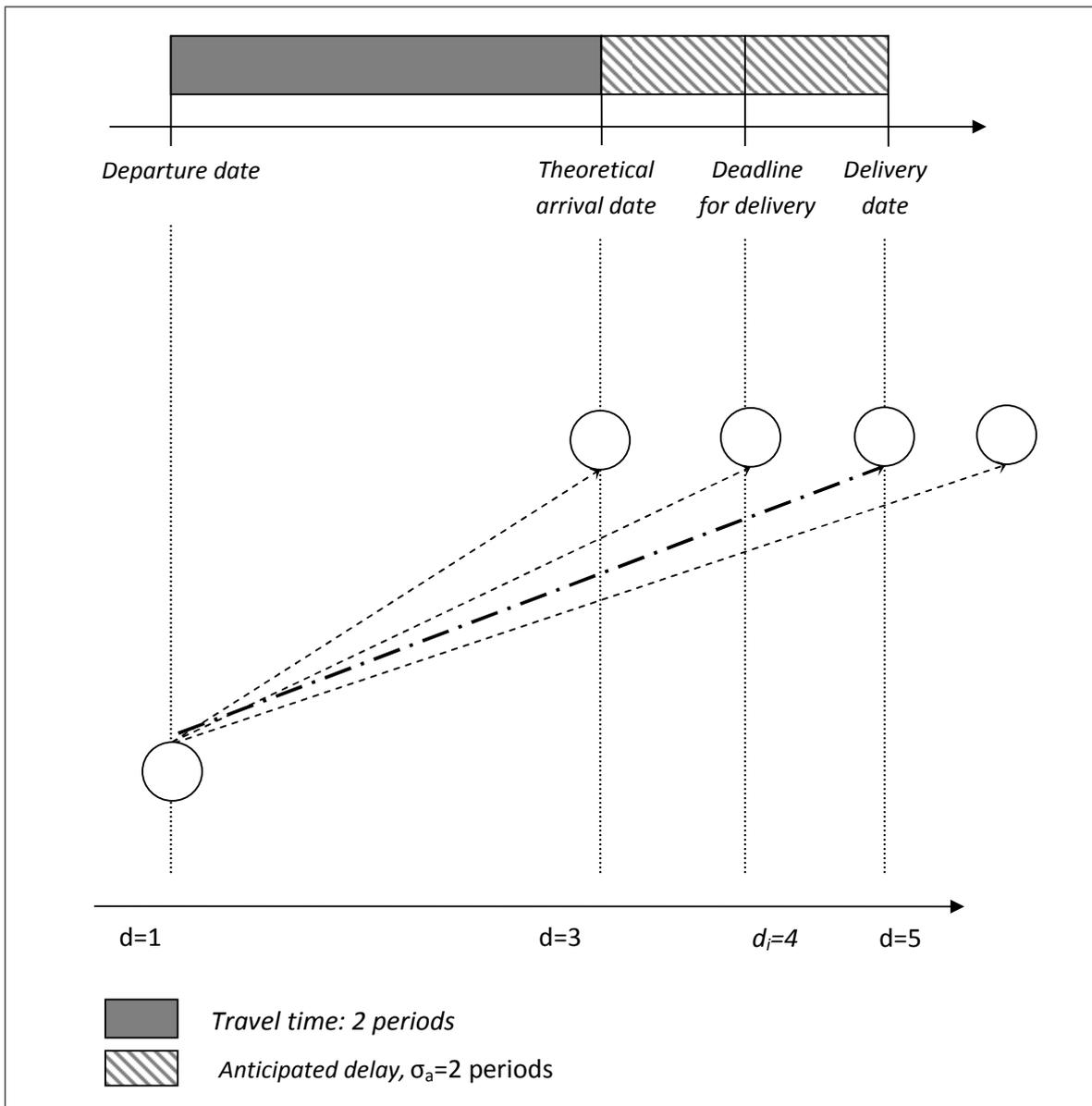


Figure 4.3: Time Decomposition of a Forbidden Anticipated Delay

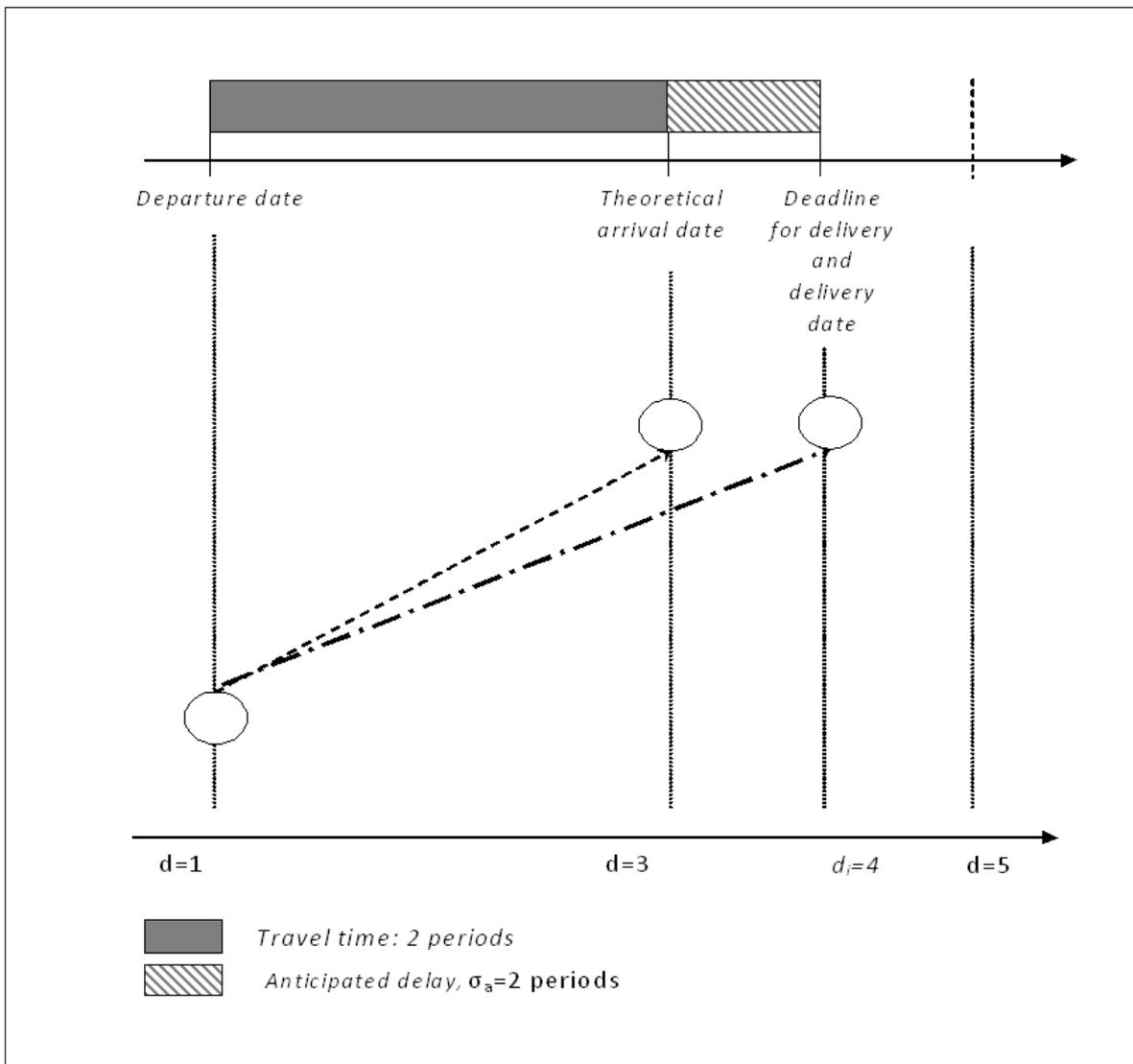


Figure 4.4: Time Decomposition of a Corrected Move

The cost  $C_{\Delta}(s(\varphi, \sigma_r))$  is therefore computed as follows:

$$\begin{aligned} C_{\Delta}(s(\varphi, \sigma_r)) = & C_{hold}(c, t_i) \times \max(0, \sigma_{lim}(\varphi) - \sigma_r) \\ & + C_{delay}(i) \times \left[ \max(0, \sigma_r - \sigma_{lim}(\varphi)) - \max(0, \sigma_a - \sigma_{lim}(\varphi)) \right] \end{aligned} \quad (4.5)$$

#### 4.1.4 Linear Program With Travel Time Uncertainty

In the linear program, the previously defined cost (see chapter 3) corresponds to the cost of the anticipated moves. An additional cost must then be added in order to take into account the travel time uncertainty. This cost is called the “Total Expected Anticipation Cost” (TEAC) and is associated to a weight  $W_{TEAC}$ :

$$TEAC = \sum_{\varphi \in \Phi} N_{tr}(\varphi) \times C_a(\varphi) \quad (4.6)$$

Let the new linear program be named  $LP_{stoc}(\alpha, \beta)$  where parameter  $\alpha$  is the set of orders processed, parameter  $\beta$  is the horizon time considered. ( $LP_{glob}$ ) constraints are then modified as follows to take into account the explicit dependence of  $N_{tr}$  variables upon  $\sigma_a$ :

- Initial state Equation (3.9) becomes

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C},$$

$$N_{av}(t, c, d^0) = \text{inventory}(t, c, d^0)$$

$$\begin{aligned} & - \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t}} \sum_{\substack{\sigma_a=0 \\ d^0 + \Delta(t, t_i) + \sigma_a \leq d^f}}^{D_{max}} N_{tr}(t, i, c, d^0, d^0 + \Delta T(t, t_i) + \sigma_a) \\ & - \sum_{i \in \mathcal{I}(t, c)} Q_{all}(i, d^0) - \sum_{i \in \mathcal{I}(t)} \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d^0) \end{aligned}$$

(4.7)

- State Equation (3.10) becomes

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H} \setminus \{d^0\},$$

$$N_{av}(t, c, d) = N_{av}(t, c, d - 1) + \text{inventory}(t, c, d)$$

$$\begin{aligned} & + \sum_{\substack{i \in \mathcal{I}(t) \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t'}} \sum_{\substack{\sigma_a=0 \\ d - \Delta T(t, t_i) - \sigma_a \geq d^0}}^{D_{max}} N_{tr}(t', i, c, d - \Delta T(t, t_i) - \sigma_a, d) \\ & - \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t}} \sum_{\substack{\sigma_a=0 \\ d + \Delta T(t, t_i) + \sigma_a \leq d^f}}^{D_{max}} N_{tr}(t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \\ & - \sum_{\substack{i \in \mathcal{I}(t, c) \\ d \geq d_i}} Q_{all}(i, d) - \sum_{\substack{i \in \mathcal{I}(t) \\ d \geq d_i \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}(i, c, d) \end{aligned} \quad (4.8)$$

- Allocation after moves constraint (3.13) becomes

$$\begin{aligned} & \forall i \in \mathcal{I}, \forall d \in \mathcal{H} \\ & Q_{all}(i, d) \geq \sum_{t \in \mathcal{T}} \sum_{\substack{\sigma_a=0 \\ d - \Delta T(t, t_i) - \sigma_a \geq d^0}}^{D_{max}} N_{tr}(t, i, c_i, d - \Delta T(t, t_i) - \sigma_a, d) \end{aligned} \quad (4.9)$$

- Substitution after moves constraint (3.14) becomes

$$\begin{aligned} & \forall i \in \mathcal{I}, \forall d \in \mathcal{H}, \forall c \in \mathcal{R}_{sub}(u_i, c_i), \\ & Q_{sub}(i, c, d) \geq \sum_{t \in \mathcal{T}} \sum_{\substack{\sigma_a=0 \\ d - \Delta T(t, t_i) - \sigma_a \geq d^0}}^{D_{max}} N_{tr}(t, i, c, d - \Delta T(t, t_i) - \sigma_a, d) \end{aligned} \quad (4.10)$$

- Integrality constraint (3.15) becomes

$$\begin{aligned} & \forall c \in \mathcal{C}, \forall d \in \mathcal{H}, \sigma_a \in [0, \dots, D_{max}] \forall t \in \mathcal{T}, \forall i \in \mathcal{I} \text{ with } t \neq t_i \\ & N_{tr}(t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \in \mathbb{R}^+ \end{aligned} \quad (4.11)$$

Note that uncertainty is not handled on already scheduled travels (variable  $N_{arr}^{BIG}$ ). In fact, already scheduled travels will mainly stand because:

- Either they were planned before the implementation of the tool,
- Or they have been implemented because, for any reason, the suggested original schedule (computed by the LP) does not fit with the scheduler's will.

In the first case, the management of these travels is not necessary as they will disappear as the tool will be used. Moreover, changing all previously scheduled travels could completely disorganize the activity. In the second case, it is assumed that these manually scheduled

travels will be a minority among all scheduled travels and the schedulers will have to evaluate the delay himself (since he or she does not agree with the computed suggestion).

## 4.2 Robustness Level Cursor

The previous refinement provides us with an optimal solution based on expected value. However, we have to offer more flexibility to schedulers, who may not want to apply a strategy based on the average outcome of a probabilistic set of events. For example, when activity is low, on-time deliveries of orders can be valued over turnover; on the contrary, risk-taking strategy will be privileged as activity increases. That is why we introduce a *robustness cursor*. This cursor is denoted by  $\gamma_{rob}$  and bounded by  $0 \leq \gamma_{rob} \leq 2$ . It should make the linear program provide solutions with:

- lower anticipated delays when  $\gamma_{rob}$  is low,
- greater anticipated delays (where required) when  $\gamma_{rob}$  is high.

For this purpose, Equation (4.5) can be changed as follows:

$$C_{\Delta}(s(\varphi, \sigma_r)) = (2 - \gamma_{rob}) \times C_{hold}(c, t_i) \times \max(0, \sigma_{lim}(\varphi) - \sigma_r) + \gamma_{rob} \times C_{delay}(i) \times \left[ \max(0, \sigma_r - \sigma_{lim}(\varphi)) - \max(0, \sigma_a - \sigma_{lim}(\varphi)) \right] \quad (4.12)$$

Similarly to  $LP_{stoc}(\alpha, \beta)$ , this new linear program is denoted by  $LP_{rob}(\alpha, \beta, \gamma_{rob})$ . When  $\gamma_{rob}$  is high, say  $\gamma_{rob} = 2$ ,  $C_{hold}(c, t_i)$  is null while  $C_{delay}(i)$  is doubled: scenarios with high risks of delay will then be more expensive than safer ones with holding risk. Note that  $LP_{rob}(\alpha, \beta, \gamma_{rob} = 1) \equiv LP_{stoc}(\alpha, \beta)$ .

### 4.3 About the Control of Equity

Because of the amount of parameters that should be taken into account (see section 3.4.2), the selection of orders incurring delays is indirectly made by the scheduler when he evaluates the delay and unfulfillment penalties. Thus, if the relation with a customer is getting bad because of repeated delays, the scheduler has the responsibility for evaluating the penalty considering the consequences of a new delay.

### 4.4 About the Pre-Processing

The current configuration of the model make possible the design of a preprocessing step selecting the arcs with the lowest cost before computing the optimal solution. However, future work on this model would consist in adding blocking constraints or capacity constraints. Selecting only lowest-cost arcs could not enforce such constraints; that is why no specific pre-processing step has been used.

# Chapter 5

## Model Evaluation

As I was not granted access to real data, I have developed a random data generator in JAVA language to compute inputs of the problem. These inputs have been used to process different tests in order to assess the model. Before all, the maximum size of solvable models have been evaluated. Then, a simulator has been used to process computed schedule in order to assess the effect of  $\gamma_{rob}$  and the contribution of the stochastic approach compared to a deterministic approach. Finally, the model output has been compared to outputs obtained by processing simple heuristics.

All tests and outputs computation have been executed on a computer running with a 2.19GHz AMD processor and 1Go of RAM.

### 5.1 Data Generator

To provide data to the model, I have programmed a data generator (in JAVA language). Inputs of this generator are the following:

- The time horizon length (default: 6);
- The amount of orders, terminals, categories and customers;
- The ratio of committing orders (default: 80%);
- The lower and upper bounds on the amount of railcar to be delivered in the scope of an order (default: 10 to 30);
- The lower and upper bounds of the unfulfillment penalties for an order (default: 50 to 150);
- The lower and upper bounds of the delay penalties for an order (default: 10 to 30);
- The maximum number of substitution permitted for a customer (default: 2);
- The upper bound of substitution costs (lower bound is 0, default: 10);
- The upper bound of travel time (lower bound is always 1, default: 3);
- The lower and upper bounds of moving cost (default: 5 to 10);
- The lower and upper bounds of holding cost (default: 3 to 5);
- The upper bound of the amount of railcar of each category present at a terminal at the beginning of the first period (default: 40);
- The maximum allowed delay for a move (default: 3);
- The delay probability for every origin-destination couples.

Data are then randomly generated following uniform distributions bounded by the corresponding lower and upper limits.

## 5.2 Size and Computing Time: 3-step LP and Global LP Comparison

I have used the CPLEX algorithm NETOPT designed to solve network problem and I have enabled the pre-process of CPLEX in order to fasten the computation of the solution eliminating needless or redundant constraints (depending on input). Changing the algorithm to TRANOPT (dual method) or PRIMOPT (primal method) does not affect the computing time.

The aim of these tests consists in quantifying the loss of optimality, the difference of computing times and the size of solvable problems when using the decision making process (3-step problem) instead of solving the global LP problem.

The program has been solved on a set of problems covering a horizon of 6 periods. Each line of Table 5.1 corresponds to an instance of the problem generated randomly according to previously described default range, columns contains following data:

- Columns (1), (2) and (3) depict the number of terminals, categories and orders considered in the model.
- Column (4) is the “offer:demand ratio”, that is to say the ratio between offer and demand. A ratio lower than 1 corresponds to a demand greater than the offer, it is often associated to a high “total cost” as many unfulfillment penalties are credited in these cases.
- Column (5) is the computing time (in seconds) of the 3-step problem.
- Column (6) is the computing time (in seconds) of the global LP problem (not sequenced and so not enforcing the decision-making process).

- The two columns (7) and (8) correspond to the result of the objective function for, respectively, the 3-step and global LP model.
- Column (9) is the gap between 3-step and global LP solution. It is the result of following equation:  $gap = \frac{(7)-(8)}{(8)}$ .

It can be drawn from tests that the 3-step program is able to solve bigger problems than the linear program. Indeed, CPLEX raise an “out of memory” error for problem involving more than 74,170 variables. The sequenced model raised the same error for problem involving more than 484,700 variables. However, on problems solvable with both methods, no significant increase in computing times has been noticed.

Analysis of results showed that the gap variation is mainly affected by the selection of these unfulfilled orders. In the 3-step LP, committing orders are priority and are fulfilled before the not-committing ones. Thus, the fulfillment of a committing order can prevent various not-committing ones from being fulfilled and therefore indirectly generate big penalties. The gap size depends then rather on the “quality” of the input than on its size. Quality is used to qualify the input in this way:

- An input has a good quality if the starting location of railcars will not imply many moves and if the ratio offer:demand is high.
- An input has a poor quality if there are few railcars to offer comparing to demand and if delivery terminals of order are different from the starting locations of orders.

Moreover, committing orders due on day  $d^0$  are highly priority and their fulfillment can prevail over other orders even if their unfulfillment penalties are higher, this can generate expensive solution (it is the case in instance 2).

Table 5.1: LP-sequence Comparison

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
1	10	1	10	0.54	1.9	0.9	11,401	8,214	38.8%
2	10	3	10	1.26	2.2	2.0	171	145	17.9%
3	10	5	10	2.56	3.0	3.1	90	90	0.0%
4	50	1	10	3.73	2.2	3.4	63	63	0.0%
5	50	3	10	9.55	6.2	7.1	344.7	344.7	0.0%
6	50	5	10	13.48	4.1	13.8	111	111	0.0%
7	10	1	50	0.19	6.5	3.6	87,536	81,353	7.6%
8	10	3	50	0.34	13.4	11.2	47,379	38,719	22.4%
9	10	5	50	0.52	11.7	15.5	35,206	30,951	13.7%
10	50	1	50	0.54	18.1	9.6	46,656	38,097	22.5%
11	50	3	50	1.57	110.3	*	438	*	*
12	50	5	50	2.81	27.9	*	377	*	*
13	10	1	100	0.06	9.1	6.1	222,198	213,706	4.0%
14	10	3	100	0.18	16.5	25.3	179,857	166,746	7.9%
15	10	5	100	0.27	22.1	*	160,235	*	*
16	50	1	100	0.31	20.3	*	130,387	*	*
17	50	3	100	0.85	32.3	*	29,049	*	*
18	50	5	100	1.56	160.7	*	846	*	*
19	12	3	100	0.21	20.7	19.9	180,278	160,804	12.1%
20	15	3	100	0.28	22.4	36.0	156,937	134,160	17.0%
21	50	3	200	0.44	271.5	*	206,276	*	*

\*: CPLEX has returned error "out of memory"

(1): terminals, (2): categories, (3): orders, (4): Offer:Demand ratio, (5): computing time of the 3-step problem, (6): computing time of the global LP problem, (7): result for 3-step, (8): result for global LP model, (9): gap.

## 5.3 Simulator

### 5.3.1 Description

To analyze the solution quality of the model, I have developed a simulator (in JAVA language). This simulator extracts the solution computed by CPLEX and processes it (as shown on figures 5.1 and 5.2). This solution is represented by allocation stacks and departure stacks for each days of the time horizon. These stacks contain the allocations and departures vectors scheduled by CPLEX; each vector contains the different parameters of an allocation (date, order, terminal, quantity, category) or a departure (date, departure and arrival terminals,

quantity, category and order targetted). During the simulation the arrival stack and orders are filled. The cost of the simulated schedule is then computed.

### 5.3.2 Transience

The inherent unstability of the system, that is to say the unpredictability of delays, can sometimes generate great variation of outcomes from one instance of a problem to another. In order to be able to draw conclusions from tests, indicators extracted must be stable. That is to say, under identical conditions, two indicators extracted from different batches of simulations should be similar. Variations are smoothed when a simulation is repeated many times. Figure 5.3 shows the total mean cost over the processed simulations. Input processed are characterized by the value of  $\gamma_{rob}$  when they were computed by CPLEX and by the distribution of the potential delay on move. This distribution is denoted by a 4-uple  $P = (a, b, c, d)$ ;  $a$  is the probability to arrive on time,  $b$  to arrive one-day late,  $c$  two-day late, and  $d$  three-day late.

- input G0 to G2 correspond to CPLEX solutions on random data where  $\gamma_{rob}$  was set equal to 0 to 2 and distribution equal to  $P = (0.6, 0.2, 0.1, 0.1)$
- input P1 correspond to CPLEX solutions where  $\gamma_{rob} = 1$  and the distribution was  $P = (0.4, 0.3, 0.2, 0.1)$
- input P2 correspond to CPLEX solutions where  $\gamma_{rob} = 1$  and the distribution was  $P = (0.9, 0.07, 0.02, 0.01)$

The mean cost reaches a stable state when at least 30 simulations have been processed. It has been therefore chosen to use mean costs computed over 30 simulations as an indicator in the following tests.

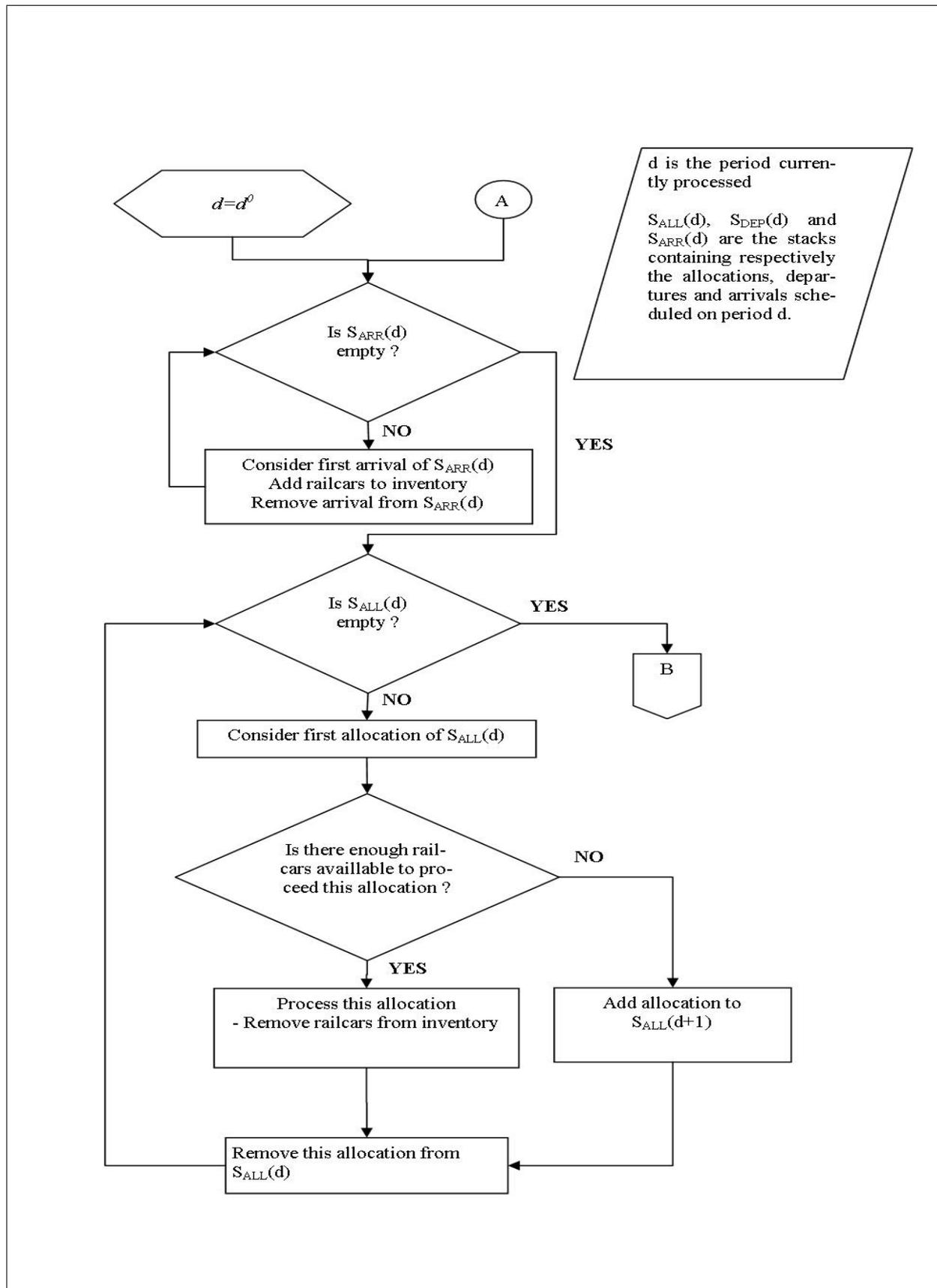


Figure 5.1: Simulator Flow Chart (Part 1)

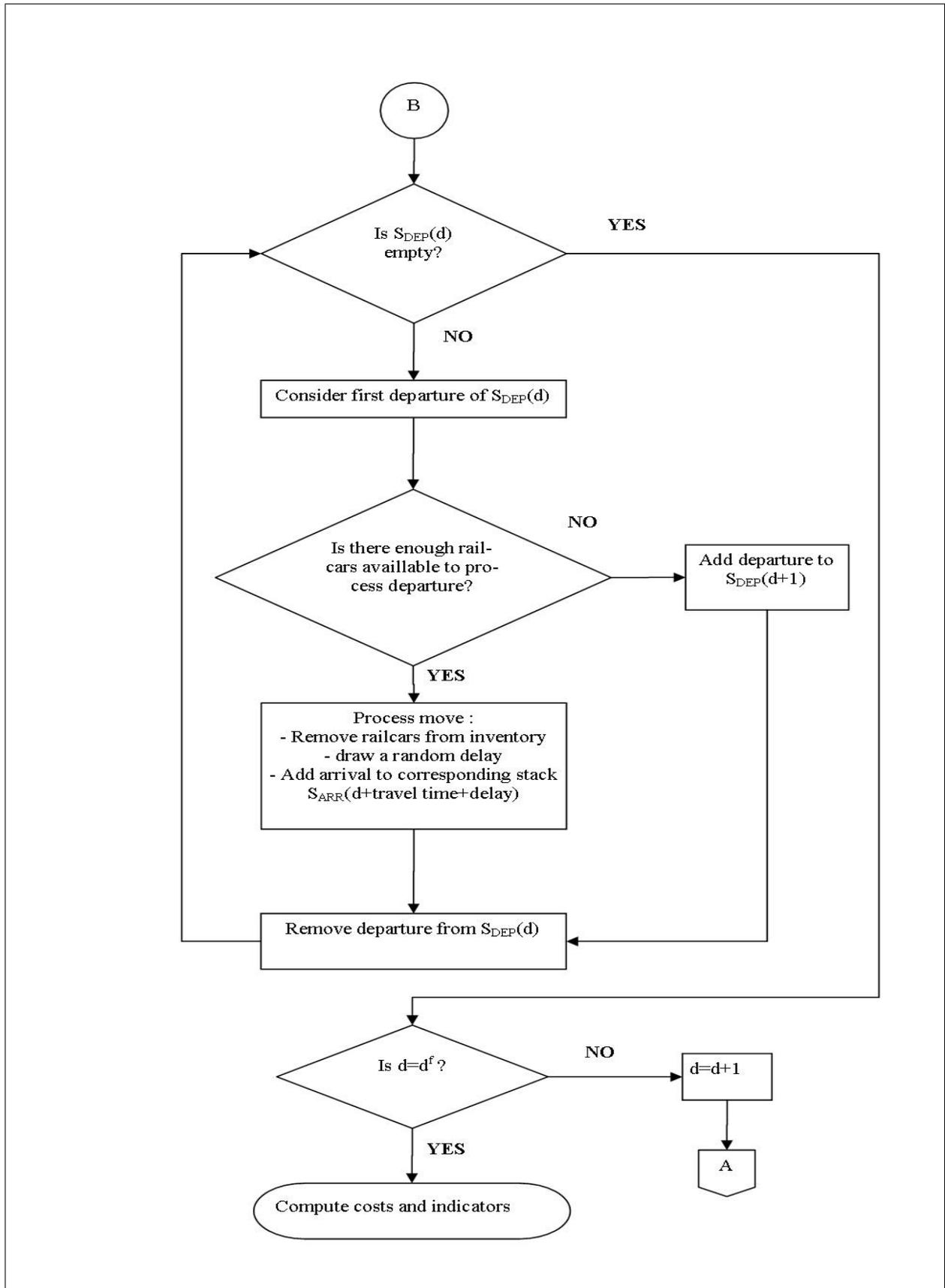


Figure 5.2: Simulator Flow Chart (Part 2)

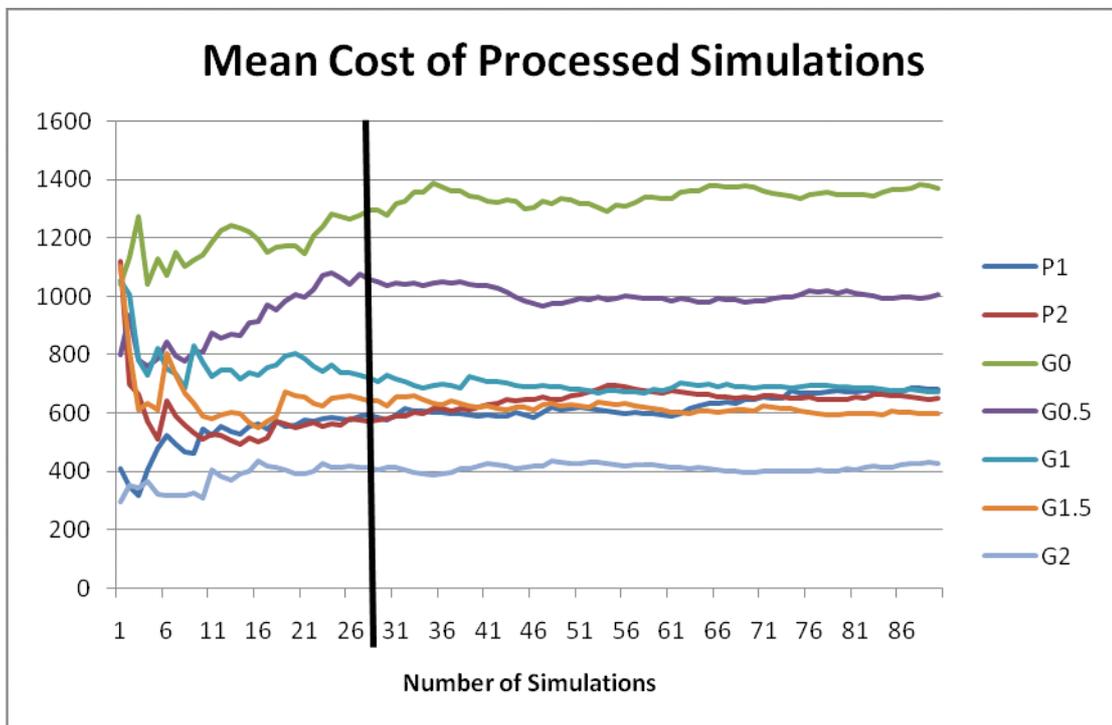


Figure 5.3: Mean Stabilization

Only tests on real data over a long time horizon could show if great outcomes generated when a big error occurs between anticipated and real delay are compensated by low outcomes generated by more accurate prediction. Considering the project duration and the time required for such a development, it was not possible to carry out such a study.

## 5.4 $\gamma_{rob}$ Cursor Test

These tests should help in quantifying the impact of the variation of  $\gamma_{rob}$  cursor and to assess its utility. Five different inputs have been used in these tests, they are characterized by the ratio Offer:Demand OD and by the distribution of the potential delay on move. Note that the ratio OD is approximated (error:  $\pm 10\%$ ) and the distribution applies to all origin-destination couples. The six different inputs are the following:

Inputs A: OD variations

- Input A1:  $OD \approx 0.5$ ,  $P = (0.6, 0.2, 0.1, 0.1)$ ; “shortage situation”
- Input A2:  $OD \approx 1$ ,  $P = (0.6, 0.2, 0.1, 0.1)$ ; “balanced situation” and “normal environment”
- Input A3:  $OD \approx 2$ ,  $P = (0.6, 0.2, 0.1, 0.1)$ ; “surplus situation”

Inputs B: Uncertainty variations

- Input B1:  $OD \approx 1$ ,  $P = (0.4, 0.3, 0.2, 0.1)$ ; “uncertain environment”
- Input B2 = input A2:  $OD \approx 1$ ,  $P = (0.6, 0.2, 0.1, 0.1)$ ; “balanced situation” and “normal environment”

- Input B3:  $OD \approx 1$ ,  $P = (0.9, 0.07, 0.02, 0.01)$ ; “certain environment”

Every inputs contains data for 100 orders, 20 terminals, 3 railcar categories over a 6-period time horizon which is a realistic size for sub-network on which the model could be applied. Following line-graphs (figures 5.4 to 5.11) depict the variations of delay cost, holding cost and sum of both depending on  $\gamma_{rob}$ .

Figures 5.4 to 5.6 highlight that in a “surplus situation”  $\gamma_{rob}$  does not have a significant impact on delay and holding strategy as few moves are required. On “balanced” and “shortage” delay costs continuously decrease as  $\gamma_{rob}$  increases while holding costs increases from  $\gamma_{rob}=0.5$  to  $\gamma_{rob}=2$ . Sum of costs is almost constant form  $\gamma_{rob} = 0$  to  $\gamma_{rob} = 1.5$  which could be a good range of use. The best value happens to be around  $\gamma_{rob} = 0.8$  for these inputs. Let us note that this depends highly on the structure of inputs.

It can be drawn from figures 5.7 to 5.9 that the contribution of cursor  $\gamma_{rob}$  seems not to depend on the probabilistic distribution of the potential delays except that the cursor has no effect on a “certain environment” (which seems to be logical as no delay really requires to be anticipated). On the same pattern the ideal range of value for  $\gamma_{rob}$  seems to be 0 to 1.5; beyond 1.5 holding cost increases dramatically overlapping the saves in delay costs.

Figures 5.10 and 5.11 emphasize the fact that  $\gamma_{rob}$  has almost no effect on moving and unfulfillment costs (comparing to other costs).

To conclude, on balanced situations and on shortage situations in an uncertain or moderate environment, the  $\gamma_{rob}$  cursor does affect the strategy by swapping delay cost to holding cost when  $0 \leq \gamma_{rob} \leq 1.5$  on other situations it has no effect and if  $\gamma_{rob} > 1.5$  the sum of delay and holding cost increases significantly. Finally, based on sum of the holding and delay cost, the best value for  $\gamma_{rob}$  seems to be 0.8.

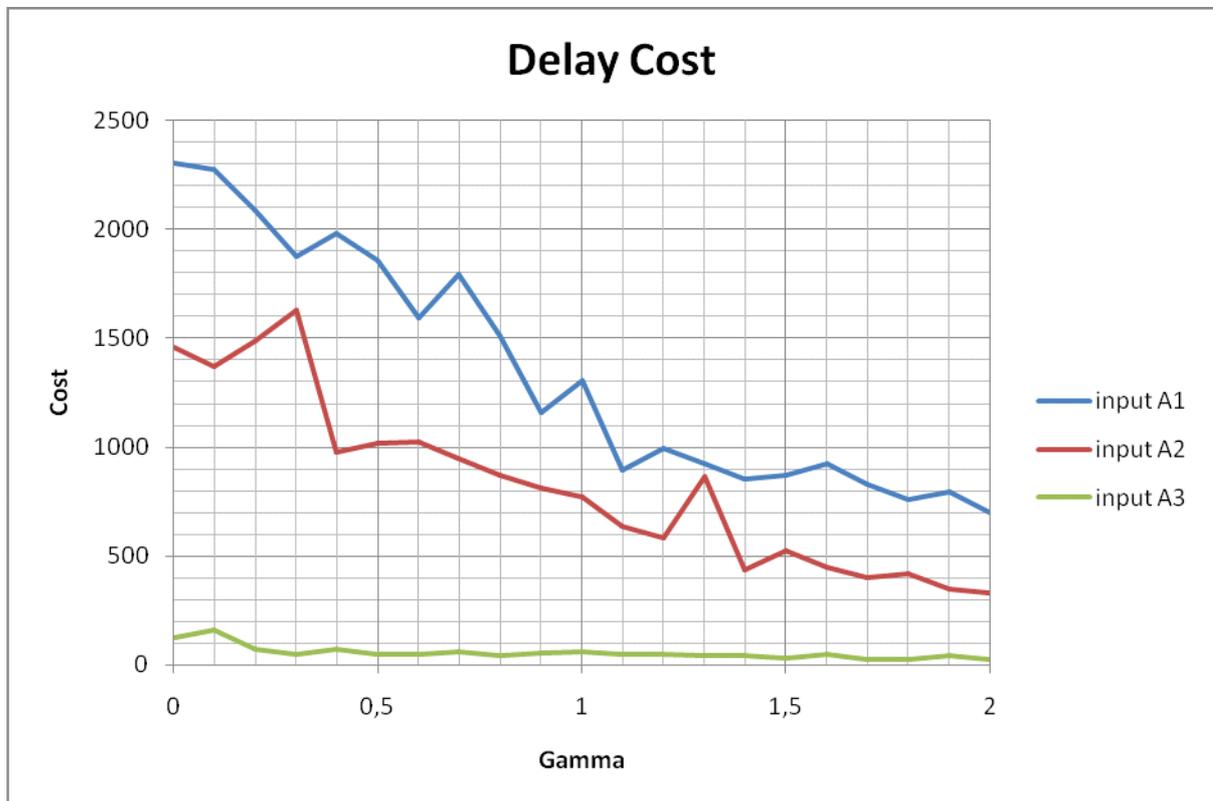


Figure 5.4: Delay Cost Variation Depending on  $\gamma_{rob}$  and Ratio OD

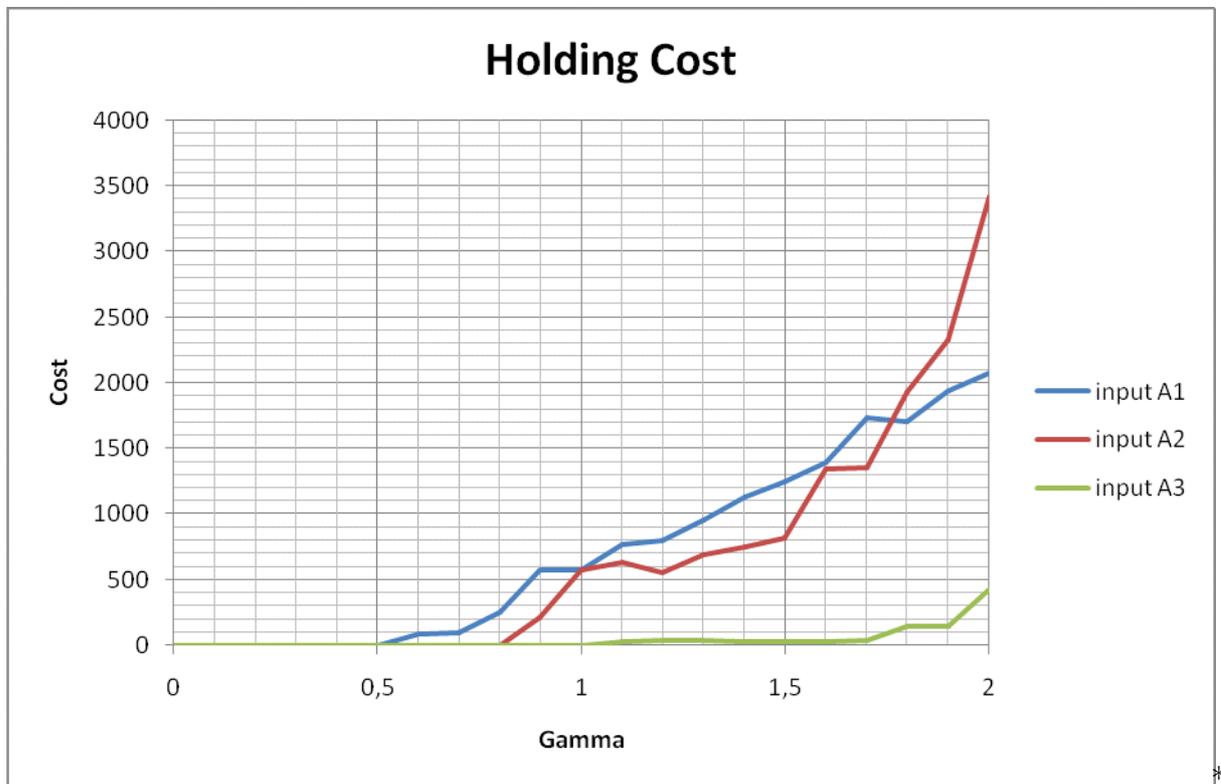


Figure 5.5: Holding Cost Variation Depending on  $\gamma_{rob}$  and Ratio OD

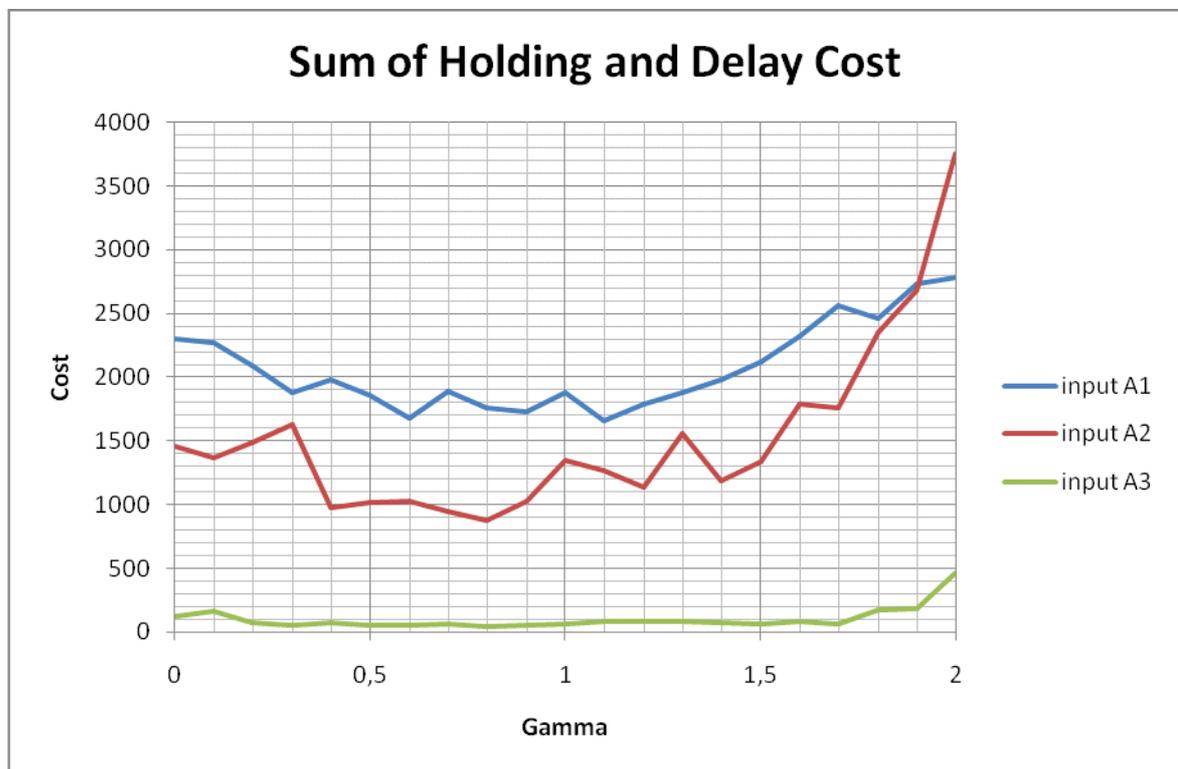


Figure 5.6: Sum of Holding and Delay Cost Variation Depending on  $\gamma_{rob}$  and Ratio OD

## 5.5 Contribution of Stochastic Approach Tests

To ensure the stochastic LP bring any contribution comparing to a similar but deterministic LP, three different inputs have been generated and solved using a deterministic LP “LPD” and the stochastic LP “LPS( $\gamma_{rob}$ )”. Their outputs have then been simulated 30 times and costs are compared in figures 5.12 to 5.14. As moving costs and substitution costs are slightly affected by the stochastic approach, they are not represented. Inputs are characterized as follows:

- Input C1: Small problem 8,000 variables
- Input C2: Medium problem 25,000 variables
- Input C3: Big problem 100,000 variables

For all sizes of problem, the stochastic LP helps to provide a more economical strategy, however figures 5.13 and 5.14 highlight the fact that an extreme  $\gamma_{rob}$  value can lead to an extreme strategy where either holding or delay cost can considerably increase, moreover  $\gamma_{rob} = 0.5$  (which is close to the 0.8 depicted above) is an interesting value. When taking into account all costs, the variation of costs between deterministic and stochastic approaches can reach 10%.

## 5.6 Anticipated Delay Pertinence Tests

The next set of tests is carried out to make sure that the linear program has a better behavior than simple method which could have been easily implemented in the linear program. These manual methods consist in simply adding 1, 2 or 3 slack periods systematically to every single

move instead of computing the anticipated delay (these methods are denoted respectively M1, M2 and M3). Following results were obtained by solving the linear program with input A2 using the method M1, M2 and M3 and using the stochastic linear programme with different values of  $\gamma_{rob}$  (0, 0.5, 1 and 1.5 corresponding to LP0, LP0.5, LP1 and LP1.5).

Figure 5.15 shows the cost decomposition for the 7 methods (unfulfillment and substitution costs are similarly low comparing to other one through the seven methods; they are therefore not shown). Although they are easy to implement, it is clear that methods M1, M2 and M3 are less economically interesting than the LP method (about 50% more expensive).

## 5.7 Conclusions

Before all, it is important to emphasize that these tests have been carried out on randomly generated data, therefore, following conclusions could vary for real data.

Tests have showed the stochastic approach gives better results than the deterministic one or than a simpler method. In addition, the sequenced model can handle problem containing up to 484,700 variables and can solve it in 271 seconds which corresponds to 50 terminals, 3 categories and 200 orders. This is greater than the 74,170 variables solvable by the global linear program. However, for small problems, the linear program gives better results. Finally, a good range of use of  $\gamma_{rob}$  cursor appears to lie from 0.5 to 1.5 and a value close to 0.8 gives good results.

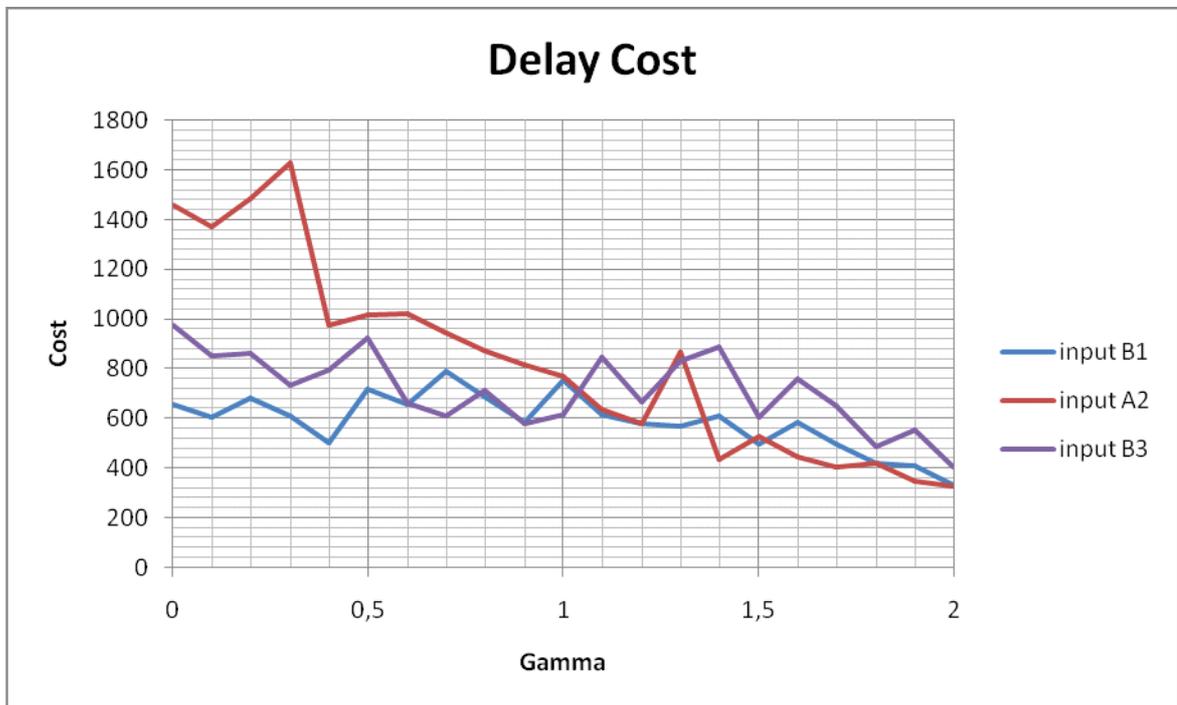
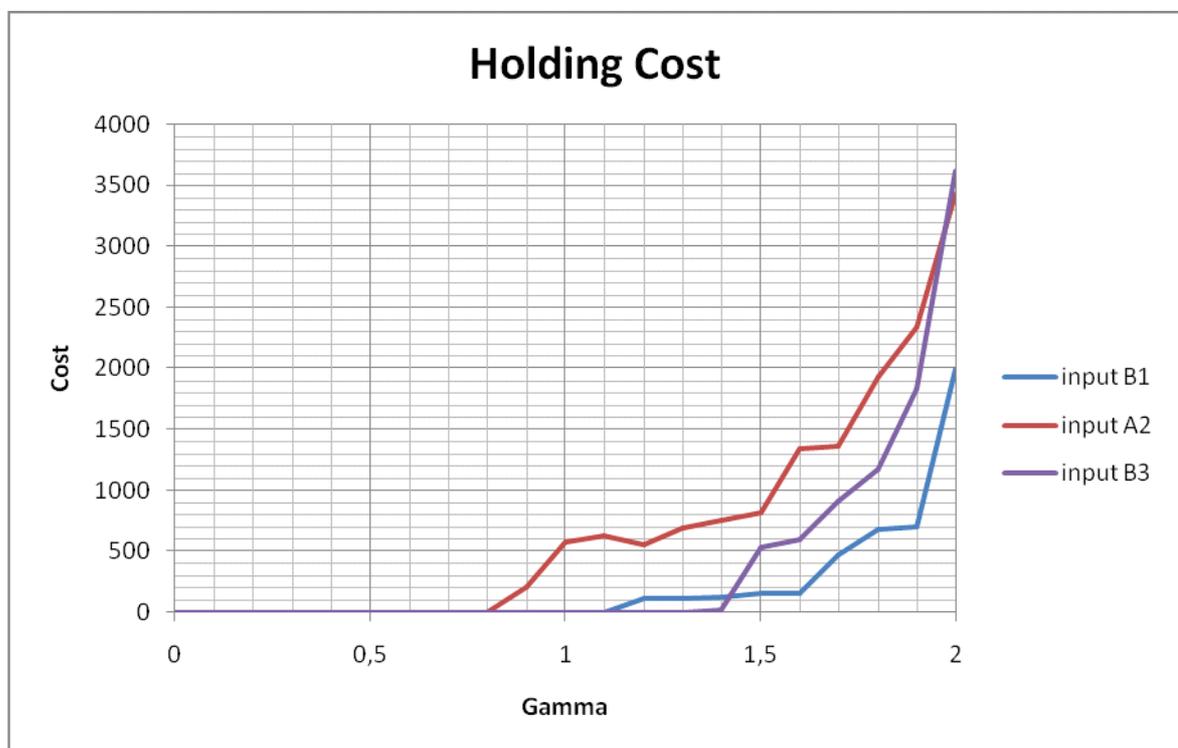


Figure 5.7: Delay Cost Variation Depending on  $\gamma_{rob}$  and Distribution

Figure 5.8: Holding Cost Variation Depending on  $\gamma_{rob}$  and Distribution

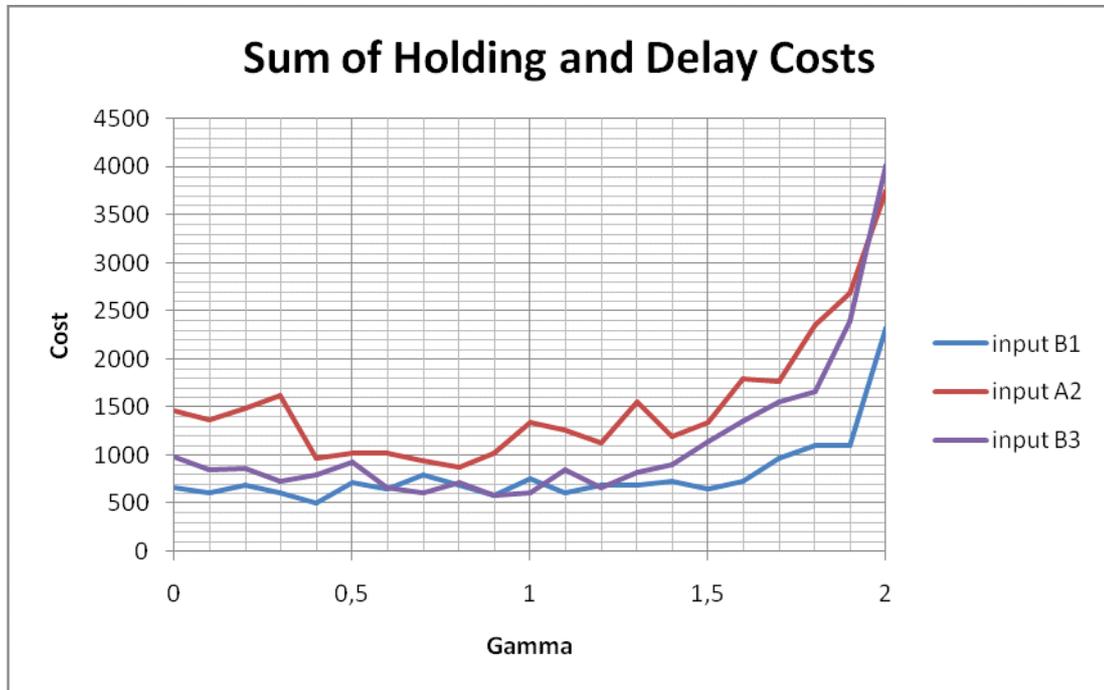


Figure 5.9: Sum for Holding and Delay Cost Variation Depending on  $\gamma_{rob}$  and Distribution

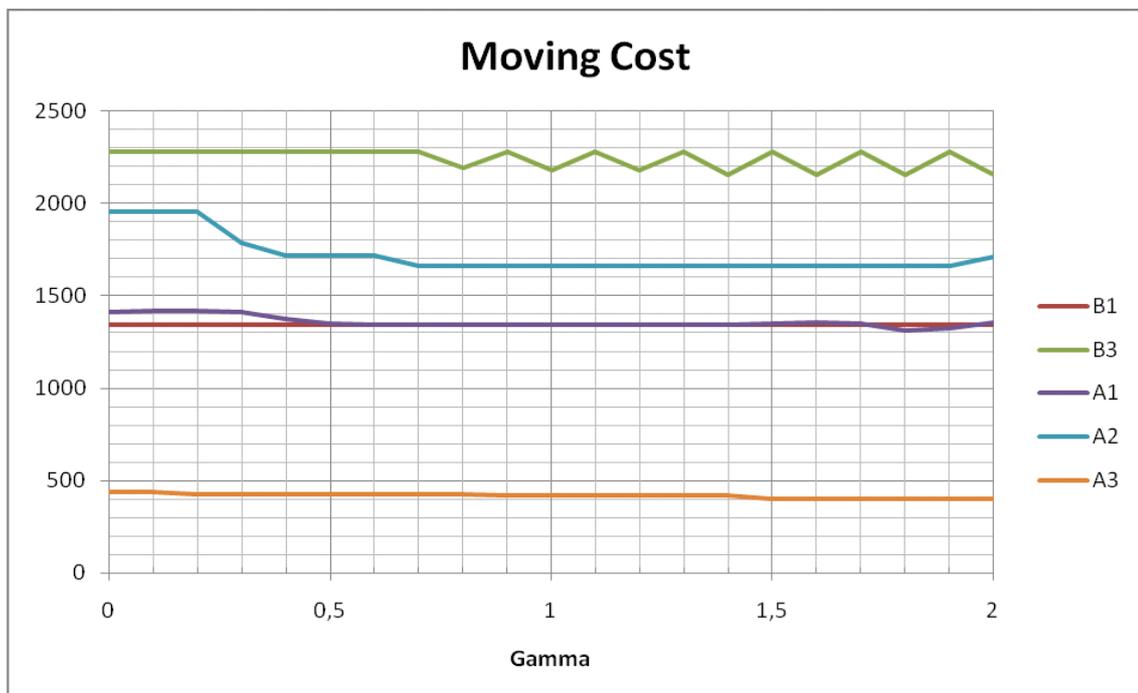


Figure 5.10: Moving Cost Variation Depending on  $\gamma_{rob}$ , ratio OD and Distribution

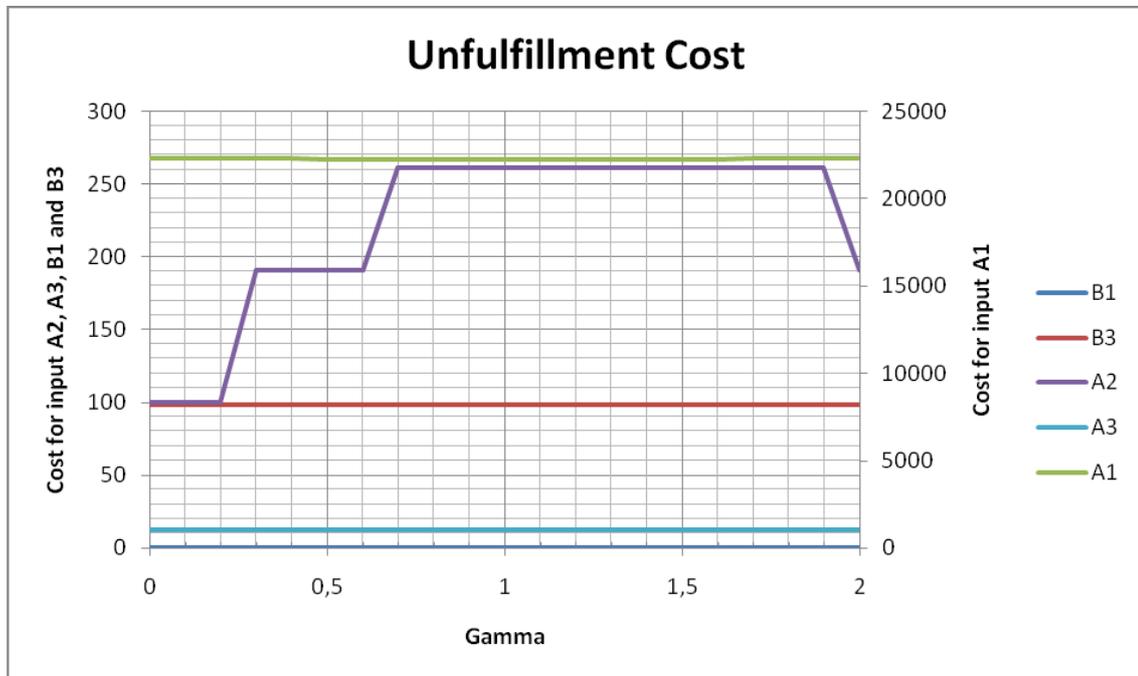


Figure 5.11: Unfulfillment Cost Variation Depending on  $\gamma_{rob}$ , ratio OD and Distribution

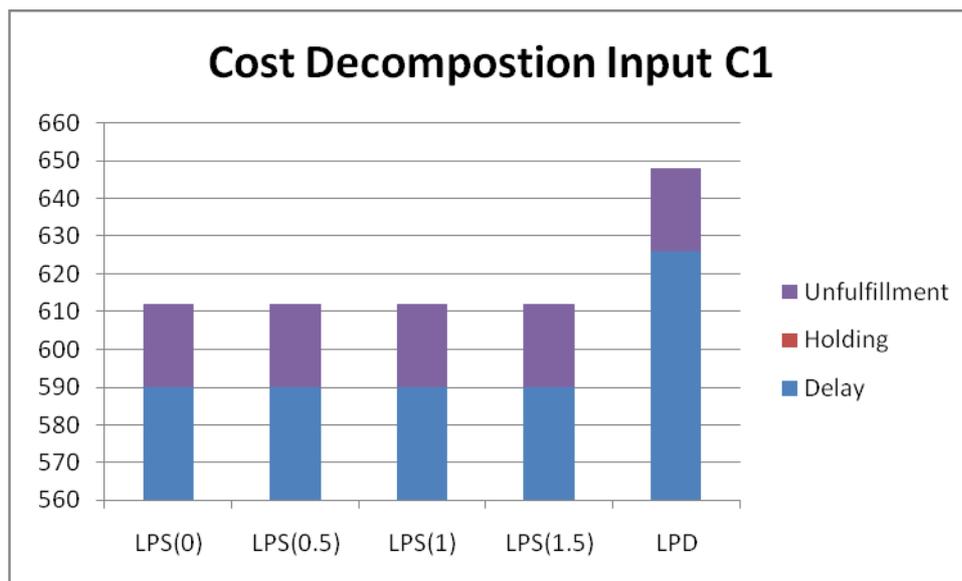


Figure 5.12: Cost Decomposition Input C1

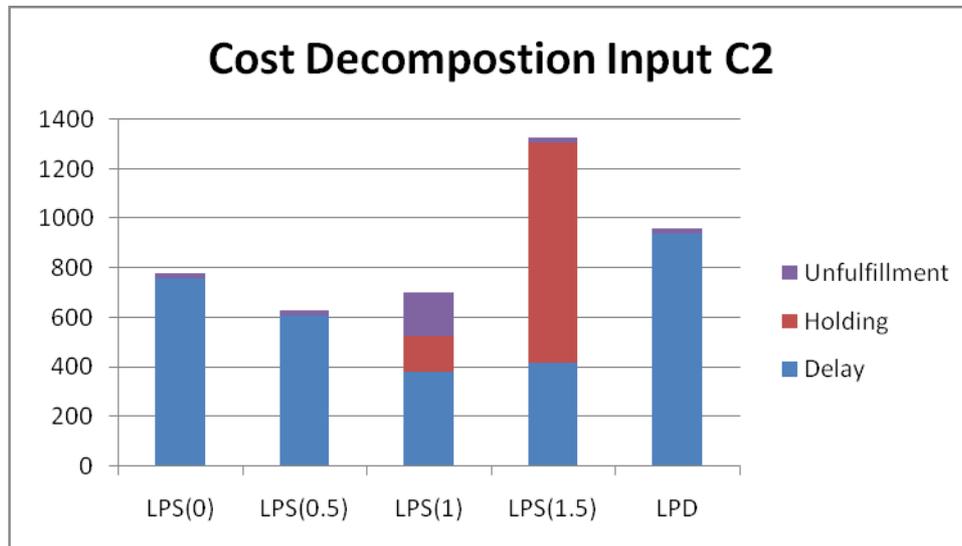


Figure 5.13: Cost Decomposition Input C2

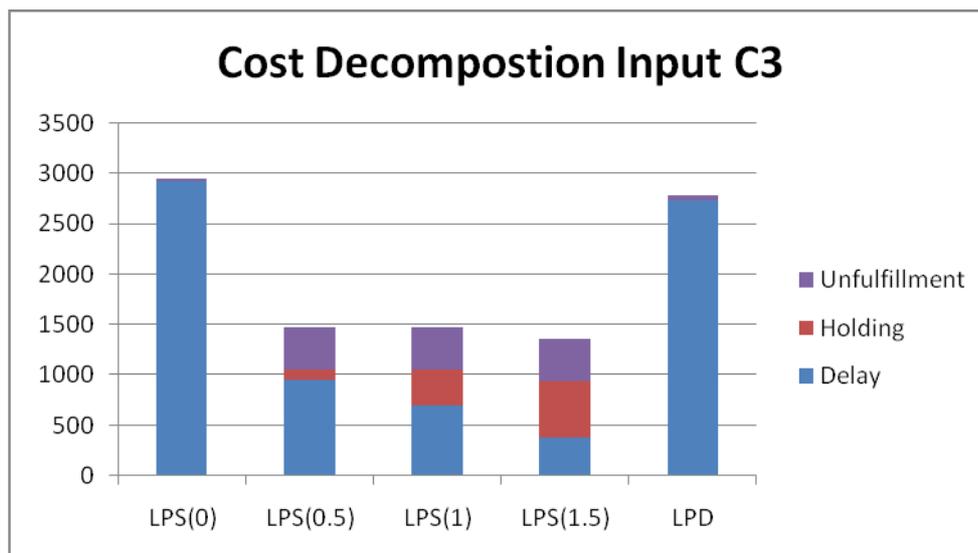


Figure 5.14: Cost Decomposition Input C3

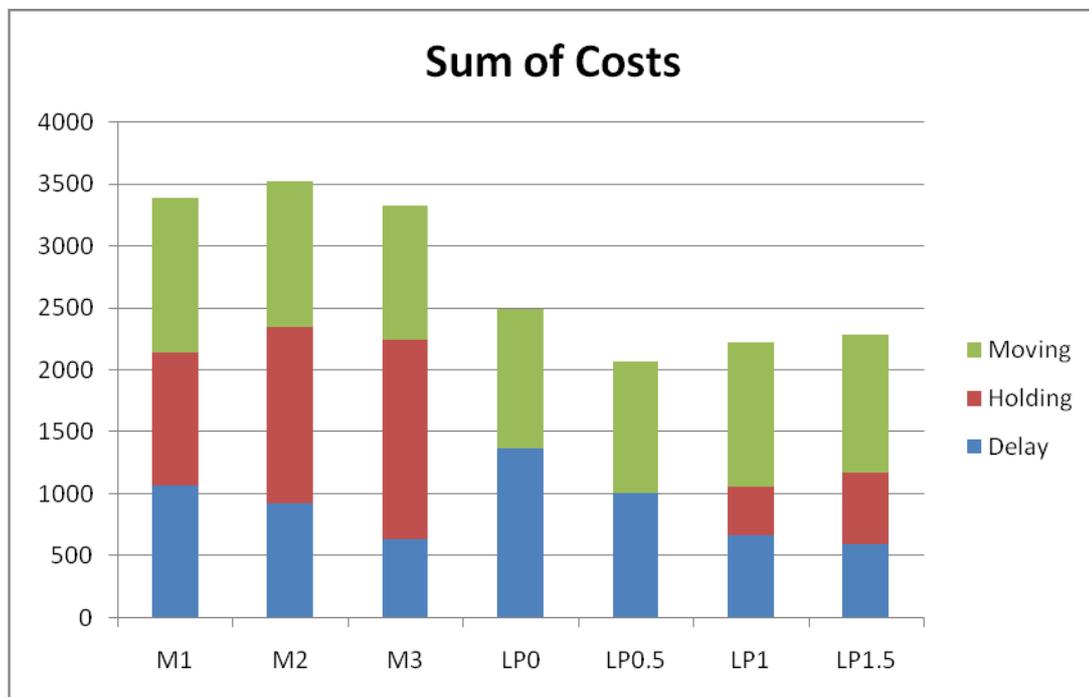


Figure 5.15: Sum of Costs for 7 Solving Methods

# Chapter 6

## Conclusions and Future Research

### 6.1 Summary

First of all in this study, a first linear program has been designed with the ability to use data extracted from information system. This linear program suggests substitutions of railcars category and minimizes moving costs and delay, substitution and unfulfillment penalties. Then, this linear program has been adapted to a decision-making process close to the schedulers' one in order to limit the changes in habits. In a third step, the linear program has been adapted to take into account the uncertainty on travel time by adding slack periods at the end of moves if a delay is probable and its financial consequences important. Then, a cursor has been implemented to modulate the rate of robustness. Finally, key aspects of this model (as the cursor or the stochastic approach interest) have been tested with a schedule simulator and the potential savings has been evaluated with data randomly generated.

All objectives set up by SNCF have been completed and the results on randomly generated data enable us to be optimistic for application on real data. The prototype is currently made

of 5 units:

- A linear program coded in OPL language (600 lines)
- A run script (the decision-making process) coded with ILOG language (1685 lines)
- A data generator developed in JAVA language (500 lines)
- A schedule simulator coded in JAVA language (2500 lines)
- A graph generator coded in JAVA language using ILOG Jviews libraries (distributed by ILOG)

The linear program and the run script have been designed with the OPL-IDE software (which work with CPLEX engine) distributed in France by ILOG.

As a prototype, this model generates satisfactory results. The stochastic approach presented in this thesis should be used to treat a similar problem dealing with ballast shipment for railroad building site. Moreover, this prototype opens a wide range of possible improvements (detailed in the next Section). The opportunity to explore them will highly depend on users' feedback and real contribution. Moreover, the Freight branch is currently restructured. It is then necessary to wait for the end of this long process before investing in further research. The next strategy and challenges faced by SNCF freight will then enable the Innovation & Research department of SNCF to chose the improvements to be carried out.

## 6.2 Perspectives for Future Research

Research and Innovation are a priority at SNCF, the will to anticipate customers' future needs impulses and the exploration of many paths of improvement. This research on the

empty railcar repositioning considering travel time uncertainty is one of those paths. Two main ideas have guided this research: the necessity to adapt the tool to the current methods and to the environment **and** the short duration of the project (8 months). Moreover, the result of the freight branch for first quarter 2007, below the objectives, lead SNCF to re-define the freight priorities. Thus, the Innovation & Research department has carried-out alone the project in a prospective way. Therefore, to design an efficient linear program in the given time, it has been necessary to set a trade-off between the level of details and the complexity of the model; not considered decisions have to be made by the scheduler. Then, complementary research could deal with the implementation of blocking strategies [20, 6]. Indeed, it is often more economical to group railcars following a similar route even if this means delivering them earlier. Moreover, because of organizational and safety considerations, trains cannot leave a station on-demand. Moves of railcars have to enforce a previously designed train plan. Using this plan as an input [7] could be a very interesting improvement from a workload viewpoint. The control of equity could also be delegated to the optimization tool by ensuring, for instance, an equal rate of unfulfillment for every customer [20] or by computing penalties automatically regarding to historical data about delivery incidents. The myopic approach of the model (see section 3.4.2) could also be improved by implementing a look-ahead feature like the rolling horizon method [20] or by attributing a salvage value to railcars at the end of the horizon depending on their location [8]. Finally, it could be interesting to evaluate the financial impact of imposing a deadline to the customers for railcar returns. This could improve the visibility on railcar inventory and, then, make the implementation of a look-ahead feature easier.

# Bibliography

- [1] J. F. Cordeau, P. Toth, and D. Vigo. A survey of optimisation models for train routing and scheduling. *Transportation Science*, 32:380–404, 1998.
- [2] T. G. Crainic and G. Laporte. Planning models for freight transportation. *European Journal of Operational Research*, 97:409–438, 1997.
- [3] A. J. Davenport and J. C. Beck. A survey of techniques for scheduling with uncertainty. Unpublished Manuscript, 2000.
- [4] P. J. Dejax and T. G. Crainic. A review of empty flows and fleet management models in freight transportation. *Transportation Science*, 21:227–247, 1987.
- [5] H. Herren. Computer-controlled empty wagon distribution on the sbb. *Rail International*, 8:25–32, 1977.
- [6] K. Hölmberg, M. Joborn, and J. T. Lundgren. Improved empty freight car distribution. *Transportation Science*, 32:163–173, 1998.
- [7] M. Joborn, T. G. Crainic, M. Gendreau, K. Hölmberg, and J. T. Lundgren. Economies of scale in empty freight car distribution in scheduled railways. *Transportation Science*, 38:121–134, 2004.
- [8] W. C. Jordan and M. A. Turnquist. A stochastic, dynamic network model for railroad car distribution. *Transportation Science*, 17:123–145, 1983.
- [9] O. K. Kwon, C. D. Martland, and J. M. Sussman. Routing and scheduling and heterogeneous freight car traffic on rail networks. *Transportation Research*, 34:101–115, 1998.
- [10] V. J. Leon, S. D. Wu, and R. H. Storer. Robustness measures and robust scheduling for job shops. *IIE Transactions*, 26:32–43, 1994.
- [11] K. N. McKay, F. R. Safayeni, and J. A. Buzacott. Job-shop scheduling theory : What is relevant ? *Interfaces*, 18:84–90, 1998.

- [12] S. C. Misra. Linear programming of empty wagon disposition. *Rail International*, 3:151–158, 1972.
- [13] C. E. Philip and J. M. Sussman. Inventory model of the railroad empty car distribution. AAR Report R-287, prepared under the AAR-FRE Freight Car Utilization Program, 1978.
- [14] W. B. Powell, A. Marar, J. Gelfand, and S. Bowers. Implementing real-time optimization models : A case application from motor carrier industry. *Operations Research*, 50:571–581, 2002.
- [15] W. B. Powell and J. A. Shapiro. A representational paradigm for dynamic resource transformation problem. *Annals of Operations Research*, 104:231–279, 2001.
- [16] W. B. Powell, J. A. Shapiro, and H. P. Simao. A dynamic management of heterogeneous resources. Technical Report SOR-98-06, Department of Civil Engineering and Operations Research, Princeton University, 1998.
- [17] W. B. Powell and H. Topaloglu. Fleet management, June 2002. Department of Operations Research and Financial Engineering, Princeton University.
- [18] N. Sadeh, S. Otsuka, and R. Schelback. Predictive and reactive scheduling with the microboss production scheduling and control system. In *Proceedings of the IJCAI-93 Workshop on Knowledge-Based Production, Planning, Scheduling, and Control*, pages 293–306, 1993.
- [19] H. El Sakkout, T. Richards, and M. Wallace. Unimodular probing for minimal perturbation in dynamic resource feasibility problems. In *Proceedings of the CP-97 Workshop on Dynamic Constraint Satisfaction*, 1997.
- [20] H.D. Sherali and A.B. Suharko. A tactical decision support system for empty railcar management. *Transportation Science*, 32:306–329, 1998.
- [21] A. Strobbe. Optimisation de la gestion des wagons vides. Technical report, SNCF, Direction de l’Innovation et de la Recherche, Unité Génie Décisionnel Appliqué, 2003.
- [22] H. Topaloglu and W. B. Powell. A distributed decision-making structure for dynamic allocation using nonlinear functional approximations. *Operations Research*, 53:281–297, 2003.
- [23] W. W. White and A. M. Bomberault. A network algorithm for empty rail management. *IBM System Journal*, 8:147–169, 1969.
- [24] M. Zweben, B. Daun, E. Davis, and M. Deale. *Scheduling and Rescheduling with Iterative Repair*, chapter 8, pages 241–256. Morgan Kaufmann, San Francisco, 1994.

# Appendix A : Deterministic Linear Program $LP_{glob}$

$$TCM = \sum_{t \in \mathcal{T}} \sum_{\substack{i \in \mathcal{I} \\ t_i \neq t}} \sum_{c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\}} \sum_{\substack{d \in \mathcal{H} \\ d \geq d_i}} N_{tr}(t, t_i, c, d, d + \Delta T(t, t_i)) \times C_{move}(t, t_i, c)$$

$$TUC = \sum_{i \in \mathcal{I}} \left( n_i - allocation(i) \right) \times C_{uff}(i)$$

$$TSC = \sum_{i \in \mathcal{I}} \sum_{\substack{d \in \mathcal{H} \\ d \geq d_i}} \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \times C_{sub}(c_i, c, u_i)$$

$$TCD = \sum_{i \in \mathcal{I}} C_{delay}(i) \times \sum_{\substack{d \in \mathcal{H} \\ d_i > d}} \left( \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right) \times (d - d_i) \right)$$

$$\min z = W_{move} \cdot TCM + W_{uff} \cdot TUC + W_{sub} \cdot TSC + W_{delay} \cdot TCD$$

Subject To

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C},$$

$$N_{av}(t, c, d^0) = inventory(t, c, d^0)$$

$$- \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t}} N_{tr}(t, i, c, d^0, d^0 + \Delta T(t, t_i))$$

$$- \sum_{i \in \mathcal{I}(t, c)} Q_{all}(i, d^0) - \sum_{\substack{i \in \mathcal{I}(t) \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}(i, c, d^0)$$

$$\begin{aligned}
& \forall t \in \mathcal{I}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H} \setminus \{d^0\}, \\
& N_{av}(t, c, d) = N_{av}(t, c, d - 1) + \text{inventory}(t, c, d) \\
& \quad + \sum_{\substack{i \in \mathcal{I}(t) \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t \neq t' \\ d - \Delta T(t', t_i) \geq d^0}} N_{tr}(t', i, c, d - \Delta T(t', t_i), d) \\
& \quad - \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t \\ d + \Delta T(t, t_i) \leq d^f}} N_{tr}(t, i, c, d, d + \Delta T(t, t_i)) \\
& \quad - \sum_{\substack{i \in \mathcal{I}(t, c) \\ d \geq d_i}} Q_{all}(i, d) - \sum_{\substack{i \in \mathcal{I}(t) \\ d \geq d_i \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}(i, c, d)v
\end{aligned}$$

$$\forall i \in \mathcal{I},$$

$$\text{allocation}(i) = \text{prevAllocations}(i) + \sum_{d \in \mathcal{H}} \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right)$$

$$\forall i \in \mathcal{I},$$

$$n_i \geq \text{allocation}(i) \tag{6.1}$$

$$\forall i \in \mathcal{I}, d \geq d_i$$

$$Q_{all}(i, d) \geq \sum_{\substack{t \in \mathcal{I} \\ d - \Delta T(t, t_i) \geq d^0}} N_{tr}(t, i, c_i, d - \Delta T(t, t_i), d)$$

$$\forall i \in \mathcal{I}, d \geq d_i, \forall c \in \mathcal{R}_{sub}(u_i, c_i),$$

$$Q_{sub}(i, c, d) \geq \sum_{\substack{t \in \mathcal{I} \\ d - \Delta T(t, t_i) \geq d^0}} N_{tr}(t, i, c, d - \Delta T(t, t_i), d)$$

$$\forall c \in \mathcal{C}, \forall d \in \mathcal{H}, \forall t \in \mathcal{T}, \forall i \in \mathcal{I} \text{ with } t \neq t_i \text{ and } d + \Delta T(t, t_i) \leq d^f \\ N_{tr}(t, i, c, d, d + \Delta T(t, t_i)) \in \mathbb{R}^+$$

$$\forall i \in \mathcal{I}, d \in \mathcal{H}, d \geq d_i, Q_{all}(i, d) \in \mathbb{R}^+$$

$$\forall i \in \mathcal{I}, \forall c \in \mathcal{R}_{sub}(u_i, c_i), d \in \mathcal{H}, d \geq d_i, Q_{sub}(i, c, d) \in \mathbb{R}^+$$

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H}, N_{av}(t, c, d) \in \mathbb{R}^+$$

$$\forall i \in \mathcal{I}, allocation(i) \in \mathbb{R}^+$$

# Appendix B : Stochastic Linear Program $LP_{stoc}$

$$TCM = \sum_{t \in \mathcal{T}} \sum_{\substack{i \in \mathcal{I} \\ t_i \neq t}} \sum_{c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\}} \sum_{\substack{d \in \mathcal{H} \\ d \geq d_i}} N_{tr}(t, t_i, c, d, d + \Delta T(t, t_i)) \times C_{move}(t, t_i, c)$$

$$TUC = \sum_{i \in \mathcal{I}} (n_i - allocation(i)) \times C_{uff}(i)$$

$$TSC = \sum_{i \in \mathcal{I}} \sum_{\substack{d \in \mathcal{H} \\ d \geq d_i}} \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \times C_{sub}(c_i, c, u_i)$$

$$TCD = \sum_{i \in \mathcal{I}} C_{delay}(i) \times \sum_{\substack{d \in \mathcal{H} \\ d_i > d}} \left( \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right) \times (d - d_i) \right)$$

$$TEAC = \sum_{\varphi \in \Phi} N_{tr}(\varphi) \times C_a(\varphi)$$

where

$$C_a(\varphi) = \sum_{\sigma_r=0}^{D_{max}} C_{\Delta}(\varphi, \sigma_r) \times p(\sigma_r, t, t_i)$$

and

$$C_{\Delta}(s(\varphi, \sigma_r)) = (2 - \gamma_{rob}) \times C_{hold}(c, t_i) \times \max(0, \sigma_{lim}(\varphi) - \sigma_r) \\ + \gamma_{rob} \times C_{delay}(i) \times \left[ \max(0, \sigma_r - \sigma_{lim}(\varphi)) - \max(0, \sigma_a - \sigma_{lim}(\varphi)) \right]$$

$$\min z = W_{move} \cdot TCM + W_{uff} \cdot TUC + W_{sub} \cdot TSC + W_{delay} \cdot TCD + W_{anticipation} \cdot TEAC$$

Subject To

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C},$$

$$N_{av}(t, c, d^0) = inventory(t, c, d^0)$$

$$\begin{aligned} & - \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t}} \sum_{\substack{\sigma_a=0 \\ d^0 + \Delta(t, t_i) + \sigma_a \leq d^f}}^{D_{max}} N_{tr}(t, i, c, d^0, d^0 + \Delta T(t, t_i) + \sigma_a) \\ & - \sum_{i \in \mathcal{I}(t, c)} Q_{all}(i, d^0) - \sum_{\substack{i \in \mathcal{I}(t) \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}(i, c, d^0) \end{aligned}$$

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H} \setminus \{d^0\},$$

$$N_{av}(t, c, d) = N_{av}(t, c, d - 1) + inventory(t, c, d)$$

$$\begin{aligned} & + \sum_{\substack{i \in \mathcal{I}(t) \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t'}} \sum_{\substack{\sigma_a=0 \\ d - \Delta T(t, t_i) - \sigma_a \geq d^0}}^{D_{max}} N_{tr}(t', i, c, d - \Delta T(t, t_i) - \sigma_a, d) \\ & - \sum_{\substack{i \in \mathcal{I} \\ c \in \mathcal{R}_{sub}(u_i, c_i) \cup \{c_i\} \\ t_i \neq t}} \sum_{\substack{\sigma_a=0 \\ d + \Delta T(t, t_i) + \sigma_a \leq d^f}}^{D_{max}} N_{tr}(t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \\ & - \sum_{\substack{i \in \mathcal{I}(t, c) \\ d \geq d_i}} Q_{all}(i, d) - \sum_{\substack{i \in \mathcal{I}(t) \\ d \geq d_i \\ c \in \mathcal{R}_{sub}(u_i, c_i)}} Q_{sub}(i, c, d) \end{aligned}$$

$$\forall i \in \mathcal{I},$$

$$allocation(i) = prevAllocations(i) + \sum_{d \in \mathcal{H}} \left( Q_{all}(i, d) + \sum_{c \in \mathcal{R}_{sub}(u_i, c_i)} Q_{sub}(i, c, d) \right)$$

$$\begin{aligned} \forall i \in \mathcal{I}, \\ n_i \geq \text{allocation}(i) \end{aligned}$$

$$\forall i \in \mathcal{I}, \forall d \in \mathcal{H}$$

$$Q_{all}(i, d) \geq \sum_{t \in \mathcal{T}} \sum_{\substack{D_{max} \\ \sigma_a=0 \\ d - \Delta T(t, t_i) - \sigma_a \geq d^0}} N_{tr}(t, i, c_i, d - \Delta T(t, t_i) - \sigma_a, d)$$

$$\forall i \in \mathcal{I}, \forall d \in \mathcal{H}, \forall c \in \mathcal{R}_{sub}(u_i, c_i),$$

$$Q_{sub}(i, c, d) \geq \sum_{t \in \mathcal{T}} \sum_{\substack{D_{max} \\ \sigma_a=0 \\ d - \Delta T(t, t_i) - \sigma_a \geq d^0}} N_{tr}(t, i, c, d - \Delta T(t, t_i) - \sigma_a, d)$$

$$\forall c \in \mathcal{C}, \forall d \in \mathcal{H}, \sigma_a \in [0, \dots, D_{max}] \forall t \in \mathcal{T}, \forall i \in \mathcal{I}$$

with  $t \neq t_i$

$$N_{tr}(t, i, c, d, d + \Delta T(t, t_i) + \sigma_a) \in \mathbb{R}^+$$

$$\forall i \in \mathcal{I}, d \in \mathcal{H}, d \geq d_i, Q_{all}(i, d) \in \mathbb{R}^+$$

$$\forall i \in \mathcal{I}, \forall c \in \mathcal{R}_{sub}(u_i, c_i), d \in \mathcal{H}, d \geq d_i, Q_{sub}(i, c, d) \in \mathbb{R}^+$$

$$\forall t \in \mathcal{T}, \forall c \in \mathcal{C}, \forall d \in \mathcal{H}, N_{av}(t, c, d) \in \mathbb{R}^+$$

$$\forall i \in \mathcal{I}, \text{allocation}(i) \in \mathbb{R}^+$$

## Appendix C : Simulator Source

```
import java.util.*;

public class Simulation
{
/**
 * This class simulate the schedule processing from
 * a Class SimulateurFactory input
 */
/**
 * Contains CPLEX results
 */
SimulateurFactory simFac;
/**
 * departure Calendar from simFac
 */
ArrayList<ArcDeplacement>[] calendrierDepart;
/**
 * Arrival Calendar from simFac
 */
ArrayList<ArcDeplacement>[] calendrierArrivee;
//taille nbPeriodes+1 derniere case pour les commandes
arrivant en dehors de l'horizon
/**
 * Allocation calendar from simFac
 */
ArrayList<ArcAffectation>[] calendrierAffectation;
/**
 * Result indicators:
 * First index
 * - 0 for estimated results
 * - 1 for real results
 * The second index is the order number
```

```
* Third index:
*- 0 for on time railcars
*- 1 for late railcars
*- 2 for substituted railcars
*- 3 Not-delivered railcars
*- 4 Ordered railcars
*- 5 early railcars
*/
double[] [] [] wagonsLivres;
/**
 * Results indicators:
 * Number of order for which all railcars are on time
 */
double[] nbCdeALheure;
/**
 * Results indicators:
 * Number of order for which any railcars is late
 */
double[] nbCdeRetard;
/**
 * sum of duration delay
 */
double[] delaiRetard;
/**
 * sum of delay penalties
 */
double[] penalites;
/**
 * number of unfulfilled orders
 */
double[] nbCdeNonLivree;
/**
 * sum of anticipated delays
 */
double[] sigma;
/**
 * rate of fulfilled orders
 */
double[] txSatis;
/**
 * rate of late orders
 */
```

```
double[] txRetard;
/**
 * rate of on-time order
 */
double[] txALheure;
/**
 * sum of unfulfilled cost
 */
double[] coutNonSatisfaction;

/**
 * sum of moving cost
 */
double coutDeplacement;
/**
 * sum of substituted orders
 */
double nbCdeSubstituees;
/**
 * sum of holding costs
 */
double immobilisations;
/**
 * sum od advance duration
 */
double delaiAvance;
/**
 * number of early railcar
 */
double qteAvance;
/**
 * rate of substituted moves
 */
double txSubstituees;
/**
 * rate of on-time moves
 */
double txDplctALheure;
/**
 * rate of late moves
```

```
    */
double txDplctRetard;
/**
 * rate of in-advance moves
 */
double txDplctAvance;
/**
 * rate of in-advance orders
 */
double txAvance;

/**
 * number of in-advance moves
 */
int nbDeplacementsEnAvance;
/**
 * number of on-time moves
 */
int nbDeplacementsALHeure;
/**
 * number of late moves
 */
int nbDeplacementsEnRetard;
/**
 * number of processed moves
 */
int nbDeplacements;
/**
 * number of in-advance orders
 */
int nbCdeAvance;
/**
 * number of currently in-transit railcars
 */
int[] enTransit;

/**
 * constructor
 * @param simFac SimulateurFac
 */
public Simulation(SimulateurFactory simFac)
```

```
{
this.simFac=simFac;
initialize();
check();
}
/**
 * check if simFac has been built with
 * SimulateurFactory.build()
 * @see SimulateurFactory#build()
 *
 */
public void check()
{
if(!this.simFac.isBuilt())
System.out.println("ATTENTION SIMFACTORY
NON CONSTRUIT!!!!");
}
/**
 * initialisation of instance variables
 *
 */
private void initialize()
{
calendrierDepart =
simFac.getCalendrierDepart();
calendrierArrivee =
simFac.getCalendrierArrivee();
calendrierAffectation =
simFac.getCalendrierAffectation();
wagonsLivres=
new double[2][simFac.getData().getNbCommandes()][6];
nbCdeALheure=new double[2];
nbCdeRetard=new double[2];
delaiRetard=new double[2];
penalites=new double[2];
nbCdeNonLivree=new double[2];
sigma=new double[2];
txSatis=new double[2];
txRetard=new double[2];
txALheure=new double[2];
coutNonSatisfaction = new double[2];
coutDeplacement=0;
```

```
enTransit=
new int[simFac.getData().getNbCommandes()];
nbCdeSubstituees=0;
immobilisations=0;
delaiAvance=0;
qteAvance=0;
txSubstituees=0;
txDplctALheure=0;
txDplctRetard=0;
txDplctAvance=0;
```

```
}
```

```
/**
 * @return the calendrierAffectation
 */
public ArrayList<ArcAffectation>[]
getCalendrierAffectation() {
return calendrierAffectation;
}
```

```
/**
 * @return the calendrierArrivee
 */
public ArrayList<ArcDeplacement>[]
getCalendrierArrivee() {
return calendrierArrivee;
}
```

```
/**
 * @return the calendrierDepart
 */
public ArrayList<ArcDeplacement>[]
getCalendrierDepart() {
return calendrierDepart;
}
```

```
/**
 * @return the simFac
```

```
    */
public SimulateurFactory getSimFac() {
return simFac;
}

/**
 * @param calendrierAffectation the
 * calendrierAffectation to set
 */
public void setCalendrierAffectation(
ArrayList<ArcAffectation>[] calendrierAffectation) {
this.calendrierAffectation = calendrierAffectation;
}

/**
 * @param calendrierArrivee the calendrierArrivee to set
 */
public void setCalendrierArrivee(
ArrayList<ArcDeplacement>[] calendrierArrivee) {
this.calendrierArrivee = calendrierArrivee;
}

/**
 * @param calendrierDepart the calendrierDepart to set
 */
public void setCalendrierDepart(ArrayList<
ArcDeplacement>[] calendrierDepart) {
this.calendrierDepart = calendrierDepart;
}

/**
 * @param simFac the simFac to set
 */
public void setSimFac(SimulateurFactory simFac) {
this.simFac = simFac;
}
```

```
/**
 * Launch the simulation and compute results
 * indicators following the flow chart
 * @param verbose display messages
 * @return true if there were no error
 */
public boolean start(boolean verbose)
{
//populate the table enTransit
Iterator<ArcDeplacement> iDplct=
simFac.getOut().getDeplacements().iterator();
while(iDplct.hasNext())
{
ArcDeplacement iDep=iDplct.next();
enTransit[iDep.getCommandeCibleId()]+=iDep.getQuantite();
}

ArrayList<ArcDeplacement> departsDuJour=
new ArrayList<ArcDeplacement>();
ArrayList<ArcDeplacement> arriveesDuJour=
new ArrayList<ArcDeplacement>();
ArrayList<ArcAffectation> affectationsDuJour=
new ArrayList<ArcAffectation>();

for(int j=simFac.getData().getj0();
j<=simFac.getData().getjf();j++)
{

if(verbose)
System.out.println("JOUR : "+j);

//##### Moves Processing
```

```
//Departures
departsDuJour=(ArrayList)
simFac.getCalendrierDepart(j).clone();

Iterator<ArcDeplacement> iterDep=
departsDuJour.listIterator();
while(iterDep.hasNext())
{
String out="";
ArcDeplacement iDep=iterDep.next();

if(simFac.getStock(iDep.getSiteDep(),
iDep.getCategorie(), j)>=iDep.getQuantite())
{
//If Inventory OK, then substract railcars
simFac.removeStock(iDep.getSiteDep(),
iDep.getCategorie(), j, iDep.getQuantite());
//set departure date

iDep.setJdep(j);
out+="dpart de "+ iDep.getQuantite()+
" wagons de "+iDep.getCategorie()
+" depuis le "+iDep.getSiteDep().getNom()+
" pour le "+iDep.getSiteCible().getNom()
+" de la commande "+iDep.getCommandeCibleId() ;

//Draw arrival date

double rnd=Math.random();
int retard=0;
boolean stop=false;
while(!stop)
{
double proba=0;
for(int i=0;i<=retard;i++)
{
proba+=simFac.getData().getProbaATemps(i,
iDep.getSiteDep(),
iDep.getSiteCible());
}
}
```

```
if(rnd<proba)
{
if(verbose)
System.out.println("rnd="+
Outils.round2(rnd)+"<"+Outils.round2(proba)
+"--> Retard="+retard);
stop=true;
}
else
{
retard++;
if(retard>simFac.getData().getRetardMax())
stop=true;
}
}
//set real arrival date
int jourArrivee=iDep.getJdep()+
simFac.getData().getTTransit(iDep.getSiteDep()
,iDep.getSiteCible()+retard;

if(jourArrivee>simFac.getData().getjf())
//if arrival date is the end of horizon
{
jourArrivee=simFac.getData().getjf()+1;
//if arrival date is out of horizon

}
iDep.setJarr(jourArrivee);
out+=" arrive relle "+ iDep.getJarr()+
" au lieu de "+iDep.getJarrPrev()+" prvu";
if(iDep.getJarrPrev()==iDep.getJarr())
//Change status
{
iDep.setStatut(2);
out+=" A L'HEURE";

}
if(iDep.getJarrPrev(>iDep.getJarr())
{
iDep.setStatut(1);
```

```
out+=" EN AVANCE";
//NOTE: arrival modified only in case of delay,
//otherwise, railcars added to stock could be used to
// process another order
calendrierArrivee[iDep.getJarrPrev()].remove(iDep);
calendrierArrivee[jourArrivee].add(iDep);
out+=" arrive arc dplacs de "
+iDep.getJarrPrev()+" "+jourArrivee;

}

if(iDep.getJarrPrev()<iDep.getJarr())
{
iDep.setStatut(-2);
out+=" EN RETARD";
//If jarr late, the arc is moved in the calendar
calendrierArrivee[iDep.getJarrPrev()].remove(iDep);
calendrierArrivee[jourArrivee].add(iDep);
out+=" arrive arc dplacs de "
+iDep.getJarrPrev()+" "+jourArrivee;

}

}
else
{
//if Inventory not OK, the departure is delayed (D+1)
iDep.setStatut(-3);
out+="IMPOSSIBLE : dpart de "+ iDep.getQuantite()+
" wagons de "+iDep.getCategorie()+" depuis le "
+iDep.getSiteDep().getNom()+" stock="
+simFac.getStock(iDep.getSiteDep(),iDep.getCategorie(), j);

int newJourDepart;
if(j+1<simFac.getData().getjf())
{
newJourDepart=j+1;
}
}
```

```

else
{
newJourDepart=simFac.getData().getjf()+1;
}
calendrierDepart[j].remove(iDep);
calendrierDepart[newJourDepart].add(iDep);

}
if(verbose)
System.out.println(out);
}
//ARRIVALS
arriveesDuJour=(ArrayList)
simFac.getCalendrierArrivee(j).clone();

Iterator<ArcDeplacement> iterArr= arriveesDuJour.iterator();

while(iterArr.hasNext())
{
String out="";
ArcDeplacement iArr=iterArr.next();
simFac.addStock(iArr.getSiteCible(),
iArr.getCategorie(),j,iArr.getQuantite());
out+="Arrive de "+ iArr.getQuantite()+
" wagons de "+iArr.getCategorie()+
" sur le site "+iArr.getSiteCible().getNom()+
" depuis le "+iArr.getSiteDep().getNom()+
" pour la commande"+ iArr.getCommandeCibleId() ;
if(verbose)
System.out.println(out);
enTransit[iArr.getCommandeCibleId()-]=iArr.getQuantite();
if(j<iArr.getJarrPrev() && iArr.getJarrPrev()
>iArr.getCommandeCible().getDueDate() &&
enTransit[iArr.getCommandeCibleId()]==0)
{
//If railcars have arrived, process allocation
PLOutput tempOut = simFac.getOut();
while(tempOut.existeAffectation(
iArr.getCommandeCible()))
{
ArcAffectation arcADeplacer =
tempOut.rechercheAffectation(

```

```
iArr.getCommandeCible());
tempOut.retirerAffectation(arcADeplacer);
calendrierAffectation[arcADeplacer.getJPrev()]
.remove(arcADeplacer);
arcADeplacer.setJ(j);
calendrierAffectation[j].add(arcADeplacer);
System.out.println("affectation de "+
arcADeplacer.getCommandeCibleId()+
" avance de "+arcADeplacer.getJPrev()+
" "+arcADeplacer.getJ() );
}
}
}

//##### ALLOCATIONS PROCESSING
affectationsDuJour=(ArrayList)
simFac.getCalendrierAffectation(j).clone();

Iterator<ArcAffectation> iterAff=
affectationsDuJour.iterator();
while(iterAff.hasNext())
{
String out="";
ArcAffectation iAff=iterAff.next();
if(simFac.getStock(iAff.getSite(),
iAff.getCategorie(), j)>=iAff.getQuantite())
{
//If inventory OK
out+="affectation de "+iAff.getQuantite()+
" wagons de "+iAff.getCategorie()+" pour la commande "
+iAff.getCommandeCible().getId()+" sur le site "
+iAff.getSite().getNom();
//set allocation date
iAff.setJ(j);
//railcars substracted from inventory
simFac.removeStock(iAff.getSite(),
iAff.getCategorie(),j,iAff.getQuantite());
//railcars allocated are reported
```

```
if(iAff.getJPrev()==iAff.getJ())
{
iAff.setStatut(1);
out+="A L'HEURE";

//computation of delay penalties

}
if(iAff.getJPrev()<iAff.getJ())
{
iAff.setStatut(-2);
out+="EN RETARD";
}
if(iAff.getJPrev()>iAff.getJ())
{
iAff.setStatut(+2);
out+="EN AVANCE";
}

}
else
{
//if Inventory not OK : delay (D+1)
calendrierAffectation[j].remove(iAff);
int newJourAff;
if(j+1>simFac.getData().getNbPeriodes())
{
//if Out of horizon, put it in last case
newJourAff=simFac.getData().getNbPeriodes()+1;

//Railcars are then denoted as not allocated

}
else
{
newJourAff=j+1;
}
calendrierAffectation[newJourAff].add(iAff);
out+="IMPOSSIBLE : affectation de "+iAff.getQuantite()
+" wagons de "+iAff.getCategorie()+" pour la commande "
```

```
+iAff.getCommandeCible().getId()+" sur le site "  
+iAff.getSite().getNom()+" stock="  
+simFac.getStock(iAff.getSite(), iAff.getCategorie(), j);  
out+=" dcale de "+j+" "+newJourAff;  
iAff.setStatut(-1);  
  
}  
if(verbose)  
System.out.println(out);  
}  
}  
System.out.println("SIMULATION OVER");  
return true;  
}  
/**  
 * Extract simulation results  
 * @return true if there were no error during simulation  
 */  
public boolean extraireResultats()  
{  
  
for(int i=0;i<getSimFac().getData().getNbCommandes();i++)  
{  
wagonsLivres[0][i][4]=getSimFac().getData()  
.getCommande(i).getQuantite();  
wagonsLivres[1][i][4]=getSimFac().getData()  
.getCommande(i).getQuantite();  
}  
List<ArcAffectation> affectations =  
simFac.getOut().getAffectations();  
Iterator<ArcAffectation> iterAff = affectations.iterator();  
ArcAffectation iAff;  
while(iterAff.hasNext())  
{  
iAff=iterAff.next();  
  
  
if(iAff.getJ()==iAff.getCommandeCible().getDueDate())  
{  
wagonsLivres[1][iAff.getCommandeCibleId()][0]  
+=iAff.getQuantite();
```

```
}
```

```
if(iAff.getJ()>iAff.getCommandeCible().getDueDate()  
&& iAff.getJ()!=simFac.getData().getNbPeriodes())  
{  
  wagonsLivres[1][iAff.getCommandeCibleId()][1]  
  +=iAff.getQuantite();
```

```
  delaiRetard[1]+=iAff.getJ()  
  -iAff.getCommandeCible().getDueDate();  
  penalites[1]+=Math.max((iAff.getJ()  
  -iAff.getCommandeCible().getDueDate()),0)  
  *iAff.getCommandeCible().getCoutRetard()  
  *iAff.getQuantite();
```

```
}
```

```
if(!iAff.getCategorie()  
.equals(iAff.getCommandeCible().getCategorie()))  
{
```

```
  wagonsLivres[0][iAff.getCommandeCibleId()][2]  
  +=iAff.getQuantite();
```

```
}
```

```
if(iAff.getJPrev()==  
iAff.getCommandeCible().getDueDate())  
{  
  wagonsLivres[0][iAff.getCommandeCibleId()][0]  
  +=iAff.getQuantite();
```

```
  delaiRetard[0]+=iAff.getJPrev()  
  -iAff.getCommandeCible().getDueDate();
```

```
}
```

```
if(iAff.getJPrev()>iAff.getCommandeCible().getDueDate())  
{
```

```

wagonsLivres[0][iAff.getCommandeCibleId()][1]
+=iAff.getQuantite();
penalites[0]+=Math.max((iAff.getJPrev()
-iAff.getCommandeCible().getDueDate()),0)
*iAff.getCommandeCible().getCoutRetard()
*iAff.getQuantite();

}
if(iAff.getJ()==simFac.getData().getNbPeriodes())
{
wagonsLivres[1][iAff.getCommandeCibleId()][3]
+=iAff.getQuantite();

}
if(iAff.getJ(<iAff.getCommandeCible().getDueDate())
{
wagonsLivres[1][iAff.getCommandeCibleId()][5]
+=iAff.getQuantite();

}

}

List<ArcDeplacement> deplacements
= simFac.getOut().getDeplacements();
Iterator<ArcDeplacement> iterDep
= deplacements.iterator();
ArcDeplacement iDep;
while(iterDep.hasNext())
{
iDep=iterDep.next();
nbDeplacements++;
this.coutDeplacement+=iDep.getQuantite()
*simFac.getData().getCoutDplct(iDep.getCategorie(),
iDep.getSiteDep(), iDep.getSiteCible());

if(iDep.getJarr(<iDep.getJarrPrev())
{
immobilisations+=Math.max(
(iDep.getCommandeCible().getDueDate()
-iDep.getJarr()),0)*simFac.getData().getCoutImmo(
iDep.getSiteCible(),iDep.getCategorie())

```

```

*iDep.getQuantite();
delaiAvance+=iDep.getCommandeCible().getDueDate()
-iDep.getJarr();
this.nbDeplacementsEnAvance++;
this.delaiAvance+=(iDep.getJarrPrev()-iDep.getJarr());
this.qteAvance+=iDep.getQuantite();
sigma[0]+=iDep.getJarrPrev()-iDep.getJdepPrev()
-simFac.getData().getTTransit(iDep.getSiteDep(),
iDep.getSiteCible());
sigma[1]+=iDep.getJarr()-iDep.getJdep()
-simFac.getData().getTTransit(iDep.getSiteDep(),
iDep.getSiteCible());
//compute moving cost

//compute holding cost
this.immobilisations+=Math.max(
(iDep.getCommandeCible().getDueDate()-iDep.getJarr()),0)
*simFac.getData().getCoutImmo( iDep.getSiteCible()
,iDep.getCategorie()*iDep.getQuantite());

//compute delay cost
//this.penalites[1]-=Math.max(iDep.getJarrPrev()
-iDep.getCommandeCible().getDueDate(),0)
*iDep.getCommandeCible().getCoutRetard()
*iDep.getQuantite();

}
if(iDep.getJarr()==iDep.getJarrPrev())
{
this.nbDeplacementsALHeure++;
}
if(iDep.getJarrPrev()<iDep.getJarr())
{
iDep.setStatut(-2);
this.nbDeplacementsEnRetard++;
}
}
}

```

```

for(int i=0;i<simFac.getData().getNbCommandes();i++)
{//0:A 1'heure, 1:en retard, 2:Substitus,
  3:non Livrs, 4:wagons commands
for(int k=0;k<=1;k++)
{
//comptage OK
if(wagonsLivres[k][i][0]==wagonsLivres[0][i][4])
this.nbcdeALheure[k]++;
//comptage retard
if(wagonsLivres[k][i][1]>0)
this.nbcdeRetard[k]++;
//comptage non livraison
if(wagonsLivres[k][i][3]>0)
{
this.nbcdeNonLivree[k]++;
}
//solde de wagons pour la commande en cours
double temp=wagonsLivres[k][i][4]
-(wagonsLivres[k][i][0]+wagonsLivres[k][i][1]
+wagonsLivres[k][i][5]);
if(temp>0)
this.coutNonSatisfaction[k]
+=simFac.getData().getCommande(i).getCoutNonSatis()*temp;

if(wagonsLivres[k][i][5]>0 && wagonsLivres[k][i][1]==0)
this.nbcdeAvance++;
}
//comptage substitution
if(wagonsLivres[0][i][2]>0)
this.nbcdeSubstituees++;

}

double[] nbCdeSatis = new double[2];

nbCdeSatis[0]=nbCdeRetard[0]+nbCdeALheure[0];
nbCdeSatis[1]=nbCdeRetard[1]+nbCdeALheure[1]+nbCdeAvance;

txSatis[0]=((double)nbCdeRetard[0]+(double)nbCdeALheure[0])

```

```

/(double)simFac.getData().getNbCommandes()*100;
txSatis[1]=((double)nbCdeRetard[1]+(double)nbCdeALheure[1])
/(double)simFac.getData().getNbCommandes()*100;

txRetard[0]=(double)nbCdeRetard[0]/(double)nbCdeSatis[0]*100;
txRetard[1]=(double)nbCdeRetard[1]/(double)nbCdeSatis[1]*100;

txSubstituees=(double)nbCdeSubstituees/(double)nbCdeSatis[0]*100;

txALheure[0]=(double)nbCdeALheure[0]/(double)nbCdeSatis[0]*100;
txALheure[1]=(double)nbCdeALheure[1]/(double)nbCdeSatis[1]*100;

txAvance = (double)nbCdeAvance/(double)nbCdeSatis[1]*100;

txDplctALheure = (double)nbDeplacementsALHeure
/(double)nbDeplacements*100;
txDplctRetard = (double)nbDeplacementsEnRetard
/(double)nbDeplacements*100;
txDplctAvance =(double)nbDeplacementsEnAvance
/(double)nbDeplacements*100;

return true;
}

/**
 * @return indicator resultes
 */
public String toString()
{
String out = "";
out += "tx Satis Prev : "+Outils.round(0,txSatis[0])+" reel : "
+Outils.round(0,txSatis[1])+"\n";
out += "tx Retard Prev : "+Outils.round(0,txRetard[0])+" reel : "
+Outils.round(0,txRetard[1])+"\n";
out += "tx Avance"+Outils.round(0,txAvance)+"tx Substituees : "
+Outils.round(0,txSubstituees)+"\n";
out += "txALHeure : "+Outils.round(0,txALheure[0])+" reel : "
+Outils.round(0,txALheure[1])+"\n";
out += "txDplctOK : "+Outils.round(0,txDplctALheure)+" txDplctRet : "
+Outils.round(0,txDplctRetard)+
" txDplctAv : "+Outils.round(0,txDplctAvance)+"\n";
out += "penalites Prev : "+penalites[0]+" reel : "+penalites[1]

```

```

+" immobilisation : "+immobilisations+"\n";
out += " cout non Satisfaction : "+this.coutNonSatisfaction[0]
+" reel : "+this.coutNonSatisfaction[1)+"\n";
out += "cout de dplacement : "+coutDeplacement+"\n";
return out;
}

/**
 *
 * @param i the line number, if i=-1 return column tags
 * @return indicator results in CSV format
 *
 */
public String toCSVLine(int i)
{
String out;
if(i===-1)
out = "#;Nb Dplct;tx retard Dplacement;
tx Avance Dplacement;tx OK Dplacement;
tx Satisfaction prev;tx Satisfaction reel;
tx Retard prev;tx Retard reel;tx Avance;
tx Substitution;tx A L'heure prev;
tx A L'heure reel;penalites prev;penalite reel;
 immobilisations";
else
{
out=i+";"+nbDeplacements+";"+
Outils.round(0,txDplctRetard)+
";" +Outils.round(0,txDplctAvance)+
";"+Outils.round(0,txDplctALheure)+ ";";

out+=Outils.round(0,txSatis[0])+
";"+Outils.round(0,txSatis[1])+";";
out+=Outils.round(0,txRetard[0])+
";"+Outils.round(0,txRetard[1])+";";
out+=Outils.round(0,txAvance)+
";"+Outils.round(0,txSubstituees)+" ";
out+= Outils.round(0,txALheure[0])+
";"+Outils.round(0,txALheure[1])+";";
out+= penalites[0]+";"+penalites[1]+
";"+immobilisations+" ";
+coutDeplacement+" ";
}
}

```

```
+this.coutNonSatisfaction[0]+";"  
+this.coutNonSatisfaction[1]+"\\n";  
}  
return out;  
  
}  
  
}
```