From a magnetic board to an interactive planning support system:

How to build a decision support system specific to a factory

Thesis

presented to the Faculty of Economics and Social Sciences
at the University of Fribourg (Switzerland)

by

Marc ULDRY
from Le Châtelard FR

in fulfilment of the requirements for the degree of
Doctor of Economics and Social Sciences

Accepted by the Faculty of Economics and Social Sciences
November 28th, 2012 at the proposal of

First supervisor: Prof. Marino Widmer
Decision Support & Operations Research
Department of Informatics
University of Fribourg

Second supervisor: Prof. Alain Hertz
Département de mathématiques et de génie industriel
École Polytechnique de Montréal

Fribourg, Switzerland, 2012

For my godchildren Camille and Samuel

Acknowledgments

First of all, I want to thank the company Ciments Vigier SA for having provided an enthralling industrial problem, which is the origin of this PhD thesis, and especially Misters Gaschen, Rion and Dysli for their collaboration and their interest, confidence, and availability throughout the carrying out of this work.

I am grateful for the regular monitoring and the valuable advices given by my first supervisor, Professor Marino Widmer. I also extend my warmest thanks to my second supervisor, Professor Alain Hertz, for having shared his knowledge and experience with relevant tips.

Thank you Yves, Oliver, Reinhard, Mila, Mariam, Micheal and Andreas, my colleagues, for your cheerfulness, the nice moments spent together, and the good ideas that have emerged during discussions.

Let me finally express my gratitude to my family and my friends for their support throughout this long journey.

Marc Uldry

Contents

Introduction

Nowadays, companies have to manage and handle a large amount of information related to their business processes. Regarding manufacturing companies, data have to be stored and processed on the one hand from the receipt of orders to their invoicing and on the other hand, from the supplying of raw materials to the delivery of finished goods, i.e. at each stage of the integrated production management. Illustrated in Figure 1 and described in detail in [58], this one presents the different modules that appear in a production system. With the development of the information technologies, data processing is more and more automated while the level of information granularity continues to increase. Indeed, the processors performance is constantly improved, as the capacity of data storage, and for an ever lower price due to a miniaturization of the components and to a mass production.

Only usable via command line some decades ago, most information systems are now user friendly and intuitive. With the constant development of mobile devices, like tactical tablets or smartphones, and the possibility to access information systems via remote connection, great potentials however still remain to improve or create efficient user interfaces. Taking advantage of this evolution, tools implementing techniques of operational research have also been developed in information systems to aid the users making decisions in their tasks.

For many years, global solutions designed for managing most of the activities of a company, called Enterprise Resource Planning, are available in the market. Although these solutions can sometimes be adapted, some specific domains may however not be covered. Indeed, while the management of orders, invoicing and inventory is relatively similar between two different companies, this is often not the case for the production management and the distribution logistics. Furthermore, most of the global solutions only handle data and does not aid the users in

making decisions in everyday's activities. For this reason, specific solutions sometimes need to be implemented, not to replace global solutions but to complete them to form more efficient information systems.

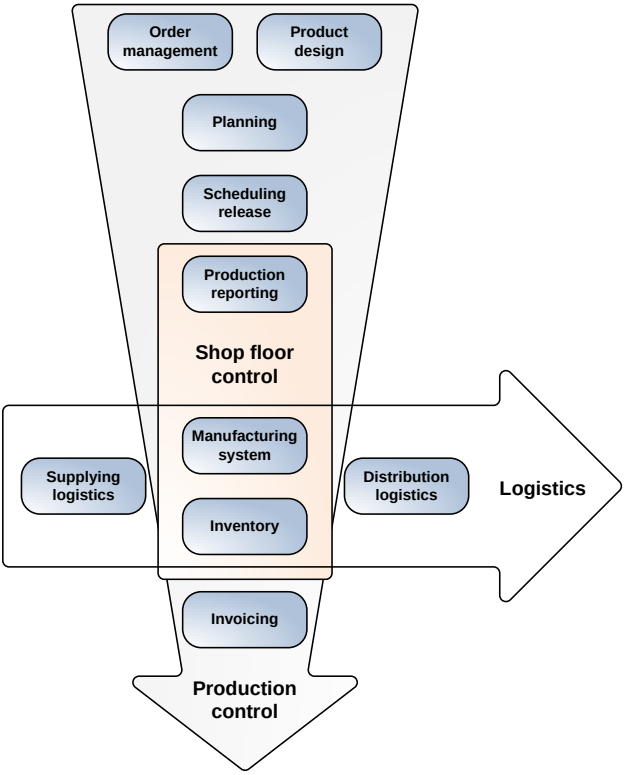

Figure 1: Integrated production management

Although presented in Figure 1 as last stage of the logistics process for manufacturing companies, the distribution logistics, which concerns all tasks involved in the distribution of goods to customers, can also be the main business of other companies specialized in this domain. Due to the use of vehicles, the distribution logistics, and more precisely the transportation planning elaboration, is not negligible, both in terms of cost and environmental impact. Indeed, most of these vehicles use fossil fuel and reject therefore carbon dioxide (CO_2) in the atmosphere. Although recent vehicles are more efficient in terms of fuel consumption, a tax is

however imposed per kilometer driven.

While goods could simply be delivered from a factory to customers visited successively, this basic pattern is often not suitable and multiple variants therefore exist. For example, products can be first transported from a factory to a central warehouse, waiting to be then transferred with other products to a retailer where customers can finally buy them and bring them home. Note that the type or size of vehicles can be different if these ones are used to transport goods between two warehouses or from a detailer to its customers. In the first case, semi-trailers of high capacity can be used due to the large amount of products transported and due to the use of highways while in the second case, a van that can access each customer can be sufficient to deliver a few products. Furthermore, more the quantities ordered reach the capacity of a vehicle, more frequently this one has to be reloaded. When vehicles are unloaded following one or more deliveries, they can sometimes be used to recover raw materials from suppliers. Note that some companies are specialized in pickups and deliveries, i.e. that their vehicles simultaneously collect goods from locations and dispatch them to other locations.

The nature of the products has also to be considered, as customers wishes according to the delivery period. If the products are perishable goods, like food, or if a customer has to receive the goods in a specified time period, corresponding constraints have to be taken into account when elaborating a transportation planning. All the features described before, which depend on the business of a company, can explain the difficulty to implement a dedicated module in global solutions.

This PhD thesis concerns on the one hand information management and on the other hand, operations research. Indeed, the development of an interactive planning system intended to facilitate the transportation planning elaboration of a cement factory will be presented in the first part of this thesis, while the implementation of an optimization module intended to complete this information system in order to form an interactive planning support system will be discussed in the second part of this thesis.

The cement factory has to deliver every day various cement types to customers with a fleet of heterogeneous vehicles having especially different capacities. Compared to examples mentioned before, the quantity ordered by a customer often exceeds the capacity of each of these vehicles, what implies the split of orders into multiple deliveries. Furthermore, only one cement type can be delivered with a vehicle. In addition to the factory, other depots can also be used to load some vehicles.

The aim of the interactive planning system to develop is to help a dispatcher in his daily activities by providing him appropriate functionalities and a high level user interface. Note that this information system has to be connected to other existing system of the company. For the conception of the interactive planning system, the way of working of the dispatcher, as tools used hitherto, i.e. a magnetic board, magnets and some spreadsheets, will be considered. The aim of the optimization module to implement is to propose different solution methods for elaborating any transportation planning related to the cement factory in order to aid the dispatcher in making decisions. To compare the performance of these solution methods, these ones will be tested on real life instances and the obtained results will be analyzed.

As mentioned before, the structure of this PhD thesis is composed of two parts. The first part concerns the development of an interactive planning system. For this purpose, a description of the problem and the methodology used to develop this information system will be presented in Chapter 1. The existing system of the cement factory and its potential improvements will then be discussed in Chapter 2. The concept and the appropriate technologies required to develop an interactive planning system will be defined in Chapter 3. Following the design of a database in Chapter 4, the graphical user interface and functionalities will be presented in Chapter 5. Finally, some logical aspects of this information system will be discussed in Chapter 6. The second part of this thesis concerns the implementation of an optimization module. Therefore, a restrictive description of the problem and the state of the art of neighboring problems will first be presented in Chapter 7. A modeling of the problem and four different solution methods to solve it will then be described in Chapter 8. Finally, comparisons between these solution methods will be presented in Chapter 9. More details about each chapter will be given in the first chapter of each part of this thesis.

Part I

From a magnetic board to an interactive planning system

Chapter 1

Preamble to the development of an information system

Although the first part of this thesis presents the development of an interactive planning system, it is beforehand useful to introduce on the one hand the context in which such a system has to be built, and to present on the other hand the methodology selected to develop this system. For this purpose, a description of the problem is first detailed in Section 1.1, which is then followed in Section 1.2 by a presentation of the selected methodology. The structure of the remaining chapters of this first part is finally described in Section 1.3.

1.1 Problem description

A factory produces different cement types. Mainly sold in bulk to concrete plants and terrace garden material manufacturers, these cement types are routed to the customers by road using silo trucks. The transportation planning and the cement deliveries are realized by the factory which has indeed its own fleet of trucks together with truck-drivers. When work overflow occurs, some vehicles can also be rented to third parties, which involves naturally additional costs.

Trucks properties

Due to the different locations of the customers and the various size of their orders, the fleet of trucks is heterogeneous. It means that each silo truck has its own capacity, between 20 and 30 tons, and can only access some customers. Note that some vehicles are older than other ones, what implies different carbon dioxide emissions.

The tractor and the trailer of a truck can be separated. This allows different possible combinations of [tractor;trailer] and therefore can modify the capacity of a truck.

Mostly, only one driver is assigned to each truck. For this reason, the daily availability of a truck is restricted, although it is authorized to drive from 05:00 to 22:00. Indeed, union rules state that the driving time must not exceed 9 hours per day with a maximum overtime of 2 hours per week per driver.

Local depots

To spare time, wagons are placed in some railway stations to create local depots. Although these depots are nearer from some customers than the factory, the quantities of cement are there however limited. Furthermore, only trucks with specific equipment can be loaded at these depots. Note that each truck has to start and end its daily activity at the factory.

Orders handling and composition of a delivery

Every day, cement types ordered the day before have to be delivered by trucks to several customers. Ordered quantities are usually between 20 and 180 tons. Due to the trucks capacity, each order is so split into one or more deliveries. The delivery process decomposes into four steps:

1. **LOADING TIME** A cement type is first loaded by a truck either at the factory, considered as the main depot, either at one of the local depots located in the railway stations. If it is the first delivery of the day for the truck, the load can only be done at the factory. Although the loading time takes 5 to 10 minutes at the main depot, 20 to 25 minutes are needed at the local depots because the trucks with specific equipment use a filter, which takes more time.

2. **TRAVEL TIME** When loaded, the truck drives from the selected depot to the customer of the delivery. The travel time needed to arrive at the customer depends not only on the location of this one, but also on the used truck. As mentioned before, the fleet of trucks is heterogeneous, which implies various routes and speed performances.

3. **UNLOADING TIME** When arrived at the customer, the cement is unloaded. Although the unloading time mainly depends on the facilities of the customer, 5 to 10 minutes are needed.

4. **BACK TRAVEL TIME** Finally, the truck has to drive back either to a depot if other deliveries have to be done, or to the factory if it is its last delivery of the day. As for the travel time, the back travel time mainly depends on the used truck.

To resume, the loading and unloading times depend on the used truck, on the cement type and on the equipment of the location. The travel and back travel times depend on the used truck and on the length of the route between the depot and the customer.

Preloads

To spare time, it frequently happens that trucks are already loaded at the end of the day before with a cement type for their first delivery of the day even if the customer of this first delivery is not yet known. Indeed, two cement types are mainly ordered by customers. Each preloaded truck is therefore preloaded with one of these two cement types. When possible, some truck-drivers drive their truck home after a preload if it reduces the travel time to the first customer of the day after.

Ashes recoveries

During the cement manufacturing process, ashes are used as solid fuel in a regular quantity. These ashes of pyrite and paper, which are in fact substitutes of combustible materials, can be recovered from suppliers in parallel to the cement deliveries as far as a truck is not loaded with a cement type.

Truck-drivers multipurpose

Because of different skill levels, each truck-driver belonging to the factory can only drive some trucks, supply some customers and handle some materials. Some trailers require indeed a special license to be driven. Furthermore, only 80% of the drivers know the location of all customers. Finally, all truck-drivers cannot transport the same combustible materials during a recovery because each combustible material requires specific knowledge.

Dispatcher's activities

The split of orders into deliveries and their planning is realized by a dispatcher. To be in accordance with the union rules, this employee is also responsible to manage the working and driving time of each truck-driver by placing necessary daily breaks. In addition to these main activities, the dispatcher has also to plan ashes recoveries from suppliers and sometimes to assign planned tasks to the trucks or to their driver. As explained before, ashes of pyrite and paper are used as combustible materials during the manufacture of cement. For this reason, after having supplied a customer, unloaded trucks can access these suppliers and load ashes before driving back to the factory. Usually, 5 to 10 ashes recoveries are planned per week. Regarding the planned tasks, they represent time consuming tasks related to the trucks or their driver that do not concern deliveries or ashes recoveries, like driver training or vehicle maintenance.

Dispatcher's activities are illustrated by a use case diagram in Figure 1.1. Use cases belong to Unified Modeling Language (UML) and are used in project planning and development to represent scenarios, i.e. interactions between actors and a system to notably highlight the needs. More information about this modeling technique can be found in [25].

1.2 Methodology

The first part of this thesis concerns information management and more precisely the implementation of an interactive planning system. To manage the life cycle of this information system including not only its development but also its operation and maintenance, a system development methodology named FAST for 'Framework for Application of Systems Thinking' has been selected. Unlike other

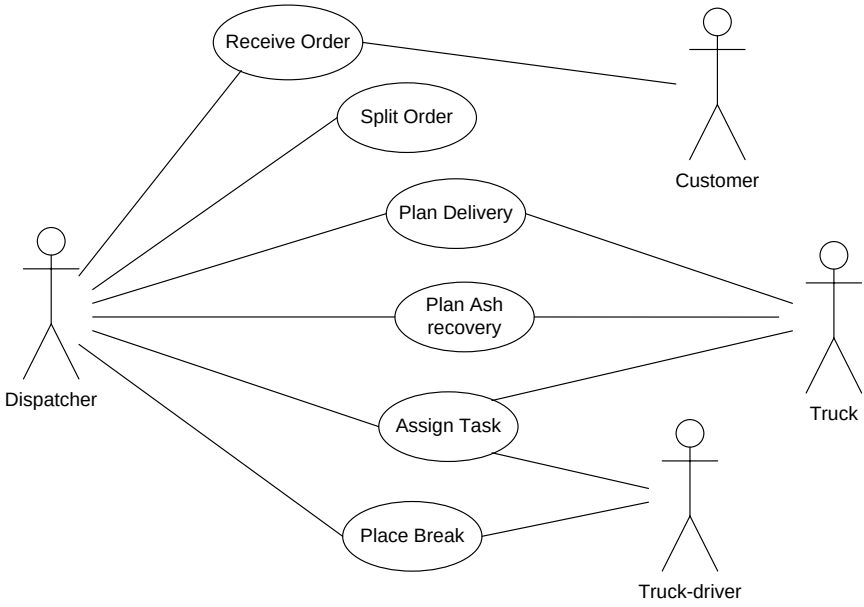

Figure 1.1: Dispatcher's activities

methodologies that are commercial, FAST is an hypothetic and flexible methodology composed of the best practices used in many commercial and reference methodologies. Described in detail in [8], the main source used to write this section, and illustrated by a schema in Figure 1.2, this methodology involves the participation of the system owners, project managers, system analysts, system designers, system builders, and system users, and consists of 9 phases succinctly presented later.

Each phase requires prerequisites, is composed by a set of activities, involves participants and describes their role, and produces outputs that are transferred to the next phase. In parallel to each phase, documentation is accumulated. A project starts with some combination of problems, opportunities and directives from the user community, i.e. the system owners and system users, and finishes with a working business solution for the user community.

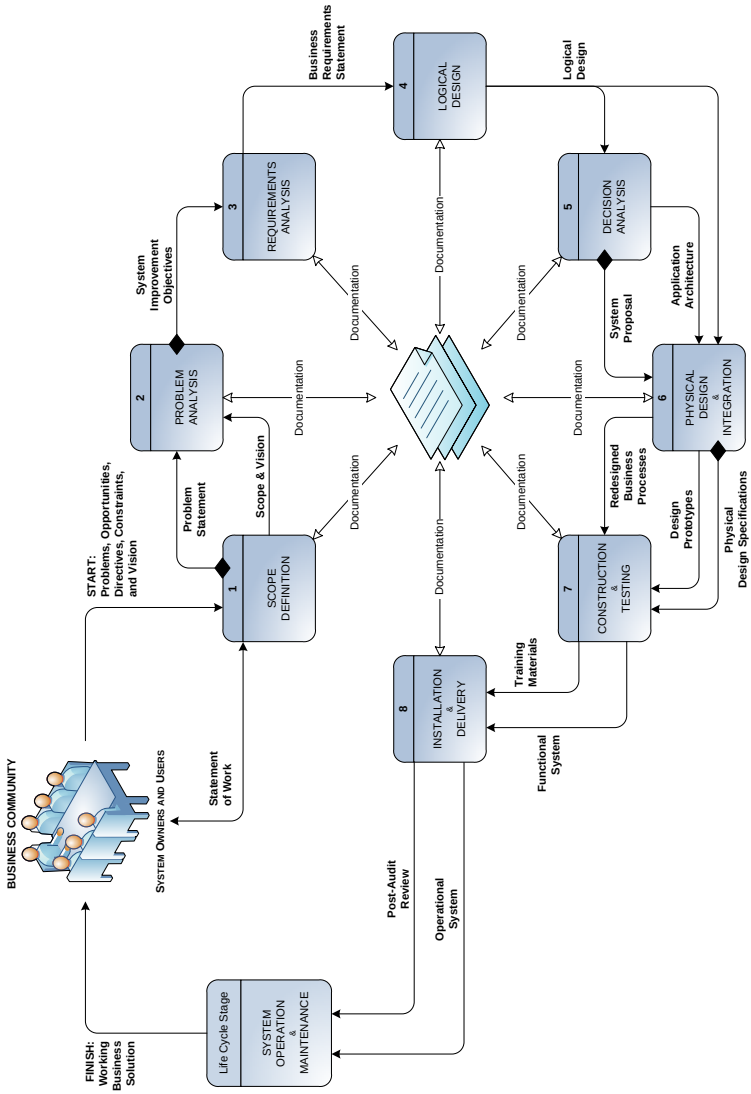

Figure 1.2: Process view of system development

1. **SCOPE DEFINITION** Given problems, opportunities, directives, constraints and vision, the scope definition phase, illustrated in Figure 1.3, first wonders about the relevance of developing a new system and, if it is pertinent, establishes then the size and boundaries of the project, the project vision, the constraints or limitations, the required project participants, the budget, and the schedule.

   The aim of this phase involving the system owners, project managers, and system analysts is to list and organize useful information taking into account budget limits, deadlines, available human resources, business policies, government regulations, and technology standards in order to produce a problem statement. This phase also defines a scope and vision of the project. While the scope focuses on the size of the project team, the budget allocated to the system development, and on the schedule preparation for the remaining phases, the vision concerns improvements of the current system including comments on data, functionalities, and interfaces needed. A statement of work, i.e. a contract related to the system development, is also realized during the scope definition phase.

   At the end of this phase, the system owners have three alternatives represented by a diamond on the schema. They can either agree with the proposed scope, budget, and schedule, either reduce the scope to decrease costs and time, or cancel the project.

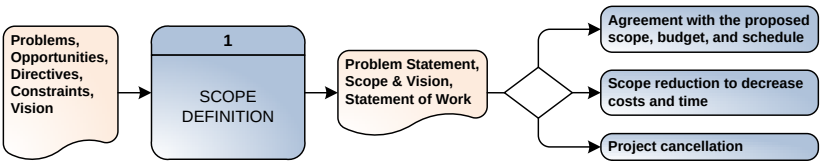

Figure 1.3: Scope definition

2. **PROBLEM ANALYSIS** As a new system always replaces an existing system, the problem analysis phase, illustrated in Figure 1.4, examines in detail the existing system. This analysis helps not only to better understand the functionalities and architecture of the existing system, but also to discover problems and opportunities that have not been detected in the scope definition phase. Given the scope and problem statement defined and approved in the scope definition phase, the problem analysis phase helps to decide if the benefits of a new system exceed its development costs.

System users, considered as business experts, are involved in this phase which produces a set of system improvement objectives related to business criteria. These objectives, presented either as a written recommendation or as an oral presentation, will be used to evaluate the new system. The choice of documenting the existing system depends on the complexity and the schedule of the project. This documentation can include analysis demonstrating inefficiencies, bottlenecks or other problems related to the business processes.

At the end of this phase, the system owners have three alternatives represented by a diamond on the schema. They can either agree with the recommended system improvement objectives and continue with the next phase, either reduce or expand the scope with budget and schedule modifications before continuing with the next phase, or cancel the project if developing a new system is no longer relevant.

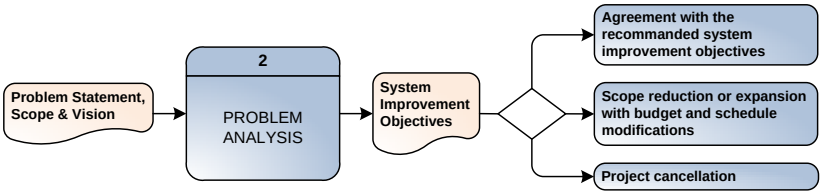

Figure 1.4: Problem analysis

3. **REQUIREMENTS ANALYSIS** Given the system improvement objectives defined and approved in the problem analysis phase, the requirements analysis phase, illustrated in Figure 1.5, defines the business requirements, i.e. what the system has to do but not how it has to operate. More precisely, this phase states the functionalities of the new system, the data processed, and finally the performance level expected.

   The participants include system users, systems analysts and project managers. While system designers are not involved in this phase to avoid focusing on technology aspects, system users are fully required to identify and prioritize their need with interviews, questionnaires and meetings. Priorities allow on the one hand to rescope the project if its schedule is short and on the other hand, to define iterations in order to create staged versions of the new system. This phase produces a business requirements statement including business data requirements, business process requirements and

business and system interface requirements. The business requirements statement can be either a document of a few pages, or a document presenting for each requirement one or more pages.

The requirements analysis phase is important. Indeed, any error or omission will result in users dissatisfaction with the new system and therefore imply costly modifications.

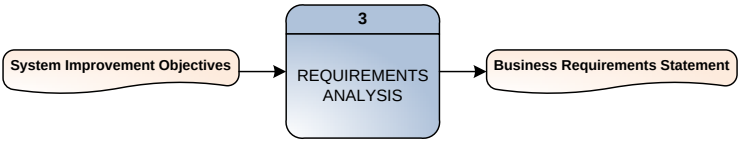

Figure 1.5: Requirements analysis

4. **LOGICAL DESIGN** Illustrated in Figure 1.6, the logical design phase translates the business requirements expressed in words in the requirements analysis phase into pictures called system models in order to be validated for completeness and consistency. The term logical design means that the pictures illustrating the system are independent of any possible technical solution. The amount and levels of detail of the system models depend on the selected methodology and on the complexity of the system to develop.

   The participants include system analysts, who draw the models, and system users, who validate the models. Project managers are also involved to ensure that standards previously defined are respected. Logical data models that depict data and information requirements, logical process models that depict business processes requirements, and logical interface models that depict business and system interface requirements can be drawn. This phase produces the logical system models and specifications.

   The requirements analysis and logical design phases usually overlap. Indeed, as business requirements are identified and documented, they can be modeled. The scope definition, problem analysis, requirements analysis, and logical design phases are considered as system analysis.

5. **DECISION ANALYSIS** Given business requirements and the logical system models, many alternatives can be selected to design a new system that fulfill these requirements. The aim of the decision analysis phase, illustrated in Figure 1.7, is to answer questions about the automatization level of the new system, the option to purchase or build in-house a solution, and the

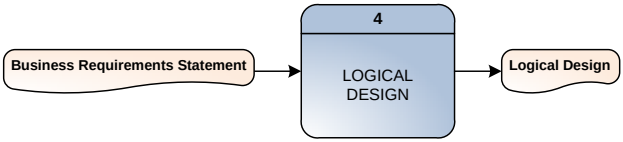

Figure 1.6: Logical design

technologies to use in order to identify and analyze feasible candidate solutions. Candidate solutions are characterized in terms of technical, operational, economic, schedule and risk feasibility. The most feasible solution, i.e. the solution that offers the best combination of previous enumerated criteria, is chosen.

System designers, considered as the technical experts in specific technologies, are involved in the decision analysis phase, as system users and system analysts who help to define and analyze the alternatives, and the system owners who have to approve or disapprove the decisions because they fund the project. The decision analysis phase produces a system proposal which can be written or presented verbally. Optionally, this phase may also produce an application architecture model for the approved solution which serves as a high-level blueprint for the system proposal.

At the end of this phase, the system owners have four alternatives represented by a pentagon on the schema. They can either approve and fund the system proposal for design and construction, either approve or fund one of the alternative candidate solutions, either reject all the candidate solutions and cancel the project or review it for new recommendations, or approve a reduced-scope version of the proposed solution.

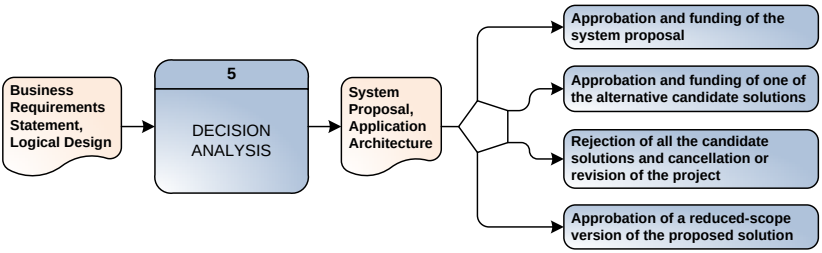

Figure 1.7: Decision analysis

6. **PHYSICAL DESIGN AND INTEGRATION** Given the system proposal approved in the decision analysis phase, the aim of the physical design and integration phase, illustrated in Figure 1.8, is to convert the business requirements including logical system models into physical design specifications to guide the system construction. This phase concerns not only technical aspects of the system, i.e. physical database design specifications, physical business process and software design specifications, and physical user and system interface specifications, but also the integration of this system with other information systems. Questions about completeness, usability, reliability, performance, and quality are also discussed during the physical design and integration phase.

   System analysts, system designers and sometimes system users are involved in this phase which produces some combination of physical design models and specifications, design prototypes, and redesigned business processes.

   At the end of this phase, there are various alternatives represented by a pentagon on the schema. Although a project is rarely canceled after the design phase, the scope may be decreased to produce a minimum acceptable product in a specified time frame, or the schedule may be extended to build a more complete solution in multiple versions. Usually, the design and construction phases overlap.

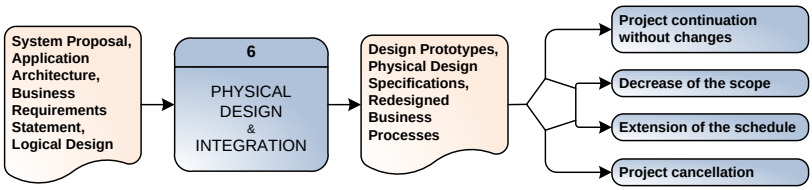

Figure 1.8: Physical design and integration

7. **CONSTRUCTION AND TESTING** Given physical design models, specifications and design prototypes defined in the physical design and integration phase, the aim of the construction and testing phase, illustrated in Figure 1.9, is to build and test a system that fulfills business requirements and physical design specifications, and to implement the interfaces between the new system and existing systems. This phase has also to provide documentation about system operation for users training and help desk support. The construction and testing phase can finally include installation of purchased

softwares including database management systems and software packages.

The participants include system builders, system analysts, system users, and project managers. System designers may also be involved to clarify design specifications. When a system component is built, users test it in order to give feedbacks. Tests have to be conducted not only on individual system components but also on the overall system. Once tested, a system or a version of a system is ready for installation and delivery.

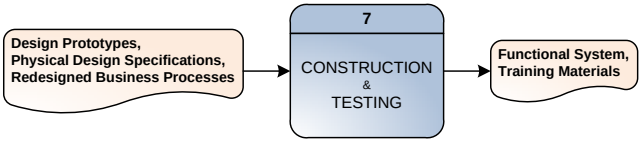

Figure 1.9: Construction and testing

8. **INSTALLATION AND DELIVERY** Given the functional system from the construction and testing phase, the aim of the installation and delivery phase, illustrated in Figure 1.10, is to install this system into the production environment and to train users with published documentation. The transition from the old system to the new system is also realized during this phase. The old system can either be terminated and replaced by the new system on a specific date, or run in parallel with the new system until the new system is considered as acceptable to replace the old system.

   The participants include system builders, who install the system, system analysts, who train system users, write various user and production control manuals, create operational databases, and perform final system testing, and system users, who provide feedbacks when a new problem occurs. The installation and delivery phase produces an operational system and a post-audit review to evaluate the success of the completed systems project. Any problem discovered in this phase can imply to rework in previous phases.

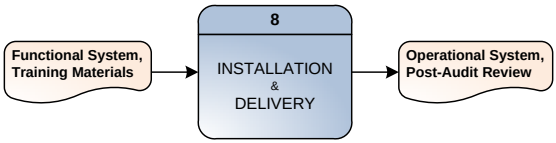

Figure 1.10: Installation and delivery

9. **SYSTEM OPERATION AND MAINTENANCE** Once the system is used in the production environment, it requires support for the remainder of its productive lifetime, i.e. until it has to be replaced by a new system due to changed business needs. Illustrated in Figure 1.11, system support consists of assisting users, fixing software bugs, recovering the system, and adapting the system to new business problems or technologies requirements.

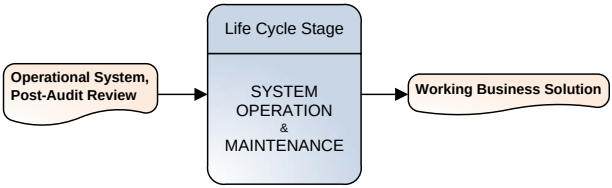

Figure 1.11: System operation and maintenance

1.3 Structure of the first part

The first part of this thesis only focuses on the activities of system developers, i.e. related to the roles of project managers, system analysts, system designers and system builders. Although system developers are involved in all phases of the FAST methodology presented in previous section, some phases are not covered in the following chapters for two reasons. On the one hand, the presentation of the scope definition and decision analysis phases is skipped because these two phases are decision points whose resolution is taken by the system owners, i.e. the cement factory. Indeed, the existence of this thesis and the properties of the remaining phases are the results of these resolutions. On the other hand, the presentation of the last three phases of the FAST methodology are skipped because the aim of this part is to present conceptual aspects of the development of an information system, and not to give details about building, testing, installing, operating and maintaining this system.

To resume, only the problem analysis, requirements analysis, logical design, and physical design and integration phases are discussed in the next chapters. However, the content of these four remaining phases and their sequence are completely restructured to be more digest for the reader. The correspondences between FAST phases and chapters of this first part, that presents the main steps of

the implementation of an interactive planning system intended to assist the dispatcher in his daily activities, can be observed in Table 1.1.

Table 1.1: Correspondences between FAST phases and chapters of this first part

Phase	Chapter
Scope definition	-
Problem analysis	Existing system analysis
Requirements analysis	Concept ~~and selected technologies~~ Graphical user interface and functionalities
Logical design	Preceding rules and frozen horizon rules
Decision analysis	-
Physical design and integration	~~Concept and~~ selected technologies Database design
Construction and testing	-
Installation and delivery	-
System operation and maintenance	-

After having given more details about the problem and its context in introduction of this chapter, a thorough analysis of the existing system, i.e. the tools used by the dispatcher to realize the transportation planning and their place in his daily work, is first done in Chapter 2. The concept of the interactive planning system and the choice of the technologies used to implement this system are then defined in Chapter 3. Following this, a relational database used to store and manage the data of the system is designed in Chapter 4. Chapter 5 presents the functionalities and focuses on the interactive aspect of the planning between the user and the system. Rules involved in some of these functionalities are finally detailed in Chapter 6.

Chapter 2

Existing system analysis

To develop an information system, the first step consists in understanding in detail the finality of the activities that will be managed or in communication with this system, as well as their relationships. The existing handling of these activities can indeed influence the design of the information system and also allow the detection of potential improvements. For this purpose, the existing system used by the cement factory to realize the transportation planning and its related activities, illustrated with a schema in Figure 2.1, is analyzed in this chapter. Following a detailed description of the tools used by the dispatcher and their involvement in his daily work in Section 2.1, potential improvements of these tools are discussed in Section 2.2. A conclusion is finally given in Section 2.3.

2.1 Daily work and used tools

Before the planning elaboration, orders are first received by phone until about 16:00 for the day after. Each customer can fix time periods for their deliveries or simply indicate a preference. The dispatcher uses for this purpose a paper diary reporting all the customers requirements.

In parallel, current day's planning is supervised and possibly adapted by the dispatcher during the day due to unforeseen situations, what can imply modifications in a weighing system presented later in Subsection 2.1.2. Occurred events

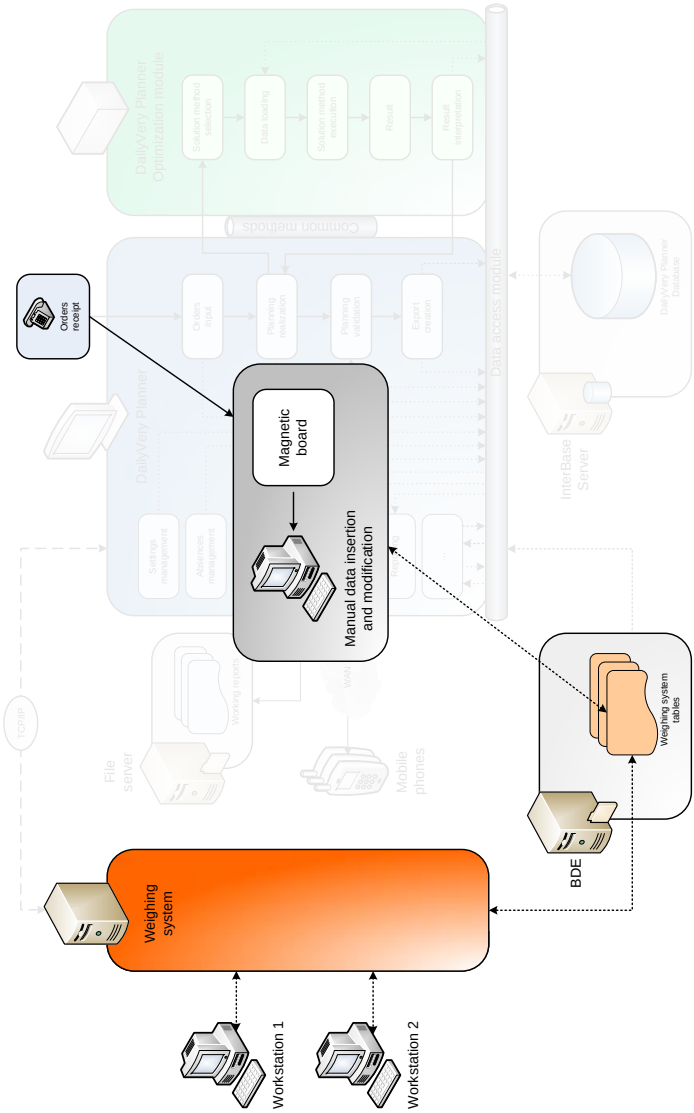

Figure 2.1: Existing system schema

can however not be modified. At the end of a workday, the planning becomes therefore reality.

When the majority of orders have been received, the dispatcher splits them into deliveries using a magnetic board and magnets. For this task, the dispatcher uses not only a map containing the locations of the customers and ashes suppliers, but also a document indicating the components of the delivery duration according to the customer and the cement type ordered. Note that each ashes supplier is allocated to a cluster of customers.

2.1.1 Magnetic board

A use example of the magnetic board is presented in a snapshot in Figure 2.2. Printed on a large sheet, a timeline schedule is fixed on this board and displays a daily period between 05:00 and 22:00. Set just at the left, schedule's resources are printed on another sheet to allow possible changes. Each resource, represented by an identification number, corresponds to a truck belonging to the cement factory. Next to each of these numbers are mentioned the maximum total weight and the payload of the truck. When displayed, 'F' indicates that the truck is equipped with a filter that allows it to access to the local depots located in the railway stations and 'K' indicates that the trailer is equipped with hydraulic jack. Space is then available to place a magnet related to the truck-driver assigned for the day. The factory has more truck-drivers than trucks and most of them always drive the same truck. For this reason, the identification number of their truck is also indicated on the magnet.

To represent the deliveries and their duration, white magnets with different lengths are available and placed in ascending order at the right of the timeline schedule on another sheet fixed for this purpose. Above the schedule, these are completed with a few yellow magnets representing ashes recoveries. Magnets related to truck-drivers being sick, on holidays and on training are placed below the schedule.

The timeline schedule is used to build and display the daily planning of each resource. In addition of the duration of the delivery, the name and the location of the customer are written on each white magnet when used by the dispatcher. These are completed by another small and round colored magnet that indicates the cement type only if the delivery does not concern the type representing most of the sales. Distances sometimes separate two magnets. These 'white spaces' represent in fact breaks. When needed, an ash recovery is planned. Therefore, a

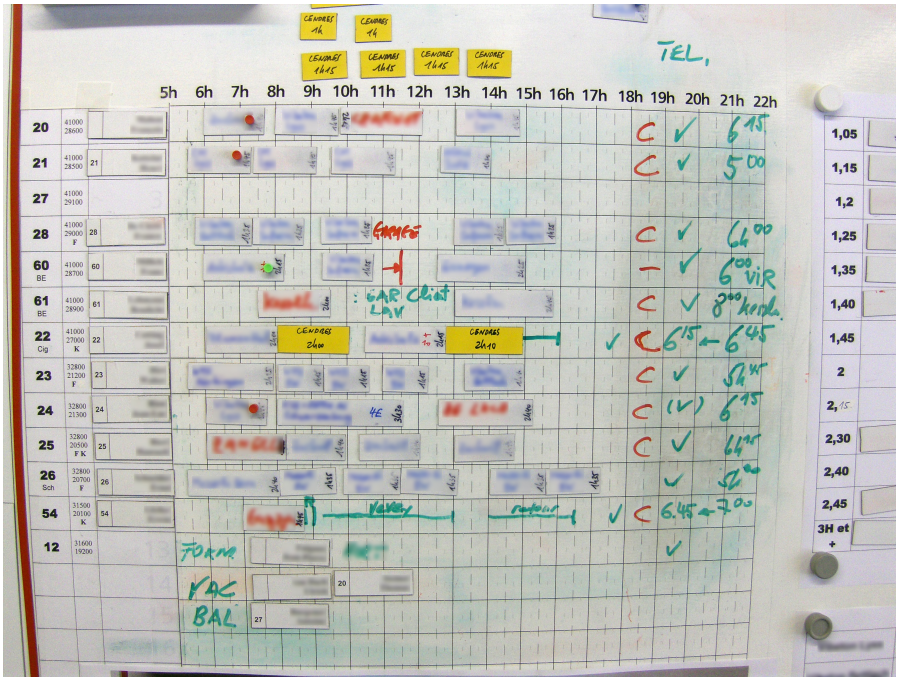

Figure 2.2: Magnetic board (Ciments Vigier SA)

yellow magnet is placed either next to a white magnet, if the truck drives from the cement factory to load ash at a supplier, or superposes the end of a white magnet to cancel the back travel time of the related delivery, if the truck does not drive back to a depot after the unload of this delivery but drives to an ashes supplier. As for the breaks, the planned tasks are not represented with magnets. They are indeed directly written on the board. At the end of the daily planning of each truck, 'C' and '✓' respectively indicate that the truck is preloaded for the first delivery of the day after and that its truck-driver is informed about the start time which is finally displayed. Note that 'C' is often replaced by a number corresponding to the cement type.

When elaborating the transportation planning, the objective is to use the less magnets as possible in order to minimize the total travel duration done by the trucks for their deliveries. If an order needs several deliveries, the same truck-driver is, if possible, every time selected. These magnets are placed on the time-

line schedule to build for each needed truck a daily planning. For this task, the dispatcher has to define which trucks and which drivers are able to deliver which orders. The size of each customer's site has also to be considered. Indeed, the smaller a site is, the more important the delivery time is because of its stock capacity. As mentioned before, breaks are represented on the planning with 'white spaces'. If a break is done during a delivery, it is reported at the end of this one due to the use of magnets. Breaks are only placed for guidance because each truck-driver decides in fact when he does his breaks. What matters is to know at what time a driver has finished his workday, which is indicated by the end of the last magnet placed on the daily planning of his truck.

When all the magnets are placed on the magnetic board, i.e. at the end of the planning elaboration which takes about 2 hours, the dispatcher enters the information of each corresponding delivery in a weighing system to allow the loads of the trucks with the appropriate cement types. After this operation, which takes about 45 minutes, the dispatcher finally calls the truck-drivers by phone and communicates them their planning of the day after and especially the start time of their first delivery.

2.1.2 Weighing system

To load cement at one of the three docks available at the cement factory, each truck-driver owns a badge related to his truck. Completed by a password, this one allows him to connect on a first workstation. If the driver belongs to the cement factory, the daily deliveries of his truck are displayed on the screen. Defined by the dispatcher, they are listed in ascending order of their execution. By default, the driver has to select the first delivery of the list before validating his choice. If the truck-driver belongs to a third party, only deliveries of customers supplied by this third party are displayed and can be chosen.

The selected delivery is then displayed in detail on the screen of a second workstation. While the drivers belonging to the factory have only to validate this delivery, drivers belonging to third parties have to select a cement type. Only one cement type can be selected and therefore loaded per delivery because of the silo trailers that are not partitioned.

The weighing system knows for each truck its tare, its maximal loading capacity, its total weight and the real quantity of the loaded cement type. The system can therefore detect if a truck, back from a customer, has a remainder of cement. The properties of the trucks belonging to third parties, which deliver a few cus-

tomers, are also known. When one of those customers makes an order, he is immediately redirected to the allocated third party. It is a total subprocess because these customers are always delivered by the same third parties.

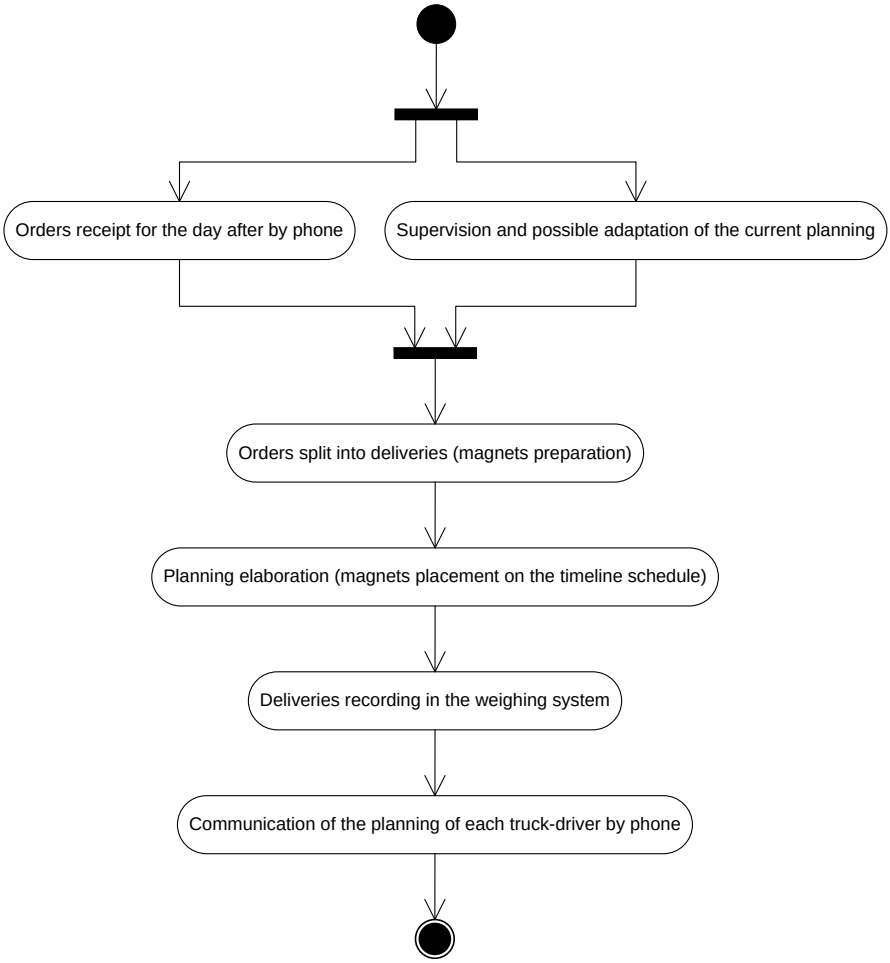

Figure 2.3: Daily work

Dispatcher's daily work is illustrated by an activity diagram in Figure 2.3. Activity diagrams belong to UML and are used to describe organization of activities of a system including conditional and parallel behaviors. More information about this modeling technique can be found in [25].

2.2 Potential improvements

The timeline schedule of the magnetic board is not only a visualization tool, but also a creating tool. Thanks to the magnets, the dispatcher can easily allocate deliveries and ashes recoveries to the trucks and make changes during the planning elaboration. Furthermore, the planning can then be adapted during the day to finally reflect reality.

No assistance

However, no tool helps the dispatcher to split the orders received by phone into deliveries. This imposes him to prepare the magnets before the planning elaboration, i.e. to split each order into a number of deliveries depending on the used trucks, which are therefore already selected before the use of the timeline schedule.

Limited scalability

Due to the properties of the magnetic board, the number of resources that can be displayed on the timeline schedule is limited. It can cause problem if the cement factory increases its fleet of trucks and has to realize the transportation planning of the trucks belonging to third parties. The magnets and their sizes are also limited. Only frequent durations exist, which sometimes implies the use of a pen to directly extend the size of some magnets on the timeline schedule.

Lack of precisions

Due to the magnets which cannot be split, breaks are not always well placed by the dispatcher. Furthermore, it is sometimes difficult to delimit the breaks represented by 'white spaces' from planned tasks or to know if a 'white space' corresponds to a break or is only a space between two deliveries.

Although the various sizes of the magnets are mostly sufficient and do not need to be adapted, their related duration does not always correspond to reality and are often approximations. Indeed, the duration of a delivery particularly depends on the used truck and the cement factory owns different types of trucks in terms of speed performances. To simplify his work, the dispatcher does not take this into account during the planning elaboration and uses for the deliveries of an order the same duration for any truck. A magnet can so be moved from a truck to another one without having to be changed. If for a same order a truck has less or more capacity compared to its predecessor, magnets related to the deliveries of this order have perhaps to be added or removed.

Approximations of durations also exist for some ashes recoveries when the related yellow magnets superpose roughly the end of a white magnet to cancel the back travel time of the related delivery. In such cases, the truck does not drive back to a depot after having unloaded its cement type at the customer but drives to a supplier to load ash.

Other problems of precision about durations can finally be observed for the last delivery of a truck when its cement type is loaded at a local depot located in a railway station. Indeed, the magnet related to the last delivery has the same size as the previous ones related to deliveries whose cement type has also been loaded at the same local depot. But for the last delivery, a truck has in reality to drive back to the cement factory and not to the local depot, which takes more time.

Validity of a solution

A planning solution is considered as *valid* if each delivery or ash recovery is compatible with the properties of its allocated truck and with the skill level of its driver. Furthermore, breaks created for each truck-driver have to be set in sufficient quantity at the right time and with the appropriate duration in order to respect the union rules. Finally, the use of each truck must not exceed its daily availability.

With the magnetic board, there is no check if a planning solution is valid or not. In case of inattention of the dispatcher, helped with a map and a document indicating the properties of the customers and ashes suppliers, many errors can therefore occur.

Weakness of history storage

To keep track of the planning, a snapshot is done every day before emptying the timeline schedule of the magnetic board for the planning elaboration of the day after. Although it is a simple method adapted for a magnetic board, it is not easy to recover and process information displayed on a large amount of snapshots.

No user interface

Although two workstations with a touch screen user interface are available for the truck-drivers to load the cement qualities related to their deliveries, no user interface exists for the dispatcher to enter beforehand the data of these deliveries in the weighing system. Indeed, each delivery has to be stored manually as a record directly in the corresponding table of the weighing system's database. The primary key of each of these records is also set manually in order to define the position of the related delivery in the daily planning of the concerned truck. If changes are made on the timeline schedule of the magnetic board after having stored the deliveries in the weighing system, the corresponding records have to be manually modified or deleted, or new records have to be manually added in this system.

Drawbacks of communication tools

At the end of the day, the truck-drivers are called by phone to be aware of the transportation planning of their truck for the day after. Due to this synchronous method, the drivers are not always reachable and some calls have to be done several times. Furthermore, if after a fixed time a truck-driver has not received a call, he has to contact the dispatcher. With phone calls, the truck-drivers do not receive a written document but only listen to the dispatcher.

2.3 Conclusion

In this chapter, the work of the dispatcher and the tools involved in his daily activities, i.e. a magnetic board and a weighing system, have been presented in detail. While the magnetic board is used to create the transportation planning with magnets representing deliveries and ashes recoveries, the weighing system is used to store information related to this transportation planning in order to allow the

truck-drivers to do the planned deliveries. Following this presentation, potential improvements of these tools have been detected and discussed.

The analysis of the tools used by the dispatcher and their potential improvements will be useful all along the project to ensure the development of an interactive planning system that satisfies the needs of the cement factory and that takes into account the improvements required to face the weaknesses of the existing system.

Chapter 3

Concept and selected technologies

Following the analysis of the existing system and the detection of potential improvements, the next step consists in defining the concept of the information system to develop, in our case an interactive planning system, and to select the technologies needed for its implementation. The concept is defined in Section 3.1 while the selected technologies are presented in Section 3.2. The chapter ends with a conclusion in Section 3.3.

3.1 Concept definition

The aim of the concept is to highlight the main characteristics of the interactive planning system expected by the cement factory to replace its existing system without presenting in detail the required specifications and functionalities.

Complete software tool

The interactive planning system has to present itself as a software tool that manages not only all activities of the dispatcher between the receipt of orders by phone

and the cement deliveries or ashes recoveries by trucks, but also all required information to manage these activities. As mentioned in previous chapter, the timeline schedule of the magnetic board allows the planning of deliveries, ashes recoveries, breaks and planned tasks. The core functionalities of the software tool have therefore to be designed to handle these main activities. While the properties of the cement types, the customers, the ashes suppliers, the trucks, the drivers and their absences are only mentioned on the magnetic board, they have to be explicitly considered in the interactive planning system. In other words, these properties have to be stored in a structured way to be processed by the core functionalities.

User friendly interface

The user interface of the software tool must be 'user friendly'. This interface has to be on the one hand intuitive and easy to use, and on the other hand, effective to process tasks with a limited number of steps and in an automatic manner.

Considering the magnetic board, the use of a timeline schedule and magnets is an appropriate way to create and visualize the transportation planning of the trucks. A same approach has therefore to be adopted for the software tool. Although the timeline schedule displaying a daily period and resources has to be kept as it is, the magnets have to be replaced by a software equivalent, i.e. planner items, also allocated to resources for a time period. These planner items have however to be extended to include not only the deliveries and ashes recoveries, but also the breaks and the planned tasks to avoid confusion with 'white spaces'. Unlike the magnetic board where the magnets have to be prepared in advance, the planner items have to be created directly in the timeline schedule of corresponding trucks using the core functionalities of the software tool. This is also valid for the split of orders into deliveries. When possible, i.e. for each delivery or ash recovery, the duration of the corresponding planner item has to be automatically computed using the appropriate data stored in the interactive planning system.

Regarding the interactivity of the user interface, the access to the core functionalities of the information system and the use of these ones has to be easy. For this purpose, popup menus or even drag and drop options have to be integrated when possible in this interface. Unlike a main menu that is statically set at the top of a software window, popup menus appear at the cursor position by right-clicking on a software object, i.e. an element of the user interface. Considering the interactive planning system, this element could be a planner item representing a delivery. Thanks to popup menus, the dispatcher can quickly access the function-

alities that are only available for the selected software object. While popup menus only concern a software object, drag and drop options always involve two different software objects and represent therefore an interaction between these two software objects. Considering the interactive planning system, the first software object could be one of the daily orders while the second software object could be the timeline schedule of a truck. By dragging and dropping a software object on another software object, the dispatcher triggers an underlying functionality involving these two software objects. For example, by dragging and dropping an order on the timeline schedule of a truck, the dispatcher could create a delivery of this order and allocate it to that truck.

Concerning the effectiveness of the user interface, input supports have to assist the dispatcher to store data in the interactive planning system by selecting appropriate default values that can however be changed and by completing the initial content of some fields. Furthermore, the input of deliveries in the weighing system, as the transmission of the daily planning to the truck-drivers, has to be realized using a dedicated functionality in a transparent manner, i.e. without worrying about communication aspects.

High level details

The software tool has to present high level details. This concerns not only the amount of information displayed on the screen but also the granularity level for modeling the reality.

As for the magnetic board, the interactive planning system has to present on the screen a dashboard containing a lot of information useful for the dispatcher. In addition to a timeline schedule, the orders and the remaining quantity of cement that has to be split into deliveries have to be displayed, as truck-drivers being absent. Note that each available truck-driver not yet allocated to a planner item has also to be presented. In parallel to the planning, the reality has to be mentioned, i.e. if and at what time a cement delivery is loaded, together with the start time of the unload, the start time of the back travel and the end time of the back travel, which have to be estimated thanks to the data stored in the information system.

Regarding the granularity level, the four durations required to represent the total duration of a delivery have to be separately taken into account. Unlike the existing system, a different data has to be considered for each truck or more precisely for each trailer in the interactive planning system. Concerning the timeline

schedule, the granularity of a time unit must be adapted to 5 minutes. As mentioned in Chapter 1, the trucks can only access some customers and the truck-drivers can only drive some trucks, supply some customers and handle some materials. The multipurpose of the trucks and the truck-drivers have to be separately considered in order to be combined together to cope with all possible scenarios may happening during the planning elaboration.

Weighing system interconnection

The interactive planning system has to be interconnected with the weighing system to synchronize and exchange information. Indeed, many data required by the software tool to operate are common with data used by the weighing system. It concerns the cement types, the trucks, the customers and the deliveries. At the end of a transportation planning elaboration, the deliveries have to be communicated to the weighing system. This is also valid if changes are made to the planning. During the workday, the weighing system stores the start and end time of the cement load of each delivery. When such information is available, the corresponding delivery has to be updated in the interactive planning system not only to modify his status but also to define its real time period.

Communication with truck-drivers

To avoid the dispatcher to call by phone each truck-driver sequentially, the interactive planning system has to communicate with the truck-drivers in an asynchronous way via Short Message Service (SMS) to indicate not only the time of their first delivery but also the transportation planning of their truck. With the software tool, the dispatcher must also be able to print a work report for each truck-driver. This document, published before the first delivery of the day, has to present on the one hand the details of the transportation planning of the used truck and on the other hand, possible comments related to each delivery.

Check of solution

The validity of each transportation planning solution has to be checked. Due to the planner items that have to be created directly in the timeline schedule of corresponding trucks, many checks can be done during their creation because of the core functionalities that have to detect unauthorized operations thanks to the

properties of the deliveries or ashes recoveries and the multipurpose of the trucks and the drivers. Furthermore, checks have also to be done in real time when moving planner items, and just before the input of the deliveries in the weighing system to verify that all required information are available and to avoid inconsistencies.

Efficient history storage

Unlike the snapshots of the magnetic board, the history storage of the interactive planning system must be efficient. It has not only to be directly integrated in the software tool in order to be transparent for the user, but it has also to allow information search, statistics establishment and reporting creation.

3.2 Technologies selection

When developing an information system, the principles of robustness, scalability and consistency have to be kept in mind throughout the project. An information system has indeed to be robust, i.e. it has to face any possible users action. Note that a user can be a person or another information system. If unauthorized actions are done by users, the system does not have to crash but has to stay operational and even inform users about their mistakes. Furthermore, an information system has to be scalable, i.e. it has to be designed such a way that it has to be able to manage a maximum number of possible cases, and to store a large amount of data. In other terms and due to its lifetime that is often uncertain, an information system has to be generic and flexible enough to include future needs and to absorb increasing workload. Finally, the data stored in an information system have to stay consistent at any time, even if an unexpected interruption of this system occurs.

The three principles presented before combined with the concept defined in previous section allow the selection of the technologies needed to implement the interactive planning system. This selection concerns not only the choice of a programming language and a way to store the data used by the information system, but also the choice of an information system architecture from the physical design point of view.

3.2.1 Programming language

The choice of a programming language depends not only on the characteristics of the solution that has to be implemented, i.e. the interactive planning system, but also on other information systems used by the factory, on the number of employees that are involved in the maintenance and improvement of these systems and on their skill level. In our case, the cement factory uses not only purchased solutions to manage for example its invoicing, but also develops its own solutions. This is the case with the weighing system, which has been developed in Delphi.

Delphi is not only a programming language, but also an Integrated Development Environment (IDE), i.e. a tool that helps the developer to manage the development of a software application. In fact, Delphi is an extension of Pascal, a programming language developed in 1971 in order to provide a language suitable for teaching programming thanks to a clear syntax, and also reliable and efficient on computers. This programming language has evolved over the years to adopt first the concept of object oriented programming under the name of Object Pascal, and finally the possibility to build graphical user interfaces including windows and fields under the name of Delphi. More information about the history of Delphi and its use can be found in [11, 12, 45], while the concept of object oriented programming is presented in detail and independently of any programming language in [10].

Although Delphi has been mainly used for years to develop software solutions, new programming languages like Java or C# are adopted today due to new principles that have been implemented. Unlike Delphi, code of these programming languages is no longer compiled in machine language, but is interpreted through a framework that translates instructions in a language adapted for the operating system used. In other terms, an application realized with Delphi can only operate on one kind of operating system and machine architecture. This is not the case of applications realized with Java or C# where a framework appropriate to any operating system and machine architecture can be used. In addition to this concept, Integrated Development Environments designed for these new programming languages are more advanced in terms of tasks automatization, and can sometimes correspond to Computer Aided Software Engineering (CASE) tools that manage all the development processes of a software application from its design to its installation and update utility. The Java and C# programming languages are presented in depth in [24, 1] and the concept of CASE tools is discussed in [9].

Despite its weaknesses, Delphi is imposed by the cement factory because of the skill level of employees responsible for the maintenance and improvement of

the interactive planning system to develop. As explained before, an information system has to be robust. Thanks to its syntax that allows error handling, i.e. the ability to detect and handle any error of usage with appropriate instructions that keep the system operational, Delphi is suitable from this point of view. Of course, its Integrated Development Environment does not correspond to a CASE tool, but additional available softwares can easily be combined together to fulfill the lacks. Indeed, tools like Inno Setup and ISTool can be used for building the installation and update utility. Like many Integrated Development Environment, Delphi provides a wide variety of fields to realize a graphical user interface, but does not allow the possibility to add a timeline schedule. Note that the timeline schedule is the core of the interactive planning system to develop, and therefore the possibility to add such an element in the user interface is essential. Fortunately, packages can be added to Integrated Development Environments to improve their initial potentials. After the analysis of schedule packages available for Delphi on the market, the most appropriate one, presented hereafter, is selected.

Schedule package

The selected schedule package is TMS Planner. Provided by TMS Software, this package is composed of a set of components designed to be used for a wide range of planning and scheduling applications, from a single Personal Information Manager (PIM) to a time planning including multiple days and multiple resources. Components useful for the development of the interactive planning system are illustrated in Figure 3.1 and shortly presented hereafter. For more details about all components of the TMS Planner package, the reader can refer to the website of TMS Software.

- **TPLANNER / TDBPLANNER** This component corresponds to the timeline schedule of the magnetic board presented in previous chapter. In addition to the selection of the time axis that can be horizontal or vertical, a day view, week view, month view, day period view, half day period view, multi month view, or timeline view can be selected to display the schedule. Note that the granularity of the time unit can be defined for the timeline. Although a resource can only be identified by a unique value with this component, the planning grid can display in a balloon several information about this resource by flying over a corresponding cell with the cursor. Furthermore, a right-click on a cell related to a resource and a time period allows the use of popup menus and therefore the use of functionalities that can take into account this two informations.

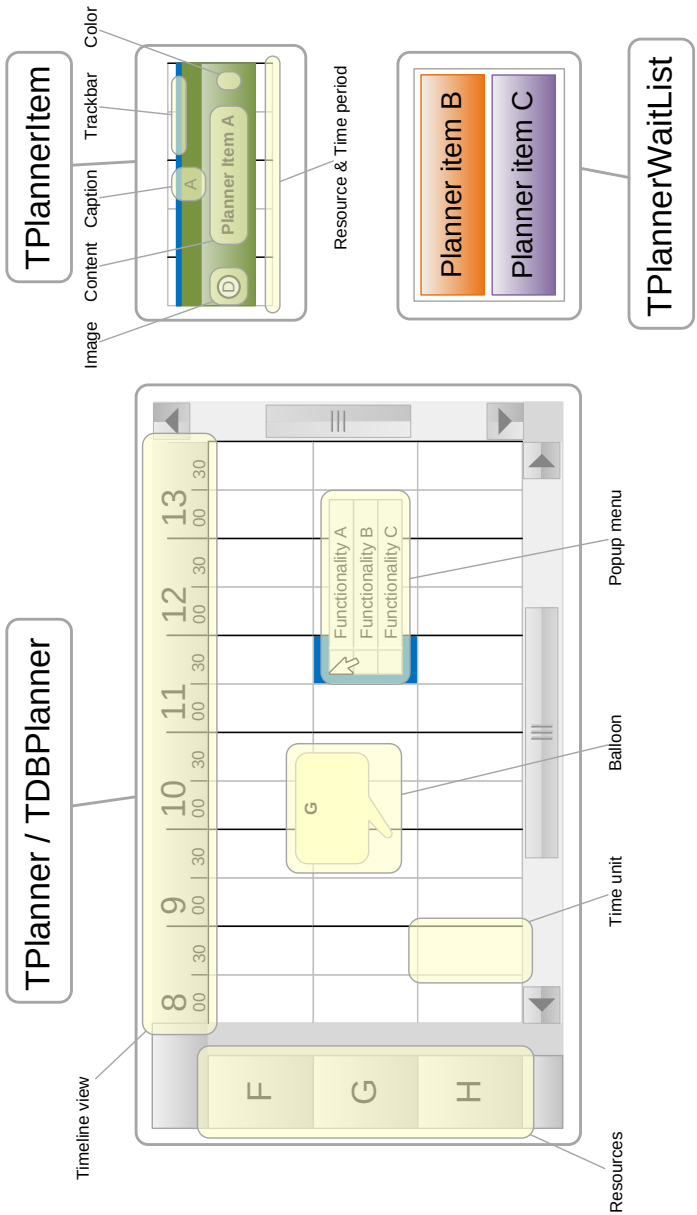

Figure 3.1: TMS Planner package

- **TPLANNERITEM** This component corresponds to a delivery, an ash recovery, a break or a planned task of the magnetic board. Each instance of this component, i.e. each software object, is defined by a resource, a time period, an image, a caption, a content, a color and a trackbar. Note that a planner item can be resized and moved directly on the planner if such an operation is authorized. Furthermore, a right-click on a planner item also allows the use of popup menus and therefore the use of functionalities that can consider the resource and the time period of this item.
- **TPLANNERWAITLIST** This component has no equivalent in the magnetic board. It is a list intended to contain planner items, which can be dragged and dropped in any cell of the planning grid. This component can be useful to list orders and to create deliveries of these orders. Note that popup menus can not be used with this component. However, double-clicking on a planner item of such a list can be linked to a functionality.

Two possibilities exist to save and load planner items displayed in a planner. It can be either programmed manually by the developer (cf. TPlanner), or done automatically with the use of a database (cf. TDBPlanner). Note that with a database, whose characteristics will be discussed later, modifications done in the planner are reported in real time in the database. Similarly, modifications done in a database are reported in real time in the planner.

3.2.2 Data storage

Various alternatives exist to store persistent data of an information system, which have to be retained even if this system is not running. These alternatives can however be separated into two groups. Indeed, either a filesystem or a database, which have both their advantages and drawbacks, can be used to store, retrieve or update these data.

Using a filesystem

The use of a filesystem is an easy way to manage data because these ones are saved in text files arranged hierarchically in folders. Due to Integrated Development Environments that provide functionalities to create such files, and to read and write their content, the installation of additional softwares is therefore not required. Different types of text files, illustrated in Figure 3.2, can be observed:

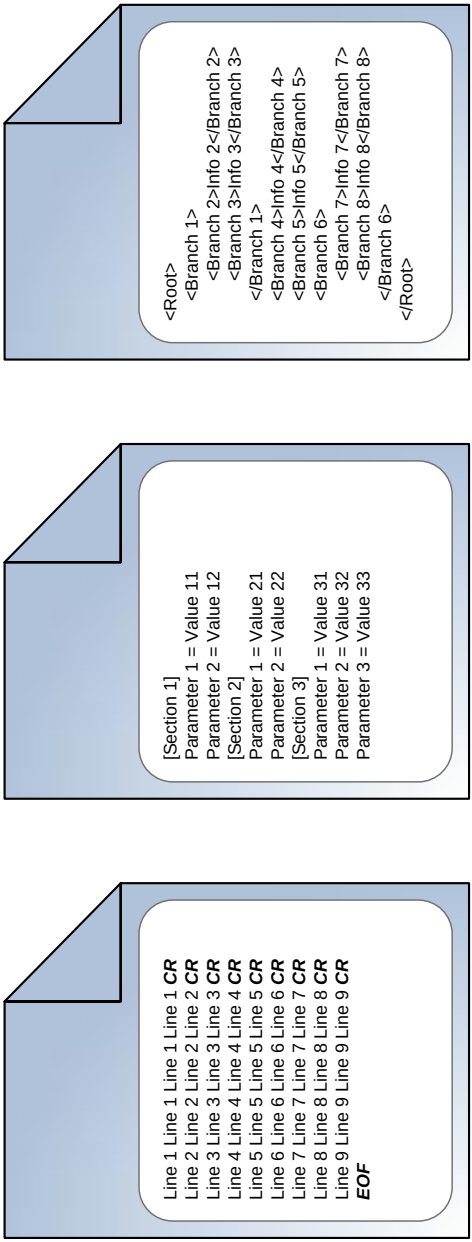

Figure 3.2: Structure of the data stored in different text files of a filesystem

- **STANDARD TEXT FILES (TXT)** This type of file stores data as a sequence of lines. To retrieve specific information stored in a such file, it is necessary to explore its content from the beginning to the end, line after line, until this information is found. Note that the end of a line is specified with a carriage return (CR) and that the end of a text file is specified with an end of file (EOF). To resume, this standard text file format does not provide an easy access to data.

- **INITIALIZATION TEXT FILES (INI)** This type of file stores data in a structured way because it is divided in sections, themselves composed of a set of parameters. A value can be allocated to each of these parameters. To retrieve specific information stored in a such file, it is sufficient to know the names of the section and the parameter of the corresponding value. To resume, two steps are required with initialization text file format to access data. Although appropriate to store settings, this type of text file is however limited to store complex structured data.

- **EXTENSIBLE MARKUP LANGUAGE TEXT FILES (XML)** This type of file stores data in a more structured way as a tree with a root and branches. Unlike initialization text files where there are only two levels (sections and parameters), the length of the tree and the number of branches of an XML text file are not restricted, i.e. many levels are allowed. To retrieve specific information stored in a such file, it is necessary to know the names of the tree branches that have to be browsed. To resume, the number of steps required by extensible markup language file format to access data corresponds to the level of the branch where are stored these data compared to the tree root. Additional information about XML technologies can be found in [36].

Despite their ease of use, any of the three types of text file presented before have however some drawbacks. When a file is opened and used by several users, only the first user having opened this file has the permission to save the modifications brought to its content. In other terms, the content of a file cannot be modified simultaneously by different users with a filesystem, what can be problematic if an information system has to be used by many users. Furthermore, a filesystem does not satisfy the principles of scalability and consistency defined at the beginning of this section. Although efficient when there is a few data, the use of a filesystem is not appropriate to manage a large amount of complex information due to the functionalities provided by any Integrated Development Environment, which are too basic to be efficient. In case of interruption of a system, modified and unsaved

content of text files opened by this system is not stored with a filesystem, what can compromise the integrity of its data. To learn in detail the characteristics of a filesystem, the reader can refer to [54, 55].

Using a database

The use of a database is a more complex way to store data. Indeed, it requires the installation of a database management system, which can be resumed as an additional software layer added on the operating system to manage information stored in a database. However, this additional layer allows a more efficient management of data by taking into account the principles of robustness, scalability and consistency whose implementation will be shortly described later.

Although the content of a database is physically stored in one or many files as with a filesystem, this content can only be accessed and modified through functionalities provided by a database management system. The manner the data are organized by such a system is independent of their physical storage and is defined by a database model. Note that only the relational model is considered here due to its widely use. With this model, each information is stored as a record in a table. Databases using the relational model are called relational databases because each one of them is composed of a set of tables that can be linked together. To access functionalities of a database management system, interfaces are available for any Integrated Development Environment.

- **ROBUSTNESS** With a filesystem, it is possible to store any type of value (integer, boolean, string of characters,...) in text files even if it does not correspond to the expectation of the information system that uses them. This can therefore compromise the operation of this system and imply the implementation of rigorous checks before any data processing. With a database, each information, i.e. each record, is composed of a set of values whose type is explicitly predefined and checked before any addition or modification. This task is directly done by the database management system, which ensures at this point of view a perfect stability or robustness of the system operation. To resume, an information system does not need to care if processed data are correctly typed or not with a database.

- **SCALABILITY** Relational databases are designed and implemented such a way that they can store and retrieve a large amount of data with a great performance. To read, modify and add records to a relational database, two

different methods are provided by any database management system. The first method consists in parsing the records of a selected table with dedicated functionalities. Unlike standard text files where it is necessary to explore their content line after line from the beginning to the end, tables of a relational database are more evolved. Indeed, they can also be explored from the end to the beginning and allow a quick access to specific information thanks to keys that are allocated to one or several values of each record. The value of a key allows to directly position on the first or last record having a same value. The second method consists in using a query language, which allows not only the merge of the content of different tables, but also the aggregation of values of same type. Relational databases use Structured Query Language (SQL), which is a declarative language. It means that it is only necessary to formulate what information is needed and not how to retrieve this information.

- **CONSISTENCY** As mentioned before, unsaved content of text files is not stored with a filesystem in case of a system interruption. To avoid the consequences of such a situation, database management systems allow the use of transactions when managing data. A transaction can be seen as a set of data modifications that has either to succeed or to fail. If a problem occurs during a transaction, all data involved in this transaction recover their initial state as it was at the beginning of the transaction. Otherwise, the new state of all involved data is stored. Thanks to transactions, the integrity of an information system, i.e. the consistency of its data, remains intact.

Unlike the content of text files, the content of tables of a database can be modified simultaneously by many users because the access to a table can be locked only when a user applies modifications to a record. More information about database management systems and the structured query language can be found in [27, 48].

Storage choice

Following the analysis of the characteristics of a filesystem and of a database, and considering the properties of the selected schedule package, it is obvious that the use of a relational database is required to develop an interactive planning system. Note that many providers of relational databases exist. Although the weighing system uses a Paradox relational database, the use of such a database is however not possible because it does not provide the mechanism of transactions. Furthermore, a Paradox relational database corresponds to a set of files that are not easy

to identify and move. Even if the relationships between the data of the interactive planning system can be complex (this will be discussed in Chapter 4), the amount of data that has to be managed is not huge. For many reasons that are presented hereafter, an InterBase relational database is chosen. First, this relational database is easy to install and manage, and provides the mechanism of transactions. Then, this relational database is developed by the same provider as Delphi and is therefore fully compatible. Finally, all information of an InterBase relational database is stored in only one file, what ensures an easy move of the data.

In addition to an InterBase relational database, two initialization text files are however required. The first one has to store the settings that allow the connection to the relational database, while the second one has to store preference settings related to the user interface of the interactive planning system.

3.2.3 Architecture

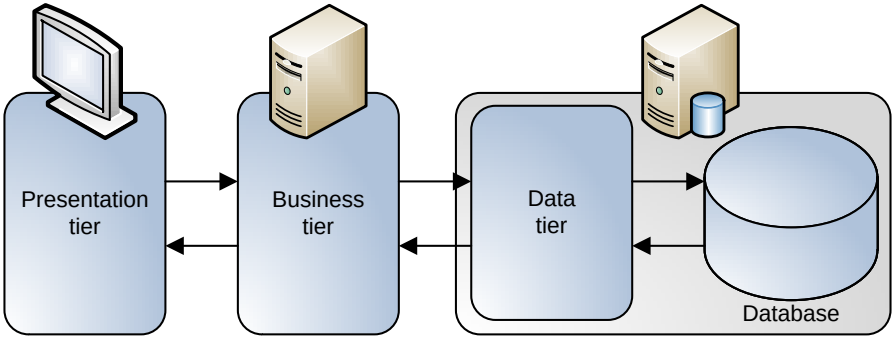

Figure 3.3: 3-tiers architecture

The architecture of an information system can be compared to an internal view of the operation of this system. To ensure performance when using functionalities and when communicating with other information systems, the required data of the interactive planning system to develop have to be logically structured and separated from their presentation and business processing. The '3-tiers' architecture model, illustrated in Figure 3.3, is adapted to these needs because it divides an information system into three levels named 'tiers':

1. **PRESENTATION TIER** This tier corresponds to the graphical user interface of the interactive planning system to develop. On the one hand, this interface displays information about daily transportation plannings and properties needed by the functionalities of the information system. On the other hand, this interface allows users to access these functionalities especially through popup menus and drag and drop options in order to interact with the information system. When a functionality is triggered by a user, although a few parameters can be collected by the presentation tier, the entire business process of this functionality is done by the business tier. For example, when a user creates a delivery with a drag and drop option, the presentation tier first collects the order, the resource and the start date time defined with this option, and then communicates to the business tier to create a delivery related to these three parameters.

2. **BUSINESS TIER** This tier corresponds to the operational part of the information system. Indeed, it processes the instructions of functionalities triggered from the presentation tier. Despite possible parameters collected by the presentation tier, the business tier often communicates with the data tier either to retrieve additional information needed by these functionalities, or to store persistent data. Following this, the business tier communicates the results to the presentation tier, which updates the content displayed by the graphical user interface. Regarding the creation of a delivery, the business tier requires from the data tier not only the duration of the four components of the delivery to create, but also the properties of the related truck and the multipurpose of its truck-driver to check beforehand if this creation is authorized. In case of success, a delivery is created and the business tier communicates to the presentation tier to update the content of the timeline schedule displayed on screen.

3. **DATA TIER** This tier is responsible for the access and storage management of the persistent data used by the interactive planning system to develop. It has not only to retrieve efficiently any information needed by the business tier when executing instructions, but also to add, modify or remove information of the system if this is defined in these instructions. If a database is used, the data tier can correspond to a set of queries formulated in the query language provided by the database management system. Regarding the creation of a delivery, the data tier especially executes on the one hand a query to recover in an aggregate manner the four durations that compose this delivery, and stores on the other hand information about this delivery as records in the appropriate tables of the database.

The use of a '3-tiers' architecture, which is a variant of the N-tiers architecture discussed in [34], offers several advantages. The separation between the graphical user interface and the functionalities of the information system allows the creation of other interfaces that can directly use these functionalities without having to redefine them. New interfaces can for example be built to give access to the functionalities of the interactive planning system from the command line, or to make them available for other information systems in order to complete the initial information system with additional modules intended to satisfy new requirements. The development of such a module will be discussed in the second part of this thesis.

A '3-tiers' architecture also allows to split an information system on several machines thanks to the associated 'client-server' principle. Indeed, an interface is the client of functionalities, considered as services, provided by the operational part of the system. Similarly, a functionality is the client of data, considered as services, provided by the storage management part of the system. For example, an interface could be installed on the workstation of each user, while the operational and storage management parts could be installed on a machine with great computing power. Although such a configuration is possible when using web technologies, i.e. a web browser and a web server, this is not possible with the Delphi programming language because this one is only designed to create software applications containing windows forms, i.e. executables including simultaneously the interface, the operational part, and the storage management part.

From the developers point of view, a '3-tiers' architecture allows on the one hand a better tasks repartition by creating teams of specialists focused on a particular tier and on the other hand, an easier evolution of the information system when adding new functionalities due to its architecture, what ensures therefore its durability.

3.3 Conclusion

In this chapter, the concept of the interactive planning system to develop has first been defined. Considering the analysis of the existing system and therefore the detected potential improvements, this concept also includes additional properties and functionalities expected by the cement factory. Following this definition, the technologies available and needed to develop the interactive planning system have been discussed and the most appropriate ones have been selected. It

concerns the choice of a programming language used to build the information system, a way to store the persistent data of this system, and an architecture that decomposes this system into several parts.

In practical terms, the programming language Delphi has been imposed by the cement factory, not only for its properties, but mainly due to the skill level of the employees responsible for the maintenance and improvement of the interactive planning system to develop. To complete this programming language or more precisely this Integrated Development Environment, the TMS Planner package, appropriate to develop planning and scheduling applications, has been selected, as the tools Inno Setup and ISTool, used for building installation and update utilities. Concerning the data storage, the use of a relational database is required due to the complexity of the data that have to be processed and stored. For this reason and due to its adequacy with the characteristics of the information system to develop and with Delphi, the database management system InterBase has been adopted. Regarding the system architecture, the '3-tiers' architecture which separates the data of their presentation and business processing seems appropriate and has therefore been selected.

The definition of the concept of the interactive planning system can be considered as a complement of the analysis of the existing system detailed in Chapter 2 that has allowed the selection of the required technologies for its development. This information will now be helpful to really begin the development of the information system exposed in the following chapters with the use of the programming language, the database management system, the system architecture, and additional tools presented and selected here for this purpose.

Chapter 4

Database design

After having defined the concept of the interactive planning system and selected the technologies appropriate for its development, the next step consists in analyzing the data that will be managed and processed by this system. It concerns the lower layer of the system architecture, i.e. the data tier, and consists in practical terms to design the logical structure of the relational database that will be used by the information system. For this purpose, an entity-relationship diagram which links the data together is first presented in Section 4.1. This diagram is used then in Section 4.2 to describe a relational database schema that gives more precisions about these data and their relationships. A conclusion is finally given in Section 4.3.

4.1 Entity-relationship diagram

The elaboration of the entity-relationship diagram related to the persistent data of the interactive planning system is done using the entity-relationship model. This model considers two different types of elements, entities and relationships, which are briefly described hereafter:

- **ENTITY** An entity is a specific object that stands out from other objects and can correspond to a person, an object, a concept or an event. Entities of

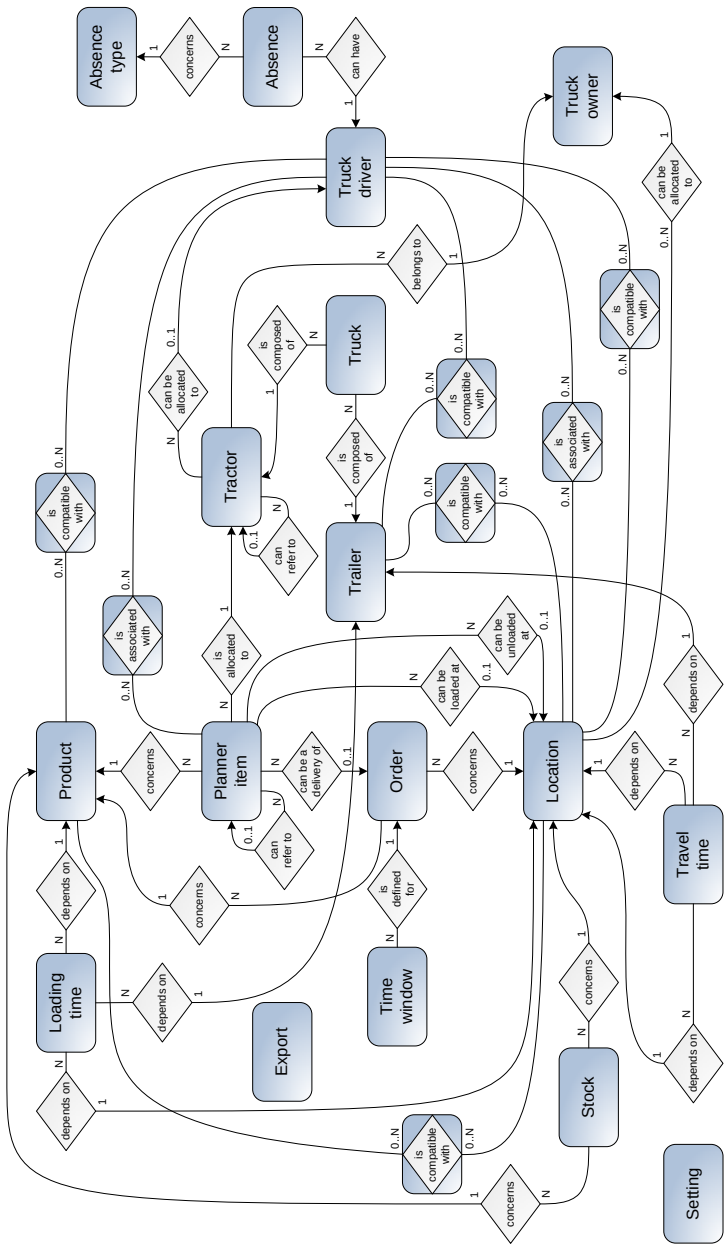

Figure 4.1: Entity-relationship diagram

same type form a set of entities and are characterized by one or many attributes according to their properties. Among these attributes, one of them is defined as an identification key, which allows to distinguish without ambiguity the entities of a same set. An entity can for example be one of the truck-drivers belonging to the cement factory who are characterized and identified by a first name and a last name.

- **RELATIONSHIP** A relationship from a set of entities to another set of entities defines an oriented link in this direction. For example, if the orders and the cement types are considered as two different sets of entities, a relationship from the orders to the cement types means that each order concerns a cement type. The type of relationship from a set of entities to another set of entities indicates the number of entities of each of these sets that are linked with the entities of the other set. Four types of relationships exist: a single relationship (1) where exactly one entity of a set is linked with entities of another set; a conditional relationship (0..1) where at most one entity of a set is linked with entities of another set; a multiple relationship (N) where many entities of a set are linked with entities of another set; and a conditional or multiple relationship (0..N) where no, one or many entities of a set are linked with entities of another set.

More information about entities, relationships, and the entity-relationship model can be found in the second chapter of [46] that discusses about the construction phases of a data model. Following this description, it is now possible to present the entity-relationship diagram, which is illustrated in Figure 4.1. Sets of entities are represented by blue rectangles and relationships by grey diamonds. When diamonds are combined with rectangles, it corresponds to a couple of multiple relationships. Although this information is not relevant in this section, it will be useful for the description of the relational database schema. Note that this diagram does not include attributes and identification keys.

The cement types, ash types, task types and the break can be grouped to form the set of entities **Product**. This set considers therefore not only physical materials, but also intangible elements. The factory and third parties that own the trucks used to do the cement deliveries or ashes recoveries form the set of entities **Truck owner**. Located at different places, the cement factory, the local depots, the customers, the suppliers, and the residences of drivers who can park their truck home form the set of entities **Location**. The truck-drivers belonging to the cement factory form the set of entities **Truck driver**. The trucks belonging either to the cement factory or to third parties, composed of a tractor and a trailer, form the sets

of entities **Tractor**, **Trailer** and **Truck**. Note that third parties drivers are implicitly allocated to third parties trucks. The durations between two different locations form the set of entities **Travel time** and the times needed to load or unload the cement types or ash types at corresponding locations form the set of entities **Loading time**. The stocks of cement types available in the local depots form the set of entities **Stock**. Periods when the truck-drivers are not available form the set of entities **Absence** and the possible reasons of these unavailabilities form the set of entities **Absence type**. The daily orders of the customers form the set of entities **Order** and the periods when deliveries of these orders have to be done form the set of entities **Time window**. Deliveries, ashes recoveries, breaks and planned tasks, which represent time events, form the set of entities **Planner item**. Various settings required to the operation of the information system form the set of entities **Setting**, and the data needed by the weighing system form the set of entities **Export**.

Each truck (**Truck**) is composed of a tractor (**Tractor**) and a trailer (**Trailer**). Each tractor belongs to a truck owner (**Truck owner**), can be allocated as default to a truck-driver (**Truck driver**), and can refer to another tractor. Each trailer can only access some customers and some local depots (**Location**). Furthermore, each trailer needs a special license to be driven that do not own all truck-drivers. Note that each customer is allocated to a truck owner whose fleet of trucks is responsible for the deliveries of all his orders. Due to different skill levels, each truck-driver can only access some customers and some suppliers. Note that the traceability of these accesses is kept for each truck-driver. Each travel time (**Travel time**) depends not only on the departure and arrival locations, but also on the trailer that is used for this travel. Each loading time (**Loading time**) depends on the one hand on the loaded or unloaded cement type or ash type (**Product**) and on the other hand, on the concerned trailer and on the location where the product is loaded or unloaded. Note that the cement types and ash types can only be handled by truck-drivers having the appropriate skill level. Furthermore, each customer and each supplier respectively orders and supplies only some cement types and ash types. Each available stock (**Stock**) concerns a cement type (**Product**) and a local depot (**Location**). Each absence (**Absence**) concerns a truck-driver and a predefined reason of unavailability (**Absence type**). Each order (**Order**) concerns a cement type and a customer. Note that periods can be defined for each order to indicate when the deliveries of this order have to be done (**Time window**). Each delivery, ash recovery, break or planned task (**Planner item**) concerns a product and is allocated to a tractor. Note that no, one or many truck-drivers can be associated to each of these planner items, and that each planner item can refer to

another planner item. Each delivery concerns an order, is loaded at the cement factory or at one of the local depots, and is unloaded at a customer. Each ash recovery is loaded at a supplier and is unloaded at the cement factory.

4.2 Relational database schema

The entity-relationship diagram can now be used to describe the corresponding relational database schema that allows to represent the sets of entities and their relationships with tables, attributes and primary keys. First, each set of entities has to be translated in one distinct table composed of attributes and a unique primary key, i.e. an identification key, that corresponds to one or many of these attributes. Then, each couple of multiple relationships (0..N,0..N), represented in the entity-relationship diagram with diamonds combined with rectangles, has also to be translated in one distinct table. Such a table is composed of two foreign keys, i.e. two attributes related to the primary keys of the two tables involved in this relationship. The primary key of the table is formed either with the two foreign keys, either with an arbitrary primary key corresponding to one or many attributes. Finally, each couple of other relationships between two tables can be expressed without having to create an additional table. Indeed, a foreign key has to be defined in the table having the single relationship in order to link it with the other table.

Additional information concerning the translation of an entity-relationship diagram in a relational database schema can also be found in the second chapter of [46]. Following this description, it is now possible to present the relational database schema, which is illustrated in Figure 4.2, by discussing hereafter the role and content of each table. Properties of the attributes of each table are presented in a table that not only gives information about the label and type of these attributes, but also indicates which of these attributes correspond to the primary key (PK), are foreign keys (FK), and always require a value (RV). Note that for each foreign key, the table corresponding to the relationship is also indicated.

Product

This table stores the products that can be allocated to the planner items of the timeline schedule. These products include the cement types that can be delivered to the customers, the ash types that can be recovered from the suppliers, the break

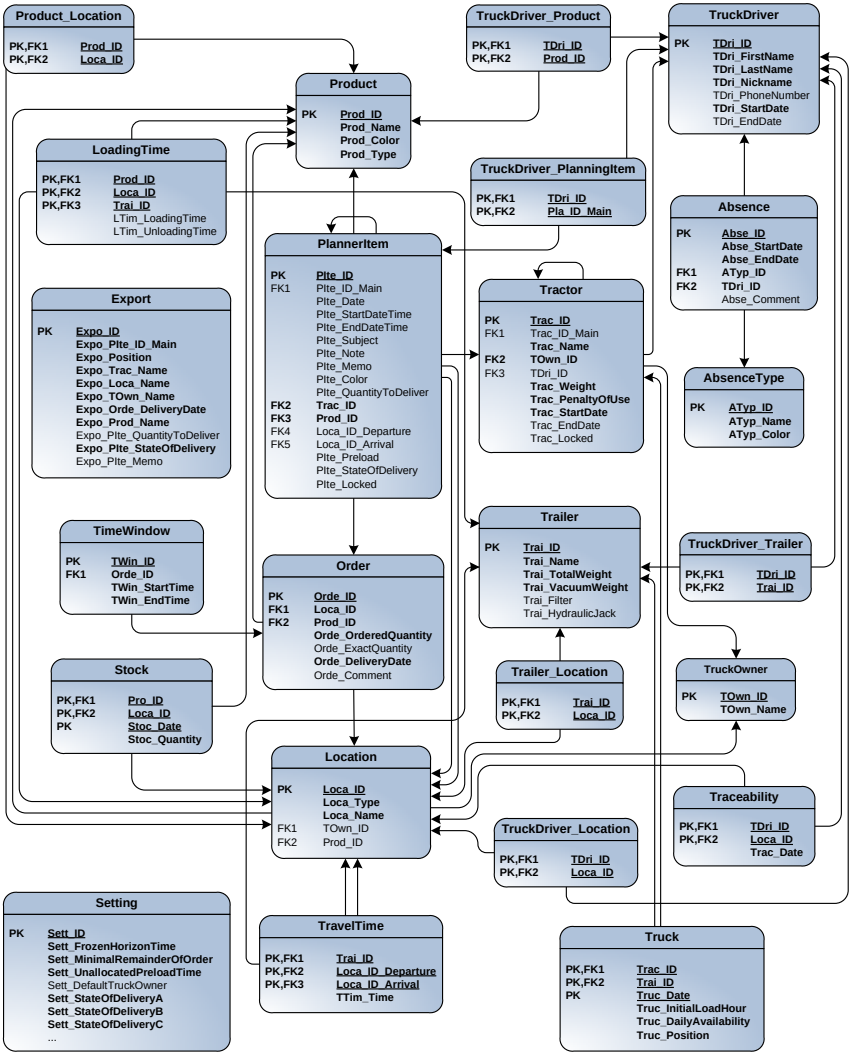

Figure 4.2: Relational database schema

that can be allocated to the truck-drivers, and the task types that can be allocated either to the trucks or to their driver. Identified by an arbitrary primary key, each record is composed of a name, a color, and a type. The type allows to determine which functionalities of the interactive planning system can be used with a planner item. Only four product types are available: 'delivery', 'ash recovery', 'break' and 'planned task'. Unlike the other types, 'break' can be assigned to only one product. Properties of the attributes listed above are presented in Table 4.1.

Table 4.1: Product

Label	Type	PK	FK	RV	Relationship
Prod_ID	integer	✓		✓	
Prod_Name	varchar(50)			✓	
Prod_Color	integer			✓	
Prod_Type	varchar(50)			✓	

Truck owner

This table stores the owners of the trucks used by the cement factory to do the cement deliveries and ashes recoveries. These truck owners include the cement factory and third parties. Identified by an arbitrary primary key, each record is composed of a name. Properties of the attributes listed above are presented in Table 4.2.

Table 4.2: Truck owner

Label	Type	PK	FK	RV	Relationship
TOwn_ID	integer	✓		✓	
TOwn_Name	varchar(50)			✓	

Location

This table stores the locations that can be allocated to the planner items of the timeline schedule if these ones correspond to deliveries or ashes recoveries. These locations concern the customers of cements, the suppliers of ashes, the main and

local depots used to supply the customers, i.e. the cement factory and the railway stations, and the residences of the truck-drivers who can drive their truck home for the first delivery of the day after. Identified by an arbitrary primary key, each location is composed of a type, a name, a truck owner, and a product. The type allows to distinguish the locations that can be involved in functionalities of the information system. Only five location types are available: 'customer', 'supplier', 'factory', 'depot', and 'truck-driver'. Unlike the other types, 'factory' can be assigned to only one location. The truck-owner and the default product can only be selected if the type of location corresponds to 'customer'. Unlike the selection of the default product that is optional, the selection of the truck-owner is required. Properties of the attributes listed above are presented in Table 4.3.

Table 4.3: Location

Label	Type	PK	FK	RV	Relationship
Loca_ID	integer	✓		✓	
Loca_Type	varchar(50)			✓	
Loca_Name	varchar(50)			✓	
TOwn_ID	integer		✓		Table 4.2
Prod_ID	integer		✓		Table 4.1

Truck-driver

This table stores the truck-drivers belonging to the cement factory. Identified by an arbitrary primary key, each record is composed of a first name, a last name, a nickname, a phone number, a start date, and an end date. The phone number allows the information system to communicate via SMS the transportation planning of the trucks related to the truck-driver. The start date and end date define the commitment period of the truck-driver. If the end date is not set, it means that the end of the commitment period is not defined. Properties of the attributes listed above are presented in Table 4.4.

Tractor

This table stores the tractors belonging either to the cement factory or to third parties. Identified by an arbitrary primary key, each record is composed of a main identifier, a name, a truck owner, a truck-driver, a weight, a penalty of use, a start

Table 4.4: Truck-driver

Label	Type	PK	FK	RV	Relationship
TDri_ID	integer	✓		✓	
TDri_FirstName	varchar(50)			✓	
TDri_LastName	varchar(50)			✓	
TDri_Nickname	varchar(50)			✓	
TDri_PhoneNumber	varchar(50)				
TDri_StartDate	date			✓	
TDri_EndDate	date				

date, an end date, and a locking status. The main identifier allows a record to refer to another record of the same table. A tractor can therefore refer to another existing tractor. This allows to represent in the timeline schedule two different resources related to the same truck. More precisions about the use of this attribute will be given in Chapter 5. Each tractor belongs to a truck owner. The selection of a truck-driver can be done to define the default truck-driver to use when creating planner items related to the truck. The input of a weight allows to define the payload of the truck that uses the tractor. The penalty of use concerns here the carbon dioxide emissions of the tractor. The start date and end date define the using period of the tractor. If the end date is not set, it means that the end of the using period is not defined. The locking status allows to prevent that the planner items related to a locked tractor are modified by external information systems. Properties of the attributes listed above are presented in Table 4.5.

Table 4.5: Tractor

Label	Type	PK	FK	RV	Relationship
Trac_ID	integer	✓		✓	
Trac_ID_Main	integer				Table 4.5
Trac_Name	varchar(50)			✓	
TOwn_ID	integer			✓	Table 4.2
TDri_ID	integer				Table 4.4
Trac_Weight	integer			✓	
Trac_PenaltyOfUse	double			✓	
Trac_StartDate	date			✓	
Trac_EndDate	date				
Trac_Locked	boolean				

Trailer

This table stores the trailers combined with tractors to create trucks. Identified by an arbitrary primary key, each record is composed of a name, a total weight, a vacuum weight, a status related to a filter, and a status related to an hydraulic jack. The input of a total weight and a vacuum weight, which respectively correspond to the weight of the trailer when it is fully loaded and empty, allows to define the payload of the truck that uses the trailer. Note that a tractor can be equipped with a filter and with an hydraulic jack. While a filter allows the load of cement types at the local depots, an hydraulic jack is appropriate for some customers. Properties of the attributes listed above are presented in Table 4.6.

Table 4.6: Trailer

Label	Type	PK	FK	RV	Relationship
Trai_ID	integer	✓		✓	
Trai_Name	varchar(50)			✓	
Trai_TotalWeight	integer			✓	
Trai_VacuumWeight	integer			✓	
Trai_Filter	boolean				
Trai_HydraulicJack	boolean				

Truck

This table stores the trucks used to do the cement deliveries and ashes recoveries. Identified by a primary key composed of the identifier of the used tractor, the identifier of the used trailer, and a date, each record is composed of an initial load hour, a daily availability, and a position. The date concerns the day when the truck has to be displayed as resource of the timeline schedule. The initial load hour corresponds to the start time of the first delivery if this delivery has to be automatically created. The daily availability concerns the availability of the truck in hours during a day. The position corresponds to the position of the truck, i.e. the resource, compared with other trucks in the timeline schedule. Properties of the attributes listed above are presented in Table 4.7.

Table 4.7: Truck

Label	Type	PK	FK	RV	Relationship
Trac_ID	integer	✓	✓	✓	Table 4.5
Trai_ID	integer	✓	✓	✓	Table 4.6
Truc_Date	date	✓		✓	
Truc_InitialLoadHour	time			✓	
Truc_DailyAvailability	double			✓	
Truc_Position	integer			✓	

Multipurpose truck-driver / trailer

This table stores the relationships between the truck-drivers and the trailers. Each record is only identified by a primary key composed of the identifier of a truck-driver, and the identifier of a trailer. These relationships correspond to the multi-purposes of the drivers concerning their ability to drive the trailers. Each record means that the driver owns the required license to drive the trailer. Properties of the attributes listed above are presented in Table 4.8.

Table 4.8: Multipurpose truck-driver / trailer

Label	Type	PK	FK	RV	Relationship
TDri_ID	integer	✓	✓	✓	Table 4.4
Trai_ID	integer	✓	✓	✓	Table 4.6

Multipurpose truck-driver / location

This table stores the relationships between the truck-drivers and the locations. These locations only concern the customers of cements, and the suppliers of ashes. Each record is only identified by a primary key composed of the identifier of a truck-driver, and the identifier of a location. These relationships correspond to the multipurposes of the drivers concerning their ability to access the customers and the suppliers. Each record means that the driver has the required skill level to access the location. Properties of the attributes listed above are presented in Table 4.9.

Table 4.9: Multipurpose truck-driver / location

Label	Type	PK	FK	RV	Relationship
TDri_ID	integer	✓	✓	✓	Table 4.4
Loca_ID	integer	✓	✓	✓	Table 4.3

Traceability

This table also stores the relationships between the truck-drivers and the locations. These locations only concern the customers of cements, and the suppliers of ashes. Identified by a primary key composed of the identifier of a truck-driver and the identifier of a location, each record is composed of a date. These relationships correspond to the traceability between the drivers and their accesses to the customers and the suppliers. Each record means that the driver has accessed the location for the first time at the specified date. Properties of the attributes listed above are presented in Table 4.10.

Table 4.10: Traceability

Label	Type	PK	FK	RV	Relationship
TDri_ID	integer	✓	✓	✓	Table 4.4
Loca_ID	integer	✓	✓	✓	Table 4.3
Trac_Date	date			✓	

Multipurpose truck-driver / product

This table stores the relationships between the truck-drivers and the products. These products only concern the cement types, and the ash types. Each record is only identified by a primary key composed of the identifier of a truck-driver, and the identifier of a product. These relationships correspond to the multipurposes of the drivers concerning their ability to handle the cement types and the ash types. Each record means that the driver has the required skill level to handle the product. Properties of the attributes listed above are presented in Table 4.11.

Table 4.11: Multipurpose truck-driver / product

Label	Type	PK	FK	RV	Relationship
TDri_ID	integer	✓	✓	✓	Table 4.4
Prod_ID	integer	✓	✓	✓	Table 4.1

Multipurpose trailer / location

This table stores the relationships between the trailers and the locations. These locations only concern the customers of cements, and the local depots located in the railway stations. Each record is only identified by a primary key composed of the identifier of a trailer, and the identifier of a location. These relationships correspond to the multipurposes of the trailers concerning their possibility to access the customers and the local depots. Each record means that the trailer has the required characteristics to access the location. Properties of the attributes listed above are presented in Table 4.12.

Table 4.12: Multipurpose trailer / location

Label	Type	PK	FK	RV	Relationship
Trai_ID	integer	✓	✓	✓	Table 4.6
Loca_ID	integer	✓	✓	✓	Table 4.3

Multipurpose product / location

This table stores the relationships between the products and the locations. These products only concern the cement types, and the ash types. Furthermore, those locations only concern the customers of cements, and the suppliers of ashes. Each record is only identified by a primary key composed of the identifier of a product, and the identifier of a location. These relationships correspond to the multipurposes of the cement types and ash types that can respectively be delivered to the customers and recovered from the suppliers. Each record means that the product can be recovered/delivered from/to the location. Properties of the attributes listed above are presented in Table 4.13.

Table 4.13: Multipurpose product / location

Label	Type	PK	FK	RV	Relationship
Prod_ID	integer	✓	✓	✓	Table 4.1
Loca_ID	integer	✓	✓	✓	Table 4.3

Travel time

This table stores the durations needed by the trailers to travel between two different locations. Identified by a primary key composed of the identifier of a tractor, the identifier of a location related to the departure, and the identifier of a location related to the arrival, each record is composed of a travel time. Although concerning all trailers, the travel times can only be defined between the cement factory and the customers, the customers and the cement factory, the local depots and the customers, the customers and the local depots, the home of the truck-drivers and the customers, the cement factory and the suppliers, the suppliers and the cement factory, and between the customers and the suppliers. Properties of the attributes listed above are presented in Table 4.14.

Table 4.14: Travel time

Label	Type	PK	FK	RV	Relationship
Trai_ID	integer	✓	✓	✓	Table 4.6
Loca_ID_Departure	integer	✓	✓	✓	Table 4.3
Loca_ID_Arrival	integer	✓	✓	✓	Table 4.3
TTim_Time	double			✓	

Loading time

This table stores the durations needed by the trailers to load or unload their products at the locations. All locations are concerned except the residences of the truck-drivers who can drive their truck, loaded at the cement factory, home for the first delivery of the day after. Identified by a primary key composed of the identifier of a product, the identifier of a location, and the identifier of a trailer, each record is composed of a loading time, and an unloading time. Note that only one type of duration, i.e. a loading time or an unloading time, can be stored for

each record. The loading times can only be defined for the cement factory and the local depots with cement types, and for the suppliers with ash types. The unloading times can only be defined for the customers with the cement types, and for the cement factory with the ash types. Properties of the attributes listed above are presented in Table 4.15.

Table 4.15: Loading time

Label	Type	PK	FK	RV	Relationship
Prod_ID	integer	✓	✓	✓	Table 4.1
Loca_ID	integer	✓	✓	✓	Table 4.3
Trai_ID	integer	✓	✓	✓	Table 4.6
LTim_Loading_Time	double			✓	
LTim_Unloading_Time	double			✓	

Stock

This table stores the stocks of products available in the locations. The products only concern the cement types and the locations only concern the local depots located in the railway stations. Identified by a primary key composed of the identifier of a product, the identifier of a location, and a date, each record is composed of a quantity of available stock. The specification of a date allows at irregular interval to realize intermittent inventories. For each date of planning, the stock of the closest date of inventory can therefore be taken into account. Properties of the attributes listed above are presented in Table 4.16.

Table 4.16: Stock

Label	Type	PK	FK	RV	Relationship
Prod_ID	integer	✓	✓	✓	Table 4.1
Loca_ID	integer	✓	✓	✓	Table 4.3
Stoc_Date	date	✓		✓	
Stoc_Quantity	integer			✓	

Absence type

This table stores the absence types used to briefly describe the reasons of unavailabilities of the truck-drivers. Identified by an arbitrary primary key, each record is composed of a name, and a color. Properties of the attributes listed above are presented in Table 4.17.

Table 4.17: Absence type

Label	Type	PK	FK	RV	Relationship
ATyp_ID	integer	✓		✓	
ATyp_Name	varchar(50)			✓	
ATyp_Color	integer			✓	

Absence

This table stores the absences of the truck-drivers. Identified by an arbitrary primary key, each record is composed of a start date, an end date, an absence type, a truck-driver, and a comment. The start date and end date correspond to the period when the truck-driver is absent. The absence type concerns one of the possible reasons of this unavailability, and the truck-driver refers to a driver belonging to the cement factory. Note that a comment can eventually be written. Properties of the attributes listed above are presented in Table 4.18.

Table 4.18: Absence

Label	Type	PK	FK	RV	Relationship
Abse_ID	integer	✓		✓	
Abse_StartDate	date			✓	
Abse_EndDate	date			✓	
ATyp_ID	integer		✓	✓	Table 4.17
TDri_ID	integer		✓	✓	Table 4.4
Abse_Comment	varchar(250)				

Order

This table stores the orders of cement done everyday by the customers. Identified by an arbitrary primary key, each record is composed of a location, a product, an ordered quantity, a status related to this ordered quantity, a delivery date, and a comment. The locations correspond to the customers and the products concern the cement types. The status related to the exact quantity allows to mention if the delivery of a surplus of cement is authorized or not. Properties of the attributes listed above are presented in Table 4.19.

Table 4.19: Order

Label	Type	PK	FK	RV	Relationship
Orde_ID	integer	✓		✓	
Loca_ID	integer		✓	✓	Table 4.3
Prod_ID	integer		✓	✓	Table 4.1
Orde_OrderedQuantity	integer			✓	
Orde_ExactQuantity	boolean				
Orde_DeliveryDate	date			✓	
Orde_Comment	varchar(250)				

Time window

This table stores the time windows used to define the periods when the deliveries of an order have to be done. Identified by an arbitrary primary key, each record is composed of an order, a start time, and an end time. If no time window is defined for an order, its deliveries can be done during the whole day. Properties of the attributes listed above are presented in Table 4.20.

Table 4.20: Time window

Label	Type	PK	FK	RV	Relationship
TWin_ID	integer	✓		✓	
Orde_ID	integer		✓	✓	Table 4.19
TWin_StartTime	time			✓	
TWin_EndTime	time			✓	

Planner item

This table stores the planner items, or more precisely their content, that can be displayed in the timeline schedule. These planner items include the cement deliveries intended for the customers, the ashes recoveries coming from the suppliers, the breaks allocated to the truck-drivers, and the planned tasks allocated either to the trucks or to their driver. Identified by an arbitrary primary key, each record is composed of a main identifier, a date, a start time, an end time, a subject, a note, a memo, a color, a quantity to deliver, a tractor, a product, a departure location, an arrival location, a status related to the delivery, a status related to the state of this delivery, and a locking status. The main identifier allows a record to refers to another record of the same table. A planner item can therefore refer to another existing planner item. This allows to represent in the timeline schedule successive planner items related to a same planner item. More precisions about the use of this attribute will be given in Chapter 5. The date indicates the day when the planner item has to be displayed in the timeline schedule, and the start time and end time correspond to the time period of this item. The subject, note, memo and color allow to graphically represent the planner item in the timeline schedule. Note that the quantity of product that has to be delivered only concerns planner items related to deliveries. The tractor corresponds to the resource of the timeline schedule for which the planner item is created, and the product concerns a cement type, an ash type, a break or a task. The departure and arrival locations are only required for planner items related to deliveries or ashes recoveries. The status related to the delivery and to the state of this delivery can only be used for deliveries. While the first status indicates if a delivery is preloaded or not, the second one mentions if a delivery is not loaded, in loading, or loaded. Appropriate functionalities of the information system can therefore be used according to the state of the delivery. The locking status allows to lock or unlock a planner item, i.e. to unauthorized or not its moving in the timeline schedule. Properties of the attributes listed above are presented in Table 4.21.

Truck-driver / Planner item

This table stores the relationships between the truck-drivers and the planner items. Each record is only identified by a primary key composed of the identifier of a truck-driver, and the main identifier of a planner item. These relationships correspond to the drivers that are associated to the planner items. Each record means that the driver is involved in the event related to the planner item. Properties of

the attributes listed above are presented in Table 4.22.

Table 4.21: Planner item

Label	Type	PK	FK	RV	Relationship
PIte_ID	integer	✓		✓	
PIte_ID_Main	integer		✓		Table 4.21
PIte_Date	date				
PIte_StartDateTime	timestamp				
PIte_EndDateTime	timestamp				
PIte_Subject	varchar(250)				
PIte_Note	varchar(250)				
PIte_Memo	varchar(250)				
PIte_Color	integer				
PIte_QuantityToDeliver	integer				
Trac_ID	integer		✓	✓	Table 4.5
Prod_ID	integer		✓	✓	Table 4.1
Loca_ID_Departure	integer		✓		Table 4.3
Loca_ID_Arrival	integer		✓		Table 4.3
PIte_Preload	boolean				
PIte_StateOfDelivery	integer				
PIte_Locked	boolean				

Table 4.22: Truck-driver / Planner item

Label	Type	PK	FK	RV	Relationship
TDri_ID	integer	✓	✓	✓	Table 4.4
Pla_ID_Main	integer	✓	✓	✓	Table 4.21

Setting

This table stores the settings needed by the interactive planning system to operate. Unlike tables presented before, only one record has to be stored in this table. Identified by an arbitrary primary key, this record, whose number of attributes corresponds to the number of required settings, is especially composed of a duration related to a frozen horizon, a quantity related to the minimal remainder of an order, a duration related to the non allocated preload time, a default truck owner, and three different states of the delivery. The frozen horizon can be described as a

period where it is not possible to modify concerned planner items. This period is generally included between the past and the actual time. With the frozen horizon time, it is possible to extend the frozen horizon beyond the actual time by defining an additional duration that is added to the actual time. The frozen horizon and its rules will be presented in detail in Section 6.2 of Chapter 6. The minimal remainder of an order corresponds to the quantity of cement that does not need to be delivered to entirely satisfy an order. The unallocated preload time concerns the default duration of a delivery that is not yet allocated to a customer. The default truck owner corresponds to the truck owner related to the cement factory and allows to distinguish him from third parties. Each different state of the delivery allows to define a set of numbers separated by semicolons. Reported in the planner items related to deliveries, each of these numbers, concerning one of the three different states of the delivery (not loaded; in loading; loaded), allows the use of the appropriate functionalities. Properties of the attributes listed above are presented in Table 4.23.

Table 4.23: Setting

Label	Type	PK	FK	RV	Relationship
Sett_ID	integer	✓		✓	
Sett_FrozenHorizonTime	double			✓	
Sett_MinimalRemainderOfOrder	integer			✓	
Sett_UnallocatedPreloadTime	double			✓	
Sett_DefaultTruckOwner	varchar(6)				
Sett_StateOfDeliveryA	varchar(50)			✓	
Sett_StateOfDeliveryB	varchar(50)			✓	
Sett_StateOfDeliveryC	varchar(50)			✓	
...					

Export

This table stores the data of the transportation planning that are required by the weighing system to allow the load of the trucks with the appropriate cement types at the available docks. These data only concern the planner items related to the deliveries of the transportation planning. Identified by an arbitrary primary key, each record is composed of a main identifier, a position, a tractor, a location, a truck owner, a delivery date, a quantity to deliver, a state of the delivery, and a memo. The content of all these attributes comes from the tables 'Planner item',

'Tractor', 'Location', 'Truck owner', and 'Order', except for the third attribute that corresponds to the position of the main planner item, i.e. the delivery, in the timeline schedule of the truck compared with other deliveries of this truck. Properties of the attributes listed above are presented in Table 4.24.

Table 4.24: Export

Label	Type	PK	FK	RV	Relationship
Expo_ID	integer	✓		✓	
Expo_PIte_ID_Main	integer			✓	
Expo_Position	integer			✓	
Expo_Trac_Name	varchar(50)			✓	
Expo_Loca_Name	varchar(50)			✓	
Expo_TOwn_Name	varchar(50)			✓	
Expo_Orde_DeliveryDate	date			✓	
Expo_PIte_QuantityToDeliver	integer				
Expo_PIte_StateOfDelivery	integer			✓	
Expo_PIte_Memo	varchar(250)				

4.3 Conclusion

In this chapter, the data needed by the interactive planning system to operate have been analyzed and grouped together in order to design the logical structure of a relational database usable by this system. For this purpose, an entity-relationship diagram has first been elaborated thanks to the entity-relationship model, which allows to link the required data together using entities and relationships. This entity-relationship diagram has then been translated in a relational database schema, which gives especially more information about the entities by mentioning their attributes. Indeed, each entity and multiple relationship is translated in a table, which presents in detail its characteristics involved in the future functionalities of the information system.

The logical structure of a relational database intended for the interactive planning system will be useful for the remainder of the project. Indeed, it will allow to create a database with the appropriate tables and attributes in InterBase, the selected relational database management system. Furthermore, it will be helpful to create the user interface and to implement the expected functionalities in Delphi, the chosen Integrated Development Environment, thanks to the required data that are now delimited and provided in a structured way.

Chapter 5

Graphical user interface and functionalities

Following the design of a relational database usable by the interactive planning system in previous chapter, the next step focuses on the two upper and remaining layers of the system architecture, i.e. the presentation tier and the business tier. In practical terms, it consists in building the software tool with the use of the programming language and the tools selected for this purpose. This chapter does not discuss about the technical aspects related to the construction of the interactive planning system, but presents the obtained result, i.e. its graphical user interface and its functionalities, as they are perceived by the user. The graphical user interface is first presented in Section 5.1. The functionalities are then described in Section 5.2. A conclusion is finally given in Section 5.3.

5.1 Graphical user interface presentation

As defined in the concept, the graphical user interface is user friendly. Indeed, the use of the software tool is intuitive thanks to the integration of high level software objects and to the availability, where possible, of input supports. Due to the richness of this interface, not all the content of the windows that compose the interactive planning system are presented here, but only the general layout of

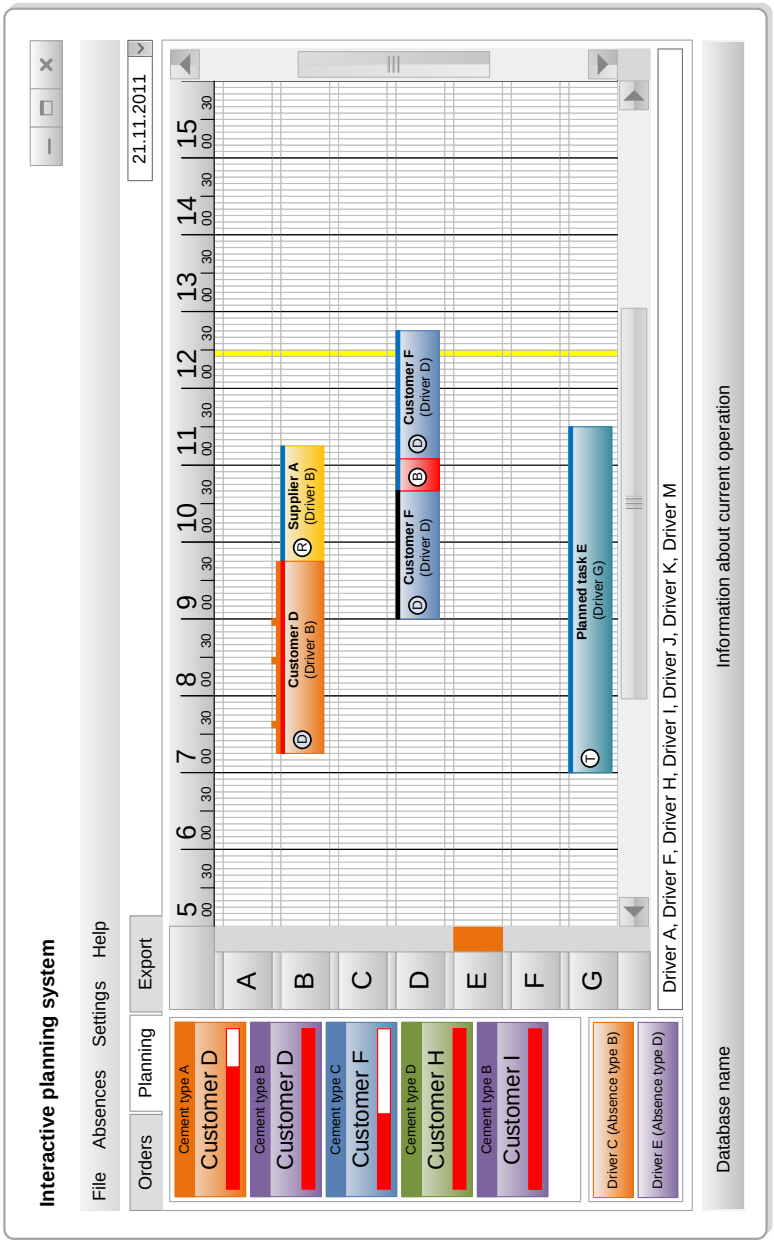

Figure 5.1: Main window of the interactive planning system

the main window, discussed in Subsection 5.1.1, and the content of the tab 'Planning', detailed in Subsection 5.1.2, which displays the transportation planning and gives access to the core functionalities of the software tool. Additional information about the graphical user interface that are not mentioned in this section can however be found in [56], the user manual of the interactive planning system.

5.1.1 Main window

Illustrated in Figure 5.1, the main window that is opened when starting the interactive planning system is composed of a main menu, a date picker, three tabs, and a status bar.

Considering the main menu, the 'File' option opens a sub-menu whose sub-options allow not only to print or quit the application, but also give access to functionalities that do not directly concern the transportation planning elaboration. Indeed, the content of some tables of the database used by the interactive planning system can be updated from the weighing system, as the state of the deliveries displayed in the tab 'Planning', which can be not loaded, in loading, or loaded. Furthermore, working reports and messages sent by SMS can be created for the truck-drivers. Finally, statistics can be generated from the content of the used database, which can also be archived or purged if needed. While the 'Absences' option allows to manage the absences of the truck-drivers, the 'Settings' option opens a sub-menu whose sub-options allow to store all information required to do the transportation planning elaboration. Finally, the 'Help' option gives information about the software tool and the access to its user manual.

The three tabs correspond to the three successive steps that allow the truck-drivers to load their truck with the appropriate cement type for each of their deliveries, i.e. the reception of orders from the customers, the split of these orders into deliveries, and the transfer of those deliveries in the weighing system. Indeed, the first tab 'Orders' allows to store the daily orders of each customer and displays the content of the tables **Order** and **Time window** presented in previous chapter. The second tab 'Planning', presented in detail in the following subsection, allows not only to create deliveries of these orders, but also to create ashes recoveries, planned tasks and breaks. Information displayed in this tab is stored in the tables **Planner item** and **Truck-driver / Planner item** presented in previous chapter. The third and last tab 'Export' displays the information of the tab 'Planning' that is required by the weighing system and corresponds to the table **Export** presented in previous chapter.

The date picker allows to select the date of the transportation planning displayed on the tab 'Planning' and therefore the date of the corresponding orders displayed on the tab 'Orders', while the status bar displays on the one hand the name of the database that is currently used and on the other hand, detailed information about a current operation related to the transportation planning elaboration or about any error that can occur.

5.1.2 Tab 'Planning'

The tab 'Planning' is composed of a list of orders, a list of truck-drivers that are not available, a timeline schedule, and of a list of truck-drivers that are available but not allocated to a planner item of the timeline schedule.

List of orders

The list of orders contains the orders related to the current transportation planning, which have been created in the interactive planning system through the tab 'Orders'. Each order, illustrated in Figure 5.2, is characterized by the name of its customer, the color associated to the ordered cement type, and by a red progress bar indicating in percent the remainder of the order, which has not yet been split into deliveries. Note that for more visibility, the name of the ordered cement type can be displayed or not for each order. The selection of this display option, reachable through the popup menu of the planning grid presented later, strongly depends on the number of orders displayed in the list and on the resolution of the used screen.

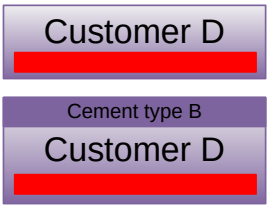

Figure 5.2: Orders detail display

The content of the list of orders can be sorted in ascending or descending order according to the name of the customer, the total quantity to deliver, or the

remaining quantity to deliver. If a minimal remaining quantity of cement that does not have to be delivered to entirely satisfy an order is specified in the settings of the information system, each order disappears from the list of orders when this remaining quantity is achieved. Note that the ordered quantity is also adapted. However, this minimal remainder does not affect orders that have to be delivered with an exact quantity. The ordered quantity of any order displayed in the list can also be adapted by double-clicking on the order. The order disappears from the list and the ordered quantity corresponds then to the quantities of the related deliveries displayed in the timeline schedule. Note that the content of the list of orders is dynamically modified when creating or deleting deliveries. This will be discussed later.

List of absences

The list of absences, illustrated in Figure 5.3, contains the truck-drivers of the cement factory that are not available for the current transportation planning. For example, a truck-driver can be on vacation and another one on sick leave. Each absence is characterized by the name of the truck-driver, the reason of his unavailability, i.e. the absence type, and by the corresponding color. Note that this list can be displayed or not. The selection of this display option, reachable through the popup menu of the planning grid presented later, allows to decrease or increase the size of the list of orders.

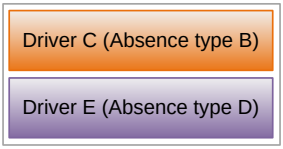

Figure 5.3: Absences display

Timeline schedule

The timeline schedule is composed of a daily timeline whose granularity is set to 5 minutes, resources which correspond to the trucks that can be used for the current transportation planning, and of a planning grid intended to display planner items and whose each cell corresponds to a specific resource and time period. Note that

the cells of the planning grid related to the current time period are displayed in yellow.

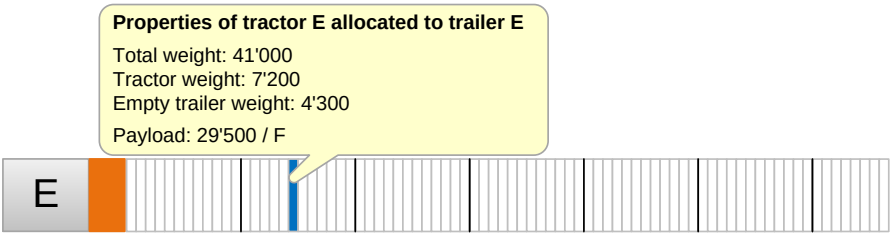

Figure 5.4: Truck properties

Each resource or truck, illustrated in Figure 5.4, is only characterized by a name. However, additional information about a resource can be displayed in a balloon by clicking on a cell related to this resource. This balloon displays the tractor and the trailer that compose the truck, the total weight of the truck when it is full loaded, the weight of the tractor, the weight of the trailer when it is empty, the payload of the tractor calculated from the weights exposed before, and finally information indicating if the truck is equipped with a filter (F) and with an hydraulic jack (K). If the truck is preloaded with a cement type for the day after, a color related to the selected cement type is displayed next to the name of the resource.

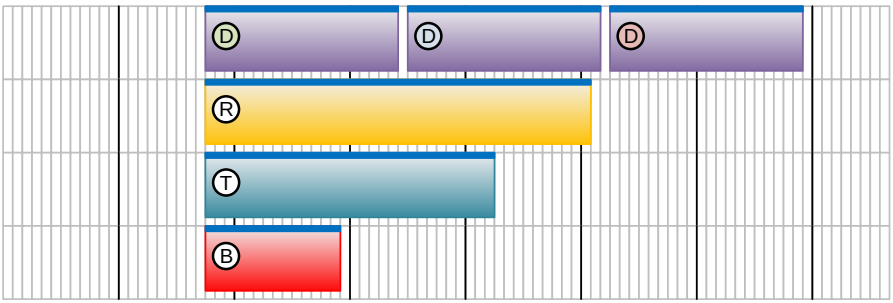

Figure 5.5: Various planner items display

Different types of planner item, illustrated in Figure 5.5, can be created in the planning grid of the timeline schedule. In fact, these planner item types correspond to the four different product types related to the table **Product** presented

in Chapter 4. To distinguish them, a different image is defined for each planner item type. Indeed, the deliveries of cement are characterized by a (D), the ashes recoveries by a (R), the planned tasks by a (T), and the breaks by a (B). Unlike the other planner item types, three different categories of deliveries exists: the deliveries that are preloaded the day before (colored in green), the deliveries that are loaded at the main depot, i.e. the cement factory (colored in blue), and the deliveries that are loaded at one of the local depots located in the railway stations (colored in red). Note that the color of each planner item corresponds to the color related to the corresponding product.

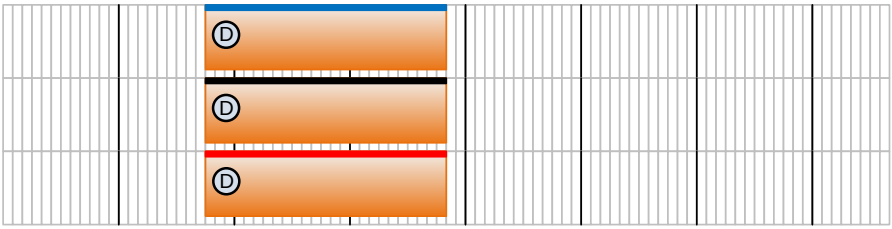

Figure 5.6: Various states of a delivery display

Although a trackbar is displayed for each planner item belonging to the planning grid of the timeline schedule, its utility only concerns planner items related to deliveries. As mentioned before, a delivery can have three different states, which have an influence on the functionalities that can be used with the corresponding planner item. Illustrated in Figure 5.6, the different states of a delivery are therefore represented by different colors of trackbar. Indeed, a blue trackbar indicates that the delivery is not loaded, a black trackbar that the delivery is in loading, and a red trackbar that the delivery is loaded.

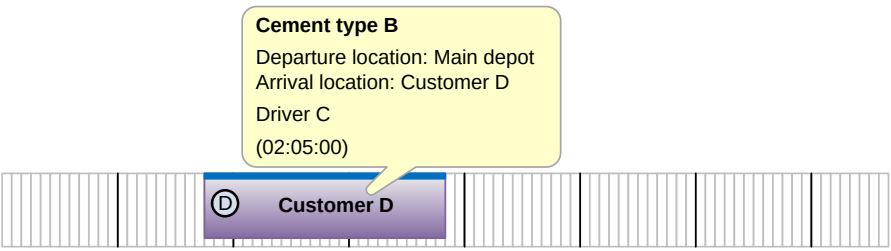

Figure 5.7: Delivery properties

Illustrated in Figure 5.7, a delivery is not only characterized by an image, a trackbar, and the color related to the loaded cement type, but also by a content containing the name of the customer. Additional information about a delivery can be displayed in a balloon by positioning the cursor on the delivery. This balloon displays the name of the loaded cement type, the departure and arrival locations, the drivers that are allocated to the delivery, and finally the duration of this delivery. The departure location corresponds to the main depot, a local depot, or to the residence of a driver who can park his truck home, while the arrival location corresponds to a customer.

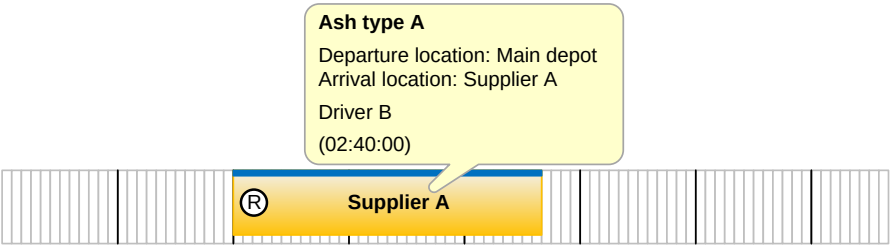

Figure 5.8: Ash recovery properties

Illustrated in Figure 5.8, an ash recovery is not only characterized by an image, a trackbar, and the color related to the loaded ash type, but also by a content containing the name of the supplier. Additional information about an ash recovery can be displayed in a balloon by positioning the cursor on the ash recovery. This balloon displays the name of the loaded ash type, the departure and arrival locations, the drivers that are allocated to the ash recovery, and finally the duration of this ash recovery. The departure location corresponds to the main depot or to a customer, while the arrival location corresponds to a supplier.

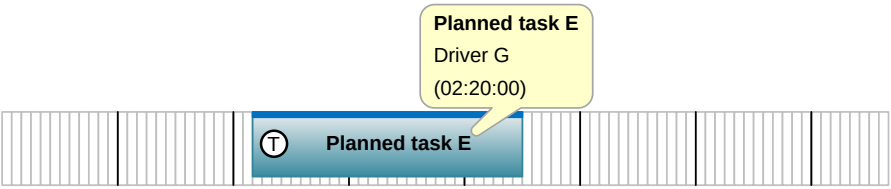

Figure 5.9: Planned task properties

Illustrated in Figure 5.9, a planned task is not only characterized by an image, a trackbar, and the color related to the task type, but also by a content containing the name of this task type. Additional information about a planned task can be displayed in a balloon by positioning the cursor on the planned task. This balloon displays the name of the task type, the drivers that are allocated to the planned task, and finally the duration of this planned task.

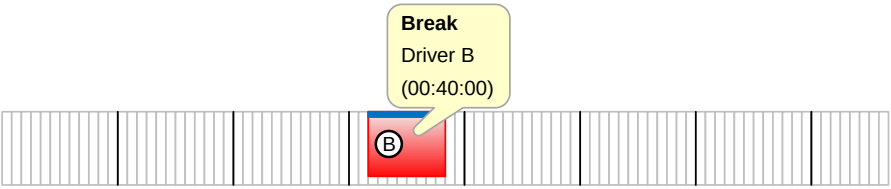

Figure 5.10: Break properties

Illustrated in Figure 5.10, a break is only characterized by an image, a trackbar, and the color related to this type of planner item. Additional information about a break can be displayed in a balloon by positioning the cursor on the break. This balloon displays the type of the planner item, the drivers that are allocated to the break, and finally the duration of this break.

Display options reachable through the popup menu of the planning grid can be selected to improve the readability of the transportation planning. For Figures 5.11 to 5.15 whose content is discussed hereafter, the inactive view, i.e. when the display option is not selected, is presented at the top, while the active view, i.e. when the display option is selected, is presented at the bottom.

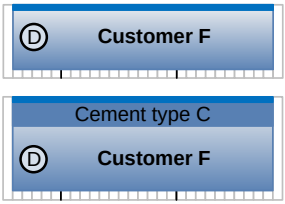

Figure 5.11: Planner items product display

Although the name of the cement type, ash type, task type, or break is available in the balloon related to a planner item, this information can be directly displayed or not in a caption for each planner item as illustrated for a delivery in Figure 5.11.

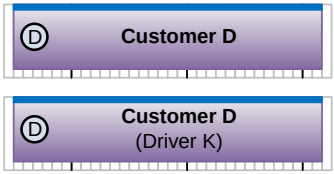

Figure 5.12: Planner items drivers display

The selection of this display option allows to decrease or increase the number of resources of the timeline schedule displayed on the screen. The truck-drivers allocated to a planner item can also be directly displayed in the content of each planner item as illustrated in Figure 5.12.

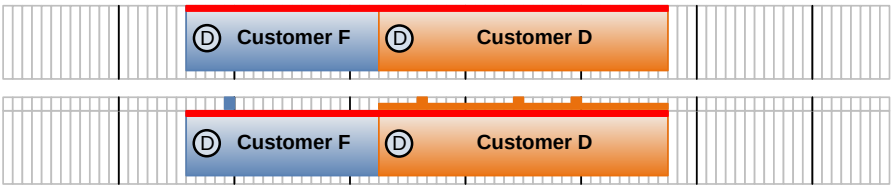

Figure 5.13: Reality display

Thanks to its communication with the weighing system, the interactive planning system knows when a delivery is loaded at a dock of the cement factory. This time corresponds to the end of the loading time and therefore to the start of the travel time related to the delivery. This information can be displayed in the timeline schedule by adding for each existing resource an additional resource intended to display the 'reality'. This explains why the table **Tractor** presented in previous chapter has as attribute a main identifier. Indeed, the planned deliveries and their 'reality' concern a same resource, i.e. a same main tractor. As illustrated in Figure 5.13, two variants are available to display the 'reality'. The first one (cf. delivery of customer F) only presents the end of the loading time, while the second one (cf. delivery of customer D) also presents the estimated begin and end of the unloading time, and the estimated begin and end of the delivery, thanks to the use of the appropriate durations stored in the information system. The selection of this display option and its variants allows to adapt the current transportation planning during the day in order to reflect the reality.

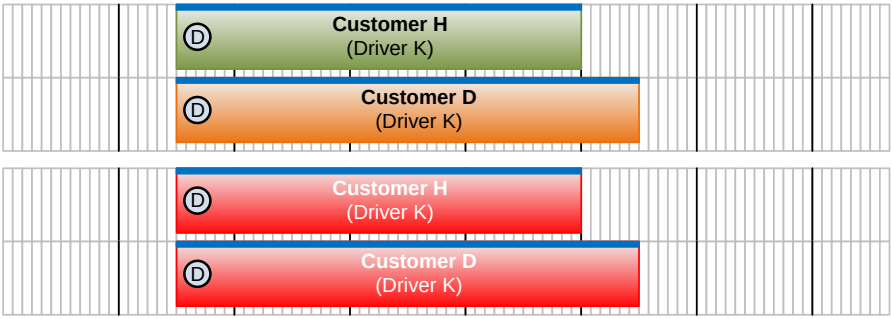

Figure 5.14: Conflicts display

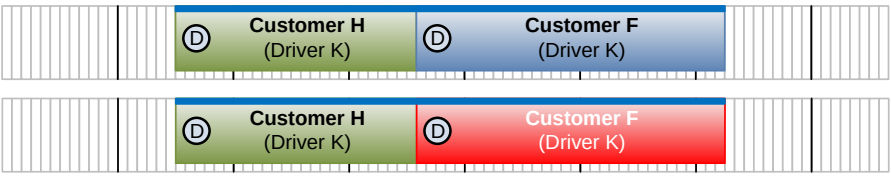

Figure 5.15: Lack of breaks display

Two types of problem can occur for truck-drivers: 'conflicts', when truck-drivers are allocated simultaneously to many planner items, and 'lack of breaks', when truck-drivers don't have enough or well placed breaks to respect the union rules. To detect conflicts and lack of breaks, the interactive planning system can highlight planner items concerned by such situations in red as illustrated in figures 5.14 and 5.15. The selection of this display option allows the dispatcher to easily detect conflicts and lack of breaks in order to apply the appropriate modifications. Note that the reason of each type of problem and the solution to adopt is displayed in the tag related to the planner item concerned by this type of problem.

Unallocated truck-drivers

The unallocated truck-drivers are displayed in a text field, illustrated in Figure 5.16, and correspond to truck-drivers that are available for the current transportation planning but that are not allocated to a planner item of the timeline schedule. Note that the content of the text field is modified dynamically when creating or deleting a planner item if a default truck-driver is allocated to the corresponding

truck, or during the selection of truck-drivers. Note that this list can be displayed or not. The selection of this display option, reachable through the popup menu of the planning grid, allows to decrease or increase the size of the timeline schedule.

Driver A, Driver F, Driver H, Driver I, Driver J, Driver K, Driver M

Figure 5.16: Unallocated truck-drivers display

5.2 Functionalities description

All the functionalities of the interactive planning system are illustrated in Figure 5.17. Split into four groups, they concern the handling of the settings and absences, the elaboration of any transportation planning, and the handling of miscellaneous other tasks. For the same reason as for the presentation of the graphical user interface, not all these functionalities are described in this section, but only the core functionalities concerning the elaboration of the transportation planning. These ones are colored in red in the figure. The functionalities related to the trucks, i.e. the resources of the timeline schedule to which the planner items are allocated, are described in Subsection 5.2.1, while those related to the deliveries, ashes recoveries, planned tasks and breaks are respectively described in Subsections 5.2.2, 5.2.3, 5.2.4 and 5.2.5. The functionalities related to the whole transportation planning are finally described in Subsection 5.2.6. Additional information about the functionalities that are not mentioned in this section can however be found in [56], the user manual of the interactive planning system.

5.2.1 Functionalities related to trucks

Truck moving

By default, the layout of the trucks or resources of the timeline schedule related to a transportation planning corresponds to the layout related to the planning that precedes this transportation planning. Although it is possible to modify this layout or to create a new layout in the settings of the interactive planning system, a functionality has however been implemented in order to move a truck up or down directly in the timeline schedule.

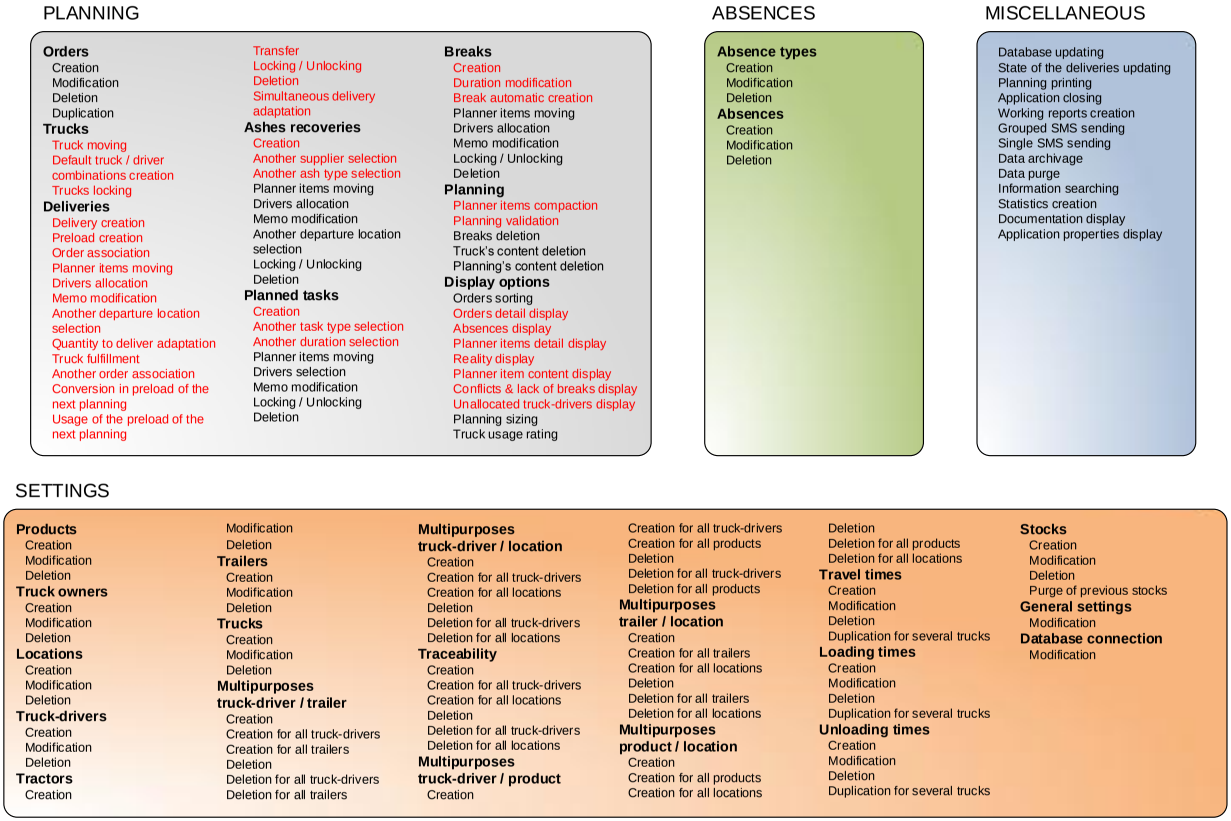

Figure 5.17: Functionalities

Three parameters are required to move a truck: a *date*, a *resource*, and a *direction*. It is provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Move the truck' and then the sub-option 'Up' or 'Down' in the popup menu of this planning grid, as illustrated in Figure 5.18.

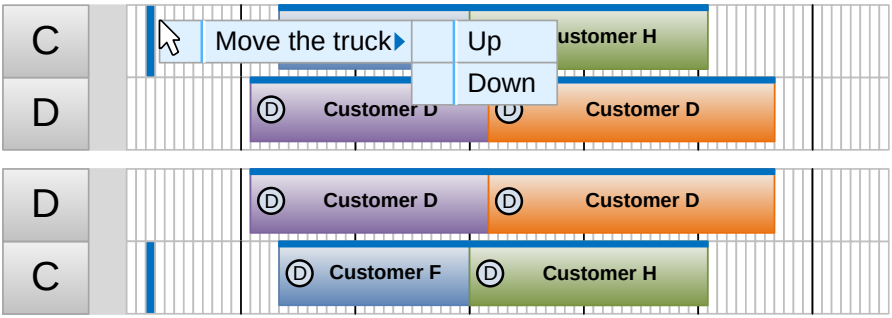

Figure 5.18: Truck moving

The interactive planning system moves the truck related to the current transportation planning and all its allocated planner items in the selected direction. Note that the sub-option 'Up' has no effect if the truck is the first resource of the transportation planning. Similarly, the sub-option 'Down' has no effect if the truck is the last resource of the transportation planning.

Default truck / driver combinations creation

Although it is possible to allocate a default truck-driver to each truck in the settings of the interactive planning system, a functionality has however been implemented in order to allocate or deallocate a default truck-driver to each truck directly in the timeline schedule.

Two parameters are required to allocate or deallocate a default truck-driver to each truck: a *set of trucks*, and a *set of truck-drivers*. They are provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Create the combinations default truck / driver…' in the popup menu of this planning grid, as illustrated in Figure 5.19. This option opens a window allowing the selection of a truck and a driver to

create a combination, and the selection of a combination to delete it in order to give back the corresponding truck and driver.

The interactive planning system applies the modifications.

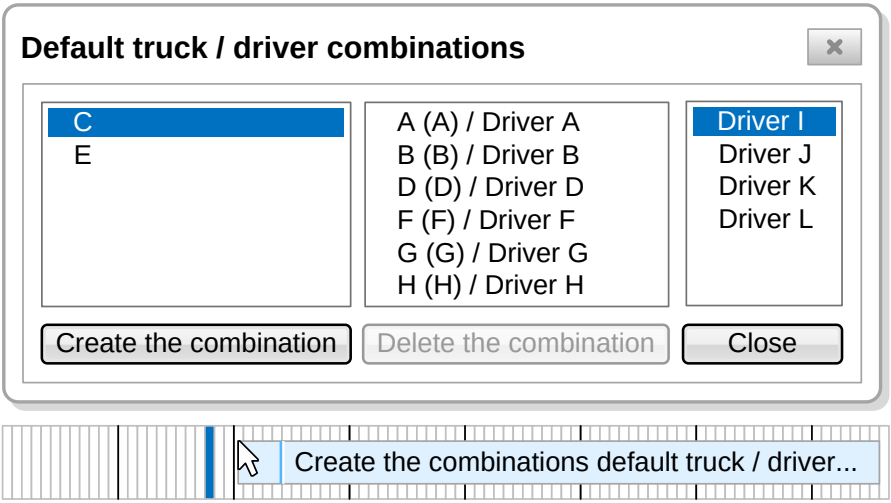

Figure 5.19: Default truck / driver combinations creation

Trucks locking

Although it is possible to lock or unlock trucks in the settings of the interactive planning system, a functionality has however been implemented in order to lock or unlock trucks directly in the timeline schedule.

Only one parameter is required to lock or unlock trucks: a *set of trucks*. It is provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Lock the trucks...' in the popup menu of this planning grid, as illustrated in Figure 5.20. This option opens a window allowing the check of trucks having to be locked.

The interactive planning system applies the modifications.

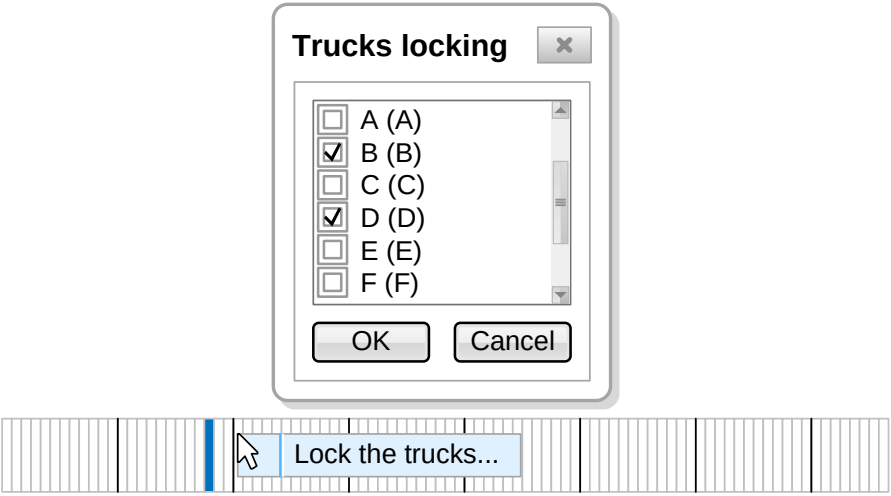

Figure 5.20: Trucks locking

5.2.2 Functionalities related to deliveries

Delivery creation

Three parameters are required to create a delivery: a *start date and time*[1], a *resource*, and an *order*. They are provided through the graphical user interface by dragging and dropping an order of the list of orders on a cell of the planning grid belonging to the timeline schedule, as illustrated in Figure 5.21. The resource related to the selected cell corresponds to the truck used to do the delivery, while the start date and time of the time period of this cell defines the start date and time of the delivery.

Before creating the delivery, the interactive planning system checks that the selected truck is compatible with the customer of the order. In this case, the total duration of the delivery is computed from the aggregation of the appropriate loading time, travel time, unloading time and back travel time, and the delivery is created. Otherwise, the delivery is not created. Note that the departure location of a delivery created with this functionality corresponds to the main depot, i.e. the cement factory. Furthermore, the delivered quantity corresponds to the payload

[1] This parameter stores simultaneously a date and a time. It can be considered as a timestamp.

of the truck. If a default truck-driver is specified for the truck, he is automatically allocated to the delivery, but only if he is compatible with the truck and with the customer and the cement type of the order. Following the creation of the delivery, the red progress bar indicating in percent the remainder of the order is updated.

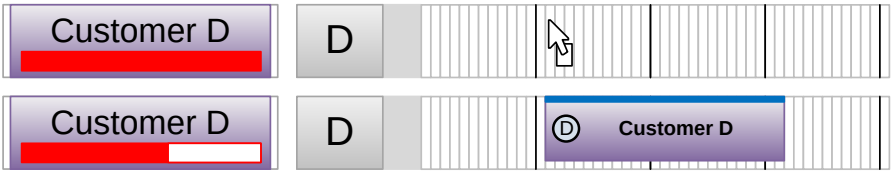

Figure 5.21: Delivery creation

Preload creation

Unlike 'normal' deliveries, 'preloaded' deliveries are not directly allocated to an order following their creation, but only to a cement type. This especially allows to define, modify or delete later the order allocated to these deliveries without having to delete and recreate the corresponding planner items. Note that the interactive planning system does not allow to delete a loaded delivery. With 'preloaded' deliveries, it is however possible to adapt the allocated order even if they are loaded.

Three parameters are required to create a preload: a *start date and time*, a *resource*, and a *cement type*. They are provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Create a preload...' in the popup menu of this planning grid, as illustrated in Figure 5.22. This option opens a window allowing the selection of a cement type. The resource related to the selected cell corresponds to the truck used to do the 'preloaded' delivery, while the start date and time of the time period of this cell defines the start date and time of the 'preloaded' delivery.

Before creating the preload, the interactive planning system checks that no preload has already been created for the truck in the transportation planning. In this case, the default duration of a preload, defined in the settings, is used, and the preload is created. Otherwise, the preload is not created. Note that a preload has neither a departure location, nor an arrival location. If a default truck-driver is specified for the truck, he is automatically allocated to the preload, but only if he is compatible with the truck and with the cement type.

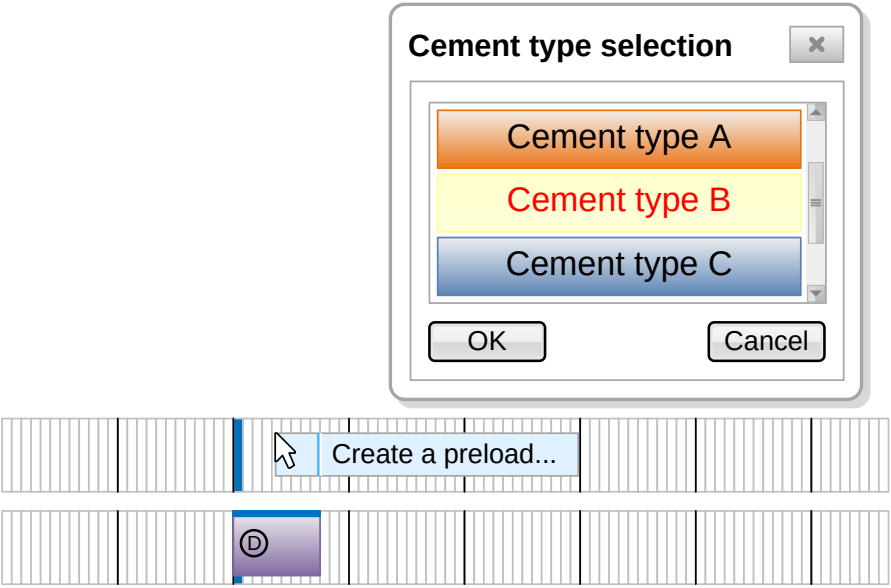

Figure 5.22: Preload creation

Order association

Three parameters are required to associate an order to a preload: the *start date and time* and the *resource* related to a preload, and an *order*. They are provided through the graphical user interface by dragging and dropping an order of the list of orders on a preload of the timeline schedule, as illustrated in Figure 5.23.

Figure 5.23: Order association

Before associating the order to the preload, the interactive planning system checks that the truck and the truck-drivers allocated to the preload are compatible with the customer of the order. If they are, the default duration of the preload is replaced by a new duration related to the order and computed from the aggregation of the appropriate travel time, unloading time and back travel time, and the order is associated to the preload. Otherwise, the order is not associated to the preload. Note that the departure location of a 'preloaded' delivery created with this functionality corresponds to the main depot, i.e. the cement factory. Following the association of the order to the preload, the red progress bar indicating in percent the remainder of the order is updated.

Planner items moving

Although it is possible to directly move a selected delivery of the timeline schedule with the cursor, the interactive planning system however limits this moving to ensure the integrity of its data. Indeed, the information system forbids on the one hand to move any planner item from a truck to another truck and on the other hand, to move any planner item beyond the start time of a next planner item and beyond the end time of a previous planner item. Furthermore, this moving restriction also depends on the state of the selected planner item, if this one is a delivery, and on the frozen horizon, which will be presented in detail in Section 6.2 of Chapter 6. To avoid to move each planner item separately, a functionality has however been implemented to allow the moving of a set of consecutive planner items related to a truck.

Five parameters are required to move a set of planner items: the *start date and time* and the *resource* related to a planner item, a *direction*, a *duration*, and an *option* indicating if this duration has to be maximal or not. They are provided through the graphical user interface by selecting the option 'Move the planner items...' in the popup menu of the first planner item of the set, as illustrated in Figure 5.24. This option opens a window allowing the selection of a direction and of a duration. If 'maximum' is selected for the duration, the set of planner items is moved as far as possible in the planning grid belonging to the timeline schedule.

Before moving the set of planner items, the interactive planning system checks that this moving is authorized. In this case, the planner items of the set are moved. Otherwise, they are not moved.

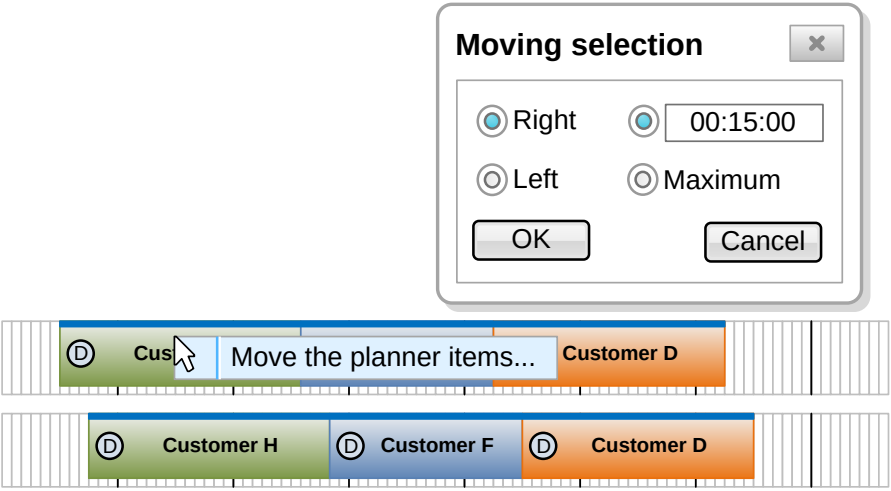

Figure 5.24: Planner items moving

Drivers allocation

Three parameters are required to manually allocate drivers to a planner item or to a set of planner items: the *start date and time* and the *resource* related to a planner item or to the planner items of a set, and *drivers*. They are provided through the graphical user interface by selecting the option 'Allocate drivers...' either directly in the popup menu of a planner item, or by selecting first a set of cells related to a set of planner items and by selecting then the same option in the popup menu of the planning grid belonging to the timeline schedule, as illustrated in Figure 5.25. This option opens a window allowing the selection of drivers, which presents in a first list the drivers already allocated to the planner item or to the set of planner items, and in a second list the available drivers that are not yet allocated to the planner item or to the set of planner items. Note that only drivers that are compatible with the planner item or with the set of planner items are displayed in this second list. Each driver can be moved from a list to another list thanks to provided buttons.

Before allocating drivers to a planner item, the interactive planning system checks that the selected drivers are compatible with the planner item. If they are, the drivers are allocated to the planner item and are displayed in its content if this

display option has been selected before. Otherwise, they are not allocated to the planner item.

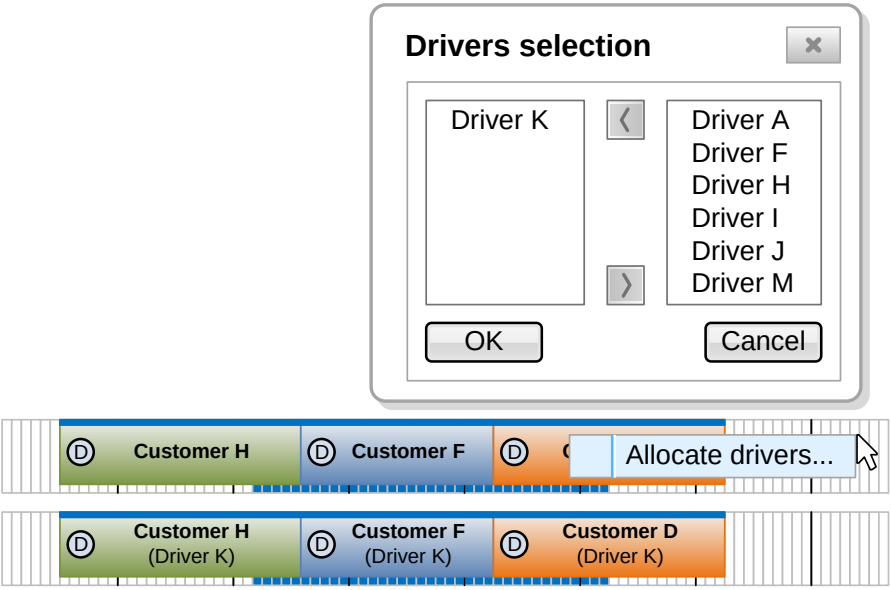

Figure 5.25: Drivers allocation

Memo modification

When the default data of a planner item are not sufficient, additional information related to this planner item can be added by the dispatcher. Called 'memo', this additional information is usually mentioned for the first deliveries of the transportation planning and is reported on the working reports of concerning truck-drivers when these ones are created. To allow the addition, modification and deletion of memos, a dedicated functionality has been implemented.

Three parameters are required to modify a memo: the *start date and time* and the *resource* related to a planner item, and a *memo*. They are provided through the graphical user interface by selecting the option 'Modify the memo…' in the popup menu of a planner item, as illustrated in Figure 5.26. This option opens a window allowing the modification of the existing memo, which can be empty if no

memo has been written before for the planner item.

If the memo is not empty after the modification, a 'post-it' is added to the image related to the planner item. Furthermore, the memo is displayed in the balloon of the planner item after its duration.

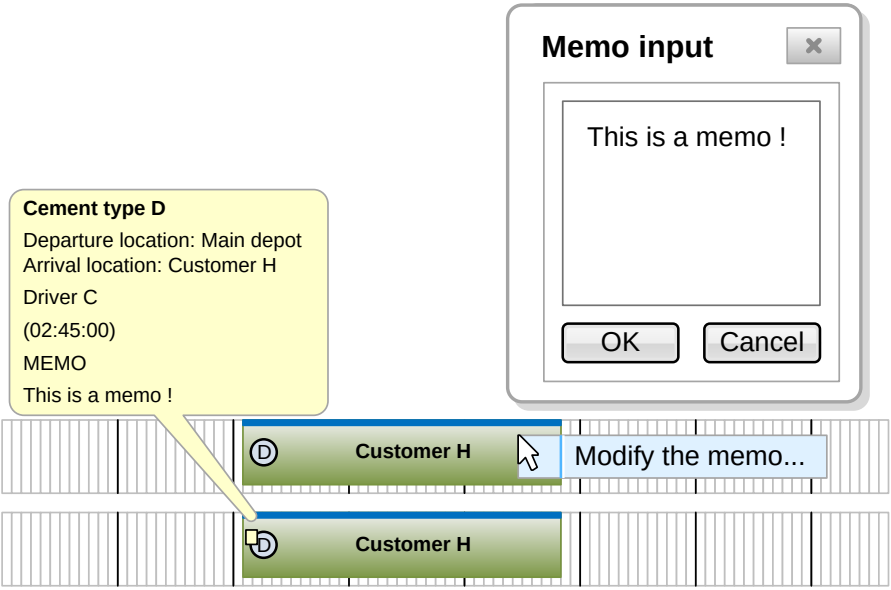

Figure 5.26: Memo modification

Another departure location selection

As mentioned before, the default departure location of a created delivery corresponds to the main depot, i.e. the cement factory. However, this departure location can be modified thanks to a dedicated functionality in order to reduce the total duration of the delivery. Considering a 'preloaded' delivery that is associated to an order, the default departure location can be changed by the residence of a truck-driver who can park his truck home for this first delivery. Considering an 'normal' delivery that does not correspond to the first delivery of a truck, the default departure location can be changed by one of the local depots located in the railway stations.

Three parameters are required to select another departure location of a delivery: the *start date and time* and the *resource* related to a delivery, and a *departure location*. They are provided through the graphical user interface by selecting the option 'Select another departure location...' either directly in the popup menu of a delivery, as illustrated in Figure 5.27, or by selecting first a set of cells related to a set of deliveries and by selecting then the same option in the popup menu of the planning grid belonging to the timeline schedule. This option opens a window allowing first the selection of a location type that filters a list of available departure locations, and then the selection of a new departure location of this list. Note that only the location types 'Factory', 'Depot' and 'Truck-driver' can be selected.

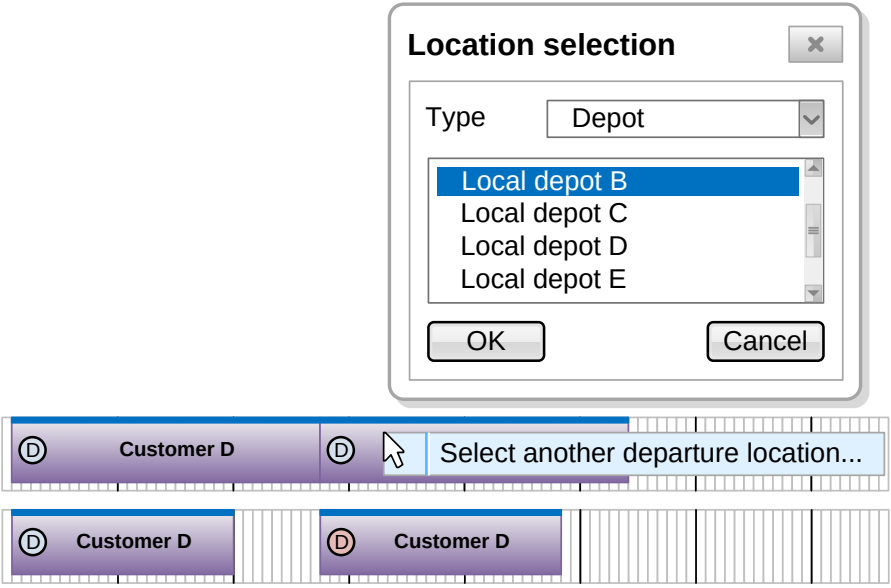

Figure 5.27: Another departure location selection

Before selecting another departure location for a delivery, the interactive planning system checks that this new departure location is compatible not only with the delivery, but also with the possible delivery or ash recovery that precedes this delivery in order to be in accordance with the preceding rules, which will be presented in detail in Section 6.1 of Chapter 6. In this case, the actual departure location of the delivery is replaced by the new departure location, and the total duration of the delivery is recomputed with a new loading time and with a new travel

time. The total duration of the possible delivery or ash recovery that precedes the delivery is also recomputed with a new back travel time.

Quantity to deliver adaptation

As mentioned before, the quantity of cement that is loaded for a delivery corresponds by default to the payload of the truck used to do this delivery. Due to this rule, it frequently happens that the sum of the quantities of cement related to the deliveries of an order is greater than the ordered quantity defined when creating this order. Although this is not a problem because the customers are able to stock additional and not ordered quantities of cement types for future uses, some orders however require to be delivered with an exact quantity of cement. For this purpose, the quantity of cement that is loaded for such deliveries must be adaptable in order to exactly satisfy the ordered quantity.

Two parameters are required to adapt the quantity of cement related to a delivery: the *start date and time* and the *resource* related to a delivery. It is provided through the graphical user interface by selecting the option 'Adapt the quantity to deliver' in the popup menu of this delivery, as illustrated in Figure 5.28.

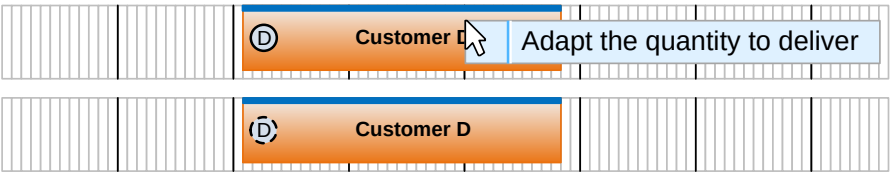

Figure 5.28: Quantity to deliver adaptation

Before adapting the quantity of cement that has to be delivered, the interactive planning system sums the quantities of cement of all existing deliveries related to the order of the selected delivery, and computes the difference between the obtained value and the ordered quantity specified when creating the order. If this difference is positive, this one is removed from the payload of the truck. Note that a new image is displayed for the delivery, which indicates that the truck is no more fulfilled.

Truck fulfillment

Two parameters are required to fulfill the truck related to a delivery: the *start date and time* and the *resource* related to a delivery. It is provided through the graphical user interface by selecting the option 'Fulfill the truck' in the popup menu of this delivery, as illustrated in Figure 5.29.

Figure 5.29: Truck fulfillment

The interactive planning system replaces the actual quantity of cement of the delivery with the payload of the truck related to this delivery. Note that a new image is displayed for the delivery, which indicates that the truck is fulfilled.

Another order association

Three parameters are required to associate another order to a preload: the *start date and time* and the *resource* related to a 'preloaded' delivery, and an *order*. They are provided through the graphical user interface by selecting the option 'Associate to another order...' in the popup menu of this 'preloaded' delivery, as illustrated in Figure 5.30. This option opens a window allowing the selection of another order of the list of orders having the same cement type.

Before associating another order to the preload, the interactive planning system checks that the truck, the cement type and the allocated truck-drivers of the preload are compatible with the customer of the new order. If they are, the actual duration of the 'preloaded' delivery is replaced by a new duration related to the new order and computed from the aggregation of the appropriate travel time, unloading time and back travel time, and the order is associated to the preload. Otherwise, the new order is not associated to the preload, which remains therefore allocated to the previous order. Following the association of another order to the preload, the red progress bar indicating in percent the remainder of the order is updated.

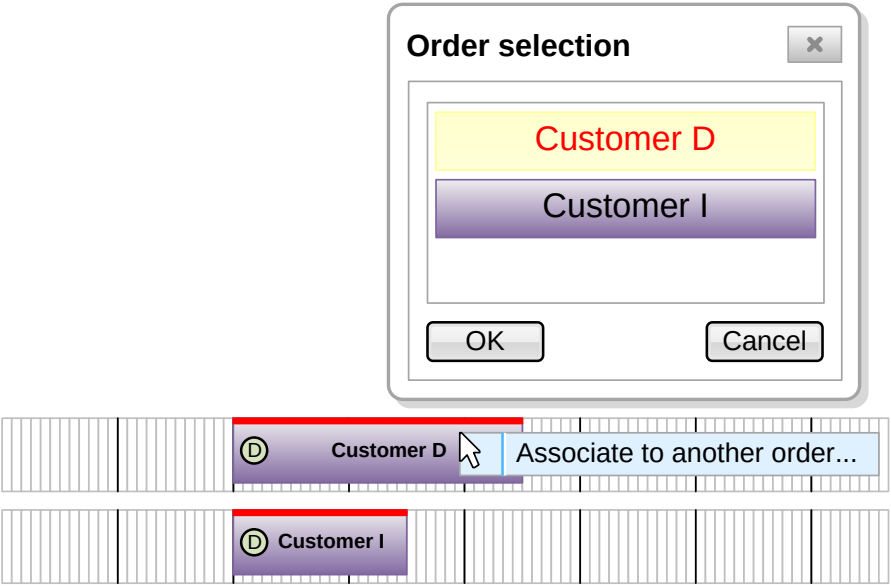

Figure 5.30: Another order association

Conversion in preload of the next planning

It sometimes happens that the last delivery of a truck is not delivered even if it is loaded. Indeed, the truck can have a problem on the way that prevents it to reach the customer, or the order can simply be cancelled at last minute. In such cases, the truck is not unloaded but keeps the cement type for the first delivery of the next planning. The truck is therefore preloaded. Due to the impossibility to delete an order that is loaded, a functionality is required to convert an order in a preload related to the next planning.

Only one parameter is required to convert a delivery in a preload related to the next planning: the *main identifier* of a delivery. It is provided through the graphical user interface by selecting the option 'Convert in preload of the next planning' in the popup menu of this delivery, as illustrated in Figure 5.31.

Before converting the delivery in a preload related to the next planning, the interactive planning system checks that the delivery is 'normal', loaded and not followed by other deliveries. Furthermore, the system checks that no planner items

are allocated to the truck related to this delivery for the next planning. If all the constraints enumerated before are respected, the delivery is deleted from the current planning and a preload is created for the next planning. This preload, which is loaded, starts at the same time and has the same cement type as the deleted delivery. Following the conversion of the delivery in a preload, the red progress bar indicating in percent the remainder of the order is updated. Furthermore, a color indicating the preloaded cement type for the next planning is displayed next to the name of the truck.

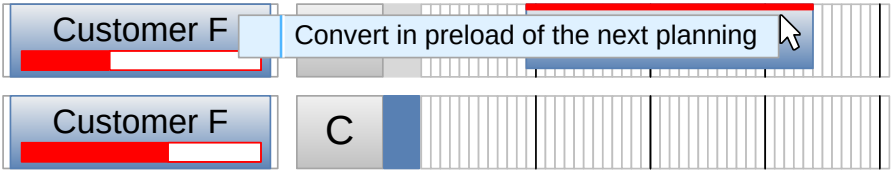

Figure 5.31: Conversion in preload of the next planning

Usage of the preload of the next planning

Although a truck is still preloaded for the next planning, it sometimes happens that a delivery of last minute concerning a same cement type has to be done for the current day. In such cases, the truck is not unloaded but allocates its preloaded cement type to this delivery. Due to the impossibility to delete a loaded preload, a functionality is required to use the preload related to the next planning for a delivery of last minute.

Only one parameter is required to use the preload of the next planning for a delivery: the *main identifier* of a delivery. It is provided through the graphical user interface by selecting the option 'Use the preload of the next planning' in the popup menu of this delivery, as illustrated in Figure 5.32.

Before using the preload related to the next planning for a delivery, the interactive planning system checks that the cement type of the preload is compatible with the delivery, and that this preload is loaded and not associated to an order. If all the constraints enumerated before are respected, the preload is deleted from the next planning and the delivery is stated as loaded. Although the red progress bar indicating in percent the remainder of the order is unchanged following the usage of the preload of the next planning, the color indicating the preloaded cement type for the next planning is deleted.

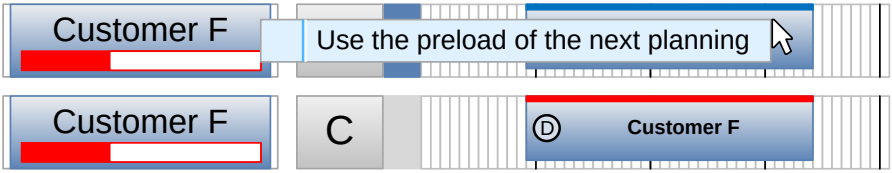

Figure 5.32: Usage of the preload of the next planning

Transfer

Two parameters are required to transfer a preload to the next or previous planning: the *main identifier* of a 'preloaded' delivery, and *one of the two available sub-options*. It is provided through the graphical user interface by first selecting the option 'Transfer' and then the sub-option 'To the next planning' or 'To the previous planning' in the popup menu of this delivery, as illustrated in Figure 5.33.

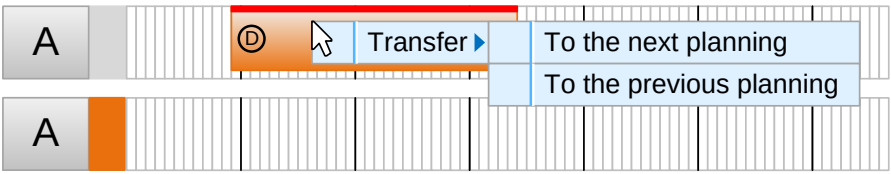

Figure 5.33: Transfer

Before transferring the delivery to the next or previous planning, the interactive planning system checks that the delivery is 'preloaded', loaded, not allocated to an order, and not followed by other deliveries. Furthermore, the system checks that no planner items are allocated to the truck related to this delivery for the next or previous planning. If all the constraints enumerated before are respected, the preload is deleted from the current planning and a same preload is created for the next or previous planning. Following the transfer of the delivery to the next planning, a color indicating the preloaded cement type for the next planning is displayed next to the name of the truck.

Locking / Unlocking

Two parameters are required to lock or unlock a planner item: the *start date and time* and the *resource* related to a planner item. It is provided through the graphical user interface by selecting the option 'Lock' and by checking or unchecking this option in the popup menu of this planner item, as illustrated in Figure 5.34.

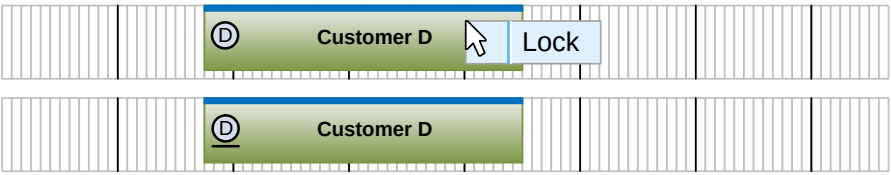

Figure 5.34: Locking / Unlocking

If the option is checked, the interactive planning system locks the planner item. Otherwise, the interactive planning system unlocks the planner item. A locked planner item cannot be moved neither directly with the cursor, nor with the functionality 'Planner items moving' presented before. Note that the image of a planner item is underlined when this one is locked.

Deletion

Two parameters are required to delete a delivery: the *start date and time* and the *resource* related to a delivery. It is provided through the graphical user interface by selecting the option 'Delete' in the popup menu of this delivery, as illustrated in Figure 5.35.

Figure 5.35: Delivery deletion

Before deleting the delivery, the interactive planning system checks that the deletion is authorized. In this case, the delivery is deleted. Otherwise, the deliv-

ery remains created. As mentioned before, a loaded delivery cannot be deleted and only the order associated to a 'preloaded' loaded delivery can be deleted. Following the deletion of the delivery, the red progress bar indicating in percent the remainder of the order is updated.

Simultaneous deliveries adaptation

It sometimes happens that deliveries related to a same customer are delivered at the same time, i.e. simultaneously, by different trucks. To avoid such situations, a functionality has been implemented to adapt the start time of concerned deliveries in such a way that their periods of cement unloading do not overlap.

Only one parameter is required to adapt the simultaneous deliveries of a transportation planning: a *date*. It is provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Adapt the simultaneous deliveries' in the popup menu of this planning grid, as illustrated in Figure 5.36.

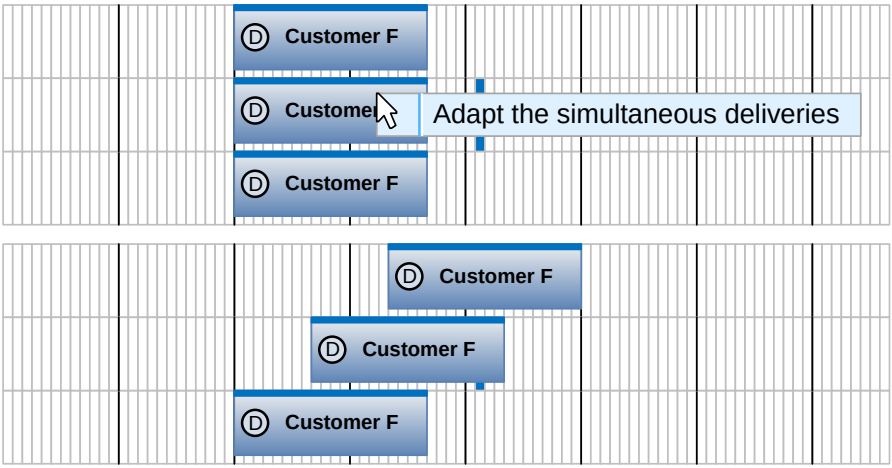

Figure 5.36: Simultaneous deliveries adaptation

Before adapting the simultaneous deliveries of the transportation planning, the interactive planning system checks that this adaptation is authorized. In this case, the start time of concerned deliveries are adapted, as the start time of fol-

lowing planner items if they are overlapped by these deliveries. Otherwise, the simultaneous deliveries of the transportation planning are not adapted.

5.2.3 Functionalities related to ashes recoveries

Ash recovery creation

Four parameters are required to create an ash recovery: a *start date and time*, a *resource*, the *location* of a supplier, and an *ash type*. They are provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Create an ash recovery...' in the popup menu of this planning grid, as illustrated in Figure 5.37. This option opens a window allowing the selection of a supplier and of an ash type. The resource related to the selected cell corresponds to the truck used to do the ash recovery, while the start date and time of the time period of this cell defines the start date and time of the ash recovery.

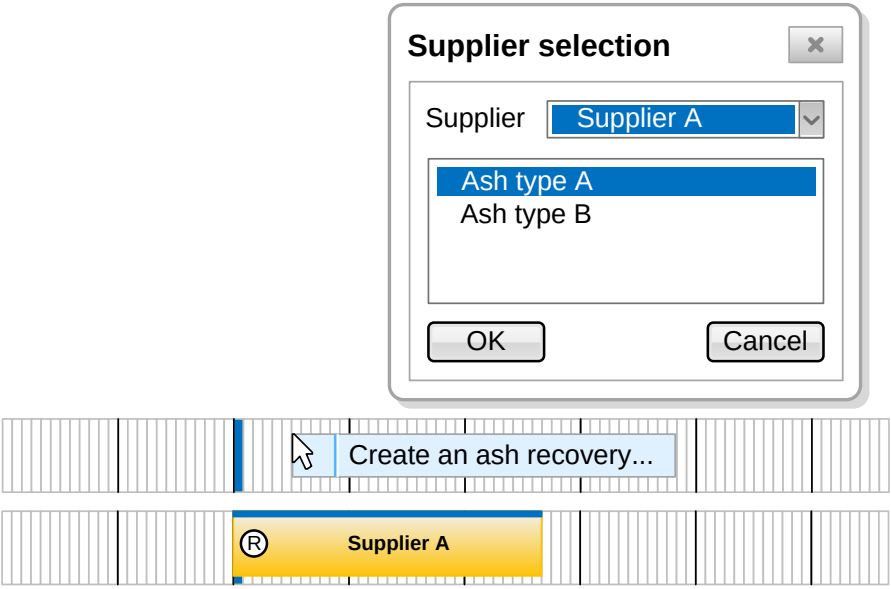

Figure 5.37: Ash recovery creation

Before creating the ash recovery, the interactive planning system checks that the selected truck is compatible with the supplier of the ash recovery. In this case, the total duration of the ash recovery is computed from the aggregation of the appropriate travel time, loading time, back travel time and unloading time, and the ash recovery is created. Otherwise, the ash recovery is not created. Note that the departure location of an ash recovery created with this functionality corresponds to the main depot, i.e. the cement factory. If a default truck-driver is specified for the truck, he is automatically allocated to the ash recovery, but only if he is compatible with the truck and with the supplier and the ash type of the ash recovery.

Another supplier and/or ash type selection

Four parameters are required to select another supplier and/or ash type of an ash recovery: the *start date and time* and the *resource* related to an ash recovery, the *location* of a supplier, and an *ash type*. It is provided through the graphical user interface by selecting the option 'Select another supplier and/or ash type…' in the popup menu of this ash recovery, as illustrated in Figure 5.38. This option opens the same window as for the creation of an ash recovery, which allows the selection of another supplier and the selection of another ash type.

Before selecting another supplier and/or ash type for the ash recovery, the interactive planning system checks that the new supplier and/or ash type is/are compatible with the truck and with the allocated truck-drivers of the ash recovery. In this case, the duration of the ash recovery is replaced by a new duration related to the new supplier and/or ash type, and is computed from the aggregation of the appropriate travel time, loading time, back travel time and unloading time, and the supplier and/or ash type is/are selected for the ash recovery. Otherwise, the new supplier and/or ash type is/are not selected for the ash recovery. Following the selection of another supplier and/or ash type, the content and/or color of the ash recovery is/are updated.

Other functionalities

As for the deliveries, it is possible to directly move a selected ash recovery and to move a selected ash recovery with the set of planner items that follows it. Furthermore, it is also possible to allocate drivers to an ash recovery, to modify the memo of an ash recovery, to select another departure location for an ash recovery, to lock/unlock an ash recovery, and to delete an ash recovery. Note that only the

location types 'Factory' and 'Customer' can be selected when selecting another departure location.

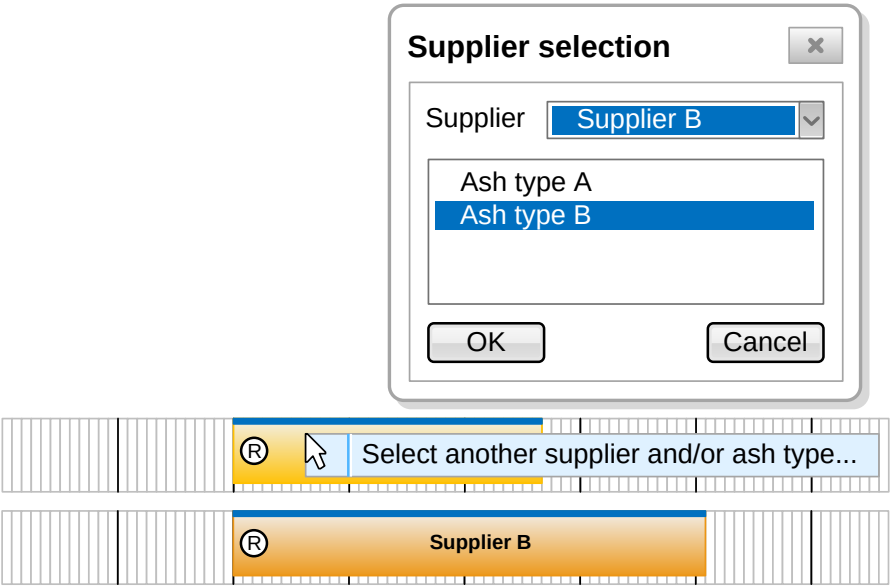

Figure 5.38: Another supplier and/or ash type selection

5.2.4 Functionalities related to planned tasks

Planned task creation

Four parameters are required to create a planned task: a *start date and time*, a *resource*, a *duration*, and a *task type*. They are provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Create a planned task...' in the popup menu of this planning grid, as illustrated in Figure 5.39. This option opens a window allowing the input of a duration and the selection of a task type. For example, a truck can be at the garage or at a washing station for a few hours. The resource related to the selected cell corresponds to the truck concerned by the planned task, while the start date and time of the time period of this cell defines the start date and time of the planned task.

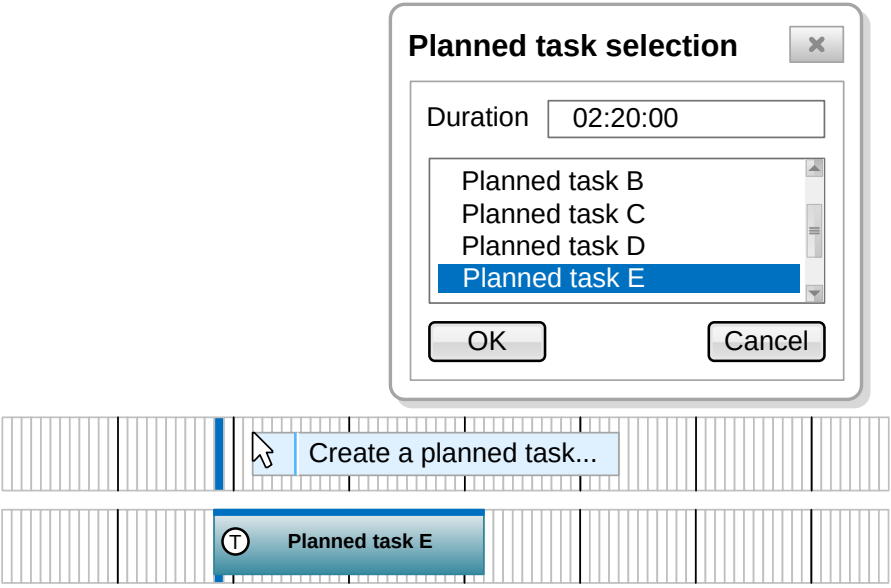

Figure 5.39: Planned task creation

Before creating the planned task, the interactive planning system checks that this creation is authorized. In this case, the planned task is created. Otherwise, it is not created. If a default truck-driver is specified for the truck, he is automatically allocated to the planned task.

Another task type and/or duration selection

Four parameters are required to select another task type and/or duration of a planned task: the *start date and time* and the *resource* related to a planned task, a *duration*, and a *task type*. It is provided through the graphical user interface by selecting the option 'Select another task type and/or duration...' in the popup menu of this planned task, as illustrated in Figure 5.40. This option opens the same window as for the creation of a planned task, which allows the selection of another task type and the input of another duration.

Before modifying the duration of the planned task, the interactive planning system checks that this modification is authorized. In this case, the planned task

is modified. Otherwise, it is not modified. Following the selection of another task type and/or duration, the content, color and/or duration of the planned task are updated.

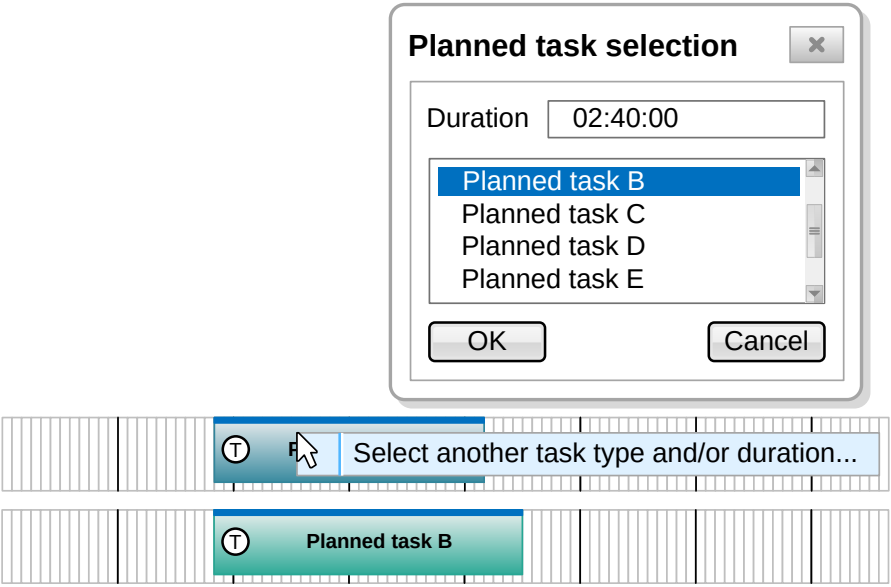

Figure 5.40: Another task type and/or duration selection

Other functionalities

As for the deliveries, it is possible to directly move a selected planned task or to move a selected planned task with the set of planner items that follows it. Furthermore, it is also possible to allocate drivers to a planned task, to modify the memo of a planned task, to lock/unlock a planned task, and to delete a planned task.

5.2.5 Functionalities related to breaks

Break creation

Three parameters are required to create a break: a *start date and time*, a *resource*, and a *duration*. They are provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Create a break…' in the popup menu of this planning grid, as illustrated in Figure 5.41. This option opens a window allowing the input of a duration. The resource related to the selected cell corresponds to the truck concerned by the break, while the start date and time of the time period of this cell defines the start date and time of the break.

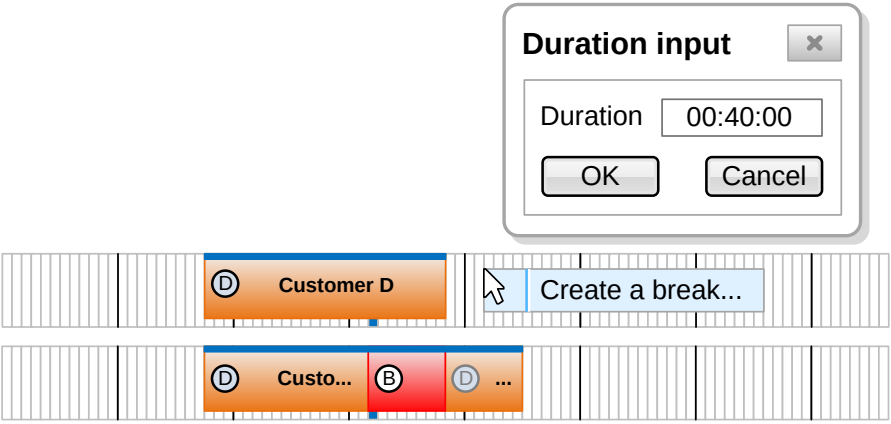

Figure 5.41: Break creation

Before creating the break, the interactive planning system checks that this creation is authorized. In this case, the break is created. Otherwise, it is not created. If a default truck-driver is specified for the truck, he is automatically allocated to the break. Note that a break can be created inside a delivery or ash recovery in order to satisfy the union rules. In such cases, the planner item related to the delivery or ash recovery is split in two planner items. This explains why the table **Planner item** presented in previous chapter has as attribute a main identifier. Indeed, the planner items related to a same delivery or ash recovery concern a same main planner item, which corresponds to the first planner item related to this delivery or ash recovery. The image of a planner item that refers to another main planner item is shaded.

Duration modification

Three parameters are required to modify the duration of a break: the *start date and time* and the *resource* related to a break, and a *duration*. It is provided through the graphical user interface by selecting the option 'Modify the duration...' in the popup menu of this break, as illustrated in Figure 5.42. This option opens the same window as for the creation of a break, which allows the input of another duration.

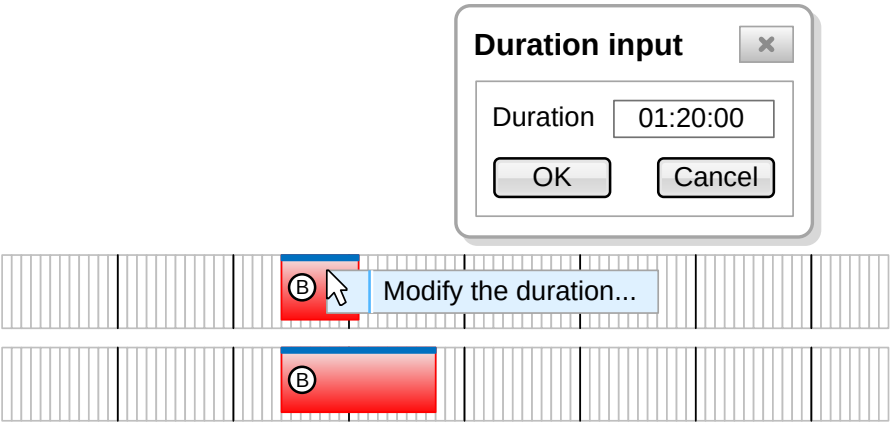

Figure 5.42: Duration modification

Before modifying the duration of the break, the interactive planning system checks that this modification is authorized. In this case, the break is modified. Otherwise, it is not modified.

Break automatic creation

Only one parameter is required to automatically create breaks for a transportation planning in order to respect the union rules: a *date*. It is provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Create the breaks automatically' in the popup menu of this planning grid, as illustrated in Figure 5.43.

If it is authorized, the interactive planning system creates a break when the drivers allocated to the trucks have to take breaks as imposed by the union rules.

The duration of each break varies if it is an 'normal' break or a 'lunch' break. Note that the number of breaks depends on the deliveries, ashes recoveries and planned tasks allocated to the trucks. If a default truck-driver is specified for a truck, he is automatically allocated to its breaks.

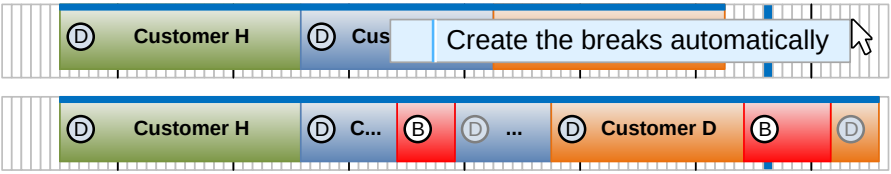

Figure 5.43: Break automatic creation

Other functionalities

As for the deliveries, it is possible to directly move a selected break or to move a selected break with the set of planner items that follows it. Furthermore, it is also possible to allocate drivers to a break, to modify the memo of a break, to lock/unlock a break, and to delete a break.

5.2.6 Functionalities related to the planning

Planner items compaction

During or at the end of the transportation planning elaboration, some planner items may not be compacted. To avoid the dispatcher to move successively each concerned planner item with the cursor, a dedicated functionality has been implemented to compact a set planner items.

Two parameters are required to compact a set of planner items: the *start date and time* and the *resource* related to the planner items of a set. It is provided through the graphical user interface by selecting a set of cells of the planning grid belonging to the timeline schedule related to a set of planner items, and by selecting the option 'Compact the planner items' in the popup menu of this planning grid, as illustrated in Figure 5.44.

Before compacting the set of planner items, the interactive planning system checks that this compaction is authorized. In this case, the planner items of the

set are compacted with the first planner item of the set. Otherwise, they are not compacted.

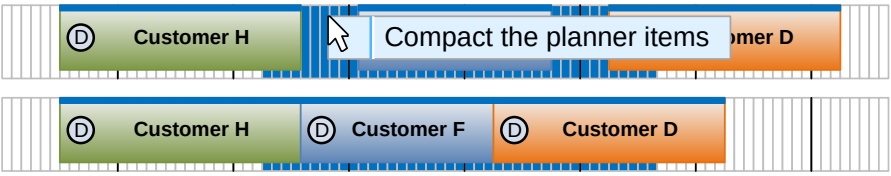

Figure 5.44: Planner items compaction

Planning validation

Although checks are made during the transportation planning elaboration by the interactive planning system, either when creating, modifying and deleting planner items, or when moving planner items, a final check has to be made at the end of this elaboration before exporting the data needed by the weighing system to ensure that there is no inconsistency and that all required information is provided. A functionality has been implemented for this purpose.

Only one parameter is required to valid a transportation planning: a *date*. It is provided through the graphical user interface by selecting a cell of the planning grid belonging to the timeline schedule, and by selecting the option 'Validate the planning' in the popup menu of this planning grid, as illustrated in Figure 5.45.

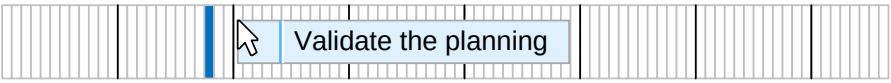

Figure 5.45: Planning validation

Before validating the transportation planning, the interactive planning system checks that each 'preloaded' delivery not associated to an order is not followed by planner items. Furthermore, the information system checks that at least one truck-driver is allocated to each delivery, ash recovery and break, and that the stocks of cement types available at the used local depots are sufficient to do the concerned deliveries. If the planning is validated, the data needed by the weighing system are exported and presented is the tab 'Export'. Otherwise, no export is

done. Following this validation, the traceability of the truck-drivers related to their access to the customers is updated.

Other functionalities

Although each break or more generally each planner item can be deleted following its selection, it is however possible to delete all the breaks or all the content, i.e. planner items, of a transportation planning in one step thanks to two dedicated functionalities related to the transportation planning.

5.3 Conclusion

In this chapter, the graphical user interface and the functionalities related to the interactive planning system have been successively discussed. Due to the completeness of the information system, the presentation of the graphical user interface has been limited to the main window of the application, and more especially to its tab 'Planning', while the description of the functionalities has been restricted to the core functionalities of the information system. Following a short presentation of the layout related to the main window, the software objects that compose the tab 'Planning' have first been described in detail. It concerns a list of orders that have to be split into deliveries for the current transportation planning, a list of truck-drivers that are not available for this planning, a timeline schedule used to display the deliveries, ashes recoveries, planned tasks, and breaks related to this planning, and a text field containing the available truck-drivers that are not yet allocated to a planner item related to this planning. The core functionalities have then been described in detail and illustrated with examples. They allow not only the creation, modification, and deletion of the four types of planner item mentioned before, but also the preparation and finalization of the transportation planning elaboration. These functionalities are accessible either through the popup menu of the planning grid belonging to the timeline schedule of the tab 'Planning', or through the popup menu of a planner item displayed in this timeline schedule.

Although the three layers of the system architecture that decompose the interactive planning system have now been presented, further information related to the second layer, i.e. the business tier, has however to be discussed. Indeed, the use of each core functionality related to a planner item has been presented in this

chapter independently of its impact on planner items that precede and follow the planner item, and without worrying if the time is up to use the functionality. For this purpose, the different cases related to the precedences of the planner items and to the possibilities to use or not the core functionalities are enumerated and treated in the following chapter.

Chapter 6

Preceding rules and frozen horizon rules

After having presented the main elements of the graphical user interface and described the core functionalities in previous chapter, information about the impact of creating, modifying or deleting a planner item in the timeline schedule of the interactive planning system has to be explained in detail. This impact does not only concern that planner item, but also the eventual planner item that just precedes and follows it. Depending on the current date and time, such operations can however be forbidden by the information system. Rules that ensure for each truck the construction of a feasible route are first presented in Section 6.1. Rules that authorize or not the use of each core functionality are then discussed in Section 6.2. Finally, a conclusion is given in Section 6.3.

6.1 Preceding rules

When elaborating a transportation planning, it is fundamental that the route of each truck is always feasible from its first planner item to its last planner item. To avoid unfeasible situations, the interactive planning system therefore assists the dispatcher when this one creates, modifies or deletes a planner item by first detecting the corresponding case and by applying then the appropriate handling.

The set of all different cases that may happen when elaborating a transportation planning is called 'preceding rules', which ensure the maintenance of a feasible route for each truck whatever the core functionality used by the dispatcher. Note that the breaks and planned tasks are not concerned by these rules because those kinds of planner item do not include travels, but only durations. For this purpose, only deliveries and ashes recoveries are implied in the 'preceding rules' and therefore considered as planner items in current section. Before describing how cases are handled in Subsection 6.1.2, the composition of a planner item is first presented in Subsection 6.1.1.

6.1.1 Composition of a planner item

As mentioned before, only planner items related to deliveries and ashes recoveries are considered in this section. The duration of these planner items does not only include time related to the load and unload of material, but also time related to the travels between different locations. Although introduced in Chapter 1, the composition of a delivery is presented again hereafter, but in another way. This one is different than the composition of an ash recovery, also presented.

Composition of a delivery

The duration of a delivery, illustrated in Figure 6.1, is composed of a *loading time*, a *travel time*, an *unloading time*, and a *back travel time*. Note that a 'preloaded' delivery does not include a *loading time* due to the corresponding truck, which is already loaded the day before.

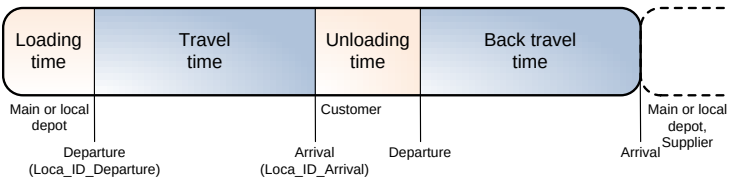

Figure 6.1: Composition of a delivery

The load of the cement type is done either at the main depot, i.e. the factory, or at one of the local depots located in the railway stations, while its unload is done at the customer of the order related to the delivery. Note that a local depot can only

be used for the load of the cement type if another delivery assigned to an order directly precedes the actual delivery. The *loading* and *unloading time* depend on the used truck, on the cement type, and on the location. The *travel time* depends on the used truck, on the departure location of the travel, i.e. the main or local depot, and on the arrival location of the travel, i.e. the customer. The *back travel time* depends on the used truck, on the departure location of the back travel, i.e. the customer, and on the arrival location of the back travel, which corresponds to the departure or arrival location of the following planner item if this one exists, or to the main depot otherwise.

To compute the total duration of a delivery, it is necessary to know the used truck, the cement type, the depot used to load this cement type, the customer, and the departure or arrival location of the following planner item. All this information is stored for each delivery as a unique record in the table **Planner item** described in Chapter 4, except for the departure or arrival location of the following planner item, which can however easily be found when positioning on the corresponding record of this table.

Composition of an ash recovery

The duration of an ash recovery, illustrated in Figure 6.2, is composed of a *travel time*, a *loading time*, a *back travel time*, and an *unloading time*. Note that some ashes recoveries preceded by a delivery do note include a *travel time* because this time is already included in the *back travel time* of the delivery.

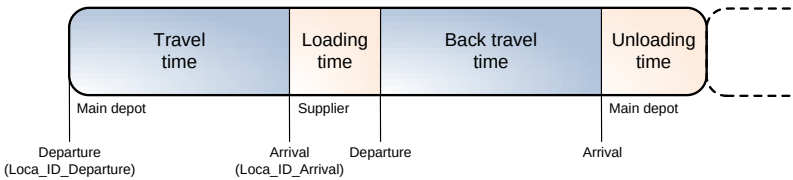

Figure 6.2: Composition of an ash recovery

The load of the ash type is done at the supplier, while its unload is done at the main depot, i.e. the factory. The *loading* and *unloading time* depend on the used truck, on the ash type, and on the location. If included, the *travel time* depends on the used truck, on the departure location of the travel, i.e. the main depot, and on the arrival location of the travel, i.e. the supplier. The *back travel time* depends

on the used truck, on the departure location of the back travel, i.e. the supplier, and on the arrival location of the back travel, i.e. the main depot.

To compute the total duration of an ash recovery, it is necessary to know the used truck, the ash type, the departure location if mentioned, and the supplier. All this information is stored for each ash recovery as a unique record in the table **Planner item** described in Chapter 4.

6.1.2 Cases handling

To deal with any situation when elaborating a transportation planning with the interactive planning system, 36 different cases to handle can be detected. These cases concern the creation, modification and deletion of a delivery or ash recovery, and can therefore be allocated into 6 distinct groups. In Figure 6.3, an activity diagram related to one of these groups illustrates the paths allowing to detect each of the 6 different cases that may be handled when creating a delivery. Due to the significant number of different cases, only the handling of 'Case 1' will be described in this section. For further information, Appendix A presents the activity diagrams related to the 6 distinct groups mentioned before, while Appendix B describes the handling of the 36 different cases that can be detected.

To facilitate the description of any case handling, the departure location of a travel and the arrival location of a back travel are sometimes simply named 'departure location' and 'arrival location'. In a same way, the arrival location of a travel, i.e. the departure location of a back travel, is simply named supplier or customer depending on the type of planner item concerned. Illustrated in Figure 6.4, 'Case 1' concerns the creation of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is inserted between two planner items (1 & 3). Note that grey planner items indicate either deliveries or ashes recoveries, while blue planner items correspond to 'normal' deliveries.

When creating such a delivery (2), the default depot used to load the cement type always corresponds to the main depot, i.e. the factory. This implies eventually a modification of the duration of the planner item that precedes the created delivery (1) if this one is a 'normal' delivery or a preloaded delivery assigned to an order whose truck does not drive back to the main depot after having unloaded its cement type at the customer. To compute the duration of the created delivery (2), it is also necessary to know if the following planner item (3) is a delivery or an ash recovery. If it is a delivery, the truck drives back to the main or local depot used for this delivery after having unloaded the cement type of the created delivery at

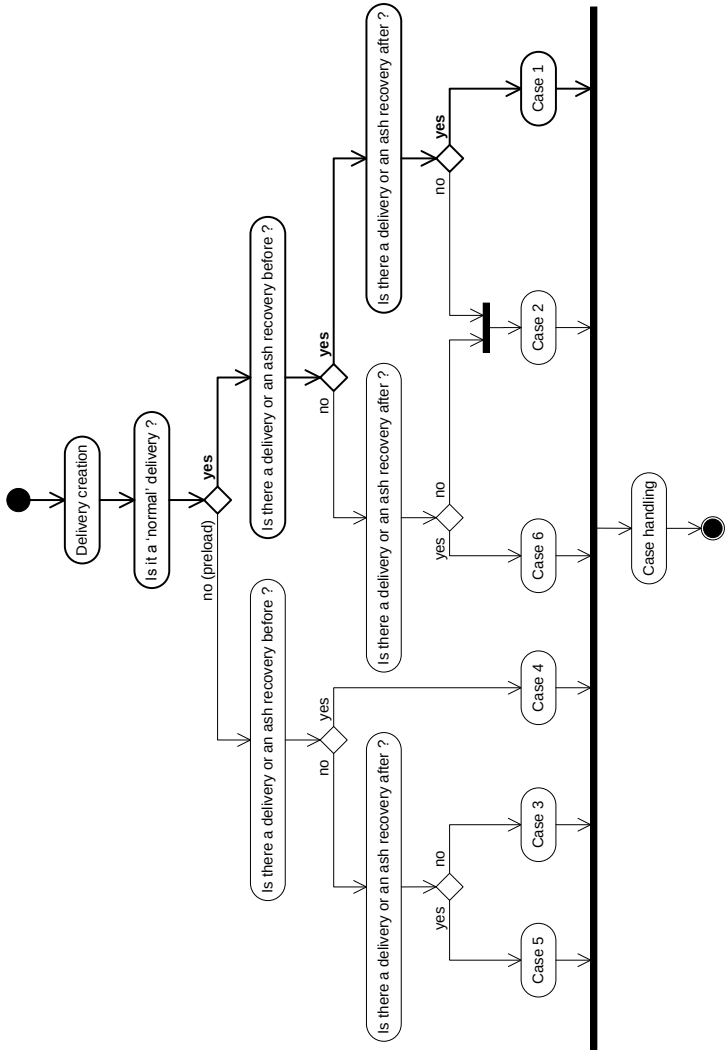

Figure 6.3: Delivery creation

the customer. Otherwise, i.e. if it is an ash recovery, the truck directly drives to the ash supplier.

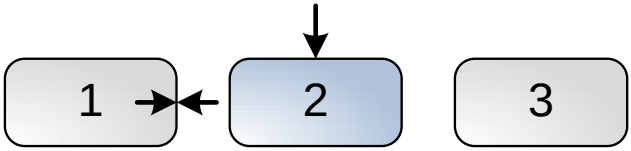

Figure 6.4: Case 1

6.2 Frozen horizon rules

Also implemented in the interactive planning system, the 'frozen horizon rules' can be considered as an additional layer of rules that detect if a core functionality can be used or not by the dispatcher depending on the current date and time and on the properties of the concerned planner item. The concept of the frozen horizon and its implementation in the interactive planning system are first described in Subsection 6.2.1. The impact of the frozen horizon on the availability of the core functionalities is then presented in Subsection 6.2.2.

6.2.1 Frozen horizon

Thanks to the interactivity provided by the interactive planning system, it frequently happens that a planning is adjusted by the dispatcher due to unforeseen situations. When changes have to be made, it is however senseless to modify events that are already finished. To avoid such operations, the interactive planning system includes the concept of 'frozen horizon'. This concept is based on 'time fences' presented in [40]. Used in master production scheduling, the purpose of 'time fences' is to maintain a reasonably control flow through a production system. Each time fence corresponds to a period in which only changes specified by rules can be made. Three basic time fences can be defined although their number and their rules depend on the context of their use: a 'frozen time fence' where no change is authorized, a 'moderately firm time fence' where only some changes are authorized, and a 'flexible time fence' where any change is authorized.

The purpose of the 'frozen horizon' implemented in the interactive planning system is to ensure the consistency of a planning. Including simultaneously the notions of 'frozen time fence' and 'moderately firm time fence' described before, the 'frozen horizon' corresponds to the period that precedes the current date and time of the information system. To avoid short-term modifications, this period can eventually be extended with an additional duration defined in the settings of the system. The rules of the 'frozen horizon', which concern all types of planner item, state which functionalities are available during this period.

6.2.2 Functionalities availability

Presented in Chapter 5, the core functionalities that the dispatcher can use in the interactive planning system to elaborate a transportation planning are inventoried in Table 6.1 and grouped by type of planner item. Note that functionalities related to all types of planner item are first mentioned. This table presents the 'frozen horizon rules', i.e. the availability or unavailability of each functionality if the involved planner item is not in the frozen horizon (N), is partially in the frozen horizon (P), or is totally in the frozen horizon (T). These three different situations and their incidence on the functionalities are described hereafter. Note that different types of delivery exist. For this purpose, Table 6.2 indicates what types of delivery are concerned by an available functionality mentioned in Table 6.1.

No frozen horizon

As illustrated in Figure 6.5, a planner item is not in the frozen horizon if its start date and time and its end date and time are subsequent to the date and time related to the end of the frozen horizon.

Figure 6.5: No frozen horizon

If a planner item is not in the frozen horizon, all its functionalities are available, except however if its type corresponds to a delivery. Indeed, an order can be

Table 6.1: Frozen horizon rules

Planner item type	Functionality	Frozen horizon N	Frozen horizon P	Frozen horizon T
All	Planner item moving	✓		
	Drivers allocation	✓		
	Memo modification	✓		
Delivery	Delivery / Preload creation	✓[1]		
	Order association	✓[2]	✓[2]	
	Another departure location selection	✓[3]		
	Quantity to deliver adaptation	✓[3]	✓[3]	✓[3]
	Truck fulfillment	✓[3]	✓[3]	✓[3]
	Another order association	✓[4]	✓[4]	
	Conversion in preload of the next planning	✓[3]	✓[3]	✓[3]
	Usage of the preload of the next planning	✓[5]		
	Delivery locking / unlocking	✓[1]		
	Delivery deletion	✓[6]	✓[7]	
Ash recovery	Ash recovery creation	✓		
	Another supplier selection	✓	✓	
	Another ash type selection	✓	✓	
	Another departure location selection	✓		
	Ash recovery deletion	✓		
Planned task	Planned task creation	✓		
	Another task type selection	✓	✓	
	Another duration selection	✓	✓	
	Planned task locking / unlocking	✓		
	Planned task deletion	✓		
Break	Break creation	✓		
	Another duration selection	✓	✓	
	Break deletion	✓		

Table 6.2: Types of delivery concerned by an available functionality

	Preloaded delivery				Normal delivery	
	Without order		With order			
	Not loaded	Loaded	Not loaded	Loaded	Not loaded	Loaded
1	✓	✓	✓	✓	✓	✓
2	✓	✓				
3			✓	✓	✓	✓
4				✓		✓
5			✓	✓	✓	
6	✓		✓	✓	✓	
7	✓		✓		✓	

associated to a delivery only if this delivery is initially not assigned to an order and corresponds therefore to a preloaded delivery. Inversely, the selection of another departure location, the adaptation of the quantity to deliver, the fulfill of the truck, and the conversion of a delivery in preload of the next planning can only be done if the delivery is beforehand assigned to an order. Moreover, it is possible to associate another order to a delivery only if this one is loaded and already assigned to an order. Furthermore, the preload of the next planning can be used for a delivery only if this one is beforehand either a preloaded delivery, or a 'normal' delivery that is not loaded. Finally, a delivery can be deleted only if it is not loaded, or if it corresponds to a preloaded delivery assigned to an order that is loaded. For this last alternative, only the order is disassociated from the delivery, which becomes again a preloaded delivery not assigned to an order.

Partial frozen horizon

As illustrated in Figure 6.6, a planner item is partially in the frozen horizon if its start date and time is prior to the date and time related to the end of the frozen horizon and if its end date and time is subsequent to the date and time related to the end of the frozen horizon.

Figure 6.6: Partial frozen horizon

If a planner item is partially in the frozen horizon, only some functionalities are available. For example, the departure location of a delivery or ash recovery cannot be changed because this one is included in the frozen horizon. As discussed in Section 6.1, such a modification could furthermore have an impact on the end date and time of an eventual planner item that precedes this delivery or ash recovery. To resume, only functionalities that does not affect the start date and time of a planner item are still available. However, a delivery or a planned task can no more be locked or unlocked. Moreover, a delivery can be deleted even if its departure location is in the frozen horizon, but only if this delivery is not loaded.

Total frozen horizon

As illustrated in Figure 6.7, a planner item is totally in the frozen horizon if its start date and time and its end date and time are prior to the date and time related to the end of the frozen horizon.

Figure 6.7: Total frozen horizon

If a planner item is totally in the frozen horizon, none of its functionalities is available. Exceptions however exist if the type of a planner item corresponds to a delivery and if the functionality does not affect the start date and time and the end date and time of the planner item.

6.3 Conclusion

In this chapter, the mechanism implemented in the interactive planning system to ensure for each truck the creation of a feasible route has first been presented. For this purpose, the composition of deliveries and ash recoveries, i.e. planner items whose duration includes travel times, has been described in detail. Furthermore, the handling of one of the cases that may happen when elaborating a transportation planning, called 'preceding rules', has been discussed. The concept of the 'frozen horizon' and its implementation in the interactive planning system have then been described. Following this description, the 'frozen horizon rules', i.e. the availability or unavailability of each functionality used by the dispatcher to elaborate a transportation planning if the involved planner item is not, partially or totally in the frozen horizon, have been presented.

The end of this chapter marks the end of the first part of this thesis, which has presented the main steps of the development of an interactive planning system. This intermediate system is illustrated by a schema in Figure 6.8. Indeed, an additional module has now to be developed in order to transform this actual interactive planning system into an interactive planning support system that helps

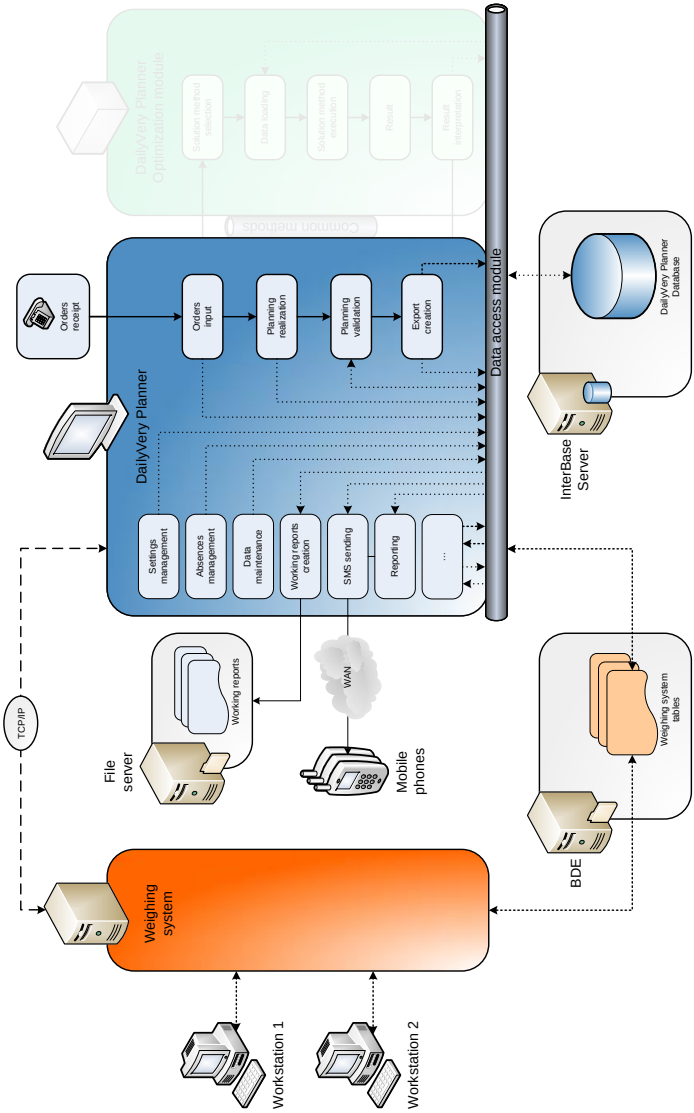

Figure 6.8: Intermediate system schema

the dispatcher to elaborate any transportation planning by providing different solution methods. The second part of this thesis focuses therefore on this module and more precisely on the modeling and performance of its implemented solution methods.

Part II

From an interactive planning system to an interactive planning support system

Chapter 7

Preamble to the development of an optimization module

The second part of this thesis presents the development of an interactive planning support system, or more precisely the implementation of an optimization module intended to be combined with the interactive planning system discussed in the first part. This final system is illustrated by a schema in Figure 7.1. The optimization module has to provide solution methods that can elaborate for any problem instance a transportation planning of high quality. The modeling of such methods however requires to restrict the problem area. For this purpose, a simplified version of the problem, called cement delivery problem, is first given in Section 7.1. Existing solution methods related to neighboring problems are then presented in Section 7.2. The structure of the remaining chapters of this second part is finally described in Section 7.3.

7.1 Cement delivery problem description

A factory produces different cement types. Every day, cement has to be delivered by trucks to several customers. Due to the trucks capacity, each order is split into one or more deliveries. These deliveries are supplied either from a main depot, the factory, where quantities of cement types are unlimited or, for special trucks with

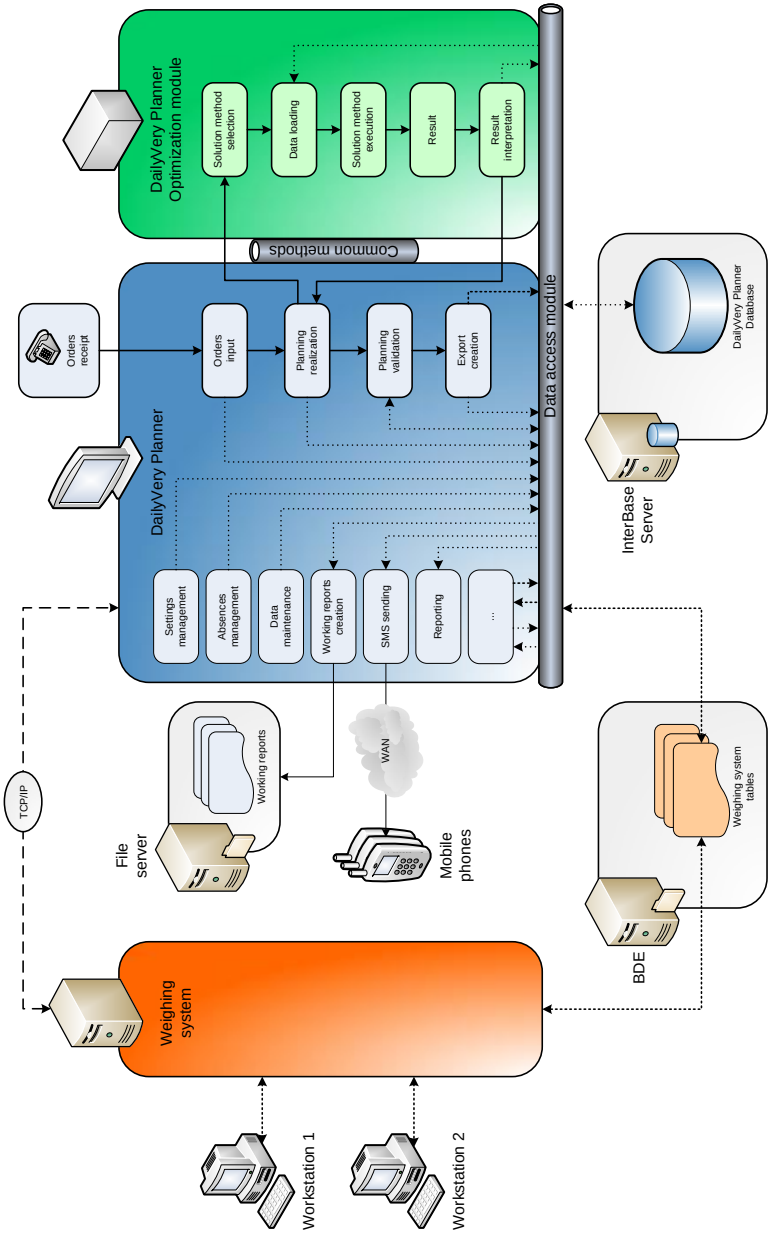

Figure 7.1: Final system schema

specific equipment, from local depots located in railway stations where quantities of cement types are limited. Each delivery has a *loading time* at a depot, a *travel time* between this depot and the concerned customer, an *unloading time* at this customer and a *back travel time* between this customer and a depot. The loading and unloading times depend on the used truck, on the cement type and on the equipment of the location. The travel and back travel times depend on the used truck and on the type of road to the locations. Each truck starts and ends its daily activity at the factory such that the first delivery of a truck is loaded at the factory and after its last delivery, each truck has to drive back to the factory.

The fleet of vehicles is heterogeneous: this means that each truck has its own capacity and can only deliver some customers. To spare time, the trucks, which have also a daily availability, can be 'preloaded' the previous day with a cement type that will be probably ordered by a customer. Note that the dispatcher can impose a first delivery and fix various tasks during the scheduling to some trucks like maintenance operations, for example. However, ashes recoveries are not taken into account in the problem description. Such extension is left for future developments. Although most of the vehicles belongs to the factory, some trucks can be rented to a third party if necessary which involves naturally additional costs. Each truck-driver belonging to the factory can only drive some trucks, supply some customers and handle some cement types. Every day, each available truck is assigned to an available truck-driver by the dispatcher.

Although the natural objective of the cement delivery problem is to determine a set of routes that minimizes the total travel duration, the factory also wants to minimize the number of trucks supplying a same customer and the number of trucks used to do the deliveries. Note that ashes recoveries and breaks are not considered in the problem. After a solution is created in the interactive planning system by a solution method of the optimization module, planner items related to ashes recoveries and breaks can however be created by the dispatcher to complete the transportation planning.

A solution related to an instance of the cement delivery problem is illustrated in Figure 7.2. In this problem instance, the factory, represented by a black square, uses 4 different trucks ($T1$, $T2$, $T3$ and $T4$) to do cement deliveries. In addition to the main depot, i.e. the factory, 3 local depots located in railway stations, represented by black triangles, can also be used. Although 5 cement types, represented by 5 colored squares, are available in unlimited quantity at the main depot, only 3 cement types are available in limited quantity at each local depot. Unlike truck $T1$, which can only load cement at the main depot, trucks $T2$, $T3$ and $T4$ have

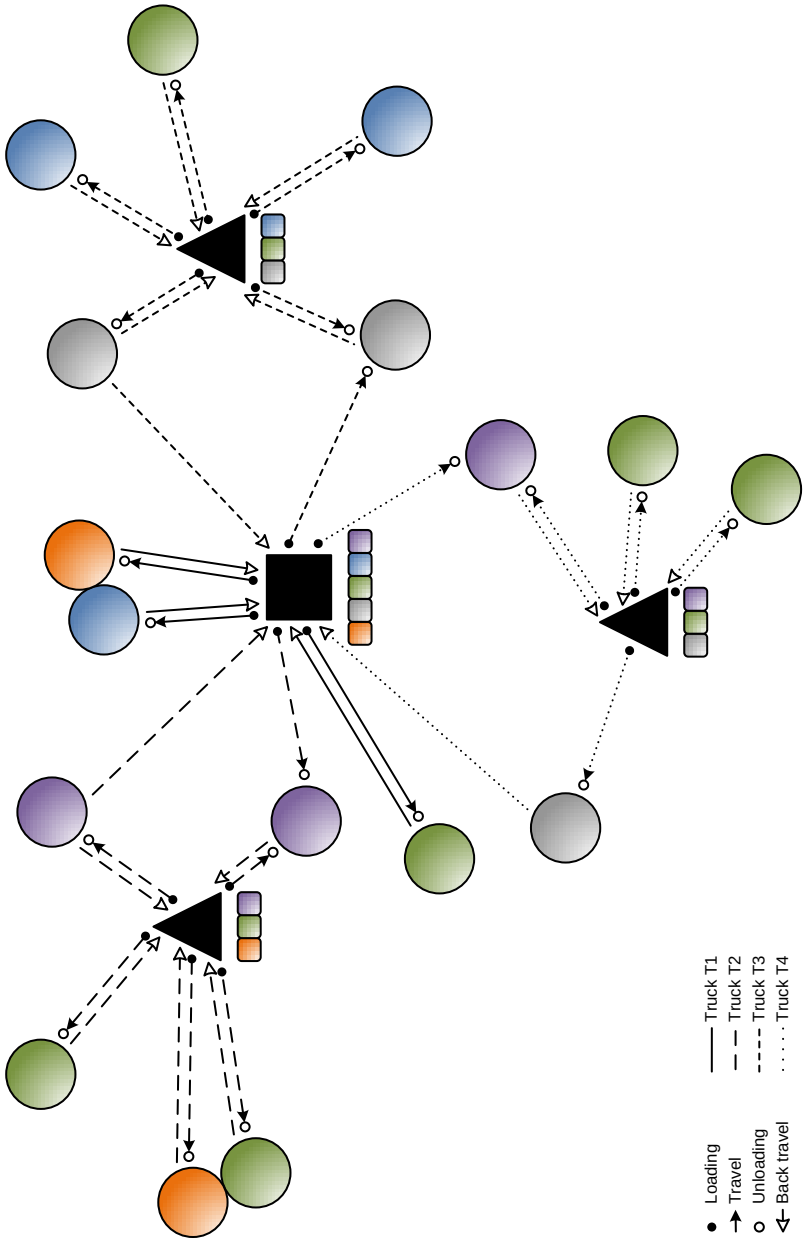

Figure 7.2: Cement delivery problem

the specific equipment required to load cement at local depots. 17 orders, represented by colored circles related to the cement types, have been placed by 15 customers. Note that orders related to a same customer are adjacent. For each of these orders, the appropriate truck loads the cement type at a depot, drives to the customer of the order, unloads its content at the customer and drives back to a depot either to do another delivery for the same order, or to do a delivery for another order, or to end its route. Note that each truck starts and ends its daily activity at the factory. The number of deliveries needed by each order is not mentioned in the figure but depends on the ordered quantity and on the capacity of the truck used for each of these deliveries.

7.2 State of the art

This section first presents a literature review for the Capacitated Vehicle Routing Problem and the Split Delivery Vehicle Routing Problem, and then indicates some similarities and differences between these problems and the Cement Delivery Problem.

Capacitated Vehicle Routing Problem

In the classical Capacitated Vehicle Routing Problem (CVRP), an homogeneous fleet of vehicles with limited capacity has to serve a set of customers from a single depot with the objective of minimizing the total traveled distance. Each customer has to be visited exactly once and the total demand of the customers visited by a vehicle cannot exceed its capacity. An overview of exact and approached solution methods for the CVRP can be found in [41], [30] and [44]. Some of these methods are presented hereafter.

Regarding exact algorithms, Christofides, Mingozzi and Toth (1981) [14] developed a direct tree search algorithm for a symmetrical CVRP defined on a non oriented graph. This solution method is based on the k-degree center tree relaxation of a Multiple Traveling Salesman Problem (MTSP) where k corresponds to the number of edges linked to the depot in the solution. Note that k does not necessarily correspond to the number of available vehicles that is fixed. Any feasible solution is composed of a set of edges partitioned in three subsets: edges forming a k-degree center tree, i.e. a spanning tree over the graph starting from the depot; edges not belonging to the solution and adjacent to the depot; edges not belong-

ing to the solution and not adjacent to the depot. This CVRP is formulated as an integer linear programming model. The objective consists in minimizing the sum of all edges costs in the solution. The constraints define the k-degree center tree and impose especially the number of edges incident to the depot, the number of edges not incident to the depot and the degree of every vertex, i.e. customer. To simplify the resolution of the problem, the objective and some constraints are redefined using Lagrangian relaxation. A lower bound on the optimal CVRP solution is also given in order to solve the problem in a polynomial time.

In [15], Christofides, Mingozzi and Toth utilized dynamic programming and proposed a method providing lower bounds on the cost of the optimal solution. This solution method considers a CVRP with a fixed number of available vehicles. A recursive function computes the minimum cost achievable using these vehicles and delivering to a subset of customers. Note that the number of recursions of this function depends not only on the number of vehicles required, but also on the different subsets of customers that can be created for each vehicle. Due to the number of computations that can be excessive, the application of dynamic programming allows a significative reduction of the number of states thanks to a state-space relaxation procedure. This procedure provides a lower bound on the cost of the optimal solution. The optimum can then be reached by embedding the bounding procedure in an enumerative scheme.

In addition to the two techniques described before, integer linear programming is also employed. Indeed, Balinski and Quandt (1964) [6] proposed a set partitioning formulation for the CVRP. Due to the huge number of binary variables involved in real life instances of this formulation, a column generation algorithm can be used to solve the problem. With this algorithm, a reduced problem containing only a restricted subset of all possible variables is repeatedly solved. Note that this algorithm has to be combined with a branch-and-bound algorithm because solutions of the CVRP are integer. In [23], Fisher and Jaikumar (1981) developed a three-index vehicle flow formulation for the CVRP with time windows and no stopping times. Compared to a two-index formulation where dedicated variables can only indicate if a vehicle passes on an arc or edge, a three-index formulation improves the precision of these variables that can henceforth indicate which vehicle passes on an arc or edge. Fisher and Jaikumar also developed an algorithm based on this formulation providing a heuristic solution, i.e. a not necessarily optimal solution. Laporte, Nobert and Desrochers (1985) [43] proposed a two-index vehicle flow formulation for the CVRP. The third index related to the vehicle is dropped in order to have a more compact formulation. Dedicated variables indicate how many times an edge is traversed by a vehicle. The number of

vehicles needed to visit the customers is defined by a lower bound in the constraints related to the subtours elimination. To solve their model, Laporte, Nobert and Desrochers developed a constraint relaxation algorithm.

In terms of heuristic algorithms, Clarke and Wright (1964) [16] proposed a constructive method to solve the CVRP. Starting with vehicle routes containing the depot and one customer, two routes are merged at each step of the method according to the largest saving that can be generated. For this purpose, all possible savings obtained by merging two routes are first computed and ordered in a non-increasing way. Note that only positive savings are considered. Then, starting with the first ordered saving, the two concerned routes are tentatively merged by introducing the arc related to this saving and by deleting the two useless arcs. If the resulting route is feasible, the merge is implemented. This step is repeated until no further improvement can be found.

Wren and Holliday (1972) [59] and Gillett and Miller (1974) [31] developed a two-phase method called 'sweep algorithm' that first defines clusters of customers and then optimizes the vehicle route for each cluster. With this method, a cluster of customers is determined by rotating a ray centered at the depot and by assigning customers hit by the ray to a vehicle as long as its capacity is not exceeded. A vehicle route is then obtained for each cluster by solving a Traveling Salesman Problem (TSP) with an exact or approached solution method. A postoptimization phase in which customers are first exchanged between adjacent clusters and then routes are reoptimized can eventually be included in the implementation of this method.

Metaheuristics, i.e. solution methods that explore a solutions space to identify good solutions, also exist to solve the CVRP. Before presenting simulated annealing and tabu search algorithms, i.e. two different metaheuristics, the descent method has beforehand to be introduced. This heuristic method first constructs an initial solution and then searches at each step the best neighbor of the current solution until there is no more improvement, i.e. until having reached a local minimum or maximum.

Simulated annealing algorithms, inspired from metallurgy, improve the descent method to avoid being trapped in a local optimum. For this purpose, a best neighbor that is not better than the actual current solution may be accepted as next current solution. The acceptance probability especially depends on a temperature, initially high, that decreases at the end of each cycle. Note that a cycle is composed of a predefined number of steps. More this temperature is low, more the acceptance probability of a neighbor that is less good than the current solution

is low. This metaheuristic ends when the best solution ever found is not improved during an entire cycle. In [49], Robusté, Daganzo and Souleyrette (1990) proposed an implementation of the simulated annealing where a neighborhood structure is defined by reversing a part of a route, by moving a part of a route into another part of the same route, and by exchanging customers between two routes. Alfa, Heragu and Chen (1991) [2] proposed a route-first, cluster second heuristic that is used to construct a first solution, followed for the search process by a 3-opt heuristic, which consists at each step in deleting first three parts of a route, reconstructing then this route in all possible ways and selecting finally the optimal route.

Tabu search algorithms, whose approach is described by Glover and Laguna in [32] and [33], also improve the descent method by including local searches with a memory structure. A neighborhood of solutions is examined as in simulated annealing, but the next current solution corresponds to the best neighbor of the actual current solution. To avoid cycling, the acceptance of solutions that were recently examined are forbidden, or tabu, for a number of steps. This metaheuristic ends when the best solution ever found is not improved after a predefined number of steps. The use of tabu search algorithms to solve the CVRP is discussed in [37] by Hertz, Taillard and de Werra (1997). In this problem, customers are often moved from one route to another to construct a neighborhood of solutions. Variants of the tabu search algorithm, such as the granular tabu search or the unified tabu search algorithm, are presented in [17] by Cordeau, Gendreau, Hertz, Laporte and Sormany (2005), and in [18] by Cordeau and Laporte (2005). The granular tabu search defines the CVRP on an undirected graph and removes from this graph edges that are unlikely to appear in an optimal CVRP solution in order to reduce the computational time. Only the edges incident to the depot and all edges whose length does not exceed a given granularly threshold are retained. Note that this threshold depends on the properties of the problem instance. The unified tabu search algorithm allows solving periodic or multi-depot CVRP including or not time windows. Unlike initial tabu search algorithms, a new diversification phase is introduced into this algorithm. Whenever the value of the best solution found has not been improved for a number of steps, the depot is moved to the first vertex of a randomly selected route and temporarily remains in this location.

In [29], Gendreau, Iori, Laporte and Martello (2008) proposed a tabu search algorithm for solving a variant of the CVRP where the demand of a customer is not represented by an integer value, but consists of a certain number of two-dimensional weighted items. Denoted 2L-CVRP, this problem combines the minimization of transportation costs with the feasible loading or unloading of the

items into vehicles. Note that this tabu search algorithm accepts solutions that are unfeasible, i.e. where the total weight exceeds the capacity of the vehicle or where the loading surface exceeds the surface of the vehicle. A penalty is however assigned to unfeasible solutions, which is proportional to the level of the violation. Infeasibilities are treated as penalties in an objective function to be minimized. To create an initial solution, the algorithm of Clarke and Wright is applied until the number of routes corresponds to the number of available vehicles. If this is not possible without violating a weight or loading constraint, an unfeasible solution is however accepted as initial solution. In order to explore promising areas of the solution space, two kinds of intensification techniques are adopted: one executed periodically, i.e. after a predefined number of steps; and one executed whenever a weight-feasible potential new solution is obtained. In the 2L-CVRP, it is assumed that goods cannot be stacked. For this purpose, Gendreau, Iori, Laporte and Martello (2006) [28] developed another tabu search algorithm to solve a combination of CVRP with three-dimensional loading constraints and other constraints frequently encountered in freight transportation. This problem can be denoted as 3L-CVRP. The demand of a customer consists of a set of rectangular boxes of given size and weight. Each vehicle requires the solution of a three-dimensional packing problem, which consists of finding a non-overlapping packing of a set of rectangular boxes into a rectangular container. In addition to the weight and packing constraints, other constraints are often encountered in real-world applications. For example, fragile goods have to be placed on the top. As for the 2L-CVRP, the algorithm accepts unfeasible solutions which are also penalized in the objective function.

In [20], Doerner, Fuellerer, Hartl, Gronalt and Iori (2007) proposed a tabu search algorithm for the CVRP with additional loading constraints related to a wood-products retailer. Indeed, chipboards of different size placed on one, two or three pallets have to be transported with forklift trucks to customers. Note that all chipboards have the same pallet width, but have in general different heights. A vehicle has the same width as a pallet and can contain up to three pallets along its length. The unloading operations have to be performed without having to move items of customers that will be visited later along the route. This problem corresponds to a Multi-Pile Vehicle Routing Problem (MP-VRP) which combines vehicle routing problems with packing and scheduling problems. For solving this variant of the 3L-CVRP, the proposed tabu search algorithm also modifies the objective function by adding a penalty term in order to penalize unfeasible solutions. However, the excess of loading is penalized by a parameter whose value is modified during the search process to give more or less importance to the loading penalization.

Zhu, Qin, Lim and Wang (2012) [60] presented a two-phase tabu search approach for the 3L-CVRP. As before, the algorithm of Clarke and Wright is used to generate an initial solution. If this initial solution is not feasible, the first phase of this tabu search focuses on finding a feasible solution. Once a feasible solution is found, the second phase focuses on minimizing the total traveling cost by exploring the neighborhood of solutions while maintaining feasibility. Five operators are used to generate neighboring solutions: a 2-opt operator which reverses the visiting orders of all customers located between a pair of customers on a route; a 2-swap operator which swaps the visiting orders of a pair of customers on a route; a move operator which transfers a customer from one route to another; a crossover operator which exchanges two sequences belonging to two different routes; and a splitting operator which splits a route into two routes.

Split Delivery Vehicle Routing Problem

In the Split Delivery Vehicle Routing Problem (SDVRP), the restriction that each customer is visited once is removed. The SDVRP has been introduced by Dror and Trudeau (1989) [22] who derived structural properties of optimal SDVRP solutions and empirically showed that allowing split deliveries can lead to substantial cost savings in terms of the total traveled distance and the number of vehicles used. To solve the SDVRP, Dror and Trudeau described a two-stage algorithm which constructs first a VRP solution and then a SDVRP solution. For the second stage of their algorithm, they proposed a k-split interchange subroutine followed by a route addition subroutine. The k-split interchange subroutine computes all the savings obtained when removing a demand point of a route and reintroducing it progressively in the $k-1$ remaining routes until the demand is entirely satisfied. Note that these remaining routes are sorted in descending order of their spare capacity. The split that generates the best saving is selected. When a demand point is already split, this one is beforehand removed from all concerned routes. The route addition subroutine adds routes to eliminate demand nodes split between several routes when this reduces the total routing cost. The heuristic algorithm ends when no improvement can be found.

Archetti, Savelsbergh and Speranza (2008) [3] also concluded that allowing split deliveries can lead to significant cost savings. Indeed, they focused on estimating the benefits of allowing split deliveries based on customer characteristics when demands are large and small relative to a vehicle capacity, and a little over half a vehicle capacity. For this purpose, they computed a ratio between the cost of an optimal VRP solution and the cost of an optimal SDVRP solution. They

observed that the largest benefits are obtained when mean customer demand is a little over half the vehicle capacity and customer demand variance is relatively small.

In [7], Belenguer, Martinez and Mota (2000) proposed a lower bound for the SDVRP where the demand of each customer is lower than the capacity of the vehicles, and the quantity delivered by the vehicles when visiting a customer is an integer number. After having defined the SDVRP as a complete and undirected graph, formulated it as an integer linear programming model and applied some relaxations, they first studied the associated polyhedron. To solve the SDVRP, they developed then a cutting-plane algorithm where violated constraints of a solution are identified using heuristic methods. This algorithm, which iteratively refines a feasible solution by adding linear inequalities, allows the resolution of small SDVRP instances.

Dror, Laporte and Trudeau (1994) [21] formulated an integer linear program for the SDVRP and described valid inequalities in addition to subtour elimination constraints. Indeed, constraints impose that each vehicle used to visit customers starts its route from the depot. When all vehicles have a same capacity, a constraint imposes the assignment of one vehicle to the customer located the furthest away from the depot. If the distance between two customers is symmetric, another constraint imposes that an arbitrary pair of customers is never visited by a same vehicle. Other constraints are also imposed to eliminate fractional cycles. To solve the SDVRP, a two-stage algorithm is developed. A relaxation of the problem, i.e. a subproblem, is first solved with simplex. If a constraint of the problem is violated, this one is added to the subproblem which is solved again. This process is repeated until no more constraint of the problem is violated. A branch-and-bound algorithm is then employed to achieve integrality.

In [19], Desaulniers (2010) formulated a multicommodity network flow with additional variables and constraints for a SDVRP with time windows where a commodity corresponds to an available vehicle. This linear programming model is decomposed into a master problem and a subproblem. The subproblem determines the quantities to deliver to each customer visited in a route. The master problem combines the routes with their corresponding delivered quantities to obtain routes with multiple split deliveries. Due to a large number of variables constraints, a branch-and-cut-and-price method is first developed to solve the master problem. Such a method is a branch-and-bound algorithm where the lower bounds are computed by a column generation algorithm and where some constraints are initially relaxed and reintroduced if needed as in a cutting-plane

method. An exact dynamic programming algorithm is then proposed to solve the subproblem.

Ho and Haugland (2004) [39] gived a mathematical formulation for a SDVRP with time windows and described a tabu search algorithm to solve it. This algorithm is decomposed into three phases. Considering travel times and waiting times, an initial feasible solution is first computed by adding on a vehicle route customers that are close to each other until the total demand exceeds the vehicle capacity. If by adding a customer the capacity of a vehicle is exceeded, the excess demand is left to another vehicle and the initial solution contains split deliveries. This initial solution is then improved with a tabu search where four move operators are used to define a neighborhood: a relocate operator removes a split delivery from a route to relocate it on another route; a relocate split operator not only removes a split delivery from a route to relocate it on another route, but also adds on the initial route a split of a delivery related to the other route; an exchange operator permutes customers between a pair of routes; a 2-opt operator exchanges descendants of two deliveries belonging to two different routes. Finally, a postoptimization phase is applied to the best solution found in the tabu search.

In [4], Archetti, Speranza and Hertz (2006) formulated a mixed integer program for the SDVRP and proposed another tabu search algorithm, also composed of three phases, to solve it. In their formulation, the demand of each customer can be greater than the capacity of the vehicles. In the first phase, an initial feasible solution is constructed by reducing the instance. This is done by making as many direct trips as possible for customers whose demand is greater than a vehicle capacity. A direct trip is a tour in which a vehicle starts from the depot, drives directly to a customer where it delivers as many units as its capacity, and drives back directly to the depot. The reduced instance is then solved as a TSP thanks to an insertion procedure and a postoptimization routine. The resulting tour is finally cut into pieces in such a way that the capacity constraints are satisfied. In the second phase, a neighborhood is obtained by first removing a customer from a set of routes and then inserting it either into a new route or into an existing route that has enough residual capacity. The algorithm also considers the possibility of inserting a customer into a route without removing it from another route. The insertion of a customer into a route is done by means of the cheapest insertion method, i.e. where the insertion is the most profitable. In the third phase, the best solution found in the tabu search is improved first by removing all t-split cycles if the distances satisfy the triangle inequality, and then by improving each route of the solution with the insertion procedure and the postoptimization routine mentioned before.

Frizzell and Giffin (1995) [26] presented a mathematical formulation and a heuristic algorithm for the SDVRP with grid network distances and time window constraints. This algorithm, which includes a construction heuristic and two improvement heuristics, requires the split of the SDVRP into sequential time slots. Considering the amount of time required to make a delivery to a customer and the amount of time remaining available before the end of that customer's time window, the main objective of the construction heuristic is to minimize the total time of the routes by allowing a large number of customers receiving split deliveries. This heuristic uses for this purpose a look-ahead approach which dynamically classify urgencies related to customers. One improvement heuristic removes a customer from a route to reinsert it in another route, while the other improvement heuristic exchanges any two customers between two routes. Note that these two improvement heuristics also try to remove any excessive amounts of split deliveries resulting from the construction heuristic.

In [35], Gulczynski, Golden and Wasil (2011) combined the SDVRP with the multiple depot vehicle routing problem (MDVRP) and described an integer programming-based heuristic composed of four phases to solve it. In the first phase, customers are assigned to depots. For this purpose, each customer is allocated to its first closest depot but only if this one is far enough from its second closest depot. Unassigned customers are then allocated to depots thanks to the cheapest insertion method. In the second phase, the SDVRP is solved separately for each depot with a two-stage heuristic. In the first stage, a mixed integer program where the delivery amounts has to be minimized is formulated and solved with a heuristic that maximizes the savings from splitting deliveries at certain customers and reallocating some or all of their demands to new routes. In the second stage, an algorithm is applied to reduce the total distance traveled by the vehicles by exchanging deliveries and roads segments. In the third phase, a mixed integer program is formulated for the MDSDVRP in order to improve the initial solution generated in the second phase. This formulation considers additional split deliveries including inter-depot split deliveries. For each route, two customers with the smallest insertion cost on another route are first considered in an inter-depot candidate set. An inter-depot mixed integer linear program is then formulated whose decision variables concern the move of all, none or some of the demand of the customers belonging to the inter-depot candidate set. The total savings among all possible moves has to be maximized. Due to the size of an instance, a run-time limit is set to solve this MIP. In the fourth phase, the routes are improved with an inter-depot routing algorithm.

Asbach, Dorndorf and Pesch (2009) [5] formulated a ready-mixed concrete

(RMC) delivery problem as a mixed integer programming model based on a graph or flow network. Compared to the standard SDVRP with time windows, new constraints are taken into account. Indeed, the ready-mixed concrete is a perishable product. Furthermore, an heterogenous fleet of vehicles and many depots are considered. Finally, only one customer can be delivered before refilling a vehicle at a depot. To solve this problem, a local search approach is proposed, which improves the current best solution by applying a neighborhood operator like tabu search or simulated annealing try to escape a local optimum. The neighborhood operator is an algorithm which does not change the variable values of a current best solution, but unfixes them. The current best solution is reoptimized with a solver which finds values for the unfixed variables and treats the fixed variables as constants. Due to its time consumption, the solver is replaced by a heuristic to compute a neighborhood.

Other approaches are used to solve a SDVRP with time windows for the delivery of RMC. Schmid, Doerner, Hartl, Savelsbergh and Stoecher (2009) [52] modeled the RMC delivery problem as an integer multi-commodity flow problem on a time-space network. To solve the problem, they proposed a hybrid method integrating optimization and heuristic techniques. The first step consists in generating an initial set of fulfillment patterns for each customer. Note that a fulfillment pattern indicates how a customer can be delivered. After solving the resulting integer multi-commodity flow problem with a multi-commodity flow optimization (MCNF) component, the solution is transferred to a variable neighborhood search (VNS) component, which locally improves the solution and also generates additional fulfillment patterns. After a certain period of time, all the patterns related to the improved solutions are collected and transferred to the MCNF component. The hybrid approach iterates between the two components, each time transferring the best solution from the MCNF component to the VNS component and the fulfillment patterns of the improved solutions encountered by the VNS component to the MCNF component. Further information about this approach can be found in [50]. In [51], Schmid, Doerner, Hartl and Salazar-González (2010) formulated the RMC delivery problem as a mixed integer linear programming model and developed a hybrid solution procedure. After having found an initial solution with a variable neighborhood search (VNS) algorithm, the mixed integer linear programming model is solved repeatedly. At each iteration, a solver is used to solve the formulation. Note that the value of most variables are fixed. Some decision variables however remain unfixed and the optimization process starts over again. The choice of which decision variables are unfixed is left to a variable-fixing approach guided by a very large neighborhood search (VLNS) algorithm which

accepts only solutions that improve the current solution. Note that the VNS and VLNS algorithms can be combined together.

Other real applications of the SDVRP like the distribution of livestock feed and the routing of helicopters for crew exchanges, reviewed by Chen, Golden and Wasil (2007) in [13], are described hereafter. Mullaseril, Dror and Leung (1997) [47] modeled the feed distribution problem as a collection of capacitated rural postman problems with split deliveries and time windows. They developed a solution algorithm composed of four phases. In the first phase, a non-split feasible solution is generated where each route is constructed one at a time until the demand is entirely satisfied. In the second phase, routes are first merged, if possible, to effect savings. The solution is then improved by road segment interchange using a 2-opt heuristic. In the third phase, split-delivery routes are generated using a k-split interchange but only if savings are obtained. In the fourth phase, the solution is modified by route addition where consolidating road segments whose demand is split among several routes into one new route realize savings. In [53], Sierksma and Tijssen (1998) modeled the helicopter routing problem as a discrete split delivery routing problem. They formulated the problem as an integer program and solved a relaxed linear program using column generation and a rounding procedure to produce integer solutions. Indeed, the application of simplex for solving the relaxed linear program provides in general a continuous solution in which the optimal values of the decision variables may be fractional. In addition, they also developed a cluster-and-route procedure where the routing and the clustering are performed simultaneously and finally proposed two improvement heuristics including respectively a 1-opt and a 2-opt heuristic.

Cement Delivery Problem

A cement delivery problem is studied in this thesis. This one can be formulated as a SDVRP with additional constraints. The problem is less complex than those related to the concrete delivery studied in [5], [50], [52] and [51]. Indeed, while these previous works assume that two deliveries at one customer may not overlap, and some vehicles with specialized unloading equipment have to be present to assist other vehicles with their unloading operations, such constraints are not imposed. Also, time windows requirements are replaced by limits on the total availability (in time units) of each vehicle. Another difference is that it is assumed that some vehicles may be loaded in advance, before the beginning of their daily activities. Moreover, the objective of the cement delivery problem differs from those studied in [5], [50], [52] and [51]. Indeed, while it is standard to minimize the total

Table 7.1: Overview of the literature on CVRP

Authors	Solution methodology and remarks
Christofides, Mingozzi and Toth [14]	Direct tree search algorithm (Lagrangian relaxation, lower bound)
Christofides, Mingozzi and Toth [15]	Dynamic programming algorithm (recursive function, state-space relaxation procedure)
Balinski and Quandt [6]	Column generation algorithm and branch-and-bound algorithm
Fisher and Jaikumar [23]	Heuristic algorithm (three-index vehicle flow formulation)
Laporte, Nobert and Desrochers [43]	Constraint relaxation algorithm (two-index vehicle flow formulation)
Clarke and Wright [16]	Heuristic algorithm (constructive method)
Wren and Holliday [59], Gillett and Miller [31]	Sweep algorithm: 1. Clusters of customers definition (rotation of a ray centered at the depot) 2. Vehicle route for each cluster optimization (TSP resolution with exact or approached method)
Robusté, Daganzo and Souleyrette [49]	Simulated annealing algorithm (neighborhood: reversing a part of a route, moving a part of a route, customers exchange between two routes)
Alfa, Heragu and Chen [2]	Simulated annealing algorithm (route-first cluster-second heuristic, neighborhood: 3-opt heuristic)
Glover and Laguna [32, 33]	Tabu search algorithm
Hertz, Taillard and de Werra [37]	Tabu search algorithm
Cordeau, Gendreau, Hertz, Laporte and Sormany [17]	Tabu search algorithm
Cordeau and Laporte [18]	Tabu search algorithm
Gendreau, Iori, Laporte and Martello [29]	Tabu search algorithm (neighborhood: customer removing and insertion)
Gendreau, Iori, Laporte and Martello [28]	Tabu search algorithm (neighborhood: customer removing and insertion)
Doerner, Fuellerer, Hartl, Gronalt and Iori [20]	Tabu search algorithm (neighborhood: customer removing and insertion)
Zhu, Qin, Lim and Wang [60]	Tabu search algorithm (neighborhood: 2-opt operator, 2-swap operator, move operator, crossover operator, splitting operator)

Table 7.2: Overview of the literature on SDVRP

Author(s)	Solution methodology and remarks
Dror and Trudeau [22]	Two-stage algorithm: 1. VRP solution construction 2. SDVRP solution construction (k-split interchange subroutine, route addition subroutine)
Archetti, Savelsbergh and Speranza [3]	Estimations about the benefits of allowing split deliveries
Belenguer, Martinez and Mota [7]	Cutting-plane algorithm
Dror, Laporte and Trudeau [21]	Two-stage algorithm: 1. Simplex algorithm 2. Branch-and-bound algorithm
Desaulniers [19]	Branch-and-cut-and-price algorithm and exact dynamic programming algorithm
Ho and Haugland [39]	Tabu search algorithm (neighborhood: split delivery relocate operator, split delivery relocate split operator, customers exchange operator, 2-opt operator)
Archetti, Speranza and Hertz [4]	Tabu search algorithm (neighborhood: customer removing and insertion, customer insertion without removing)
Frizzell and Giffin [26]	Heuristic algorithm (look-ahead approach, removing and insertion operator, exchange operator)
Gulczynski, Golden and Wasil [35]	Integer programming-based heuristic
Asbach, Dorndorf and Pesch [5]	Neighborhood algorithm
Schmid, Doerner, Hartl, Savelsbergh and Stoecher [52]	Hybrid method (multi-commodity network flow algorithm, variable neighborhood search algorithm)
Schmid, Doerner, Hartl and Salazar-González [51]	Hybrid method (variable neighborhood search algorithm, very large neighborhood search algorithm)
Mullaseril, Dror and Leung [47]	Heuristic algorithm (2-opt heuristic, k-split interchange subroutine)
Sierksma and Tijssen [53]	Heuristic algorithm (column generation, rounding procedure, cluster-and-route procedure, 1-opt heuristic, 2-opt heuristic)

duration of the deliveries while avoiding violations of time windows constraints, additional objectives are also considered such as avoiding the use of different vehicles to satisfy the demand of one customer, or minimizing the total number of vehicles used to make the deliveries.

7.3 Structure of the second part

After having described in this chapter a cement delivery problem and presented the state of the art of solution methods related to neighboring problems (cf. Tables 7.1 and 7.2 for an overview of the literature), the following chapters focus on the development and performance analysis of solution methods provided by the optimization module. For this purpose, exact and approached solution methods intended to solve the cement delivery problem are first described in Chapter 8. The implementation of these methods in the optimization module, their application on real life instances, and the analysis of the obtained results including main criticisms are then discussed in Chapter 9.

Chapter 8

Problem modeling and solution methods

To develop the solution methods provided by the optimization module, it is beforehand necessary to model the cement delivery problem described in Chapter 7. For this purpose, a simplified version of the problem is first defined and formulated as an integer linear program in Section 8.1. In order to reflect reality, additional constraints are presented in Section 8.2. Four exact and approached solution methods including these constraints are then described in Section 8.3. Among these solution methods, three methods initially discussed in [57] and [38] use integer linear programming, while one method implements a tabu search. A conclusion is finally given in Section 8.4.

8.1 Basic Model

A definition of the cement delivery problem that does not include all the characteristics of the problem presented in Section 7.1 of previous chapter is first presented in Subsection 8.1.1. This problem is then modeled using integer linear programming in Subsection 8.1.2. A solution related to an instance of this basic problem is illustrated in Figure 8.1.

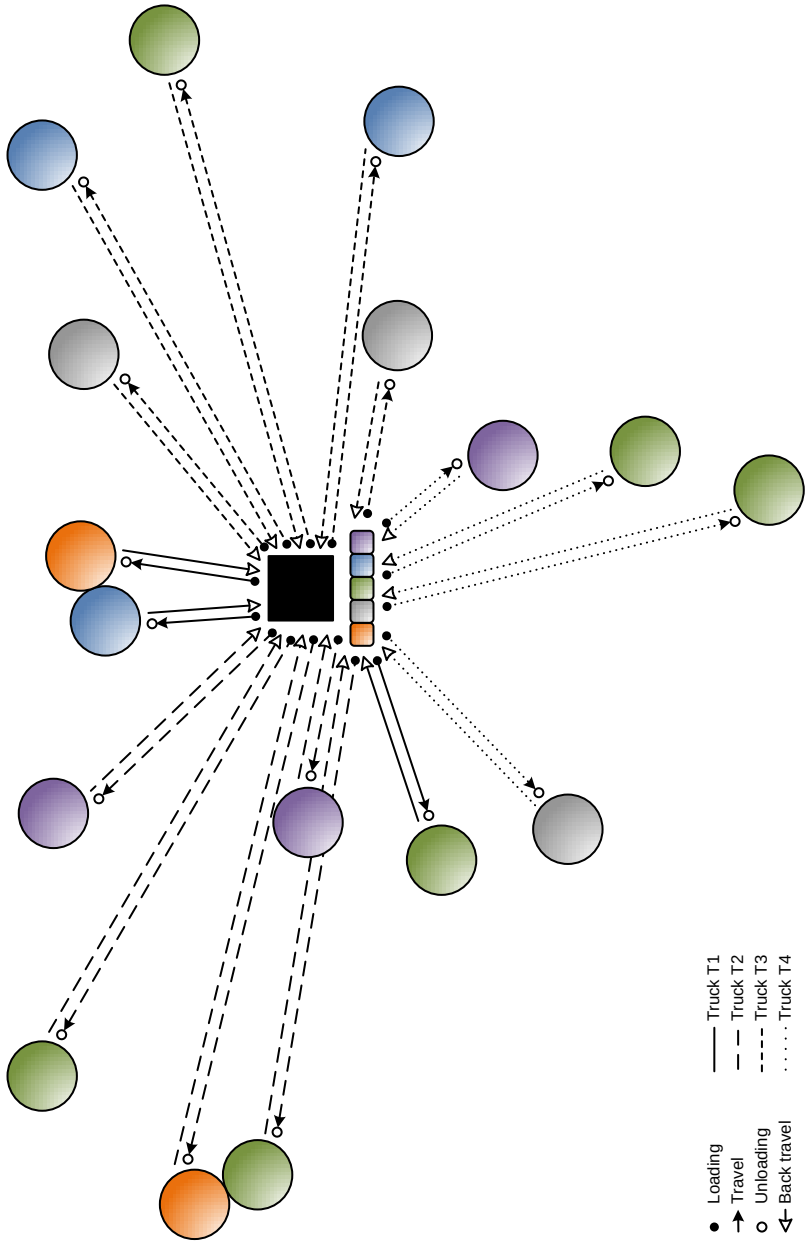

Figure 8.1: Basic cement delivery problem

8.1.1 Problem definition

A cement factory has an heterogeneous fleet V of vehicles that can be used to deliver cement to a set C of customers. Each vehicle $k \in V$ has a capacity Q_k and must start and end its daily activity at the main depot. A set I of orders is considered, each order $i \in I$ being characterized by a non-negative quantity d_i and a customer $c_i \in C$ to which d_i units of cement must be delivered. For a customer $c \in C$, let I_c denote the subset of orders i with $c_i = c$. The demands of the customers are typically larger than the capacity of the vehicles, which means that most customers must be visited several times to satisfy their demands.

When a vehicle $k \in V$ makes multiple deliveries to a customer for a same order $i \in I$, these have to appear consecutively on the vehicle route. It may also occur that the demand d_i associated with a given order $i \in I$ is small enough to be supplied with only one delivery. In such a case, a vehicle $k \in V$ with capacity Q_k strictly smaller than d_i should not be used for the delivery. Also, some vehicles cannot transport some types of cement, which explains why they cannot make some deliveries. To take such constraints into account, let V_i denote the subset of vehicles which can be used to deliver cement for order $i \in I$, and let I_k denote the subset of orders that vehicle k can handle.

No vehicle can transport cement for two different orders at the same time, which means that after a delivery to a customer, the vehicle has to travel back to the depot to load cement for the next delivery.

Loading times depend on the type of cement, and on the vehicle used. Unloading times depend on the customer where cement has to be delivered, on the type of cement, and on the vehicle used. Note that an order is related to a cement type and to a customer. Let L_{ik} denote the time needed by vehicle $k \in V$ to load cement for order $i \in I$, and let U_{ik} denote the time it needs to unload that cement at customer c_i. Travel speed depends on the vehicle used and on whether or not it is loaded. Let TL_k (respectively, TU_k) denote the time needed by vehicle $k \in V$ to travel one kilometer when it is loaded (respectively, unloaded). Also, let δ_c denote the distance from customer c to the depot. The distances are supposed to be symmetrical, which means that δ_c is also the distance from the depot to customer c. Each time unit spent by a vehicle $k \in V$ for loading, unloading or traveling has a fixed cost F_k. This cost takes into account a tax for carbon dioxide emissions as well as the use of vehicles rented by the factory. Some vehicles are possibly not available during portions of the day (for maintenance or other reasons). Let A_k denote the total availability (in time units) of vehicle $k \in V$.

Finally, it is required that no more than Nc different vehicles are used to satisfy all orders of a same customer. The cement factory can also impose that no more than Nk vehicles are used to perform all deliveries.

While a natural objective is to determine a set of vehicle routes with minimum total cost, the factory possibly has another objective which is, for example, to avoid situations where customers are supplied by more than one vehicle.

8.1.2 Basic integer linear program

The above problem can be formulated with an integer linear program. For this purpose, the variables of the proposed model are first described, starting with those related with the assignment of deliveries to the vehicles. For every order $i \in I$ and every vehicle $k \in V$, let

$$\begin{aligned} n_{ik} &= \text{number of deliveries made by vehicle } k \text{ for order } i \\ x_{ik} &= \begin{cases} 1 & \text{if at least one delivery is made for order } i \text{ by vehicle } k \\ 0 & \text{otherwise.} \end{cases} \end{aligned}$$

For every customer $c \in C$ and every vehicle $k \in V$, let

$$v_{ck} = \begin{cases} 1 & \text{if vehicle } k \text{ makes at least one delivery to customer } c \\ 0 & \text{otherwise.} \end{cases}$$

Also, for every vehicle $k \in V$, let

$$w_k = \begin{cases} 1 & \text{if vehicle } k \text{ makes at least one delivery} \\ 0 & \text{otherwise.} \end{cases}$$

As mentioned in Subsection 8.1.1, multiple deliveries made by a vehicle for an order $i \in I$ have to appear consecutively on its route. It is therefore sufficient to sequence the orders handled by a vehicle to determine its route. The following second set of variables is defined for this purpose. For each ordered pair (i, i') of elements in $I \cup \{0\}$ (where 0 denotes the depot), let

$$s^k_{ii'} = \begin{cases} 1 & \text{if } i' \text{ is the immediate successor of } i \text{ on the route of vehicle } k \\ 0 & \text{otherwise.} \end{cases}$$

Finally, for every $k \in V$ and every $i \in I \cup \{0\}$, let

$$t^k_i = \begin{cases} \text{position of } i \text{ in the route of vehicle } k & \text{if } i \in I \text{ and } x_{ik} = 1 \\ 0 & \text{otherwise.} \end{cases}$$

The constraints of the model are now defined, starting with those related with the assignment of deliveries to the vehicles. When a vehicle k delivers cement for an order i, it is known that n_{ik} deliveries are made. To compute the total time (in time units) that vehicle k needs to make all its deliveries, the loading time at the depot, the unloading time at c_i, the loaded travel time from the depot to c_i and the unloaded travel time from c_i to the depot are summed up for each delivery associated with order i. In summary, let

$$R_{ik} = L_{ik} + U_{ik} + \delta_{c_i}(TL_k + TU_k).$$

Constraints (8.1) are first imposed to ensure that no vehicle $k \in V$ is used for more that A_k time units.

$$\sum_{i \in I} R_{ik} n_{ik} \leq A_k \qquad \forall k \in V. \tag{8.1}$$

Constraints (8.2) are then imposed to ensure that all demands are satisfied.

$$\sum_{k \in V_i} Q_k n_{ik} \geq d_i \qquad \forall i \in I. \tag{8.2}$$

Constraints (8.3) impose that each order $i \in I$ is delivered by vehicles that can perform such deliveries.

$$\sum_{k \notin V_i} n_{ik} = 0 \qquad \forall i \in I. \tag{8.3}$$

Let $N_{ik} = \lceil d_i / Q_k \rceil$ denote the number of deliveries needed to satisfy order $i \in I$ if all these deliveries are performed by vehicle $k \in V$. Constraints (8.4) and (8.5) link variables x_{ik} with variables n_{ik}.

$$n_{ik} \geq x_{ik} \qquad \forall i \in I, \forall k \in V \tag{8.4}$$

$$n_{ik} \leq N_{ik} x_{ik} \qquad \forall i \in I, \forall k \in V. \tag{8.5}$$

Constraints (8.6) link variables x_{ik} with variables v_{ck} and constraints (8.7) ensure that no more than Nc vehicles are used to satisfy the demand of a customer.

$$v_{c_i k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V_i \tag{8.6}$$

$$\sum_{k \in V} v_{ck} \leq Nc \qquad \forall c \in C. \tag{8.7}$$

Constraints (8.8) link variables v_{ck} with variables w_k and constraints (8.9) impose that no more than Nk vehicles are used to make all deliveries.

$$w_k \geq v_{ck} \qquad \forall k \in V, \forall c \in C \tag{8.8}$$

$$\sum_{k \in V} w_k \leq Nk. \tag{8.9}$$

Constraints are now imposed on the delivery sequences. Constraints (8.10) and (8.11) impose that an order i has a successor and a predecessor on the route of vehicle k if and only if vehicle k makes at least one delivery for order i:

$$\sum_{\substack{i' \in I \cup \{0\} \\ i \neq i'}} s^k_{ii'} = x_{ik} \qquad \forall k \in V, \forall i \in I \tag{8.10}$$

$$\sum_{\substack{i' \in I \cup \{0\} \\ i \neq i'}} s^k_{i'i} = x_{ik} \qquad \forall k \in V, \forall i \in I. \tag{8.11}$$

Also, vehicle k leaves the depot and turns back to the depot if only if it makes at least one delivery, which is imposed by constraints (8.12) and (8.13):

$$\sum_{i \in I} s^k_{0i} = w_k \qquad \forall k \in V \tag{8.12}$$

$$\sum_{i \in I} s^k_{i0} = w_k \qquad \forall k \in V. \tag{8.13}$$

Since each route starts at the main depot, let

$$t^k_0 = 0 \qquad \forall k \in V. \tag{8.14}$$

Constraints (8.15) and (8.16) eliminate subtours:

$$0 \leq t^k_i \leq |I| \qquad \forall k \in V, \forall i \in I \tag{8.15}$$

$$t^k_i - t^k_{i'} + |I|\, s^k_{ii'} + (|I| - 2)\, s^k_{i'i} \leq |I| - 1 \;\forall k \in V, \forall i \in I \cup \{0\}, \forall i' \in I, i' \neq i. \tag{8.16}$$

Note that constraints (8.16) impose that $t^k_{i'}$ is strictly larger than t^k_i if vehicle k makes the deliveries for order i before those for order i'. But the deliveries made by vehicle k do not necessarily have consecutive positions. For example, if a problem has 10 orders and vehicle k handles only 3 of them, they may have positions

2, 7 and 9. The fact that these positions are different is however sufficient to forbid subtours.

Finally, constraints (8.17)–(8.22) define the domains of the decision variables:

$$n_{ik} \in \mathbb{N} \qquad \forall i \in I, \forall k \in V \tag{8.17}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in I, \forall k \in V \tag{8.18}$$

$$v_{ck} \in \{0,1\} \qquad \forall c \in C, \forall k \in V \tag{8.19}$$

$$w_k \in \{0,1\} \qquad \forall k \in V \tag{8.20}$$

$$t_i^k \in \mathbb{N} \qquad \forall i \in I \cup \{0\}, \forall k \in V \tag{8.21}$$

$$s_{ii'}^k \in \{0,1\} \qquad \forall i \in I \cup \{0\}, \forall i' \in I \cup \{0\}, \forall k \in V. \tag{8.22}$$

If the objective is to minimize the total cost of the vehicle routes, the following function has then to be minimized:

$$\sum_{i \in I} \sum_{k \in V_i} F_k R_{ik} n_{ik}. \tag{8.23}$$

under constraints (8.1)–(8.22).

The cement factory possibly has other objectives such as avoiding situations where more than one vehicle are used to supply the demands of a customer, or minimizing the number of vehicles used to make the deliveries. This can be achieved by minimizing $\sum_{c \in C} \sum_{k \in V} v_{ck} + \lambda \sum_{k \in V} w_k$, where λ is a parameter that gives more or less importance to the minimization of the number of vehicles used. Discrimination can then be done among solutions with the same objective value by choosing one with minimum total delivery time. For this purpose, a constant $M = \sum_{k \in V} F_k A_k$ is considered, which is obviously an upper bound on the value of a solution measured according to function (8.23). The following new objective function is then defined:

$$M(\sum_{c \in C} \sum_{k \in V} v_{ck} + \lambda \sum_{k \in V} w_k) + \sum_{i \in I} \sum_{k \in V_i} F_k R_{ik} n_{ik}. \tag{8.24}$$

and the problem to solve is to minimize objective (8.24) under constraints (8.1)–(8.22).

8.2 Additional Constraints

The real life cement delivery problem described in Section 7.1 of previous chapter has some additional constraints that make it necessary to extend the model described in the previous section. First of all, cement is supplied not only from a main depot, but also from local depots that typically correspond to railway stations where cement is brought by train. Let $D = \{0, \cdots, |D| - 1\}$ denote the set of depots where 0 denotes the main depot. The company maintains all stock levels at a value that is high enough to ensure that the total demand of the customers can be satisfied. This means that the quantity of cement available at each depot can be considered as unlimited. Some vehicles do not have the required equipment to load cement at the local depots, which means that the cement delivered by these vehicles necessarily comes from the main depot. Let V^L denote the set of vehicles which can load cement from local depots. Also, some types of cement cannot be loaded at some local depots. Let D_i denote the subset of depots where cement can be loaded for order $i \in I$.

The constraint that each vehicle must start and end its daily activity at the main depot is still valid, which means that the first delivery of each vehicle is loaded at the main depot (and not at a local depot), and every vehicle has to drive back to the main depot after its last delivery.

Loading times henceforth depend not only on the type of cement and on the vehicle used, but also on the depot where cement is loaded. Therefore, let L_{ijk} instead of L_{ik} (see Subsection 8.1.1) denote the time needed by vehicle $k \in V$ to load cement at depot $j \in D$ for order $i \in I$. Also, let δ_{cj} instead of δ_c denote the distance from customer c to depot j. All distances are supposed to be symmetrical, which means that δ_{cj} is also the distance from depot j to customer c.

To save time, some vehicles are 'preloaded' (loaded in advance) at the end of the previous day so that they can start their trip earlier in the next morning, without loosing time with the loading process. Some of these preloads can only be used for a subset of orders. A possible reason for this can be that a customer has to be delivered very early in the morning and it has to be sure that the preload is used to deliver that customer. Another reason can be that the preload contains a specific type of cement that does not correspond to some orders. Let $VP \subset V$ denote the subset of vehicles that are preloaded and by IP^k ($k \in VP$) the subset of orders $i \in I$ such that vehicle k can start its daily activity with a delivery for i.

Finally, it is required that no vehicle loads cement from more than Nd different depots.

8.3 Solution Methods

Vehicle routing problems are naturally decomposed into two components: assignment of customers to vehicle routes, and route construction. Such a decomposition, also called *cluster-first, route-second* is a standard technique used in heuristics for the solution of vehicle routing problems [42, 44]. A similar approach is used to solve this cement delivery problem. A first solution method is proposed in Subsection 8.3.1 where the problem is decomposed into three subproblems solved sequentially. Each of these subproblems is formulated as a mixed integer linear programming problem. A second solution method is proposed in Subsection 8.3.2 where two subproblems of the first solution method are merged. The problem is therefore only decomposed into two subproblems also solved sequentially. A third solution method that solves the problem with a unique mixed integer linear program is proposed in Subsection 8.3.3. After the use of exact methods, a metaheuristic is proposed in Subsection 8.3.4. Indeed, a tabu search is implemented in order to solve the problem in a single step.

8.3.1 Solving the problem in three phases with MILP

This first solution method solves the cement delivery problem in three phases. After the allocation of each order to a specific depot, a subset of deliveries is assigned to each vehicle (*cluster-first*). The sequence of deliveries is then determined on each vehicle route (*route-second*).

Allocation of each order to a specific depot

In addition to the variables specified in Section 8.1, the following one is defined for every order $i \in I$ and every depot $j \in D_i$:

$$u_{ij} = \begin{cases} 1 & \text{if order } i \text{ is allocated to depot } j \\ 0 & \text{otherwise.} \end{cases}$$

To define the total time (in time units) that a vehicle needs to load and transport a delivery associated with order i from depot j, the loading time at the depot, the loaded travel time from the depot to c_i and the unloaded travel time from c_i to the depot are first summed up for each available vehicle. The minimum of these

durations is then selected. In summary, let

$$P_{ij} = \min_{k \in V}\{L_{ijk} + \delta_{c_i j}(TL_k + TU_k)\}.$$

Constraints (8.25) and (8.26) impose that each order $i \in I$ is assigned to only one and appropriate depot $j \in D_i$:

$$\sum_{j \in D_i} u_{ij} = 1 \qquad \forall i \in I \tag{8.25}$$

$$\sum_{j \notin D_i} u_{ij} = 0 \qquad \forall i \in I. \tag{8.26}$$

Constraints (8.27) define the domain of decision variables u_{ij}:

$$u_{ij} \in \{0, 1\} \quad \forall i \in I, \forall j \in D_i. \tag{8.27}$$

To minimize the total travel time between the customers and the depot allocated to each of their orders, the following function has to be minimized:

$$\sum_{i \in I} \sum_{j \in D_i} P_{ij} u_{ij}. \tag{8.28}$$

To summarize, the problem of allocating the most appropriate depot to each order can be solved by minimizing objective (8.28) under constraints (8.25)–(8.27).

Assignment of deliveries to the vehicles

For every preloaded vehicle $k \in VP$ and every order $i \in IP^k$, let

$$y_{ik} = \begin{cases} 1 & \text{if preloaded vehicle } k \text{ makes its first delivery for order } i \\ 0 & \text{otherwise.} \end{cases}$$

Let Δ_i denote the depot $j \in D_i$ where cement has to be loaded for an order $i \in I$. For every depot $j \in D$ and every vehicle $k \in V^L$, let

$$z_{jk} = \begin{cases} 1 & \text{if vehicle } k \text{ makes at least one delivery for an order } i \text{ with } \Delta_i = j \\ 0 & \text{otherwise.} \end{cases}$$

When a vehicle k delivers cement for an order i, it is known that at least $n_{ik} - 1$ deliveries are made with cement loaded at depot Δ_i. If $k \in V^L$, the first delivery

by k for order i possibly comes from another depot, depending on the location of the previous customer visited by k. But since the sequence of deliveries on each vehicle route is not known yet, the origin of the first delivery by k for order i cannot be determined at this stage. Also, at the end of the n_{ik} deliveries, the next destination of vehicle k cannot yet be determined. To estimate the total time that vehicle k needs to make all its deliveries, it is assumed that all deliveries for order i are made from Δ_i and that the vehicle travels back to Δ_i when all its deliveries for order i are accomplished. Hence, the loading time at depot Δ_i, the unloading time at c_i, the loaded travel time from Δ_i to c_i and the unloaded travel time from c_i to Δ_i are summed up for each delivery associated with order i. All this is in fact only valid for a vehicle $k \in V^L$ that can load cement at local depots. For a vehicle $k \notin V^L$, the main depot is used to compute the exact time (i.e., not an estimation) needed by k for its deliveries. In summary, the definition of R_{ik} given in Subsection 8.1.2 is replaced by the following one:

$$R_{ik} = \begin{cases} L_{i\Delta_i k} + U_{ik} + \delta_{c_i\Delta_i}(TL_k + TU_k) & \text{if } k \in V^L \text{ and } \Delta_i \neq 0 \\ L_{i0k} + U_{ik} + \delta_{c_i 0}(TL_k + TU_k) & \text{otherwise.} \end{cases}$$

Constraints (8.1) impose that no vehicle $k \in V$ is used for more that A_k time units. As mentioned above, R_{ik} is an estimation of the time needed by k to make a delivery for order i when $k \in V^L$. Hence, if the total estimated delivery time of vehicle k is not larger than A_k, it may happen that the real delivery time is strictly larger than A_k. To avoid such situations, the availability time of all vehicles $k \in V^L$ is reduced by using a weighing factor. A parameter $\theta \in [0,1]$ is therefore considered and constraints (8.1) are replaced by the following constraints:

$$\sum_{i \in I} R_{ik} n_{ik} \leq (1-\theta) A_k \qquad \forall k \in V^L \tag{8.29}$$

$$\sum_{i \in I} R_{ik} n_{ik} \leq A_k \qquad \forall k \notin V^L. \tag{8.30}$$

In addition to constraints (8.2)–(8.9) constraints (8.31) are imposed to ensure that each preloaded vehicle $k \in VP$ performs its first delivery for an order $i \in IP^k$, and constraints (8.32) to link variables x_{ik} with variables y_{ik}.

$$\sum_{i \in IP^k} y_{ik} = 1 \qquad \forall k \in VP \tag{8.31}$$

$$y_{ik} \leq x_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{8.32}$$

Also, constraints (8.33) link variables x_{ik} with variables z_{jk}, while constraints (8.34) ensure that no vehicle in V^L loads cement from more than Nd different depots. Note that constraints (8.34) are possibly too restrictive. Indeed, if a vehicle k makes only one delivery for an order i (i.e., $n_{ik} = 1$), then $z_{\Delta_i k} = 1$ while vehicle k possibly does not load any cement from depot Δ_i. Such a situation is however not frequent since the demands of the customers are typically larger than the capacity of the vehicles, which means that most vehicles will perform multiple deliveries for a same order.

$$z_{\Delta_i k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V^L \cap V_i \tag{8.33}$$

$$\sum_{j \in D} z_{jk} \leq Nd \qquad \forall k \in V^L. \tag{8.34}$$

Constraints (8.35) and (8.36) define the domains of decision variables y_{ik} and z_{jk}:

$$y_{ik} \in \{0,1\} \quad \forall i \in IP^k, \forall k \in VP \tag{8.35}$$

$$z_{jk} \in \{0,1\} \quad \forall j \in D, \forall k \in V^L. \tag{8.36}$$

To summarize, the problem of assigning deliveries to the vehicles can be solved by minimizing objective (8.23) or (8.24) under constraints (8.2)–(8.9), (8.17)–(8.20) and (8.29)–(8.36). The complete mixed integer linear programming model of this problem is presented in Subsection D.1.2 of Appendix D.

Sequencing the orders on each route

Assuming that a set of deliveries has been assigned to each vehicle, the problem of determining the sequence of deliveries on each route is a traveling salesman problem that has to be solved for each vehicle $k \in V$. Indeed, let S_k denote the set of orders handled by vehicle k (i.e., $n_{ik} > 0$ if and only if $i \in S_k$) and consider $S'_k = S_k \cup \{0\}$, where 0 stands for the main depot. If i and i' are consecutive orders on the route of vehicle k, and if depot $j \in D_{i'}$ is chosen for the first delivery to $c_{i'}$, the unloaded travel time from c_i to j, the loading time at j, and the loaded travel time from j to $c_{i'}$ have to be added up, which gives a total time of $TU_k \delta_{c_i j} + L_{i'jk} + TL_k \delta_{c_{i'} j}$. This is to be compared with the estimation $TU_k \delta_{c_i \Delta_i} + L_{i'\Delta_{i'} k} + TL_k \delta_{c_{i'} \Delta_{i'}}$ obtained from R_{ik} and $R_{i'k}$. The estimation error due to the choice of j instead of $\Delta_{i'}$ is then the difference between these two times, and the depot $j \in D_{i'}$ with smallest error is chosen. Formally, this estimation error, denoted $T^k_{ii'}$, is defined as follows:

$$T^k_{ii'} = \min_{j \in D_{i'}} \{TU_k(\delta_{c_i j} - \delta_{c_i \Delta_i}) + (L_{i'jk} - L_{i'\Delta_{i'} k}) + TL_k(\delta_{c_{i'} j} - \delta_{c_{i'} \Delta_{i'}})\}.$$

Constraints (8.10) and (8.11) that impose a successor and a predecessor to each order i on the route of vehicle k can be simplified as follows:

$$\sum_{\substack{i' \in S'_k \\ i \neq i'}} s^k_{ii'} = 1 \qquad \forall k \in V, \forall i \in S'_k \tag{8.37}$$

$$\sum_{\substack{i' \in S'_k \\ i \neq i'}} s^k_{i'i} = 1 \qquad \forall k \in V, \forall i \in S'_k. \tag{8.38}$$

Every preloaded vehicle $k \in VP$ makes its first delivery for the order $i \in IP^k$ with $y_{ik} = 1$. This is imposed by the following constraints:

$$s^k_{0i} \geq y_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{8.39}$$

In order to forbid subtours, constraints (8.15) and (8.16) can be replaced by the following ones which are more precise:

$$0 \leq t^k_i \leq |S_k| \qquad \forall k \in V, \forall i \in S_k \tag{8.40}$$

$$t^k_i - t^k_{i'} + |S_k|\, s^k_{ii'} + (|S_k| - 2)\, s^k_{i'i} \leq |S_k| - 1 \;\forall k \in V, \forall i \in S'_k, \forall i' \in S_k, i \neq i'. \tag{8.41}$$

As mentioned before, function (8.23) only estimates the total delivery time of the vehicles. To minimize the real delivery time, the estimation error which is defined with the following function has to be minimized:

$$\sum_{k \in V} \sum_{i \in S'_k} \sum_{\substack{i' \in S'_k \\ i \neq i'}} T^k_{ii'} s^k_{ii'}. \tag{8.42}$$

In summary, the problem of finding the best sequence of orders on each route can be solved by minimizing objective (8.42) under constraints (8.37)–(8.41), (8.21), (8.22), and (8.35). $|V|$ independent problems can of course be solved, one for each k. The complete mixed integer linear programming model of this problem is presented in Subsection D.1.3 of Appendix D.

8.3.2 Solving the problem in two phases with MILP

This second solution method solves the cement delivery problem in two phases. A subset of deliveries is first assigned to each vehicle (*cluster-first*), and then the

sequence of deliveries is determined on each vehicle route (*route-second*). In this solution method, the allocation of each order to a specific depot also depends on the vehicle that is used for the deliveries. These allocations are no more done using a mixed integer linear program, but are determined separately and given as data of the first subproblem. The most significant change compared to the first solution method is in the modeling improvement of the estimated delivery time needed by any vehicle $k \in V^L$, which can load cement at local depots.

For a vehicle $k \in V$ and an order $i \in I_k$, let Δ_{ik} denote the depot $j \in D_i$ that minimizes $L_{ijk} + \delta_{c_i j} T L_k$. In words, Δ_{ik} is a depot where cement can be loaded by k for order i, and it is chosen in D_i so that it minimizes the sum of the loading time and the travel time to c_i.

As mentioned before, multiple deliveries made by vehicle k for a same order i have to appear consecutively on its route. It is always optimal, except possibly for the first delivery, to load the required cement at the depot Δ_{ik}. The use of another depot for the first delivery can however shorten the route. This is illustrated with the following example. Assume that the loading times are insignificant and consider a vehicle $k \in V^L$ that has to perform only two deliveries, one to customer A and then one to customer B. Assume that there are three local depots where cement can be loaded for these two deliveries: the local depot 1 is at distance 1 from A and 5 from B, the local depot 2 is at distance 2 from A and B, and the local depot 3 is at distance 5 from A and 1 from B. Also, it is assumed that the main depot is at distance 3 from A and B. In the shortest vehicle route, customer A receives cement loaded at the main depot while the cement delivered to B is loaded at the local depot 2. If the vehicle has to perform two deliveries to A and two to B, then the second delivery to A is made from the local depot 1 which is the closest to A, while the second delivery to B is made from the local depot 3 which is the closest to B. This is illustrated in Figure 8.2.

Assignment of deliveries to the vehicles

As mentioned before, the depot where cement can be loaded for order i also depends in this solution method on vehicle k that is used. For every depot $j \in D$ and every vehicle $k \in V^L$, the variable z_{jk} is therefore redefined as follows:

$$z_{jk} = \begin{cases} 1 & \text{if vehicle } k \text{ makes at least one delivery for an order } i \text{ with } \Delta_{ik} = j \\ 0 & \text{otherwise.} \end{cases}$$

The definitions of R_{ik} and $T^k_{ii'}$ given in Subsection 8.3.1 are also replaced by

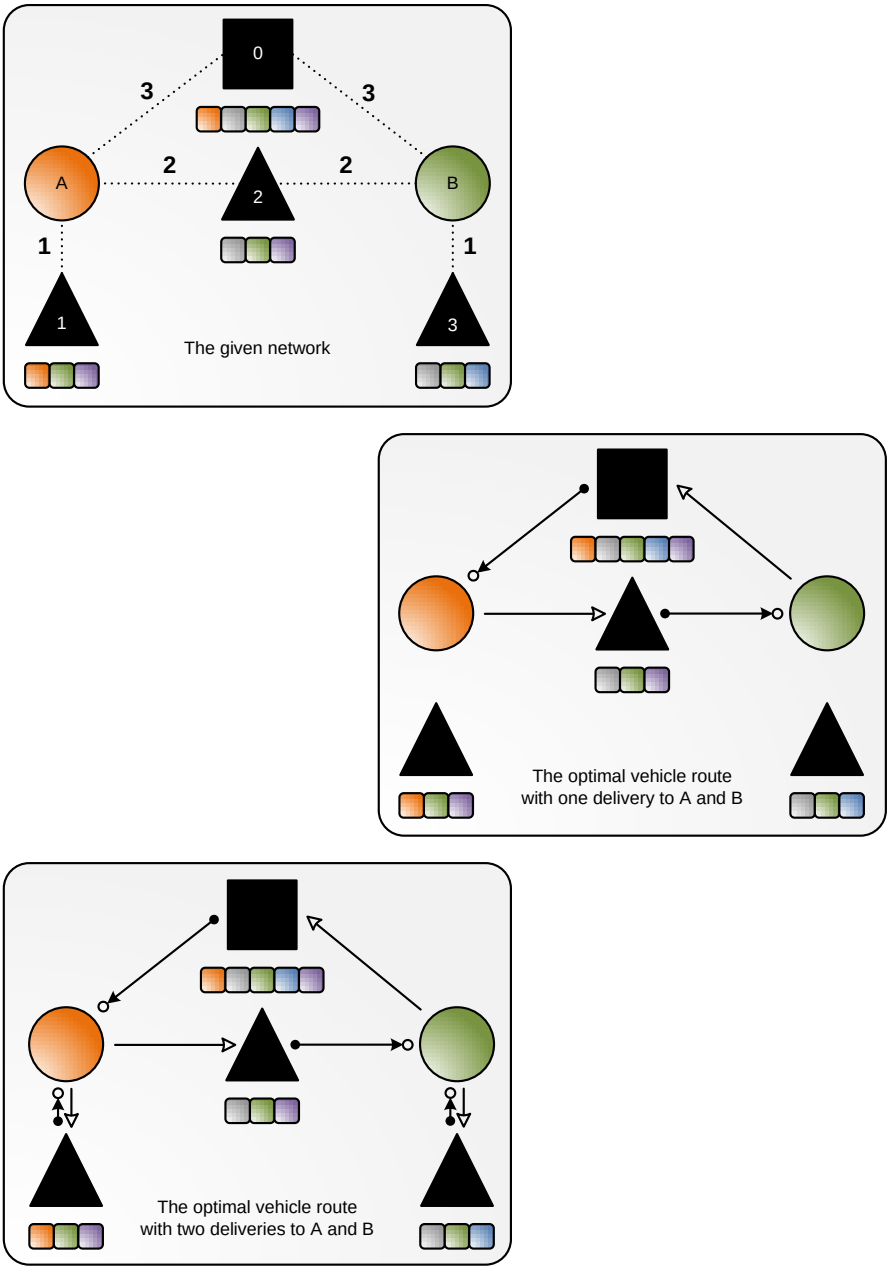

Figure 8.2: Optimal vehicle routes

the following ones:

$$R_{ik} = \begin{cases} L_{i\Delta_{ik}k} + U_{ik} + \delta_{c_i\Delta_{ik}}(TL_k + TU_k) & \text{if } k \in V^L \text{ and } \Delta_{ik} \neq 0 \\ L_{i0k} + U_{ik} + \delta_{c_i 0}(TL_k + TU_k) & \text{otherwise.} \end{cases}$$

$$T^k_{ii'} = \min_{j \in D_{i'}} \{TU_k(\delta_{c_i j} - \delta_{c_i\Delta_{ik}}) + (L_{i'jk} - L_{i'\Delta_{i'k}k}) + TL_k(\delta_{c_{i'}j} - \delta_{c_{i'}\Delta_{i'k}})\}.$$

To avoid situations where the real delivery time is strictly larger than A_k, constraints (8.29) reduce the availability time of all vehicles $k \in V^L$ with the use of a weighing factor. A more sophisticated approach is modeled in this second solution method in order to better reflect reality. This is done by bounding the estimation error. More precisely, assume that i is the first order in the route of vehicle k. The distance $\delta_{c_i 0}$ from the main depot to c_i is possibly strictly larger than the distance $\delta_{c_i\Delta_{ik}}$ used in the estimation. Also, the real loading time at the main depot is L_{i0k} if vehicle k is not loaded in advance (i.e., $k \notin VP$) and 0 otherwise, while the estimated loading time is $L_{i\Delta_{ik}k}$. In summary, assuming that i is the first order on the route of vehicle k, the estimation error T^k_{0i} due to the first delivery to i is defined as follows:

$$T^k_{0i} = \begin{cases} -L_{i\Delta_{ik}k} + TL_k(\delta_{c_i 0} - \delta_{c_i\Delta_{ik}}) & \text{if } k \in VP \\ (L_{i0k} - L_{i\Delta_{ik}k}) + TL_k(\delta_{c_i 0} - \delta_{c_i\Delta_{ik}}) & \text{if } k \notin VP. \end{cases}$$

Similarly, if i is the last order on the route of vehicle k, the estimation error T^k_{i0} due to the travel back from c_i to the main depot is defined as follows:

$$T^k_{i0} = TU_k(\delta_{c_i 0} - \delta_{c_i\Delta_{ik}}).$$

Finally, the third estimation error is on the time separating the last delivery for an order i and the first delivery for the next order i' on the route of vehicle k, and corresponds to $T^k_{ii'}$. The total estimation error on the time needed by vehicle k to make all its deliveries can however not be computed since the sequence of deliveries is not known yet. The following value E^k_1 is an upper bound on the estimation error due to the first delivery performed by vehicle k and to its last travel back to the depot:

$$E^k_1 = w_k(\max_{i \in I_k} T^k_{0i} + \max_{i \in I_k} T^k_{i0}).$$

Also, since there are exactly $\sum_{i \in I_k} x_{ik} - w_k$ pairs or consecutive orders on the route of vehicle k, the following value E^k_2 is an upper bound on the estimation error due

to consecutive orders:

$$E_2^k = \begin{cases} (\sum_{i \in I_k} x_{ik} - w_k) \max_{\substack{i,i' \in I_k \\ i \neq i'}} T_{ii'}^k & \text{if } |I_k| > 1 \\ 0 & \text{otherwise.} \end{cases}$$

In summary, $E_1^k + E_2^k$ is an upper bound on the total estimation error. Hence, by limiting the estimated delivery time of vehicle k to $A_k - E_1^k - E_2^k$, it is known that the real delivery time will not be larger than A_k. Such a reduction of the total availability of vehicle k is however possibly too restrictive since the real error can be much smaller that the upper bound. A parameter $\theta \in [0,1]$ is therefore considered and constraints (8.29) are replaced by the following constraints:

$$\sum_{i \in I} R_{ik} n_{ik} \leq A_k - \theta(E_1^k + E_2^k) \qquad \forall k \in V^L. \tag{8.43}$$

Constraints (8.33) that link variables x_{ik} with variables z_{jk} are replaced in this solution method by constraints (8.44).

$$z_{\Delta_{ik}k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V^L \cap V_i. \tag{8.44}$$

To summarize, the problem of assigning deliveries to the vehicles can be solved by minimizing objective (8.23) or (8.24) under constraints (8.2)–(8.9), (8.17)–(8.20), (8.30)–(8.32), (8.34)–(8.36), (8.43), and (8.44). The complete mixed integer linear programming model of this problem is presented in Subsection D.2.1 of Appendix D.

Sequencing the orders on each route

The problem of finding the best sequence of orders on each route is identical as the one described in Subsection 8.3.1 for the first solution method. Indeed, it can be solved by minimizing objective (8.42) under constraints (8.37)–(8.41), (8.21), (8.22), and (8.35). $|V|$ independent problems can of course be solved, one for each k. The complete mixed integer linear programming model of this problem is presented in Subsection D.2.2 of Appendix D.

8.3.3 Solving the problem in one phase with MILP

As already mentioned, by adding the total estimated delivery time (8.23) and the total estimation error (8.42), one gets the real total delivery time. For vehicle k,

this real delivery time, denoted α_k, is defined as follows:

$$\alpha_k = \sum_{i \in I} R_{ik} n_{ik} + \sum_{i \in I \cup \{0\}} \sum_{\substack{i' \in I \cup \{0\} \\ i \neq i'}} T^k_{ii'} s^k_{ii'}.$$

Constraints (8.30) and (8.43) can now be replaced by the following constraints:

$$\alpha_k \leq A_k \qquad \forall k \in V. \tag{8.45}$$

and the new objective function that combines objectives (8.23) and (8.42) is simply

$$\sum_{k \in V} F_k \alpha_k. \tag{8.46}$$

In summary, the cement delivery problem can be solved by minimizing objective (8.46) under constraints (8.2)–(8.22), (8.31)–(8.32), (8.34)–(8.36), (8.39), (8.44), and (8.45).

Again, if the company wants to avoid situations where more than one vehicle are used to supply the demands of a customer, or if it is interested in minimizing the total number of vehicles used for the deliveries, the following objective function which is similar to function (8.24) can be used:

$$M(\sum_{c \in C} \sum_{k \in V} v_{ck} + \lambda \sum_{k \in V} w_k) + \sum_{i \in I} \sum_{k \in V_i} F_k \alpha_k. \tag{8.47}$$

The complete mixed integer linear programming model of this problem is presented in Section D.3 of Appendix D.

8.3.4 Solving the problem with Tabu Search

This fourth solution method solves the cement delivery problem with a tabu search algorithm based on the MILP formulation presented in Subsection 8.3.3. Described in Procedure 1, this metaheuristic is composed of three phases: an *initialization phase* where a feasible solution is first constructed, a *tabu search phase* where better solutions are generated and a *postoptimization phase* where the best solution may be improved.

Four parameters whose use will be described later are needed to run this tabu search algorithm: the maximal number of iterations $iter_{max}$, the size of a neighborhood of solutions n_N, the maximal size of the tabu list n_{TA} and the number

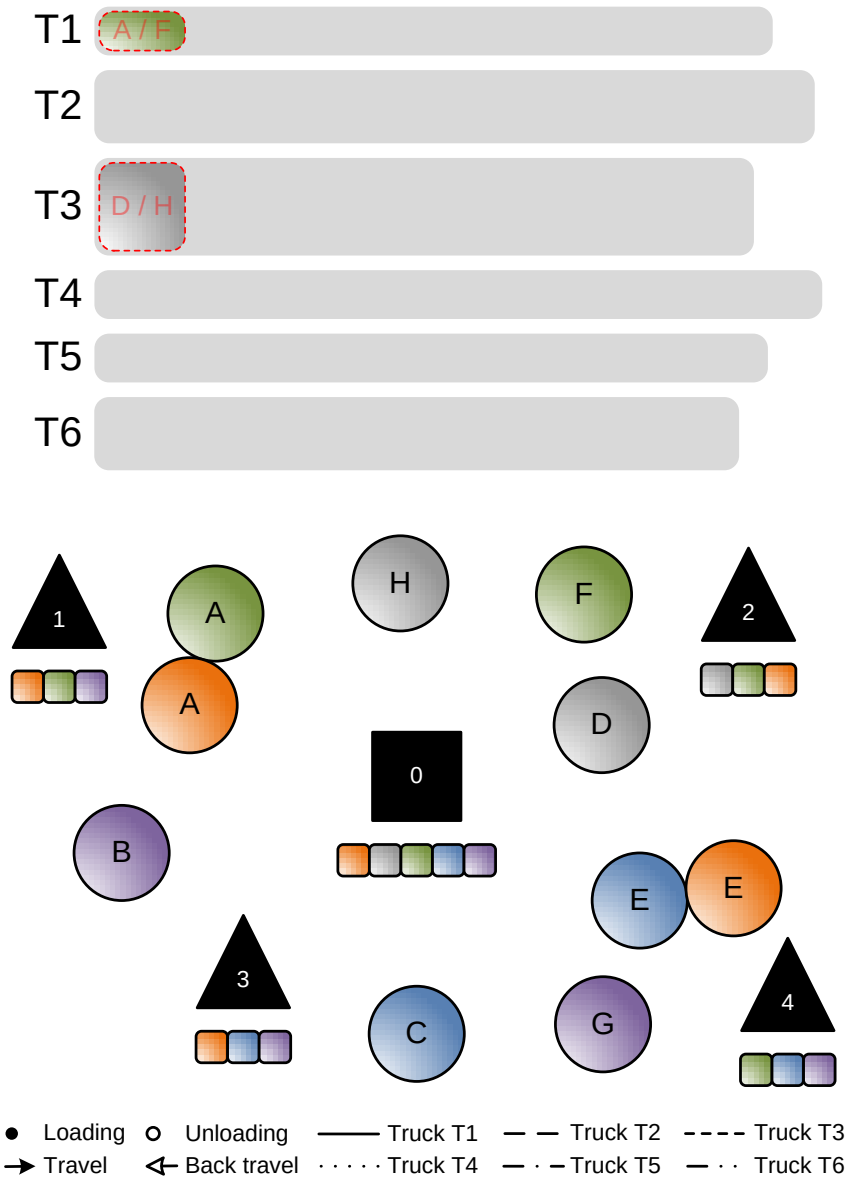

Figure 8.3: Construction of an initial solution
Initialization phase

Procedure 1 Tabu Search

1: **procedure** TABUSEARCH($iter_{max}, n_N, n_{TA}, iter_{mod}, [rt_{limit}]$)
2: $\mathfrak{s} \leftarrow$ CONSTRUCTINITIALSOLUTION
3: $\mathfrak{s} \leftarrow$ IMPROVESOLUTION($\mathfrak{s}$)
4: $\mathfrak{s}^* \leftarrow \mathfrak{s}$
5: $iter \leftarrow 0$
6: $TA \leftarrow \emptyset$
7: $iter_{best} \leftarrow 0$
8: **while** $iter - iter_{best} < iter_{max}$ [**and** rt_{limit} is not achieved] **do**
9: $iter \leftarrow iter + 1$
10: **if** $iter - iter_{best} \mod iter_{mod} = 0$ **then**
11: $\mathfrak{s} \leftarrow$ MODIFYSOLUTION($\mathfrak{s}$)
12: $\mathfrak{s} \leftarrow$ IMPROVESOLUTION($\mathfrak{s}$)
13: **end if**
14: $\{\mathfrak{s}, TA\} \leftarrow$ GENERATEBESTNEIGHBOR($\mathfrak{s}, TA, n_N, n_{TA}, \mathfrak{s}^*$)
15: **if** $f(\mathfrak{s}) < f(\mathfrak{s}^*)$ **then**
16: $\mathfrak{s}^* \leftarrow \mathfrak{s}$
17: $iter_{best} \leftarrow iter$
18: **end if**
19: **end while**
20: $\mathfrak{s}^* \leftarrow$ IMPROVESOLUTION($\mathfrak{s}^*$)
21: **return** $\mathfrak{s}^*$
22: **end procedure**

of iterations without modification of the current solution $iter_{mod}$. Note that the use of a run time limit rt_{limit} as fifth parameter is optional and therefore put in brackets.

a) Initialization phase

In this phase (cf. Procedure 1 from line 2 to line 7), an initial feasible solution $\mathfrak{s}$ is first constructed and possibly improved. Indeed, when constructing an initial feasible solution, the delivery sequences of orders are not necessarily optimal on the route of involved vehicles. An appropriate modification of the position of orders in these delivery sequences can result in time saving and can therefore provide a better flexibility during the *tabu search phase* to exchange deliveries between vehicles. This two steps will be discussed later when describing the corresponding procedures CONSTRUCTINITIALSOLUTION and IMPROVESOLUTION($\mathfrak{s}$).

Note that a solution $\mathfrak{s}$ contains not only information about the delivery sequences of every order $i \in I$ on the route of every vehicle $k \in V$, but also information about the number of deliveries made by every vehicle $k \in V$ for every

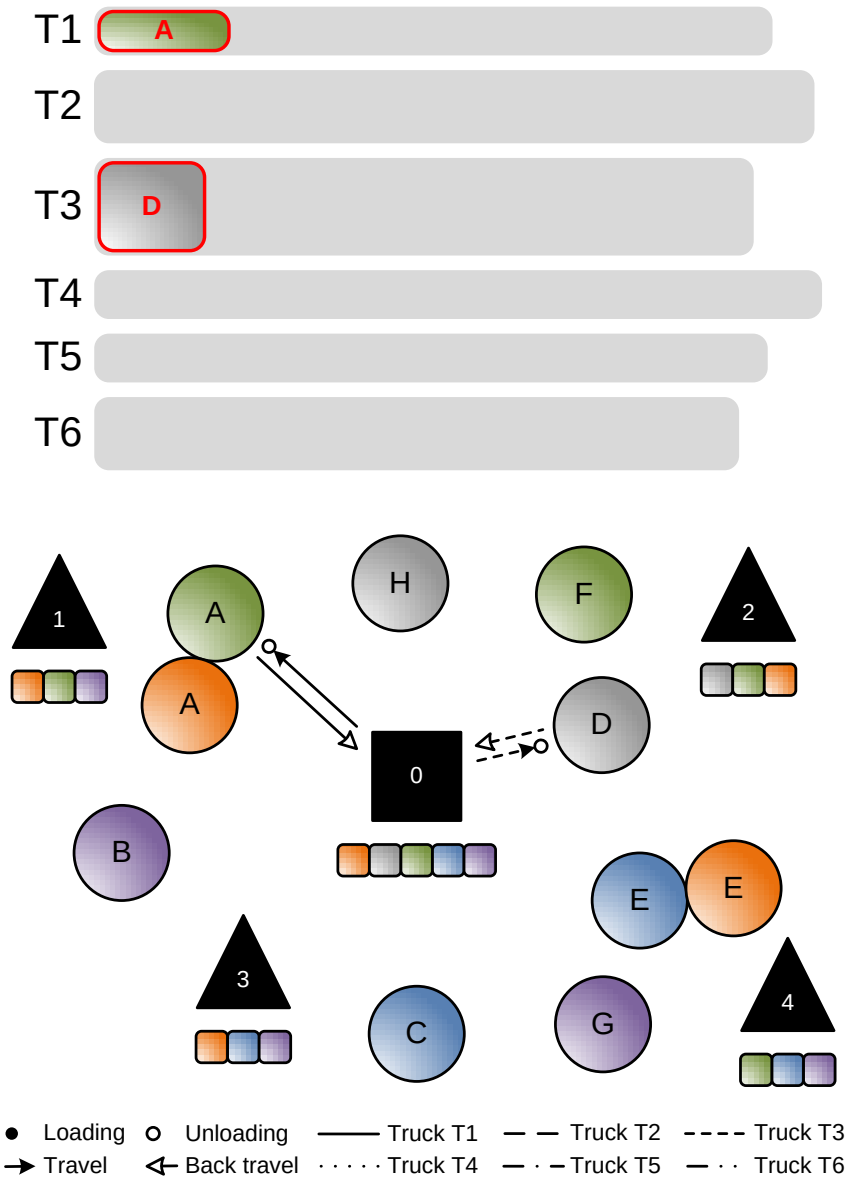

Figure 8.4: Construction of an initial solution
First construction phase

order $i \in I$:

$$\mathfrak{s} = \{s_{ii'}^{k} | k \in V, i \in I \cup \{0\}, i' \in I \cup \{0\} \text{ and } i' \neq i\} \cup \{n_{ik} | i \in I \text{ and } k \in V\}.$$

After these two steps, the possibly improved initial feasible solution $\mathfrak{s}$ is set as best solution $\mathfrak{s}^*$. Other variables needed by the *tabu search phase* are then initialized. Indeed, the current iteration $iter$ and the best iteration $iter_{best}$ are set to 0 while an empty tabu list TA is created. This list is intended to contain orders whose delivery insertion on the route of a vehicle is forbidden for a fixed number n_{TA} of iterations. For this purpose, tuples of orders and vehicles ($i \in I, k \in K$) indicating that the insertion of a delivery of order i on the route of vehicle k is forbidden will be stored in this list. Further precision on the use of this tabu list will be discussed later.

b) Tabu search phase

In this phase (cf. Procedure 1 from line 8 to line 19), the algorithm searches better solutions in the solutions space starting from the possibly improved initial feasible solution $\mathfrak{s}$. This search process continues while the best solution found is improved before the maximal number of iterations $iter_{max}$ and while the run time limit rt_{limit}, if mentioned, is not achieved.

If the best solution $\mathfrak{s}^*$ has not been improved for $iter_{mod}$ iterations, a part of the current solution $\mathfrak{s}$ is modified in order to diversify the search process. This step will be discussed in detail later when describing the procedure MODIFYSOLUTION($\mathfrak{s}$). This modified solution $\mathfrak{s}$ may then be improved with the procedure IMPROVESOLUTION($\mathfrak{s}$).

At each iteration, a neighborhood of solutions is generated from the current solution $\mathfrak{s}$ thanks to various operators and the best non-tabu solution of this neighborhood is set as new current solution $\mathfrak{s}$. Furthermore, the content of the tabu list TA is updated. If the best solution of the neighborhood is tabu, this tabu solution can however be accepted as current solution $\mathfrak{s}$ but only if this tabu solution is better than the best solution $\mathfrak{s}^*$. This is called aspiration. This step will be discussed in detail later when describing the procedure GENERATEBESTNEIGHBOR($\mathfrak{s}, TA, n_N, n_{TA}, \mathfrak{s}^*$).

If the objective function of the new current solution $f(\mathfrak{s})$ is better than the objective function of the best solution $f(\mathfrak{s}^*)$, this new current solution $\mathfrak{s}$ is set as best solution $\mathfrak{s}^*$ and the best iteration $iter_{best}$ is reinitialized as current iteration $iter$.

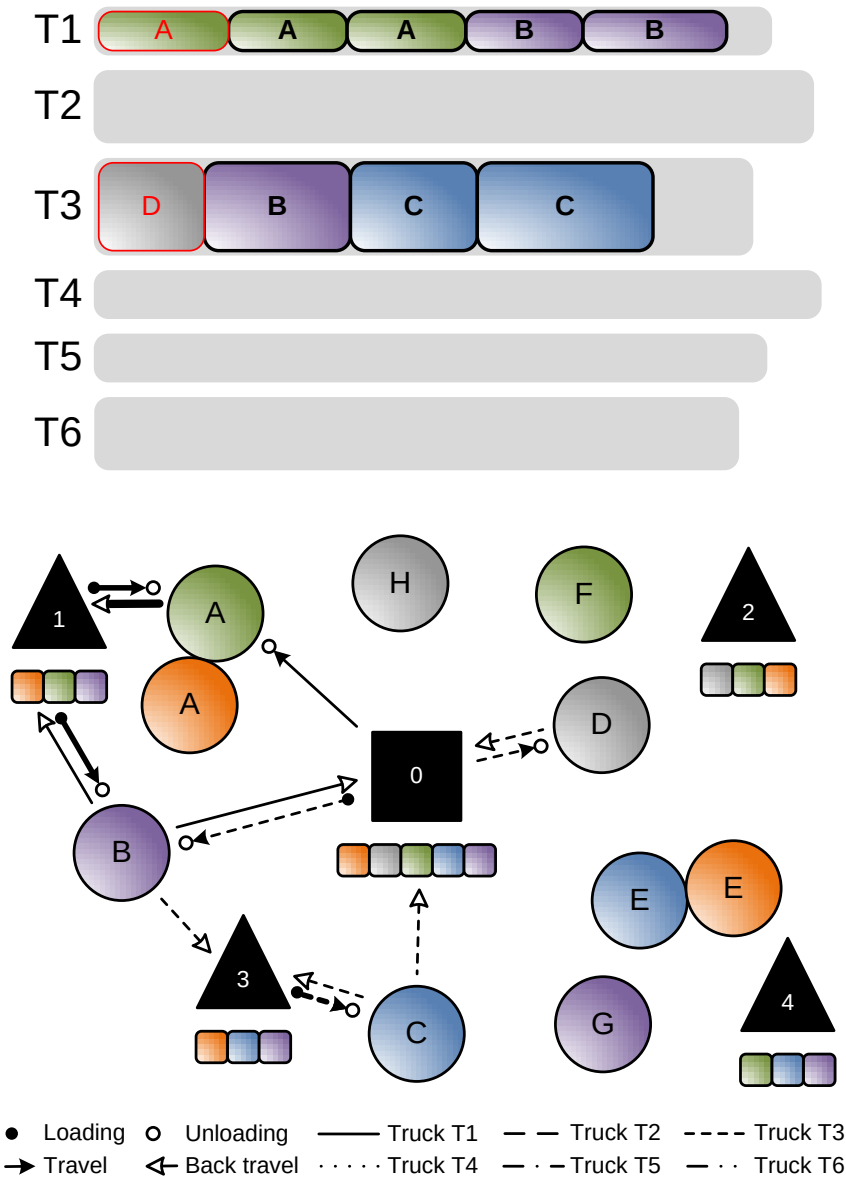

Figure 8.5: Construction of an initial solution
Second construction phase

Note that the objective function is similar to function (8.46). Again, if the company wants to avoid situations where more than one vehicle are used to supply the demands of a customer, or if it is interested in minimizing the total number of vehicles used for the deliveries, the objective function (8.47) can be used. However, v_{ck} indicating if vehicle $k \in V$ makes at least one delivery to customer $c \in C$ and w_k denoting if vehicle $k \in V$ makes at least one delivery have beforehand to be defined from n_{ik} of a solution $\mathfrak{s}$ as follows:

$$v_{c_i k} = \begin{cases} 1 & \text{if } n_{ik} \geq 1 \quad \forall i \in I, \forall k \in V_i \\ 0 & \text{otherwise} \end{cases}$$

$$w_k = \begin{cases} 1 & \text{if } \sum_{i \in I} n_{ik} \geq 1 \quad \forall k \in V \\ 0 & \text{otherwise.} \end{cases}$$

c) Postoptimization phase

In this phase (cf. Procedure 1 at line 20), the best solution $\mathfrak{s}^*$ may be improved using the procedure IMPROVESOLUTION($\mathfrak{s}^*$).

TD^k_{first}, TD^k_{inter} and TD^k_{last} which will be used later are defined as follows:

TD^k_{first}: This variable is intended to contain for vehicle $k \in V$ the loading time, the travel time and the unloading time of its first delivery. Note that the loading time is not considered if the vehicle is preloaded ($k \in VP$).

TD^k_{last}: This variable is intended to contain for vehicle $k \in V$ the back travel time of its last delivery.

TD^k_{inter}: This variable is intended to contain for vehicle $k \in V$ all other travel durations.

The aggregation of the values related to these three variables corresponds for vehicle $k \in V$ to its total travel duration.

After having briefly described the three phases of the tabu search method used to solve the cement delivery problem, each procedure called by this solution method is presented in detail in the remainder of this subsection, starting with the construction of an initial feasible solution.

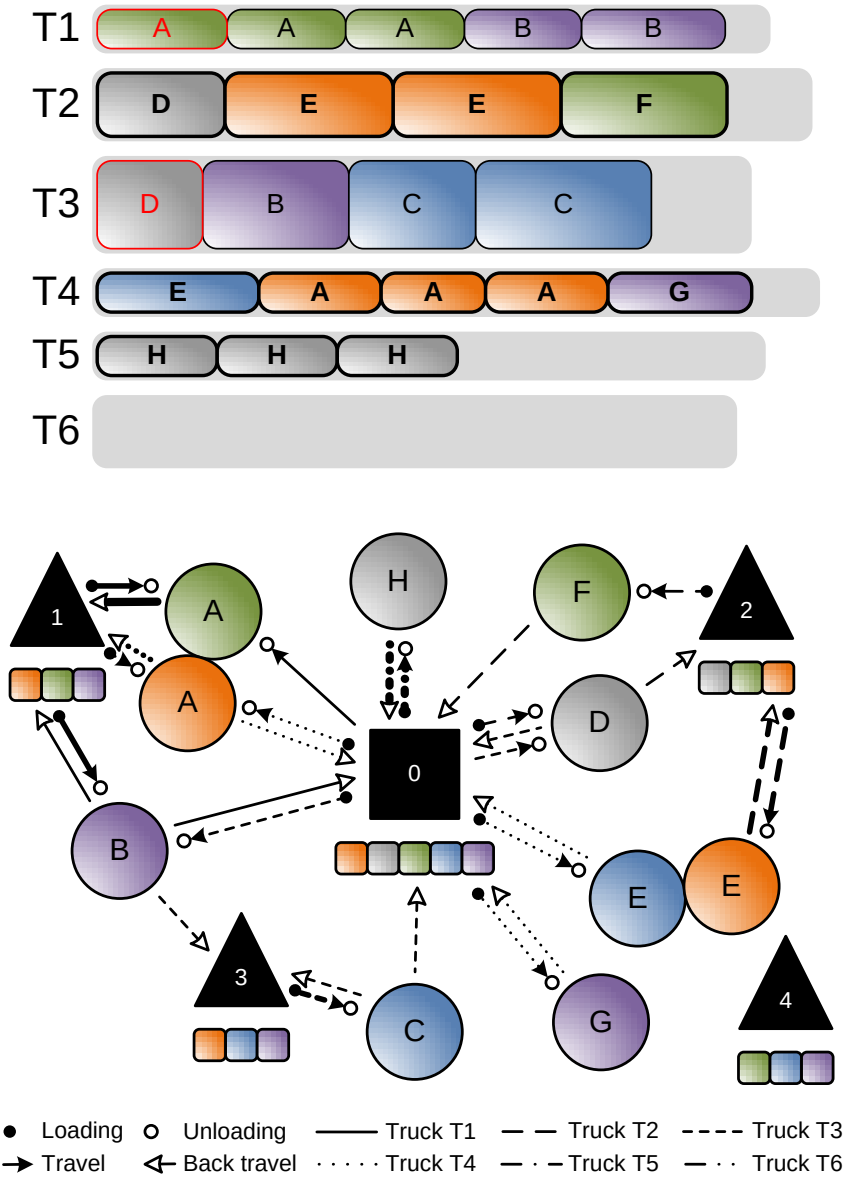

Figure 8.6: Construction of an initial solution
Third construction phase

Construction of an initial solution

Summarized in Procedure 2, the construction of an initial solution is composed of four phases: an *initialization phase* where some variables are initialized, a *first construction phase* where appropriate orders are allocated to the first delivery of preloaded vehicles, a *second construction phase* where orders are split into deliveries and assigned to preloaded vehicles and a *third construction phase* where orders are split into deliveries and assigned to vehicles that are not preloaded. A detailed pseudocode of this procedure is presented in Section E.1 of Appendix E.

Procedure 2 Construction of an initial solution (Summary)

```
procedure ConstructInitialSolution
    Initialization of some variables
    for each preloaded vehicle do
        Allocation of an appropriate order to its first delivery
    end for
    for each preloaded vehicle do
        repeat
            Split of the remainder of orders into deliveries
            Assignment of deliveries to the vehicle
        until orders are entirely satisfied or vehicle's availability is empty
    end for
    for each non preloaded vehicle do
        repeat
            Split of the remainder of orders into deliveries
            Assignment of deliveries to the vehicle
        until orders are entirely satisfied or vehicle's availability is empty
    end for
    return Initial feasible solution
end procedure
```

a) Initialization phase

This phase is illustrated in Figure 8.3 with a simplified instance of the cement delivery problem. In this problem instance, 6 different trucks ($T1$, $T2$, $T3$, $T4$, $T5$ and $T6$) can be used to do cement deliveries. The daily availability and the capacity of each truck are respectively represented by the length and the height of a grey strip. 10 orders related to 8 customers (A, B, C, D, E, F, G and H) have to be supplied from the main depot (0) and/or from 4 local depots (1, 2, 3 and 4). Initially, no delivery is allocated to a vehicle and no route exists. Furthermore, the quantity of cement delivered to each customer is 0 and the daily availability of each vehicle

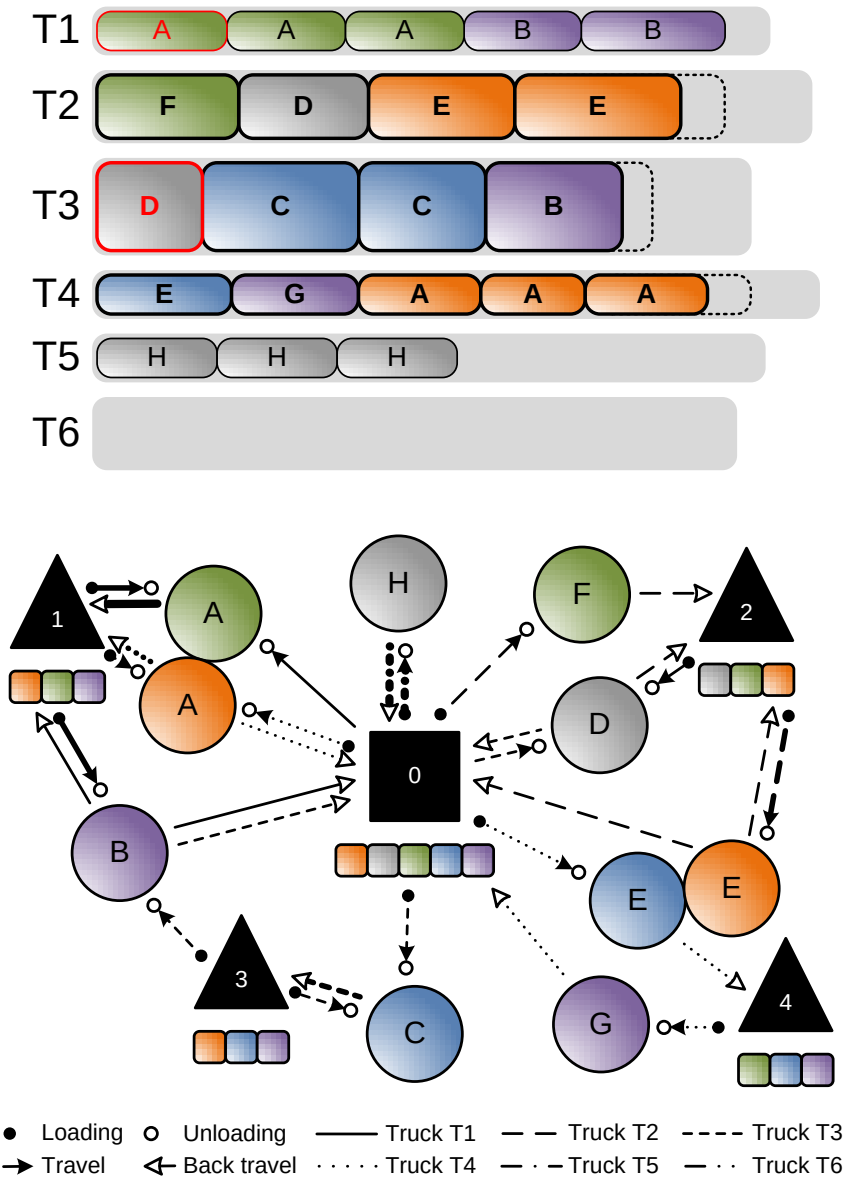

Figure 8.7: Improvement of a solution
Improvement phase

is full. In the figure, trucks *T1* and *T3* are preloaded but no order is allocated to their first delivery. Note that an order of customer *A* or the order of customer *F* can be allocated to the preload of truck *T1*, while the order of customer *D* or the order of customer *H* can be allocated to the preload of truck *T3*.

b) First construction phase

This phase is illustrated in Figure 8.4. Considering the quantity of cement already delivered to the customers and the availability of the vehicles, an appropriate order is allocated to the first delivery of each preloaded vehicle and the corresponding routes are updated. In the figure, an order of customer *A* is allocated to truck *T1* while the order of customer *D* is allocated to truck *T3*. Furthermore, the routes related to trucks *T1* and *T3* are updated. Note that there is no loading time for preloaded deliveries.

c) Second construction phase

This phase is illustrated in Figure 8.5. Considering the quantity of cement already delivered to the customers and the remaining availability of the vehicles, appropriate orders are split into deliveries and allocated to the preloaded vehicles. The corresponding routes are also updated. In the figure, 4 deliveries related to customers *A* and *B* are allocated to truck *T1* using local depot 1 while 3 deliveries related to customers *B* and *C* are allocated to truck *T3* using the main depot (0) and local depot 3. The routes related to trucks *T1* and *T3*, which start and end their daily activity at the main depot (0), are updated. Note that the number of travels or back travels between a depot and a customer is proportional to the thickness of the corresponding line.

d) Third construction phase

This phase is illustrated in Figure 8.6. Considering the quantity of cement already delivered to the customers and the remaining availability of the vehicles, appropriate orders are split into deliveries and allocated to the vehicles that are not preloaded. The corresponding routes are also updated. In the figure, 4 deliveries related to customers *D*, *E* and *F* are allocated to truck *T2* using the main depot (0) and local depot 2, 5 deliveries related to customer *E*, *A* and *G* are allocated to truck *T4* using the main depot (0) and local depot 1, and 3 deliveries related to

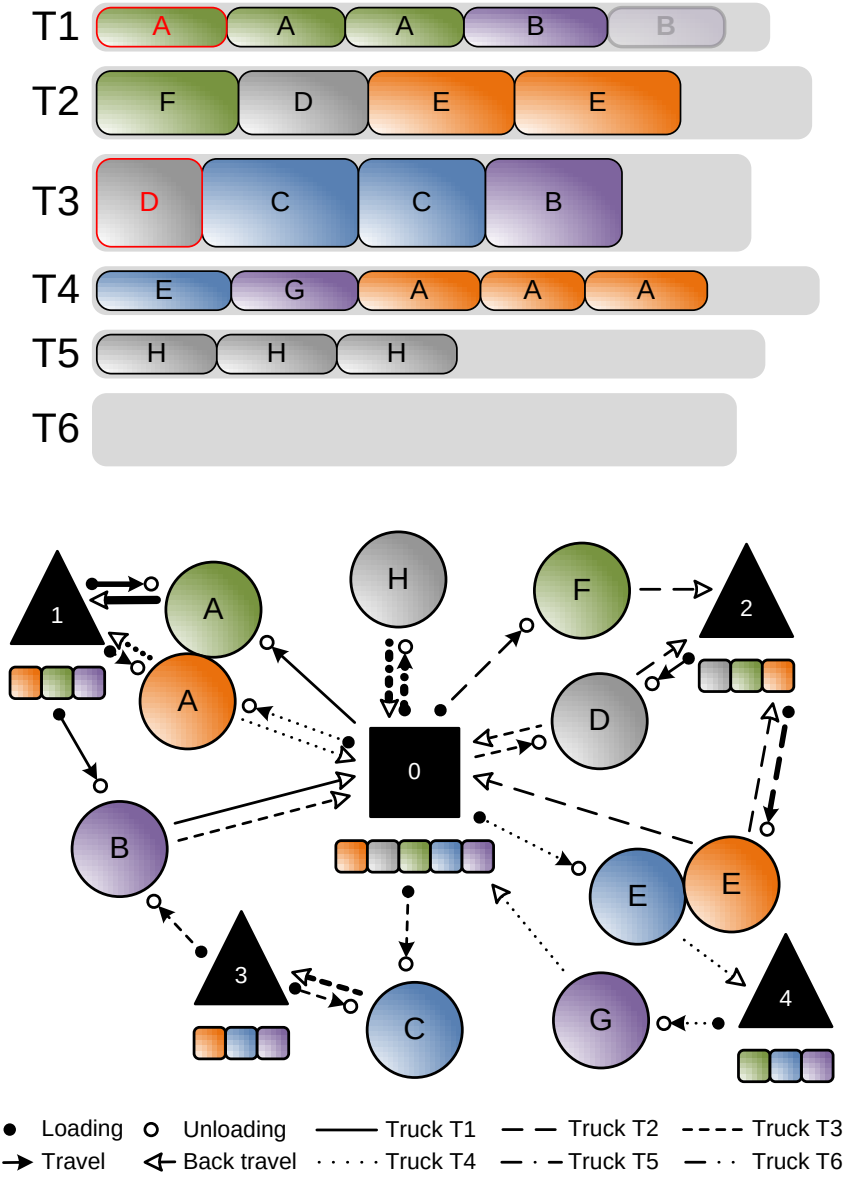

Figure 8.8: Generation of the best neighbor (Deletion & Insertion)
Deletion phase

customer H are allocated to truck $T5$ using the main depot (0). The routes related to trucks $T2$, $T4$ and $T5$ are updated.

When required by the tabu search method, i.e. in the *initialization phase*, in the *tabu search phase* and in the *postoptimization phase*, a feasible solution or a partial feasible solution related to a vehicle may be improved. The dedicated procedure called for this purpose is discussed hereafter.

Improvement of a solution

Summarized in Procedure 3, the improvement of a solution is composed of two phases: an *initialization phase* where some variables are defined or initialized and an *improvement phase* where all the possible permutations of orders involved in each delivery sequence are tested to find for each vehicle the delivery sequence that minimizes its total travel duration. Note that the number of different possibilities for each vehicle is not huge. Indeed, only a few orders can be assigned to a vehicle due to its daily availability and due to the duration of a delivery. This *improvement phase* is therefore not time consuming. A detailed pseudocode of this procedure is presented in Section E.2 of Appendix E.

Procedure 3 Improvement of a solution (Summary)

```
procedure IMPROVESOLUTION(𝔰, [k* ∈ V])
    Initialization of some variables
    for each vehicle used in 𝔰 do
        Storage of its current delivery sequence
        if the vehicle can be loaded at local depots [and the vehicle is k*] then
            Test of all the possible permutations of its orders
            Selection of the delivery sequence that minimizes its total travel duration
            Replacement of its current delivery sequence with this sequence
        end if
        Storage of the current delivery sequence as part of the improved solution
    end for
    return Improved solution
end procedure
```

One parameter is needed to run this procedure: a solution $\mathfrak{s}$. Note that the specification of a vehicle $k^* \in V$ as second parameter is optional and therefore put in brackets.

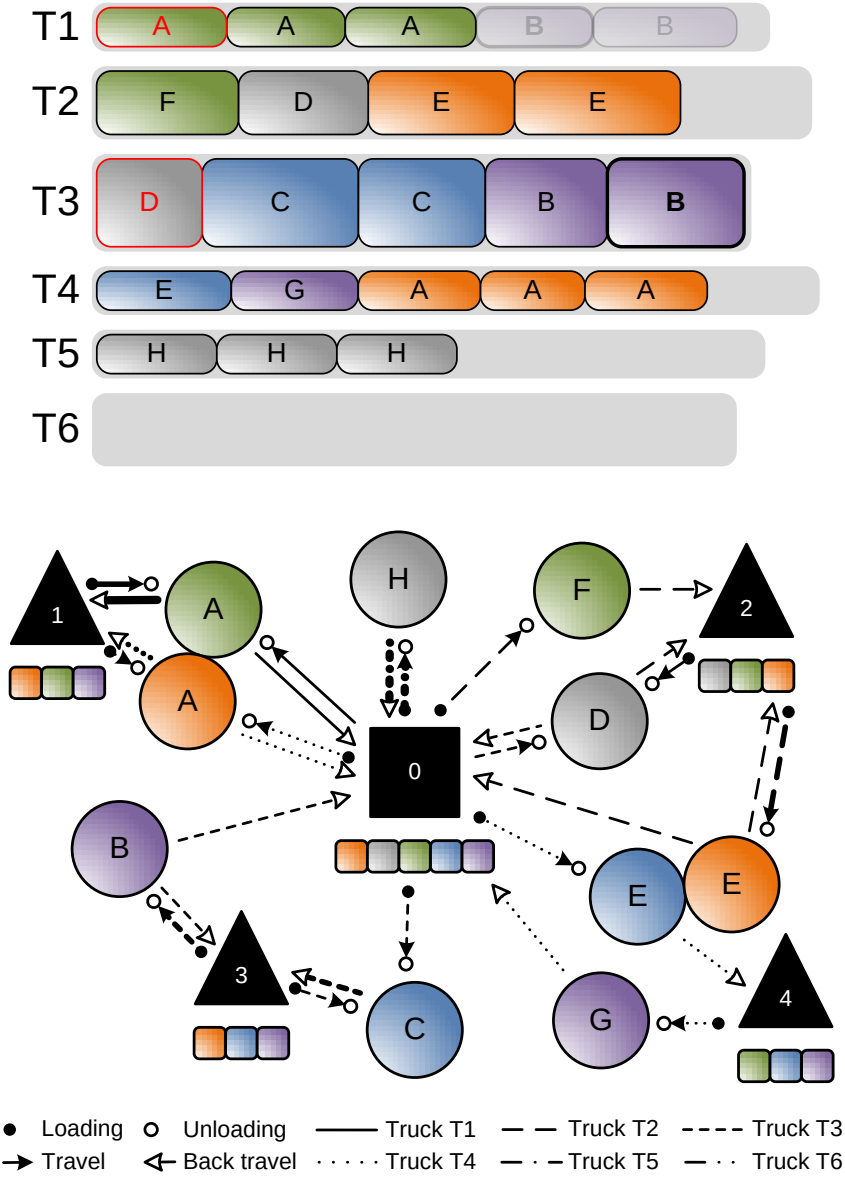

Figure 8.9: Generation of the best neighbor (Deletion & Insertion) Insertion phase

a) Initialization phase

Figure 8.6 related to the *third construction phase* of the previous procedure can also illustrate this phase. The vehicles used in the current solution as the sequence of orders on their route are considered. In the figure, trucks *T1*, *T2*, *T3*, *T4* and *T5* are used in the current solution and have respectively as delivery sequences $\{A,B\}$, $\{D,E,F\}$, $\{D,B,C\}$, $\{E,A,G\}$ and $\{H\}$.

b) Improvement phase

This phase is illustrated in Figure 8.7. All the possible permutations of orders are tested for each vehicle that can be loaded at local depots and the delivery sequence that minimizes the total travel duration of a vehicle replaces its initial sequence. In the figure, the delivery sequences of trucks *T2*, *T3* and *T4* are modified and correspond henceforth to $\{F,D,E\}$, $\{D,C,B\}$ and $\{E,G,A\}$. Note that the sequence $\{E,G\}$ on truck *T4* allows the use of local depot 4 and that truck *T5*, only loaded at the main depot (0), is not concerned by the improvement.

A first procedure called by the tabu search method in the *tabu search phase* to select the best solution among a generated neighborhood is discussed hereafter.

Generation of the best neighbor (Deletion & Insertion)

Summarized in Procedure 4, the generation of the best neighbor with operator 'Deletion & Insertion' is composed of three phases: an *initialization phase* where some variables are defined or initialized, a *deletion phase* where a delivery of each order is eventually deleted from the route of each vehicle and an *insertion phase* where each of these deliveries is reinserted on the delivery sequence of others vehicles in order to generate neighboring solutions and to select among them the best one. A detailed pseudocode of this procedure is presented in Section E.3 of Appendix E.

Five parameters are needed to run this procedure: a solution $\mathfrak{s}$, the tabu list TA presented before, the size of a neighborhood n_N, the size of the tabu list n_{TA} and the current best solution $\mathfrak{s}^*$ of the tabu search.

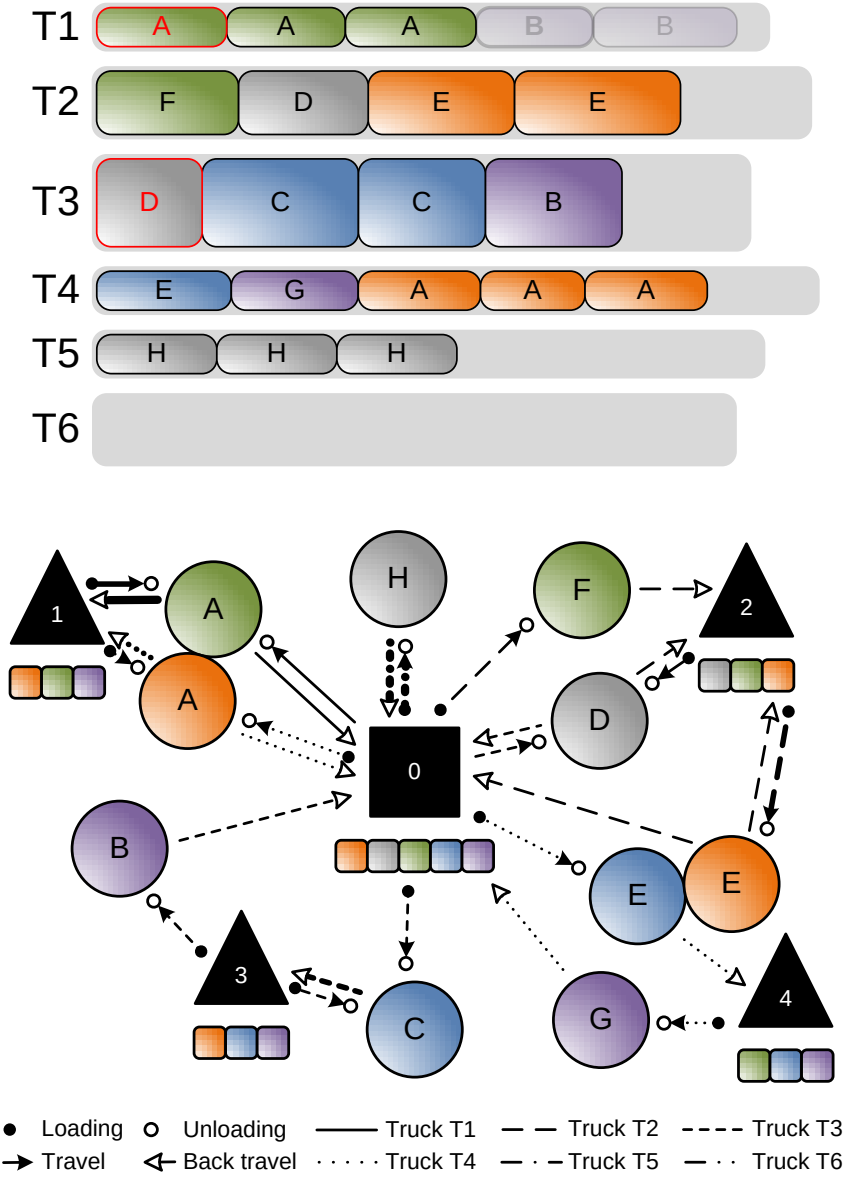

Figure 8.10: Optimization of the insertion
Cleaning phase

Procedure 4 Generation of the best neighbor (Deletion & Insertion) (Summary)

```
procedure GENERATEBESTNEIGHBOR(𝔰, TA, n_N, n_TA, 𝔰*)
    Initialization of some variables
    for each vehicle k^del used in 𝔰 do
        for each order i' allocated to vehicle k^del do
            Deletion of a delivery related to vehicle k^del and order i'
            Improvement of the delivery sequence of vehicle k^del
            for each other vehicle k ≠ k^del do
                Insertion of the required number of deliveries of the order
                Deletion of unnecessary deliveries allocated to other vehicles
                if current neighbor is better than the best one then
                    if current neighbor is not tabu or is better than 𝔰* then
                        Current neighbor is set as best one
                    end if
                end if
            end for
        end for
    end for
    Update of the tabu list TA
    return Best neighbor and updated tabu list TA
end procedure
```

a) Initialization phase

Figure 8.7 related to the *improvement phase* of the previous procedure can also illustrate this phase. A feasible solution and some of its properties are considered. In the figure, deliveries allocated to trucks $T1$, $T2$, $T3$, $T4$ and $T5$ allow the calculation of the total travel duration of these trucks and therefore the calculation of their remaining availability to possibly insert additional deliveries. Note that the remaining availability of truck $T6$ is full.

b) Deletion phase

This phase is illustrated in Figure 8.8. To generate a partial neighboring solution, a delivery of an order is deleted from the route of a vehicle but only if such an operation is allowed. Indeed, the first delivery of a preloaded vehicle can be deleted only if another delivery of the vehicle can be allocated to the preload. Note that the delivery sequence of the vehicle may then be improved due to the deletion of one of its deliveries. To generate all the partial neighboring solutions, this process is repeated for each order and for each vehicle used in the initial solution. If the entire neighborhood of solutions has not to be generated, orders and vehicles of

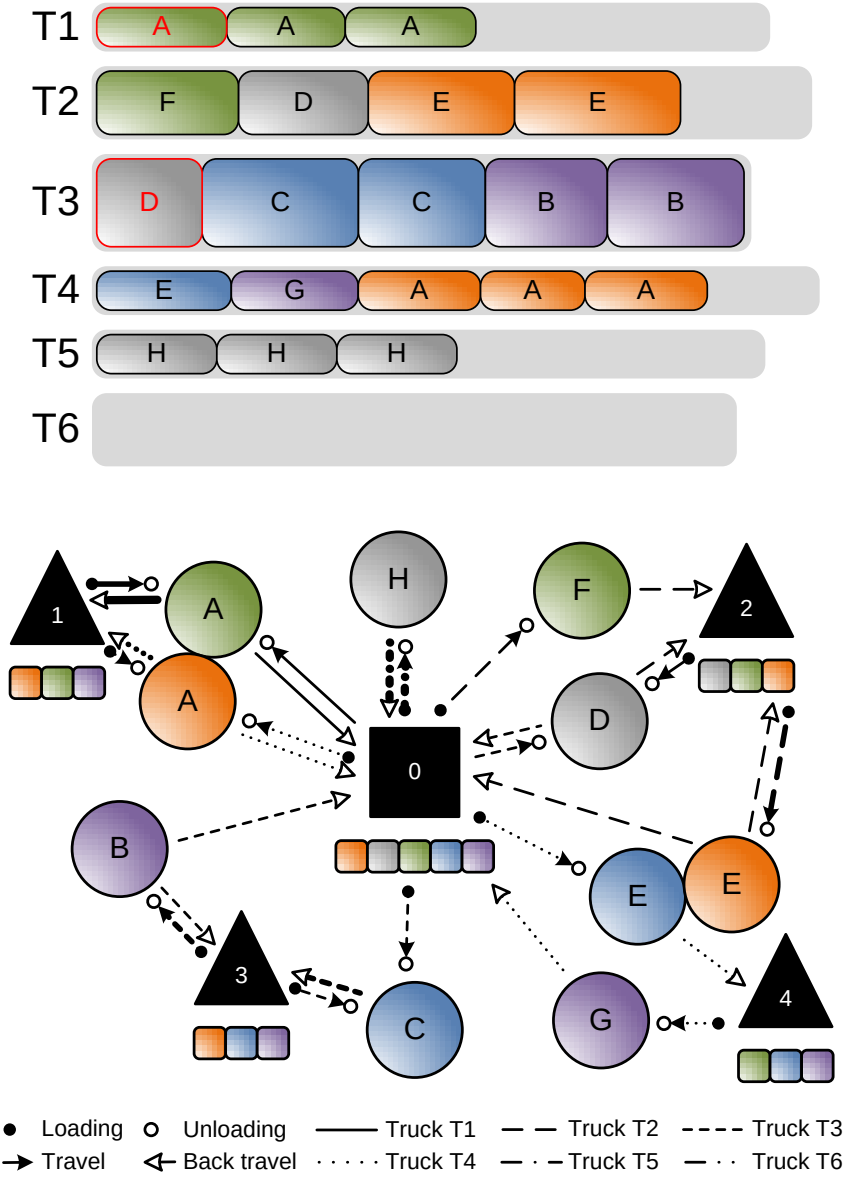

Figure 8.11: Generation of the best neighbor (Delivery swap)
Initialization phase

the initial solution are selected randomly. In the figure, a delivery related to customer B is deleted from the route of truck $T1$. Note that the delivery sequence of truck $T1$ is not improved. Indeed, a delivery of same order is still allocated to this truck.

c) Insertion phase

This phase is illustrated in Figure 8.9. After having deleted a delivery of an order from the route of a vehicle in the *deletion phase* to generate a partial neighboring solution, one or more deliveries of the same order are inserted on the route of another vehicle to form a complete neighboring solution. Indeed, the capacity of two vehicles can differ and more than one delivery may be required to keep the demand related to the order entirely satisfied. Note that the insertion is done at the best position in the delivery sequence of the vehicle. Furthermore, deliveries of the order allocated to other vehicles that are no more required due to this insertion are deleted. More precision about the best insertion and the deletion of unnecessary deliveries will be discussed later when presenting the optimization of the insertion. If the current neighboring solution is better than the best neighboring solution found, is not tabu or is tabu but better than the best solution found, this one is stored as best neighboring solution. To generate all the neighboring solutions, this process is repeated for each delivery deleted in the *deletion phase*. In the figure, the delivery related to customer B is inserted on the route of truck $T3$ after a delivery of same order. Due to the capacity of truck $T3$ that is twice more as the capacity of truck $T1$, the last delivery of same order allocated to truck $T1$ is deleted. Indeed, this delivery is no more required.

Before presenting additional procedures that generate neighboring solutions with other operators, the procedure that optimizes the insertion of a required number of deliveries on the route of a vehicle is first discussed hereafter.

Optimization of the insertion

Summarized in Procedure 5, the optimization of the insertion is composed of three phases: an *initialization phase* where some variables are defined or initialized, an eventual *cleaning phase* where unnecessary deliveries of mentioned order are deleted from the route of vehicles due to the future insertion and an *insertion phase* where one or more deliveries of this order are reinserted at the most appropriate position on the route of mentioned vehicle. A detailed pseudocode of this

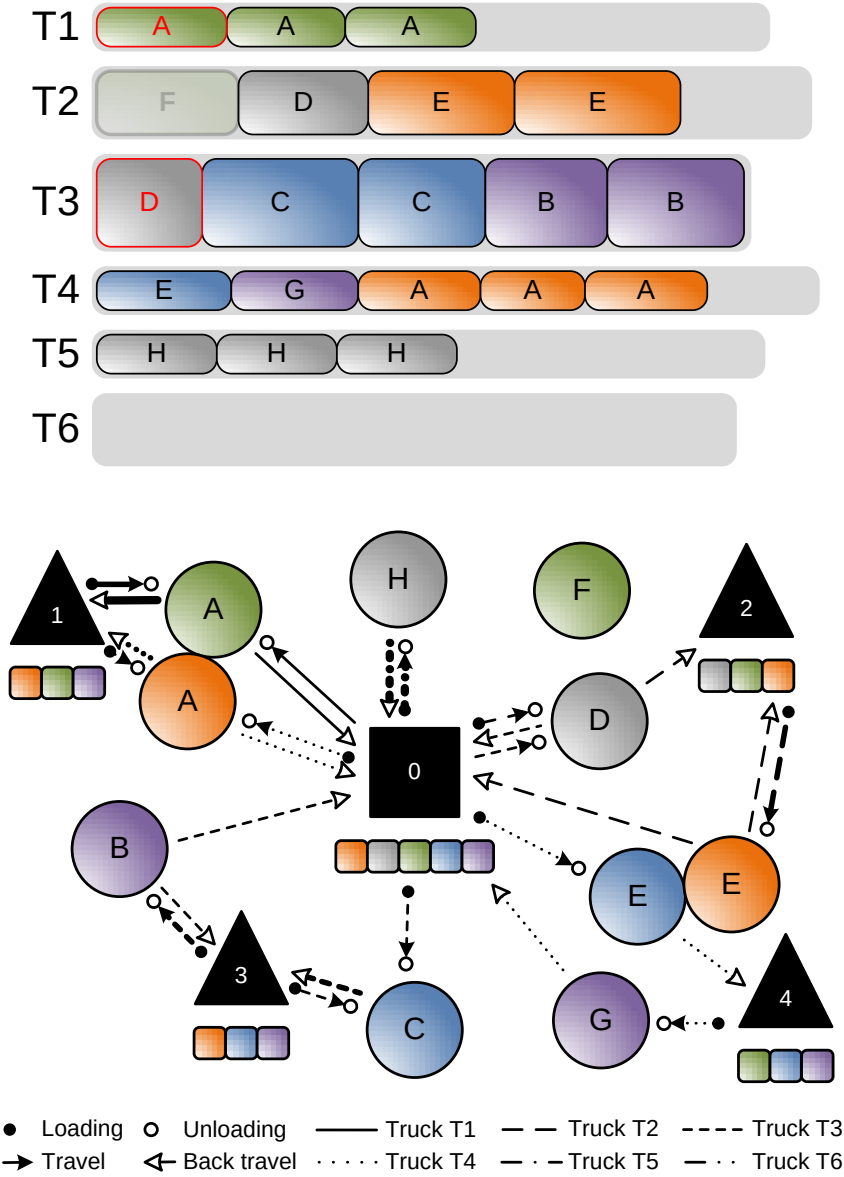

Figure 8.12: Generation of the best neighbor (Delivery swap)
First deletion phase

procedure is presented in Section E.4 of Appendix E.

Procedure 5 Optimization of the insertion (Summary)

```
procedure OptimizeInsertion(𝔰, i*, k*, η, TD^{k*}_{first}, TD^{k*}_{inter}, TD^{k*}_{last}, CP)
    Initialization of some variables
    if unnecessary deliveries of order i* have to be deleted for other vehicles then
        repeat
            for each vehicle different than k* do
                for each order i* allocated to the vehicle do
                    Storage of this delivery if it is unnecessary
                end for
            end for
            Selection of the delivery whose deletion is the most profitable
            Deletion of this delivery of the current solution
        until there is no more unnecessary delivery of order i* for other vehicles to delete
    end if
    Insertion of the required deliveries at the best position on the route of vehicle k*
    return Solution and variation of the objective function and duration variation
end procedure
```

Eight parameters are needed to run this procedure: a solution $\mathfrak{s}$, an order $i^* \in I$, a vehicle $k^* \in V$, a number of deliveries to insert η, the total travel duration of vehicle k^* before the insertion split in $TD^{k^*}_{first}$, $TD^{k^*}_{inter}$ and $TD^{k^*}_{last}$, and a boolean CP indicating if the *cleaning phase* is required or not.

a) Initialization phase

Figure 8.8 related to the *deletion phase* of the previous procedure can also illustrate this phase. An unfeasible solution (due to the deletion of a required delivery) and some of its properties are considered. In the figure, a delivery is missing on the route of truck *T1* to entirely satisfy the order related to customer B.

b) Cleaning phase

This phase is illustrated in Figure 8.10. Unnecessary deliveries of mentioned order on the route of other vehicles due to the future insertion are considered. The delivery whose deletion is the most profitable is deleted. This process is repeated until there is no more unnecessary delivery of the order to delete. In the figure, there is only one other delivery related to customer B. Allocated to truck *T1*, this delivery is deleted because the future insertion of a delivery of the order on truck

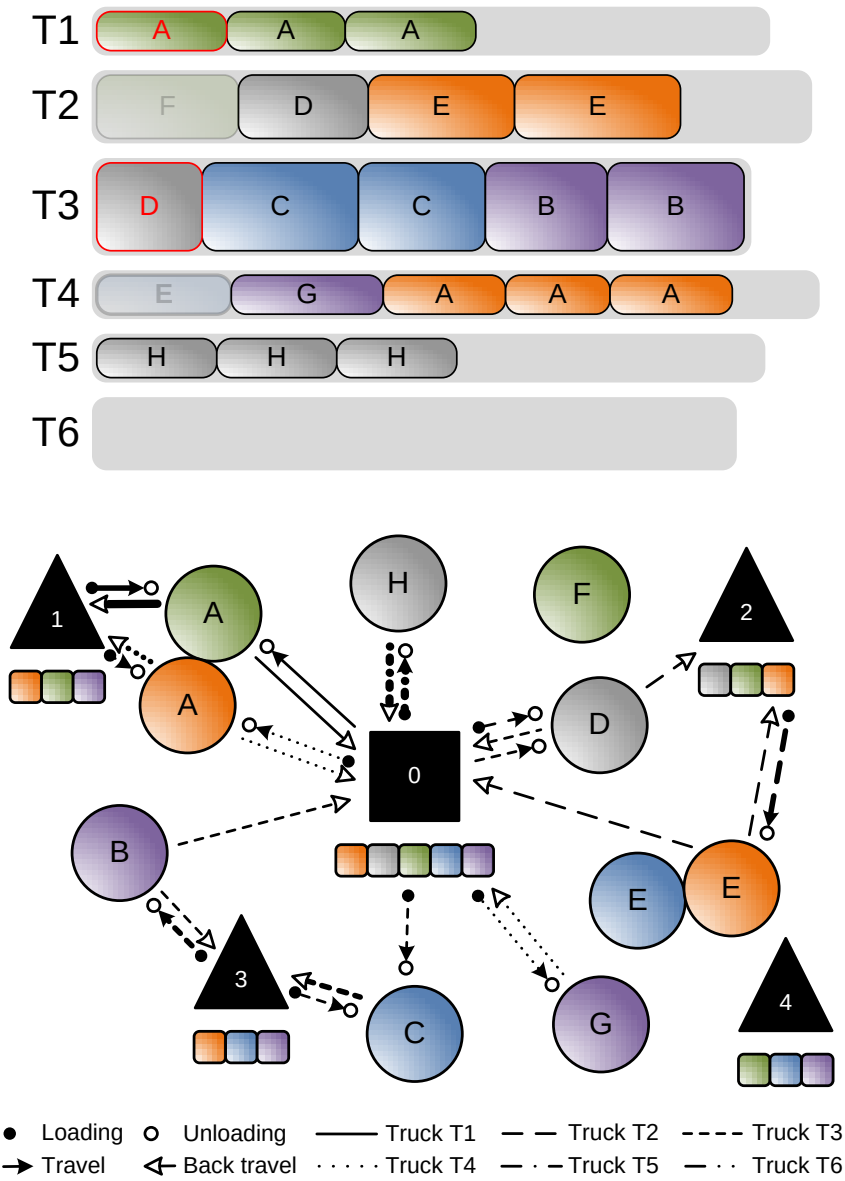

Figure 8.13: Generation of the best neighbor (Delivery swap)
Second deletion phase

T3 corresponds to 2 deliveries of the order on the route of truck *T1*. Indeed, the capacity of truck *T3* is twice as great as the capacity of truck *T1*.

c) Insertion phase

Figure 8.9 related to the *insertion phase* of the previous procedure can also illustrate this phase. One or more deliveries of mentioned order are inserted at the most profitable position on the route of mentioned vehicle. In the figure, a delivery related to customer *B* is allocated to truck *T3* at the last position of its delivery sequence. Indeed, a delivery of this order already exists.

A second procedure called by the tabu search method in the *tabu search phase* to select the best solution among a generated neighborhood is discussed hereafter. Many similarities exist between this procedure and the one that generates the best neighbor of a solution with operator 'Deletion & Insertion'.

Generation of the best neighbor (Delivery swap)

Summarized in Procedure 6, the generation of the best neighbor with operator 'Delivery swap' is composed of four phases: an *initialization phase* where some variables are defined or initialized, a *first deletion phase* where a delivery of each order is eventually deleted from the route of each vehicle, a *second deletion phase* where a delivery of each other order is eventually deleted from the route of each other vehicle and a *swap phase* where the two deleted deliveries are reinserted and swapped between the two implied vehicles in order to generate neighboring solutions and to select among them the best one. A detailed pseudocode of this procedure is presented in Section E.5 of Appendix E.

Five parameters are needed to run this procedure: a solution $\mathfrak{s}$, the tabu list TA presented before, the size of a neighborhood n_N, the size of the tabu list n_{TA} and the current best solution $\mathfrak{s}^*$ of the tabu search.

a) Initialization phase

This phase is illustrated in Figure 8.11. A feasible solution and some of its properties are considered. In the figure, deliveries allocated to trucks *T1*, *T2*, *T3*, *T4* and *T5* allow the calculation of the total travel duration of these trucks and therefore the calculation of their remaining availability to possibly insert additional deliveries. Note that the remaining availability of truck *T6* is full.

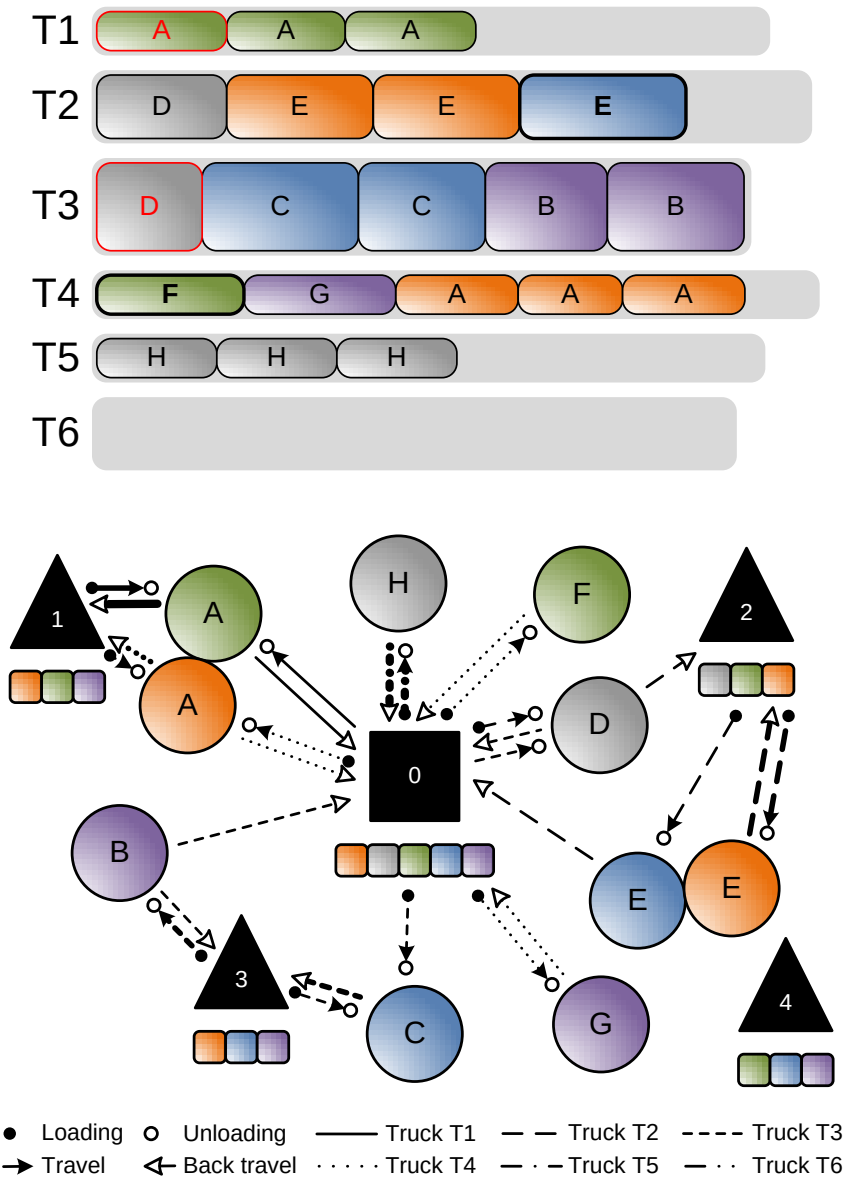

Figure 8.14: Generation of the best neighbor (Delivery swap)
Swap phase

Procedure 6 Generation of the best neighbor (Delivery swap) (Summary)

```
procedure GENERATEBESTNEIGHBOR(𝔰, TA, n_N, n_TA, 𝔰*)
    Initialization of some variables
    for each vehicle k^a used in 𝔰 do
        for each order i^a allocated to vehicle k^a do
            Deletion of a delivery related to vehicle k^a and order i^a
            Improvement of the delivery sequence of vehicle k^a
            for each other vehicle k^b do
                for each other order i^b allocated to vehicle k^b do
                    if vehicle k^a visits the customer of order i^b then
                        Deletion of a delivery related to vehicle k^b and order i^b
                        Improvement of the delivery sequence of vehicle k^b
                        for vehicle k^a and order i^b and vehicle k^b and order i^a do
                            Insertion of the required number of deliveries of the order
                            Deletion of unnecessary deliveries allocated to other vehicles
                        end for
                        if current neighbor is better than the best one then
                            if current neighbor is not tabu or is better than 𝔰* then
                                Current neighbor is set as best one
                            end if
                        end if
                    end if
                end for
            end for
        end for
    end for
    Update of the tabu list TA
    return Best neighbor and updated tabu list TA
end procedure
```

b) First deletion phase

This phase is illustrated in Figure 8.12. To generate a first partial neighboring solution, a delivery of an order is deleted from the route of a vehicle but only if such an operation is allowed. As mentioned before, the first delivery of a preloaded vehicle can be deleted only if another delivery of the vehicle can be allocated to the preload. Note that the delivery sequence of the vehicle may then be improved due to the deletion of one of its deliveries. To generate all the first partial neighboring solutions, this process is repeated for each order and for each vehicle used in the initial solution. If the entire neighborhood of solutions has not to be generated, orders and vehicles of the initial solution are selected randomly. In the figure, a delivery related to customer F is deleted from the route of truck $T2$. Note that the delivery sequence of truck $T2$ is not improved.

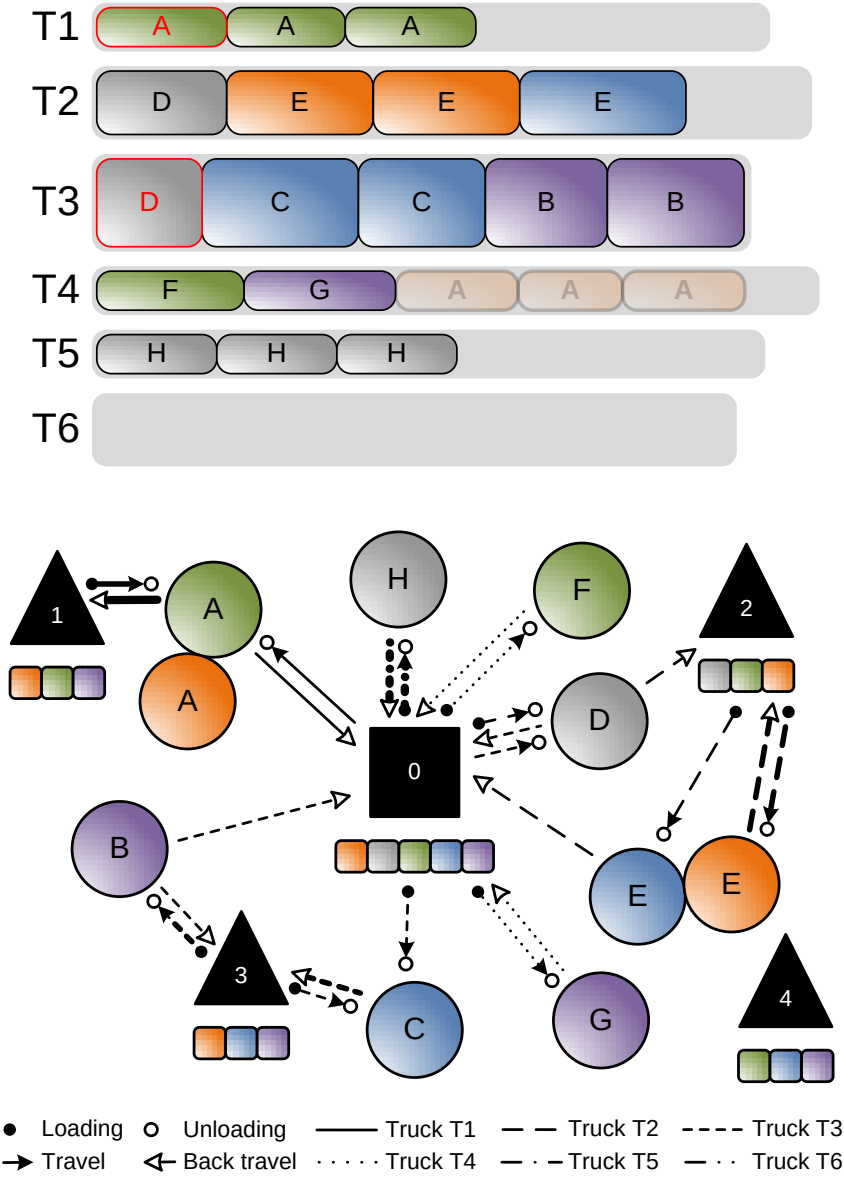

Figure 8.15: Generation of the best neighbor (Order swap)
First deletion phase

c) Second deletion phase

This phase is illustrated in Figure 8.13. To generate a second partial neighboring solution, a delivery of another order is deleted from the route of another vehicle but only if such an operation is allowed. Indeed, the first delivery of a preloaded vehicle can be deleted only if another delivery of the vehicle can be allocated to the preload. Note that the delivery sequence of the vehicle may then be improved due to the deletion of one of its deliveries. To generate all the second partial neighboring solutions, this process is repeated for each other order and for each other vehicle used in the initial solution. In the figure, a delivery related to customer E is deleted from the route of truck $T4$. Note that the delivery sequence of truck $T4$ is not improved.

d) Swap phase

This phase is illustrated in Figure 8.14. After having deleted two deliveries of different orders from the route of two different vehicles in the *first deletion phase* and *second deletion phase* to generate a first and second partial neighboring solution, one or more deliveries of each of these two orders are inserted on the route of the other vehicle to form a complete neighboring solution. Indeed, the capacity of two vehicles can differ and more than one delivery may be required to keep the demand related to the two orders entirely satisfied. Note that the insertion is done at the best position in the delivery sequence of each vehicle. Furthermore, deliveries of the order allocated to other vehicles that are no more required due to this insertion are deleted. If the current neighboring solution is better than the best neighboring solution found, is not tabu or is tabu but better than the best solution found, this one is stored as best neighboring solution. To generate all the neighboring solutions, this process is repeated for each delivery deleted in the *first deletion phase* and for each delivery deleted in the *second deletion phase*. In the figure, the delivery related to customer E is inserted on the route of truck $T2$ after a delivery related to this customer, while the delivery related to customer F is inserted at the first position on the route of truck $T4$.

A third procedure called by the tabu search method in the *tabu search phase* to select the best solution among a generated neighborhood is discussed hereafter. This procedure is almost identical to the one presented before. All deliveries of an order allocated to a vehicle are considered in the following procedure which does not generate an entire neighborhood of solutions in order to limit the tests of feasibility.

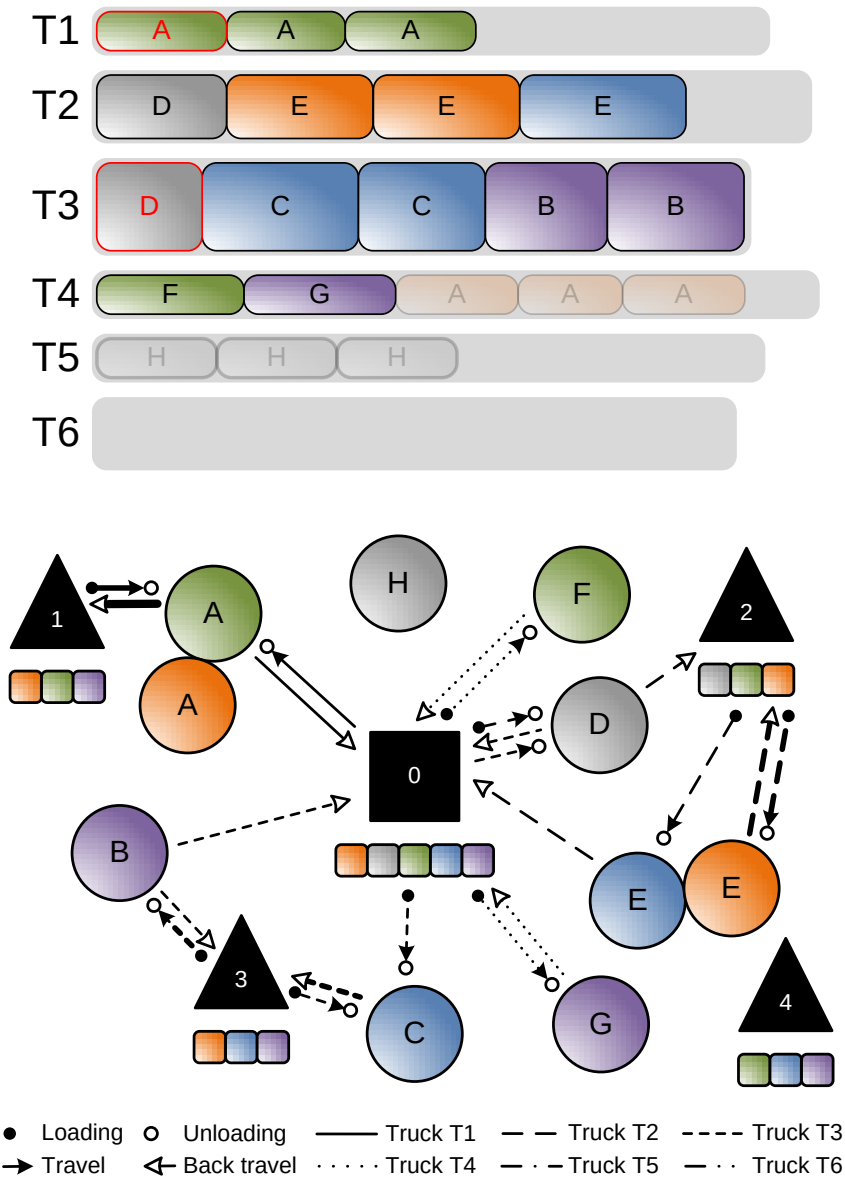

Figure 8.16: Generation of the best neighbor (Order swap)
Second deletion phase

Generation of the best neighbor (Order swap)

Summarized in Procedure 7, the generation of the best neighbor with operator 'Order swap' is composed of four phases: an *initialization phase* where some variables are defined or initialized, a *first deletion phase* where all deliveries of each order are eventually deleted from the route of each vehicle, a *second deletion phase* where all deliveries of each other order are eventually deleted from the route of each other vehicle and a *swap phase* where the deleted deliveries related to the two orders are reinserted and swapped between the two implied vehicles in order to generate neighboring solutions and to select among them the best one. A detailed pseudocode of this procedure is presented in Section E.6 of Appendix E.

Procedure 7 Generation of the best neighbor (Order swap) (Summary)

```
procedure GENERATEBESTNEIGHBOR(𝔰, TA, n_N, n_TA, 𝔰*)
    Initialization of some variables
    for each vehicle k^a used in 𝔰 having the same capacity as another vehicle do
        for each order i^a allocated to vehicle k^a not concerning a preload do
            Deletion of all deliveries related to vehicle k^a and order i^a
            Improvement of the delivery sequence of vehicle k^a
            for each other vehicle k^b having the same capacity as vehicle k^a do
                for each other order i^b allocated to vehicle k^b not concerning a preload do
                    Deletion of all deliveries related to vehicle k^b and order i^b
                    Improvement of the delivery sequence of vehicle k^b
                    for vehicle k^a and order i^b and vehicle k^b and order i^a do
                        Insertion of all deleted deliveries of the order
                    end for
                    if current neighbor is better than the best one then
                        if current neighbor is not tabu or is better than 𝔰* then
                            Current neighbor is set as best one
                        end if
                    end if
                end for
            end for
        end for
    end for
    Update of the tabu list TA
    return Best neighbor and updated tabu list TA
end procedure
```

Five parameters are needed to run this procedure: a solution $\mathfrak{s}$, the tabu list TA presented before, the size of a neighborhood n_N, the size of the tabu list n_{TA} and the current best solution $\mathfrak{s}^*$ of the tabu search.

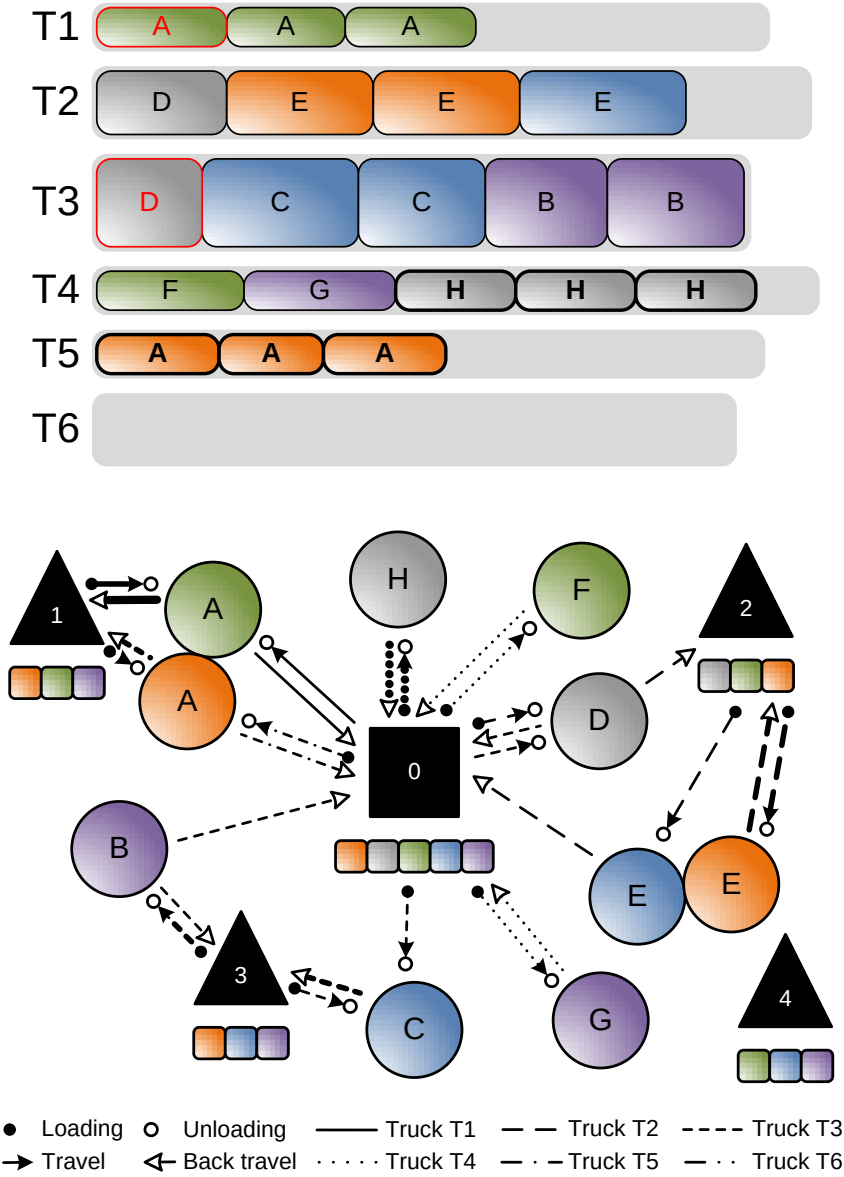

Figure 8.17: Generation of the best neighbor (Order swap)
Swap phase

a) Initialization phase

Figure 8.14 related to the *swap phase* of the previous procedure can also illustrate this phase. A feasible solution and some of its properties are considered. In the figure, deliveries allocated to trucks $T1$, $T2$, $T3$, $T4$ and $T5$ allow the calculation of the total travel duration of these trucks and therefore the calculation of their remaining availability to possibly insert additional deliveries. Note that the remaining availability of truck $T6$ is full.

b) First deletion phase

This phase is illustrated in Figure 8.15. To generate a first partial neighboring solution, all deliveries of an order are deleted from the route of a vehicle. Note that the delivery sequence of the vehicle may then be improved due to the deletion of some of its deliveries. To generate all the first partial neighboring solutions, this process is repeated for each order and for each vehicle used in the initial solution. If the entire neighborhood of solutions has not to be generated, orders and vehicles of the initial solution are selected randomly. In the figure, all deliveries related to customer A are deleted from the route of truck $T4$. Note that the delivery sequence of truck $T4$ is not improved.

c) Second deletion phase

This phase is illustrated in Figure 8.16. To generate a second partial neighboring solution, all deliveries of another order are deleted from the route of another vehicle. Note that the delivery sequence of the vehicle may then be improved due to the deletion of some of its deliveries. To generate all the second partial neighboring solutions, this process is repeated for each other order and for each other vehicle of same capacity used in the initial solution. In the figure, all deliveries related to customer H are deleted from the route of truck $T5$.

d) Swap phase

This phase is illustrated in Figure 8.17. After having deleted all deliveries of two different orders from the route of two different vehicles in the *first deletion phase* and *second deletion phase* to generate a first and second partial neighboring solution, all deleted deliveries of each of these two orders are inserted on the route of

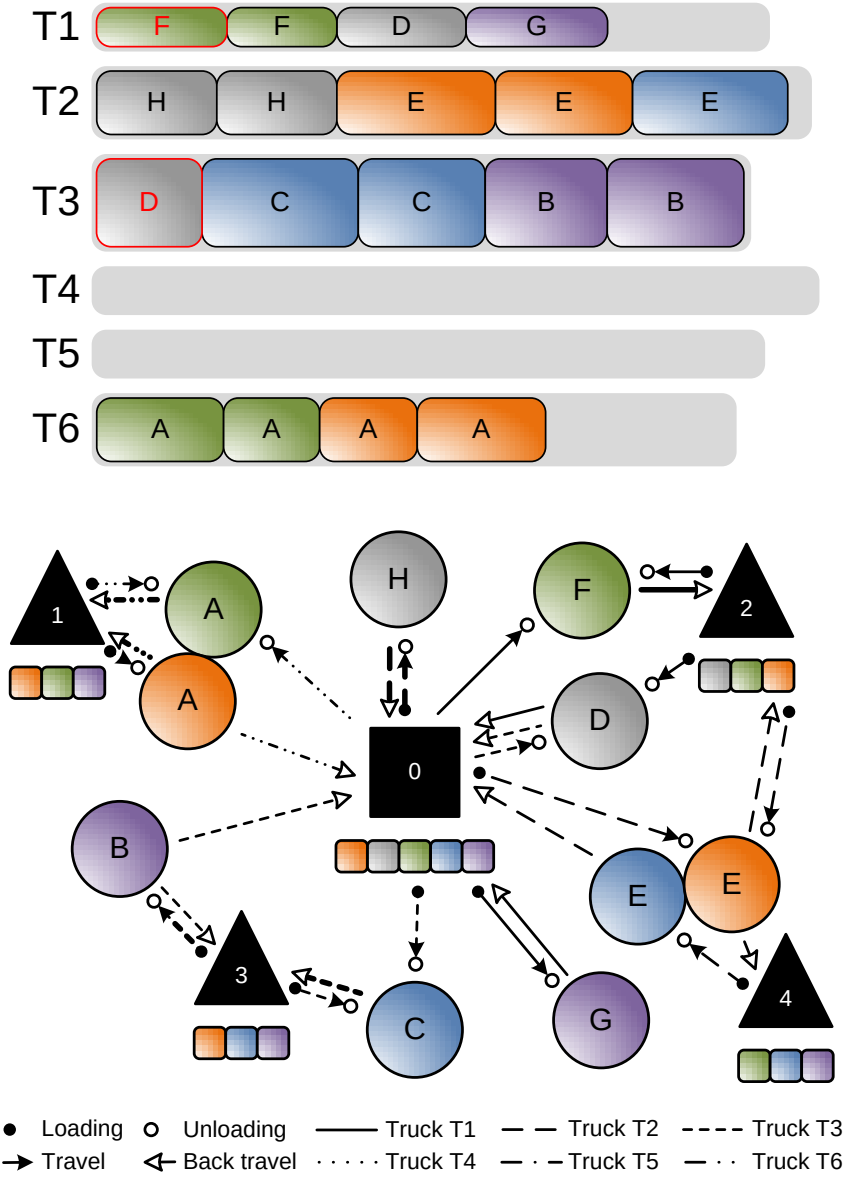

Figure 8.18: Modification of a solution
Initialization phase

the other vehicle to form a complete neighboring solution. The insertion is done at the best position in the delivery sequence of each vehicle. If the current neighboring solution is better than the best neighboring solution found, is not tabu or is tabu but better than the best solution found, this one is stored as best neighboring solution. To generate all the neighboring solutions, this process is repeated for each order whose all deliveries have been deleted in the *first deletion phase* and for each order whose all deliveries have been deleted in the *second deletion phase*. In the figure, all deleted deliveries related to customer A are inserted on the route of truck $T5$, while all deleted deliveries related to customer H are inserted on the route of truck $T4$ after a delivery related to customer G.

Following the presentation of how the best neighbor of a solution can be generated with various operators, the procedure that modifies a part of a solution after a predefined number of iterations without improvement of the best solution is finally discussed hereafter. As mentioned before, the aim of this procedure is to diversify the search process.

Modification of a solution

Summarized in Procedure 8, the modification of a solution is composed of three phases: an *initialization phase* where some variables are initialized, a *deletion phase* where deliveries allocated to vehicles selected randomly are deleted and a *construction phase* where new deliveries are allocated to unused vehicles to entirely satisfy the demand of each order. A detailed pseudocode of this procedure is presented in Section E.7 of Appendix E.

One parameter is needed to run this procedure: a solution $\mathfrak{s}$.

a) Initialization phase

This phase is illustrated in Figure 8.18. The vehicles used in the current solution are considered. In the figure, deliveries are allocated to trucks $T1$, $T2$, $T3$ and $T6$.

b) Deletion phase

This phase is illustrated in Figure 8.19. Deliveries allocated to half of the vehicles used in the solution are deleted. Some of the vehicles whose customer is also visited by another vehicle and then some of the remaining vehicles are selected

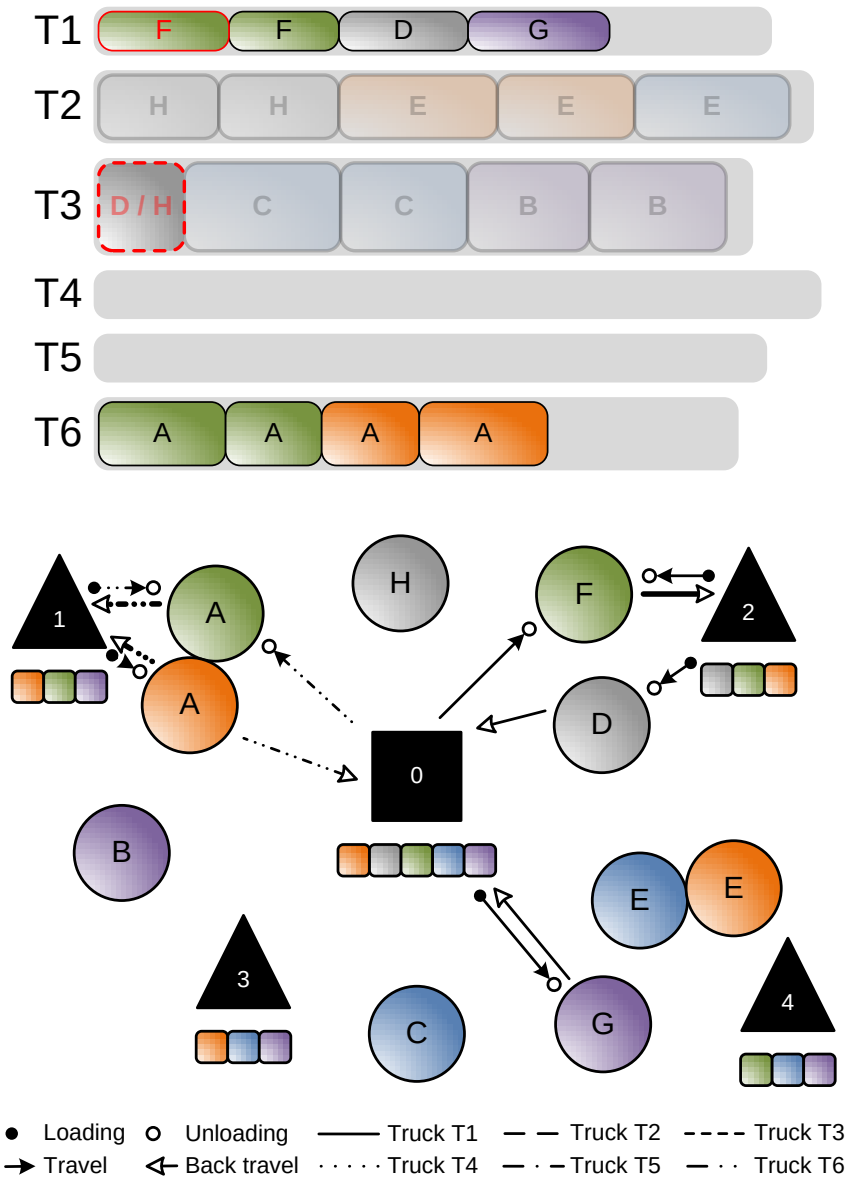

Figure 8.19: Modification of a solution
Deletion phase

Procedure 8 Modification of a solution (Summary)

```
procedure ModifySolution(s)
    Initialization of some variables
    repeat
        Random selection of vehicles whose customer is also visited by another vehicle
        Deletion of their allocated deliveries
        Random selection of remaining vehicles
        Deletion of their allocated deliveries
        Storage of the current solution as first partial solution
        for each order do
            Update of the delivered quantities
            Update of the demand whose delivered quantities are subtracted
        end for
        Consideration of vehicles that are not used in the first partial solution
        Construction of a second partial solution with these vehicles
    until the second partial solution is feasible
    Merge of the two partial solutions as modified solution
    return Modified solution
end procedure
```

randomly. In the figure, no customer is visited by many vehicles. Deliveries allocated to trucks $T2$ and $T3$ are deleted, as the corresponding routes. Note that no order is allocated to the first delivery of preloaded vehicle $T3$.

c) Construction phase

This phase is illustrated in Figure 8.20. New deliveries of orders whose demand is not entirely satisfied are allocated to unused vehicles. The corresponding routes are also updated. In the figure, 5 deliveries related to customers B and C are allocated to truck $T2$ using the main depot (0) and local depot 3, 4 deliveries related to customers H and E are allocated to truck $T3$ using the main depot (0) and local depot 2, and 3 deliveries related to customer E and D are allocated to truck $T4$ using the main depot (0). The routes related to trucks $T2$, $T3$ and $T4$ are updated. Note that customer E is visited by trucks $T3$ and $T4$.

8.4 Conclusion

In this chapter, a modeling of the cement delivery problem and solution methods provided by the optimization module to solve this problem have been presented.

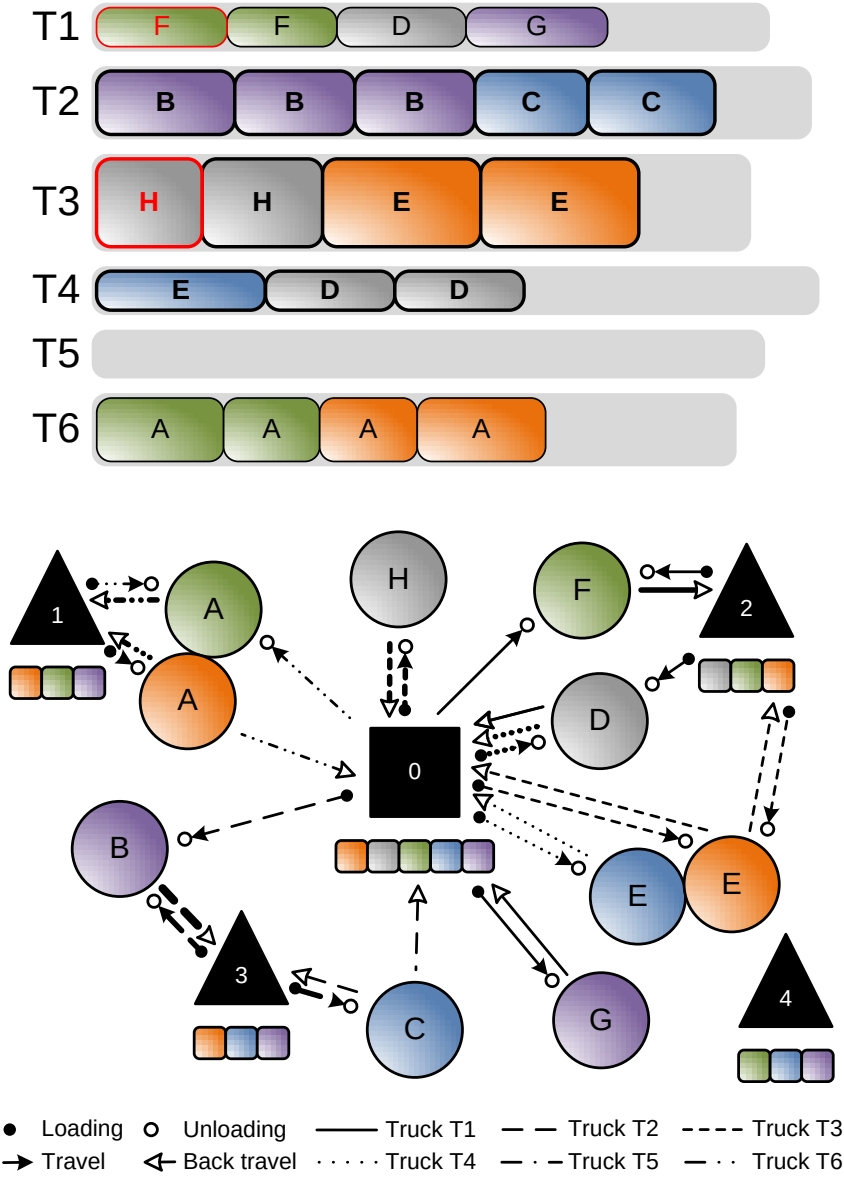

Figure 8.20: Modification of a solution
Construction phase

After a simplified definition, the cement delivery problem has been formulated as a mixed integer linear program. Additional constraints related to the use of local depots and preloaded vehicles have then been added to this basic formulation in order to reflect reality. Following this modeling, four exact and approached solution methods have been described. Composed of three phases, the first solution method solves the cement delivery problem by solving successively three mixed integer linear programs, i.e. three subproblems summarized hereafter. Each order is first allocated to a specific depot. A subset of deliveries is then assigned to each vehicle. The sequence of deliveries is finally determined on each vehicle route. The second solution method solves the cement delivery problem in two phases. Indeed, the allocation of orders to a specific depot is no more determined by solving a mixed integer linear program, but is calculated separately and set as parameter. The modeling of the problem in this method is also improved to better reflect reality. The third solution method solves the cement delivery problem in one phase, i.e. with a unique mixed integer linear program. Based on this formulation, the fourth solution method solves the cement delivery problem also in one phase with a tabu search method. To generate the neighborhood of a solution, i.e. to explore a solutions space, three different operators and additional procedures are called by this metaheuristic.

The four solution methods described in this chapter and implemented in the optimization module will be tested on real life instances of the cement delivery problem in the following chapter. The results will be compared in order to observe the performances of each of these methods.

Chapter 9

Computational experiments and analysis

After having modeled the cement delivery problem and proposed four different solution methods to solve it in Chapter 8, computational experiments have been done with these methods to evaluate their performance. The test environment and the parameter sets are discussed in Section 9.1 while the results of the computational experiments are presented and analyzed in Section 9.2. The chapter ends with a conclusion in Section 9.3.

9.1 Test environment and parameter sets

CPLEX interactive optimizer 12.2.0.0 64-bit has been used to solve the mixed integer linear programs initially generated by the three first solution methods coded in Delphi while the fourth solution method, i.e. the tabu search method, has been entirely implemented in Delphi. For the experiments, all these methods have been run on a PC (Intel Core i5-750 @ 2.66 GHz, 8M Cache, 4GB RAM) using Windows 7 Enterprise Edition 64-bit.

The data of ten instances which correspond to real life orders spread over two work weeks have been provided by the cement factory. For each instance, Table

9.1 indicates the number $|I|$ of orders, the number $|C|$ of customers, the number $|VP|$ of preloaded vehicles, and the minimum, average, and maximum total demand $\sum_{i \in I_c} d_i$ of the customers. The minimum, average and maximum quantity d_i of an order are also reported. Note that the total demand of a customer is typically larger than the quantity asked in an individual order since $|I_c| \geq 1$ for every customer c. The minimum (average and maximum) vehicle capacity is 20'360 (26'424 and 29'700). There are 14 local depots (railway stations) in addition to the central depot.

Table 9.1: Characteristics of the test set

				$\sum_{i \in I_c} d_i$			d_i		
Instance	$\|I\|$	$\|C\|$	$\|VP\|$	Min.	Avg.	Max.	Min.	Avg.	Max.
1	24	21	0	20'000	44'105	140'000	20'000	38'592	137'000
2	29	21	1	20'000	57'514	138'000	20'000	41'648	120'000
3	31	25	0	20'000	48'676	128'000	20'000	39'255	84'000
4	25	22	0	20'000	58'078	224'700	20'000	51'108	200'000
5	29	27	3	20'000	55'764	168'000	20'000	51'918	168'000
6	33	27	2	20'000	45'886	89'100	20'000	37'543	89'100
7	28	25	1	20'000	48'208	112'000	20'000	43'043	84'000
8	27	24	5	20'000	46'294	112'000	20'000	41'150	112'000
9	29	25	3	20'000	51'872	112'000	20'000	44'718	112'000
10	20	18	5	28'000	71'965	168'000	20'000	64'769	168'000

For the three phase method, two phase method and one phase method, the maximal number of different depots that a vehicle can use to do its deliveries has been set to 2 in order to create only vehicle routes centralized around a few depots, while the maximal number of different vehicles used to satisfy all orders of a same customer has been set to 10 in order to avoid being restrictive for finding feasible solutions. For this purpose, $Nd = 2$ in constraints (8.34) and $Nc = 10$ in constraints (8.7) defined in Chapter 8. The objective (8.24) has been chosen for the second (first) subproblem of the three phase method (two phase method) and the similar objective (8.47) for the one phase method. In both cases, λ has been fixed to 0.1. Constraint (8.9) has been removed (which is equivalent to set $Nk = \infty$) since the total number of vehicles used to make the deliveries is minimized in the considered objective functions.

As explained in Section 8.3 of previous chapter, parameter θ in the three phase method and two phase method helps to avoid situations where a vehicle $k \in V^L$ is used for more than A_k time units. While the use of $\theta = 1$ in constraints (8.29) and (8.43) ensures that there will be no overtime, such a setting can be too restrictive.

The three phase method and the two phase method have been run with θ varying between 0 and 1, with step 0.1. The total overtime (i.e, $\sum_{k \in V} max\{0, \alpha_k - A_k\}$) is represented in Figure 9.1, using box-and-whisker plots. More precisely, for each value of θ, the 10 total overtimes (one per instance) produced by the three phase method and the two phase method are considered and a box between the lower quartile Q_1 and the upper quartile Q_2 is constructed, with a solid line drawn across the box to locate the median. A value smaller than $(Q_1 - 1.5(Q_2 - Q_1))$ or larger than $(Q_2 + 1.5(Q_2 - Q_1))$ is defined as an outlier and is plotted using an empty circle. Two 'whiskers' are attached at the top and at the bottom of the box: the lower (upper) whisker ends at the minimal (maximal) value that is not an outlier. Note that there is no value of $\theta > 0.6$ in the figure for the three phase method and therefore no corresponding box-and-whisker plots. Indeed, some instances are infeasible with such values of θ.

For the three phase method (two phase method), no violation is observed with $\theta = 0.6$ ($\theta \geq 0.9$) while the total overtime reaches 295 (350) time units when $\theta = 0$. The company has informed that union rules allow an overtime of 120 time units per week per driver. It turns out that if one driver is assigned to each vehicle, then the solutions obtained with $0.3 \leq \theta \leq 0.6$ ($\theta \geq 0.5$) for the three phase method (two phase method) satisfy this condition. Table 9.2 provides more details on the overtime obtained when using the three phase method with $\theta = 0.3$ and the two phase method with $\theta = 0.5$. For each instance, the total (i.e. $\sum_{k \in V} max\{0, \alpha_k - A_k\}$) and the maximum overtime are indicated.

Table 9.2: Overtime

	Three phase method ($\theta = 0.3$)			Two phase method ($\theta = 0.5$)		
Instance	Total	Maximum	Average	Total	Maximum	Average
1	0	0	0.00	25	25	2.27
2	40	25	3.08	0	0	0.00
3	65	50	5.42	105	60	8.08
4	80	45	6.15	5	5	0.38
5	80	50	5.00	85	45	5.31
6	25	25	1.79	5	5	0.33
7	0	0	0.00	0	0	0.00
8	80	75	6.67	30	20	2.50
9	25	20	1.79	0	0	0.00
10	0	0	0.00	25	25	1.92

Regarding the tabu search method, the current solution has been set to be modified after 200 iterations if there is meanwhile no improvement of the best

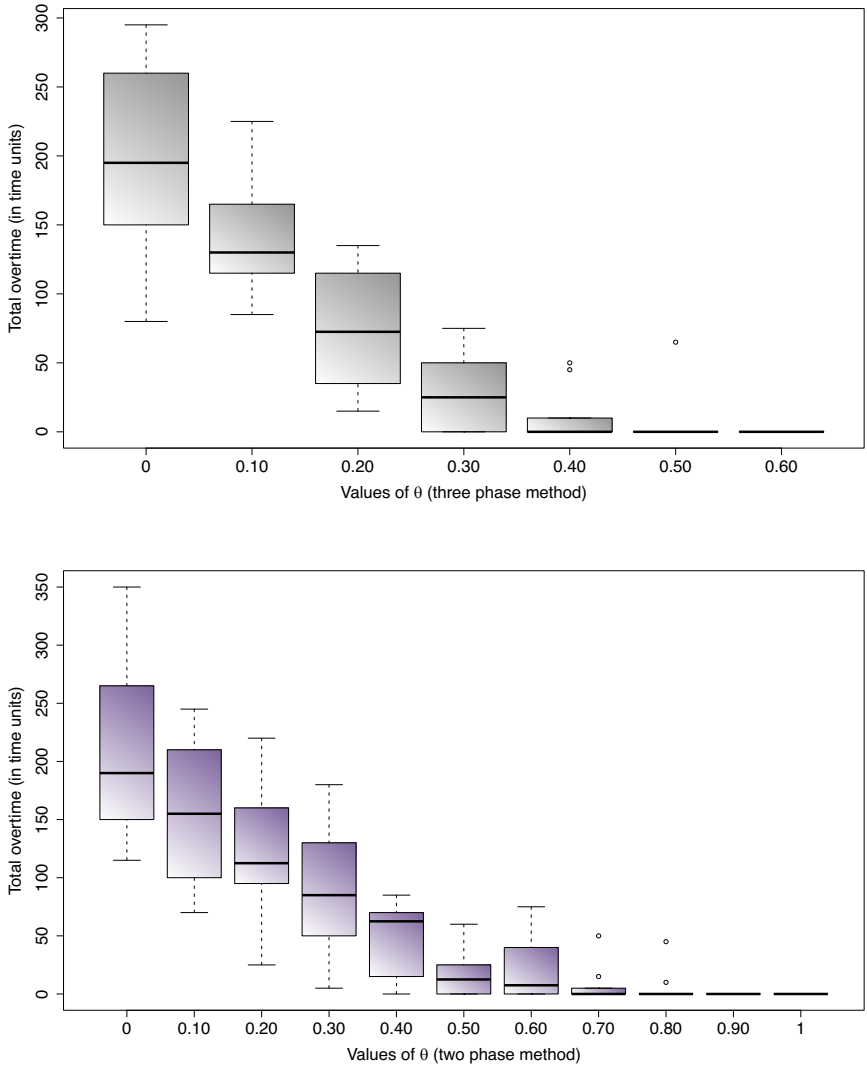

Figure 9.1: Relation between θ and the total overtime

solution and the method has been set to stop if there is no improvement of the best solution following 10'000 iterations. This allows 50 diversifications of a current solution before achieving the stopping criteria. In each iteration, the entire neighborhood has been set to be generated by the operators in order to have an intensified search process. To resume, $iter_{max}$ = 10'000, n_N = 0 (i.e. all the neighborhood) and $iter_{mod}$ = 200 in procedure (1) presented in Chapter 8. Furthermore, the tabu search method has been tested with three different tabu list sizes: n_{TA} = 8, 15 and 20. The value 15 corresponds to the average between the minimal (13) and maximal (17) number of vehicles used by the cement factory to do the deliveries for the ten instances. The values 8 and 20 respectively correspond to a lower and upper bound of this average. While the value 8 represents the maximal number of deliveries required to satisfy the demand of a customer, the value 20 represents the minimal number of daily orders among the 10 instances. Due to the stochasticity used in the tabu search method to modify a current solution, each instance has been solved ten times.

9.2 Computational results

From now on, $3P_{0.6}$ and $3P_{0.3}$ denote the three phase method used with $\theta = 0.6$ and $\theta = 0.3$ while $2P_{1.0}$ and $2P_{0.5}$ indicate the two phase method used with $\theta = 1$ and $\theta = 0.5$. 1P corresponds to the one phase method while TS_8, TS_{15} and TS_{20} denote the tabu search method used with n_{TA} = 8, 15 and 20. Following a presentation and a global analysis of the results obtained with all solution methods in Subsection 9.2.1, a further analysis related to the tabu search method (i.e. TS_8, TS_{15} and TS_{20}) is discussed in Subsection 9.2.2. A summary of these analysis is finally described in Subsection 9.2.3. The detail of the data used to write this section is presented in Appendix G.

9.2.1 Presentation and global analysis

Computing times are compared in Table 9.3. For $3P_{0.6}$ and $3P_{0.3}$ ($2P_{1.0}$ and $2P_{0.5}$), only the second (first) subproblem is considered since the optimal solutions of the other subproblems were all found in less than one second. Furthermore, only information about the best of the ten solutions obtained for a same instance is indicated for TS_8, TS_{15} and TS_{20}. For each instance, the time (in seconds) needed by CPLEX for finding an optimal solution and proving its optimality or needed by the tabu search method for finding a good feasible solution is shown. If no proof of

optimality could be obtained after 6 hours of computation, Table 9.4 indicates for the methods using CPLEX the gap between the value of the best feasible solution and the best lower bound. In nine cases (namely, instances 1, 3, 4, 5, 6, 7, 8, 9 and 10), 1P was stopped before the time limit of 6 hours because the computer ran out of memory, and the corresponding computing times are shown in italic in Table 9.3.

Table 9.3: Computing times

Instance	$3P_{0.6}$	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P	TS_8	TS_{15}	TS_{20}
1	3	299	17	295	*17'655*	11'895	7'224	9'455
2	1'963	3'717	> 6 hours	787	> 6 hours	20'919	6 hours	6 hours
3	2	121	5	1'022	*6'805*	6 hours	6 hours	6 hours
4	301	5	16	10'789	*6'226*	6 hours	14'481	6 hours
5	64	78	> 6 hours	113	*9'978*	6 hours	6 hours	21'382
6	23	100	12	15'153	*15'316*	6 hours	6 hours	6 hours
7	6	> 6 hours	5	5	*18'535*	14'444	16'122	16'921
8	158	> 6 hours	409	115	*5'296*	18'712	20'869	13'281
9	4'861	4'725	> 6 hours	2'536	*18'356*	6 hours	6 hours	6 hours
10	14	48	258	598	*4'726*	4'720	6'177	9'056

Table 9.4: Gaps to optimality

Instance	$3P_{0.6}$	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P
1	0.00%	0.00%	0.00%	0.00%	0.59%
2	0.00%	0.00%	0.16%	0.00%	0.89%
3	0.00%	0.00%	0.00%	0.00%	5.08%
4	0.00%	0.00%	0.00%	0.00%	1.95%
5	0.00%	0.00%	0.03%	0.00%	8.51%
6	0.00%	0.00%	0.00%	0.00%	1.32%
7	0.00%	0.00%	0.00%	0.00%	0.78%
8	0.00%	0.02%	0.00%	0.00%	1.29%
9	0.00%	0.32%	0.13%	0.00%	1.51%
10	0.00%	0.00%	0.00%	0.00%	8.52%

The three phase method and the two phase method have produced feasible solutions for the ten instances and most of these solutions (10 for $3P_{0.6}$, 8 for $3P_{0.3}$, 7 for $2P_{1.0}$ and 10 for $2P_{0.5}$) are proven optimal. By analyzing the output file generated by CPLEX, it can be observed that the 10 optimal solutions found by $3P_{0.6}$, the 8 optimal solutions found by $3P_{0.3}$, the 7 optimal solutions found by $2P_{1.0}$ and 8 of the 10 optimal solutions found by $2P_{0.5}$ have been produced in less than 15

minutes, the hardest and longest task being to prove the optimality. For comparison, the one phase method has generated 10 feasible solutions, and no proof of optimality could be obtained in 6 hours of computation. This however does not mean that $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$ or $2P_{0.5}$ should be preferred to 1P. Indeed, the value of the optimal solution found by $3P_{0.6}$ and $2P_{1.0}$ is an upper bound on the optimal value that 1P tries to determine. This is also the case for $3P_{0.3}$ and $2P_{0.5}$ if they deliver a solution satisfying constraints (8.45). However, if $3P_{0.3}$ and $2P_{0.5}$ generate a solution with overtime (this is the case for instances 2, 3, 4, 5, 6, 8 and 9 with $3P_{0.3}$ and for instances 1, 3, 4, 5, 6, 8, 10 with $2P_{0.5}$), then the optimal value that 1P is looking for is possibly strictly larger than the value of the solution generated by $3P_{0.3}$ and $2P_{0.5}$.

Table 9.5: Characteristics of the integer linear programs

	$3P_{0.3}$			$2P_{0.5}$			1P		
Instance	Rows	Cols	NZs	Rows	Cols	NZs	Rows	Cols	NZs
1	793	549	2'526	788	547	2'538	8'525	7'933	47'390
2	1'241	843	3'662	1'223	842	3'703	14'169	13'393	80'320
3	1'364	881	4'016	1'364	881	4'120	16'776	16'055	97'260
4	1'085	707	3'238	1'085	707	3'308	11'298	10'780	63'678
5	1'428	981	4'325	1'428	981	4'437	16'559	15'558	93'478
6	1'552	1'069	4'694	1'552	1'069	4'797	20'520	19'416	117'846
7	1'254	862	3'870	1'253	861	3'964	14'399	13'187	80'026
8	1'146	838	3'575	1'160	846	3'689	13'491	12'895	77'444
9	1'366	981	4'244	1'368	982	4'336	16'819	15'690	94'823
10	1'202	811	3'517	1'152	806	3'496	8'855	8'328	47'964
Average	1'243	852	3'767	1'237	852	3'839	14'141	13'323	80'023

The large computing times for 1P when compared to $3P_{0.3}$ and $2P_{0.5}$ can be explained by the size of the integer linear programs that have to be solved. Table 9.5 indicates the number of rows, columns and nonzero coefficients of the integer linear programs associated with the second phase of $3P_{0.3}$, with the first phase of $2P_{0.5}$ and with 1P. Note that the number of rows, columns and nonzeros are on average multiplied, respectively, by a factor 11, 16 and 21 between $3P_{0.3}$ or $2P_{0.5}$ and 1P.

The quality of the solutions produced by $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$, $2P_{0.5}$, 1P, TS_8, TS_{15} and TS_{20} are now compared. The value of the solutions implemented by the cement factory (columns CF), that have been obtained manually, are also reported. For each instance, Table 9.6 estimates this quality while Tables 9.8, 9.9 and 9.10 respectively indicate the total value of the variables v_{ck} (which is the main objec-

tive to minimize), the total number $\sum_{k \in V} w_k$ of vehicles used (which is the second most important objective), and the total weighted delivery time $\sum_{k \in V} F_k \alpha_k$ (which is the third objective).

Table 9.6: Solution values: Gap* (%)

Instance	CF	$3P_{0.6}$	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P	TS_8	TS_{15}	TS_{20}
1	20.33%	13.56%	10.20%	11.68%	10.20%	0.59%	0.59%	0.59%	0.59%
2	51.83%	18.72%	9.94%	17.42%	10.40%	0.89%	1.05%	1.44%	1.55%
3	33.12%	8.46%	1.48%	3.39%	1.78%	5.08%	4.33%	3.57%	2.68%
4	14.34%	17.16%	1.50%	6.96%	1.46%	1.95%	0.80%	0.85%	0.80%
5	19.07%	18.74%	8.14%	13.61%	4.66%	8.51%	6.16%	5.49%	5.14%
6	41.49%	4.55%	1.35%	4.08%	1.73%	1.32%	0.98%	0.98%	0.96%
7	25.30%	12.47%	1.57%	7.12%	1.57%	0.78%	1.00%	1.03%	1.03%
8	38.85%	14.92%	4.92%	14.11%	4.90%	1.29%	0.87%	0.87%	0.83%
9	40.72%	23.85%	12.76%	18.08%	5.13%	1.51%	0.91%	0.91%	0.91%
10	41.12%	23.22%	9.77%	10.18%	8.54%	8.52%	8.54%	8.54%	8.54%

Table 9.7: Number of solutions where TS_8, TS_{15} and TS_{20} are not better

	TS_8				TS_{15}				TS_{20}			
Instance	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P
1				2								
2				4				9				7
3	9	9	9		7	7	7		5	4	5	
5			10		1		9		1		7	1
7				6				7				7
10			3	8			2	8			2	9

The three combined objectives are first analyzed (cf. Table 9.6). Rather than comparing the values of the objective function related to the solutions given by the cement factory and provided by the solution methods, the gaps between these values and the lower bound estimated by CPLEX for 1P (denoted Gap*) have been computed to facilitate the analysis. Solutions obtained with $3P_{0.3}$ ($2P_{0.5}$) are strictly better than those obtained with $3P_{0.6}$ ($2P_{1.0}$). The solution methods generally produce better solutions than those given by the cement factory, except for instance 4 with $3P_{0.6}$. Note that solutions of $3P_{0.3}$ and $2P_{0.5}$ are sometimes very close. Indeed, the difference between their gap* is lower than 1% (for instances 1, 2, 3, 4, 6, 7, 8). 1P is strictly better than $3P_{0.6}$ for all instances and than $3P_{0.3}$, $2P_{1.0}$ and $2P_{0.5}$ for instances 1, 2, 6, 7, 8, 9, 10. The average of ten solutions is displayed for TS_8, TS_{15} and TS_{20}. The tabu search method is strictly better than CF, $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$

and $2P_{0.5}$, except however for instance 3 and 5. Indeed, $3P_{0.3}$ and $2P_{0.5}$ are better than TS_8, TS_{15} and TS_{20}, and $2P_{1.0}$ is better than TS_8 and TS_{15} for instance 3, while $2P_{0.5}$ is better than TS_8, TS_{15} and TS_{20} for instance 5. On average, TS_8, TS_{15} and TS_{20} are also better than 1P, except for instances 2, 7 and 10. Although the tabu search method always produces better results than the other solution methods for instances 4, 6, 8 and 9, this is not always the case for instance 1, 2, 3, 5, 7 and 10. Table 9.7 therefore presents for TS_8, TS_{15} and TS_{20} the number of solutions where $3P_{0.3}$, $2P_{1.0}$, $2P_{0.5}$ and 1P are better for these instances.

Table 9.8: Solution values: $\sum v_{ck}$

Instance	$\|I\|$	$\|C\|$	CF	$3P_{0.6}$	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P	TS_8	TS_{15}	TS_{20}
1	24	21	25	23	23	23	23	21	21	21	21
2	29	21	32	24	23	24	23	21	21	21	21
3	31	25	33	26	25	25	25	26	26	26	25
4	25	22	27	27	24	25	24	24	24	24	24
5	29	27	32	31	29	30	28	29	28	28	28
6	33	27	38	27	27	27	27	27	27	27	27
7	28	25	31	27	25	26	25	25	25	25	25
8	27	24	33	27	25	27	25	24	24	24	24
9	29	25	35	30	28	29	26	25	25	25	25
10	20	18	26	22	20	20	20	20	20	20	20

Table 9.9: Solution values: $\sum w_k$

Instance	CF	$3P_{0.6}$	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P	TS_8	TS_{15}	TS_{20}
1	13	15	11	13	11	10	10	10	10
2	16	18	13	17	14	13	13	14	13
3	16	17	12	16	13	13	12	12	12
4	15	18	13	15	13	13	12	12	12
5	17	20	16	19	16	17	16	15	16
6	17	19	14	19	15	14	13	13	13
7	17	19	15	17	15	13	14	14	14
8	17	16	12	15	12	13	12	12	12
9	17	19	14	17	14	13	13	13	13
10	17	19	14	16	13	13	13	13	13

The main objective which is to minimize the number of different vehicles used to supply a same customer ($\sum_{c \in C} \sum_{k \in V} v_{ck}$) is then analyzed (cf. Table 9.8). In other words, the cement factory would ideally like to deliver all orders of a customer with a unique vehicle. Note that TS_{20} has found 7 such ideal solutions (instances 1, 2, 3, 6, 7, 8, 9), i.e. one ideal solution more than 1P, TS_8 and TS_{15} (in-

stances 1, 2, 6, 7, 8, 9), while the three phase method has found 1 (with $\theta = 0.6$) and 3 (with $\theta = 0.3$) such solutions, and the two phase method has found 2 (with $\theta = 1$) and 3 (with $\theta = 0.5$) such solutions. For comparison, the solutions implemented by the cement supplier company are very far from such an ideal situation. Indeed, if instance 6 is for example considered, $\sum_{c \in C} \sum_{k \in V} v_{ck} = 38$ while the number $|C|$ of customers is equal to 27. TS_8, TS_{15} and TS_{20} sometimes produce lower values than 1P which produces lower values than $2P_{0.5}$ (except for instance 3 and 5) for this objective. $2P_{0.5}$ is always better than $2P_{1.0}$, $3P_{0.3}$ and $3P_{0.6}$. Although $2P_{1.0}$ is better than $3P_{0.6}$, $3P_{0.3}$ is better than $2P_{1.0}$. As mentioned before, only information about the average of ten solutions obtained for each instance is indicated for TS_8, TS_{15} and TS_{20}. However, all solutions provided by these methods are better than 1P for this main objective.

The second objective which is to use as few vehicles as possible is then analyzed (cf. Table 9.9). In 9 cases (instances 1, 2, 3, 4, 5, 6, 7, 9, 10), the solution implemented by the company uses less vehicles than the one produced by $3P_{0.6}$ and in 3 cases (instances 2, 5, 6), less vehicles than the one produced by $2P_{1.0}$. However, for this second objective, $3P_{0.3}$ is always strictly better than $3P_{0.6}$ and CF, and $2P_{0.5}$ is always strictly better than $2P_{1.0}$ and CF, while 1P is at least as good as $3P_{0.3}$ and $2P_{0.5}$ (and even better in 3 cases), with the exception of instances 3, 5 and 8. Note that 1P ran out of memory for instances 3 and 5 while the gap to optimality was larger than 5%. Finally, TS_8, TS_{15} and TS_{20} are better for this objective than all other methods and than solutions of the cement factory, except sometimes for instances 2 and 7.

Table 9.10: Solution values: $\sum F_k \alpha_k$

Instance	CF	$3P_{0.6}$	$3P_{0.3}$	$2P_{1.0}$	$2P_{0.5}$	1P	TS_8	TS_{15}	TS_{20}
1	5’940	14’780	5’010	8’660	5’050	4’535	4’555	4’535	4’535
2	7’430	19’990	6’280	14’305	6’355	5’760	6’850	7’920	7’470
3	7’445	20’950	11’025	14’025	10’365	6’225	6’390	6’390	8’150
4	7’260	19’550	6’485	11’610	6’145	9’855	5’690	5’725	5’650
5	8’180	26’960	7’645	15’680	7’860	7’795	10’265	10’880	10’190
6	7’485	18’820	6’460	14’745	6’685	6’240	6’340	6’030	6’150
7	7’195	19’860	6’065	13’730	6’050	5’790	5’765	5’700	5’675
8	6’920	9’385	5’890	6’215	5’720	5’475	5’280	5’285	5’285
9	7’860	19’295	6’550	9’540	6’460	11’075	6’220	6’160	6’185
10	7’440	15’395	10’335	6’615	5’840	5’765	5’865	5’840	5’840

The comparison of the total weighted delivery times does not go exactly in the same direction as the second objective due to the delivery time that is penalized if

vehicles belonging to a third party are used (cf. Table 9.10). Indeed, the solutions implemented by the cement factory have a strictly smaller total weighted delivery time than those produced by $3P_{0.6}$ and $2P_{1.0}$, with the exception of instances 8 and 10, where this time is larger by 705 and 825 minutes (10.2% and 11.1%) relative to $2P_{1.0}$. The total delivery time of the solutions found with $3P_{0.3}$ ($2P_{0.5}$) are always strictly better than those obtained with $3P_{0.6}$ ($2P_{1.0}$) or implemented by the cement factory, except however for instance 3 with $3P_{0.3}$ and $2P_{0.5}$, and for instance 10 with $3P_{0.3}$. The gain even reaches 1'310 minutes for instance 9 with $3P_{0.3}$ and 1'600 minutes for instance 10 with $2P_{0.5}$, which respectively corresponds to an improvement of 16.7% and 21.5%. 1P always produces better values than CF, $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$ and $2P_{0.5}$, except for instance 5 where $3P_{0.3}$ is better and for instances 4 and 9 where one or more vehicles belonging to a third party are used and where CF, $3P_{0.3}$ and $2P_{0.5}$ are better. Values of TS_8 are on average better than those obtained with 1P for instances 4, 7, 8 and 9, while values of TS_{15} and TS_{20} are on average better than those obtained with 1P for instances 1, 4, 6, 7, 8 and 9. Considering all solutions provided by the tabu search method, TS_8 is always better than 1P for instances 4, 8, 9, while TS_{15} and TS_{20} are always better than 1P for instances 4, 7, 8, 9.

9.2.2 Further analysis for TS_8, TS_{15} and TS_{20}

Although solutions provided by TS_8, TS_{15} and TS_{20} have been discussed before, the impact of the three different tabu list sizes (n_{TA} = 8, 15 and 20) has not been observed. Indeed, the solutions provided by these methods are relatively similar. Furthermore, information displayed in Tables 9.6, 9.7, 9.8, 9.9 and 9.10 only shows some characteristics on the final results and not on the progression of these results in time. For this purpose, appropriate statistics are presented hereafter for ten solutions obtained with TS_8, TS_{15} and TS_{20}.

Table 9.11 indicates the time (in seconds) needed by TS_8, TS_{15} and TS_{20} for finding solutions that are from 1% of their final value. Different time intervals ($CPU_{\mathfrak{s}}$) are displayed for each instance, followed by the number of solutions ($n_{\mathfrak{s}}$) found in these intervals. Note that between two instances or between two solution methods, the time intervals can vary in order to better partitioned the solutions. Furthermore, the time needed by 1P is also given to allow comparisons.

Table 9.12 displays the gaps between the objective function of the solutions given by the tabu search method (i.e. TS_8, TS_{15} and TS_{20}) and the lower bound estimated by CPLEX for 1P (denoted Gap*) at a run time of 900 seconds.

Table 9.11: Statistics: $CPU_{\mathfrak{s}}$ at 1% of the final value

Instance	1P $CPU_{\mathfrak{s}}$	TS_8 $CPU_{\mathfrak{s}}$	$n_{\mathfrak{s}}$	TS_{15} $CPU_{\mathfrak{s}}$	$n_{\mathfrak{s}}$	TS_{20} $CPU_{\mathfrak{s}}$	$n_{\mathfrak{s}}$
1	508	[0,900]	3	[0,900]	7	[0,900]	8
		(900,3'600]	5	(900,3'600]	2	(900,2'500]	2
		(3'600,4'700]	2	(3'600,4'200]	1		
2	1'557	[0,900]	1				
		(900,3'600]	5	(900,3'600]	4	(900,3'600]	4
		(3'600,7'200]	2	(3'600,7'200]	1	(3'600,7'200]	5
		(7'200,13'000]	2	(7'200,19'200]	5	(7'200,7'400]	1
3	1'518	[0,900]	9				
		(900,2'700]	1	(900,3'600]	7	(900,3'600]	4
				(3'600,18'500]	3	(3'600,7'200]	1
						(7'200,20'100]	5
4	1'304	[0,60]	10	[0,40]	10	[0,60]	10
5	5'913	[0,3'600]	1	[0,3'600]	2	[0,3'600]	2
		(3'600,7'200]	3	(3'600,7'200]	2	(3'600,7'200]	5
		(7'200,21'400]	6	(7'200,17'300]	6	(7'200,20'200]	3
6	7'066	[0,900]	3	[0,200]	10	[0,170]	10
		(900,3'300]	7				
7	3'542	[0,900]	5	[0,900]	10	[0,700]	10
		(900,2'700]	5				
8	535	[0,900]	5	[0,900]	9	[0,120]	10
		(900,2'400]	5	(900,1'300]	1		
9	11'532	[0,3'600]	2	[0,900]	2	[0,900]	1
		(3'600,7'200]	5	(900,3'600]	8	(900,3'600]	7
		(7'200,13'300]	3			(3'600,6'300]	2
10	489	[0,900]	10	[0,900]	6	[0,900]	9
				(900,1'400]	4	(900,1'300]	1

Table 9.12: Statistics: Gap* at 900 seconds

Instance	TS_8	TS_{15}	TS_{20}
1	< 4.6%	< 4.6%	< 4.6%
2	< 9.6%	< 8.8%	< 9.6%
3	< 3.3%	< 3.8%	< 3.8%
4	< 1.0%	< 1.0%	< 1.0%
5	< 13.3%	< 13.9%	< 11.3%
6	< 3.9%	< 0.5%	< 0.4%
7	< 4.6%	< 0.8%	< 0.9%
8	< 4.4%	< 4.4%	< 0.5%
9	< 4.1%	< 4.2%	< 4.2%
10	< 0.5%	< 5.7%	< 4.7%

Table 9.13: Statistics: $\sum v_{ck}$

Instance	1P	TS_8	TS_{15}	TS_{20}
1	21	21	21	21
2	21	21	21	[21,22]
3	26	[25,26]	[25,26]	[25,26]
4	24	24	24	24
5	29	[28,29]	[28,29]	[28,29]
6	27	27	27	27
7	25	25	25	25
8	24	24	24	24
9	25	25	25	25
10	20	20	20	20

Table 9.14: Statistics: $\sum w_k$

Instance	1P	TS_8	TS_{15}	TS_{20}
1	10	10	10	10
2	13	[12,14]	[13,14]	[12,13]
3	13	12	12	[12,13]
4	13	[11,12]	[11,12]	[11,12]
5	17	[15,16]	[15,16]	[15,17]
6	14	[12,14]	[13,14]	13
7	13	[13,14]	[13,14]	[13,14]
8	13	12	12	[11,12]
9	13	13	13	13
10	13	13	13	13

Table 9.15: Statistics: $\sum F_k \alpha_k$

Instance	1P	TS_8	TS_{15}	TS_{20}
1	4’535	[4’520,4’675]	[4’530,4’540]	[4’520,4’540]
2	5’760	[5’755,10’135]	[5’755,14’470]	[5’725,10’075]
3	6’225	[5’870,10’415]	[5’885,10’280]	[5’885,10’500]
4	9’855	[5’560,6’080]	[5’560,6’120]	[5’540,5’815]
5	7’795	[7’260,12’195]	[7’350,15’370]	[7’280,11’875]
6	6’240	[5’935,8’695]	[5’915,6’335]	[5’880,6’480]
7	5’790	[5’675,5’825]	[5’560,5’790]	[5’555,5’730]
8	5’475	[5’240,5’320]	[5’245,5’320]	[5’230,5’350]
9	11’075	[6’120,6’345]	[6’035,6’260]	[6’035,6’300]
10	5’765	[5’720,6’120]	[5’765,6’070]	[5’765,5’965]

For the solutions obtained with 1P, TS_8, TS_{15} and TS_{20}, Tables 9.13, 9.14 and 9.15 respectively indicate the number or the interval related to the different vehicles used to supply a same customer ($\sum_{c \in C} \sum_{k \in V} v_{ck}$), the vehicles used to do the deliveries ($\sum_{k \in V} w_k$), and the total weighted delivery times ($\sum_{k \in V} F_k \alpha_k$).

To better observe the performance of the tabu search method, statistics presented before are illustrated in a figure and discussed in detail hereafter for instance 4 and in Appendix F for all instances.

Statistics related to instance 4

Solutions obtained with TS_8, TS_{15}, TS_{20} and 1P for instance 4 are considered in Figure 9.2. A plot first displays the progression of the results provided by TS_8, TS_{15} and TS_{20} in time. The x-axis shows the computational time in seconds (CPU) while the y-axis displays the value of $f(\mathfrak{s})$, i.e. the objective function. The progression of a solution is displayed in orange for TS_8, in blue for TS_{15} and in green for TS_{20}. Note that for each method, 10 progressions related to 10 resolutions of an instance are drawn. The progression of the solution provided by 1P is also displayed, in red, in order to be compared with the results of the tabu search method. Furthermore, the value of the objective function related to the solution given by the cement factory (CF) and the estimated lower bound (LB) computed by CPLEX for 1P are finally mentioned in the plot with dotted lines. Box-and-whisker plots are then used to present for the final results of TS_8, TS_{15} and TS_{20} the value of the objective function and related to the three different objectives, i.e. the total value of the variables v_{ck}, the total number $\sum_{k \in V} w_k$ of vehicles used, and the total weighted delivery time $\sum_{k \in V} F_k \alpha_k$. Note that a red line indicates the value for the result obtained with 1P.

For TS_8, TS_{15} and TS_{20}, all solutions are from 1% of their final value respectively before 60, 40 and 60 seconds. Note that the solution provided by 1P reaches 1% of its final value at 1'304 seconds and is always worse than solutions provided by TS_8, TS_{15} and TS_{20} along the time progression.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are always better than the one computed by CPLEX for 1P. While the box-and-whisker plots related to the first objective ($\sum v_{ck}$) show that solutions of TS_8, TS_{15} and TS_{20} are similar from this point of view, the box-and-whisker plots related to the second objective ($\sum w_k$) indicate that solutions of TS_8, TS_{15}, TS_{20} use between 1 and 2 vehicles less than the solution given by 1P. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the

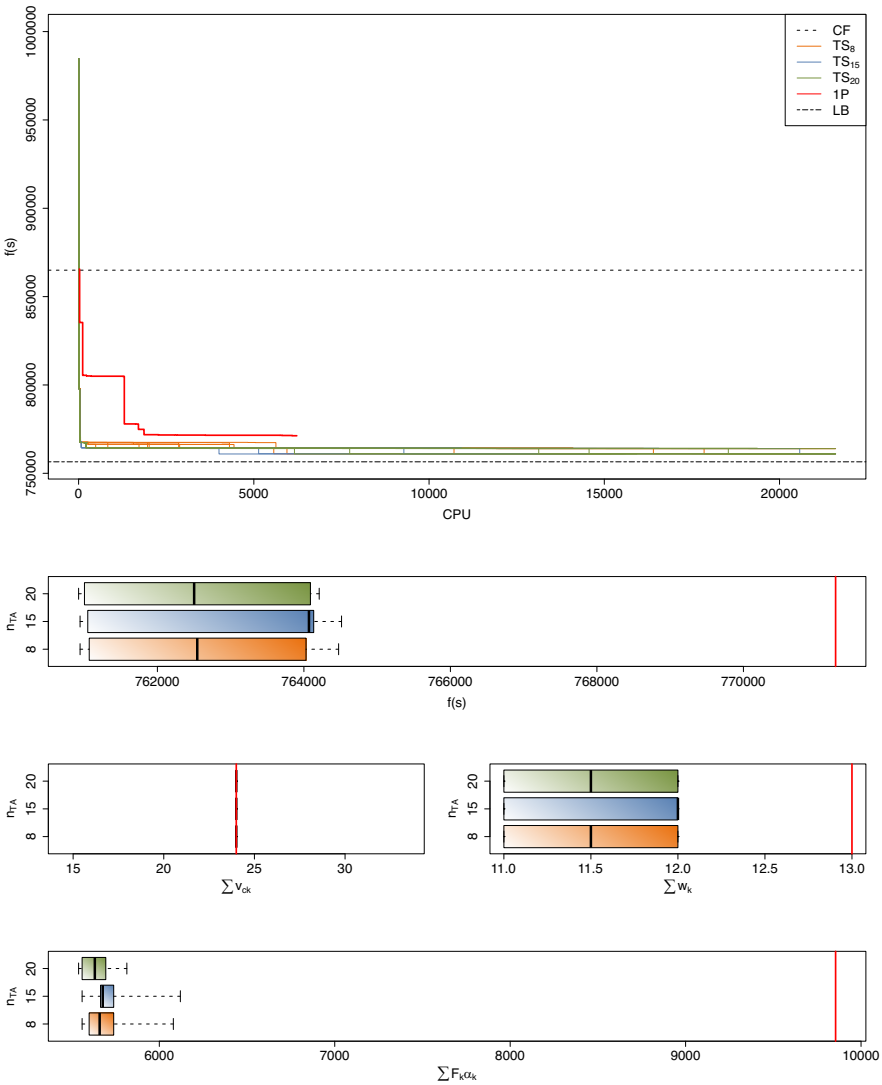

Figure 9.2: Statistics related to instance 4

total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'560 and 6'080 minutes, between 5'560 and 6'120 minutes, and between 5'540 and 5'815 minutes (while 1P, which uses third party vehicles, gives 9'855 minutes).

9.2.3 Analysis summary

The following comments summarize the analysis discussed in Subsections 9.2.1 and 9.2.2 and point out the advantages and drawbacks of each method.

- **$3P_{0.6}$** This method always produces optimal solutions which are generally obtained within a few seconds or minutes. Although these solutions are better or similar (for instance 4) than that implemented by the cement factory for the main objective ($\sum_{c\in C}\sum_{k\in V} v_{ck}$), they are worse for the second ($\sum_{k\in V} w_k$) and third ($\sum_{k\in V} F_k\alpha_k$) objective. There is however one exception. Indeed, the solution provided by $3P_{0.6}$ for instance 8 requires one vehicle less than the solution implemented by the cement factory (16 instead of 17). Considering the main objective, a maximal decrease of 11 units (28.9%) can be observed for instance 6.

- **$3P_{0.3}$** This method is strictly better than $3P_{0.6}$. Optimal solutions are generally obtained within a few minutes. These solutions are better than that implemented by the cement factory for the first ($\sum_{c\in C}\sum_{k\in V} v_{ck}$) and second ($\sum_{k\in V} w_k$) objective. However, this is not always the case for the third objective ($\sum_{k\in V} F_k\alpha_k$). Indeed, instances 3 and 10 are larger (respectively by 3'580 and 2'915 minutes) due to the use of vehicles belonging to a third party. Another drawback of $3P_{0.3}$ is that it produces solutions with a possible overtime, which means that constraints (8.45) are not necessarily satisfied. But the total violation observed typically does not exceed the limit imposed by union rules.

- **$2P_{1.0}$** This method is strictly better than $3P_{0.6}$ and always provides feasible solutions. These ones are generally obtained within a few seconds or minutes. As for $3P_{0.6}$, solutions of $2P_{1.0}$ are better than that implemented by the cement factory for the main objective ($\sum_{c\in C}\sum_{k\in V} v_{ck}$) and worse for the third objective ($\sum_{k\in V} F_k\alpha_k$) with some exceptions. For example, instances 8 and 10 are smaller (respectively by 705 and 825 minutes, i.e. 10.2% and 11.1%). Regarding the second objective ($\sum_{k\in V} w_k$), solutions of $2P_{1.0}$ are

similar or better from this point of view than that implemented by the cement factory, except for instances 2, 5 and 6 which use one or two vehicles more (respectively 17 instead 16, and 19 instead 17). However, a decrease of 8, 2 and 11 units can respectively be observed in the main objective.

- **$2P_{0.5}$** This method is strictly better than $2P_{1.0}$. Furthermore, the solutions are better than that implemented by the cement factory for all objectives, except however for the third objective ($\sum_{k \in V} F_k \alpha_k$) of instance 3 where vehicles belonging to a third party are used. If instance 3 is not considered, the total weighted delivery time of the solutions generated by $2P_{0.5}$ is in average 14.7% smaller than that implemented by the cement factory and the number of required vehicles is reduced by up to 5 units (instance 8). These good solutions are all obtained within a few minutes. The only drawback of $2P_{0.5}$ (except the use of vehicles belonging to a third party for instance 3) is that it produces solutions with a possible overtime (as for $3P_{0.3}$), which means that constraints (8.45) are not necessarily satisfied. The total violation observed typically does not exceed the limit imposed by union rules, and such a violation can induce a significant improvement in some components of the objective function. A typical example is instance 10 where $2P_{0.5}$ has produced a solution that requires strictly the same number of vehicles than other methods producing the best value for this second objective ($\sum_{k \in V} w_k$).

- **1P** This method ran out of memory in 9 out of 10 cases, has not obtained any proof of optimality, but has produced 10 feasible solutions of high quality. When comparing the solutions of the cement factory with those produced by 1P, a decrease of the total weighted delivery time by more than 22% can be observed for instances 1, 2 and 10 (the average decrease is 18.3% if instances 4 and 9 that use vehicles belonging to a third party are not considered), while the gain in the main objective ($\sum_{c \in C} \sum_{k \in V} v_{ck}$) reaches 11 units (34.4%) for instance 2, and the reduction in the number of required vehicles reaches 4 units (23.5%) for instances 7, 8, 9 and 10. Instances 3, 4 and 5 are the unique exceptions where 1P is slightly worse than $3P_{0.3}$, $2P_{1.0}$ (only for instance 3) and $2P_{0.5}$ when considering the three combined objectives, but this is because 1P ran out of memory while the gap to optimality was larger than 1.9%. The unique drawback of 1P is that the integer linear program is much harder to solve than that of the three phase method or two phase method, and a proof of optimality can hardly be produced within a few hours.

- **TS_8** Although no proof of optimality can be provided by this method, this one produced solutions of very high quality. Considering the three different objectives separately, these solutions are typically (always for instances 4, 8 and 9) better than that implemented by the cement factory and those given by $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$, $2P_{0.5}$ and 1P. Better results were however never obtained for the third objective ($\sum_{k \in V} F_k \alpha_k$) of instances 3 and 5. Despite hours of computation, good solutions are already found in a few minutes. When comparing the results of the cement factory with those obtained for TS_8, the gain in the second ($\sum_{k \in V} w_k$) and third ($\sum_{k \in V} F_k \alpha_k$) objective can respectively reach 5 units for instance 6 and 8 (29.4%), and 1'740 minutes for instance 9 (22.1%).

- **TS_{15}** This method also provided very good results which are relatively similar to those obtained with TS_8. Compared to TS_8, TS_{15} typically found for each instance better solutions than that implemented by the cement factory and those given by $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$, $2P_{0.5}$ when considering all objectives separately or combined. Furthermore, the gain in the third objective ($\sum_{k \in V} F_k \alpha_k$) can reach 1'825 minutes for instance 9 (23.2%). Despite hours of computation, good results are also produced in a few minutes and are generally better than those provided by TS_8 during the same time period. For example, all solutions of TS_{15} for instance 6 are from less than 1% of their final value before 200 seconds, while all solutions of TS_8 for this instance reach this percentage between 500 and 3'300 seconds.

- **TS_{20}** As TS_8 and TS_{15}, this method produced solutions of very high quality. Again, the results of TS_{20} are frequently (always for instances 4, 8 and 9) better than that implemented by the cement factory and those given by $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$, $2P_{0.5}$ and 1P, and relatively similar with the solutions of TS_8 and TS_{15}, when considering all objectives separately or combined. Note that the gain in the second ($\sum_{k \in V} w_k$) and third ($\sum_{k \in V} F_k \alpha_k$) objective can respectively reach 6 units for instance 8 (35.3%), and also 1'825 minutes for instance 9 (23.2%). Despite hours of computation, good results are already produced in a few minutes and are generally better than those provided by TS_8 and TS_{15} during the same time period. For example, all solutions of TS_{20} for instance 8 are from less than 1% of their final value before 120 seconds, while all solutions of TS_8 and TS_{15} for this instance reach this percentage between 200 and 2'400 seconds.

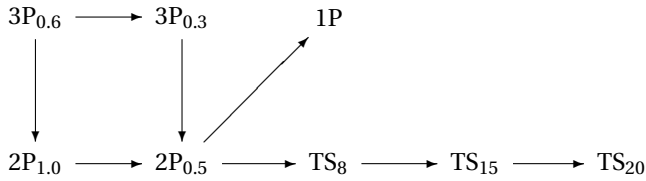

Figure 9.3: Ranking of the solution methods

Figure 9.3 presents with a diagram a ranking or a preference ordering of the solution methods summarized before. Each solution method positioned at an arrowhead is strictly (for $3P_{0.3}$, $2P_{1.0}$ and $2P_{0.5}$) or typically (for 1P, TS_8, TS_{15} and TS_{20}) better than the solution method being at the tail of the arrow.

9.3 Conclusion

In this chapter, the four solution methods implemented in the optimization module have been tested on instances representative of the daily workload of the cement factory. For this purpose, the test environment and the parameter sets have first been discussed. Although each instance has been solved with a value of θ varying between 0 and 1 for the three phase method and the two phase method, only solutions obtained with two different values of this parameter, which helps to avoid situations where a vehicle $k \in V^L$ is used more than A_k time units, have been selected. While the first value chosen for θ does not allow overtime, the second one allows a maximal overtime of 120 minutes per week, as authorized by the union rules. Concerning the tabu search method, each instance has been solved ten times with three different tabu list sizes. To resume, the results of eight different solution methods have been selected for the analysis. Following this selection, a presentation and a global analysis of the computational results have been done. Indeed, information about the computational times, the gaps to optimality, the characteristics of the integer linear programs and the solution values has been given for each instance and for each solution method in order to be evaluated and compared. To better observe the impact of the tabu list size, a further analysis has been done for each solution obtained with the tabu search method. Finally, the different analysis have been summarized for each method to highlight their advantages and drawbacks.

When considering the three combined objectives, solutions provided by $3P_{0.6}$ and $2P_{1.0}$ are on average better than that implemented by the cement factory. Furthermore, results of $3P_{0.3}$ ($2P_{0.5}$) are on average better than those of $3P_{0.6}$ ($2P_{1.0}$) because less third party vehicles are used to do the deliveries. Compared with $3P_{0.6}$ ($3P_{0.3}$), $2P_{1.0}$ ($2P_{0.5}$) is on average slightly better due to its formulation that better reflect reality. A similar remark can be done for 1P, which is on average better than $3P_{0.6}$, $3P_{0.3}$, $2P_{1.0}$ and $2P_{0.5}$. Finally, TS_8, TS_{15} and TS_{20} can produce solutions that are on average slightly better than those of 1P. Although the results of TS_8, TS_{15} and TS_{20} are relatively similar, good solutions are typically found faster when the tabu list size is large.

The following advices could be given to the company. If this one has a lot of time available to do the transportation planning elaboration and owns a solver like CPLEX, 1P should be used because this solution method always guarantees a solution of high quality. If a feasible solution has to be found in a short time or if the company does not own any solver, TS_{20} should or has to be preferred. Indeed, this solution method guarantees a feasible solution from the first second and solutions of very high quality can be found in a few minutes.

The end of this chapter marks the end of the second part of this thesis, which has presented the implementation of an optimization module from the description of the problem to the analysis of the dedicated solutions methods. This module completes the interactive planning system discussed in the first part of this thesis in order to form an interactive planning support system, as illustrated by a schema in Figure 7.1 of Chapter 7.

Conclusion

By building a decision support system specific to a factory, i.e. an interactive planning support system intended to aid the dispatcher of a cement factory in elaborating daily transportation plannings, a new, to our knowledge, problem of operations research or more precisely a new variant of the Capacitated Vehicle Routing Problem (CVRP) or Split Delivery Vehicle Routing Problem (SDVRP) has been discovered and treated in this thesis. Although different vehicles can visit a same customer in the SDVRP to entirely satisfy its order, the quantity ordered by a customer is always lower than the capacity of a vehicle, which can therefore deliver many customers successively with only one load. This is however not the case for the delivery problem specific to the cement factory, called cement delivery problem, where the quantity of cement ordered by a customer is often greater than the capacity of each vehicle that can be used. This implies the split of orders into multiple deliveries. Unlike most of neighboring problems discussed in the literature, the fleet of vehicles belonging to the cement factory or rented to third parties is heterogeneous, i.e. that each vehicle has a different capacity, emits another quantity of carbon dioxide (CO_2), and can only access some customers. To spare time, the vehicles can furthermore be preloaded the day before for the first delivery of the day. Thanks to a specific equipment, most of these ones can also be loaded at local depots located in railway stations. In parallel to the deliveries, unloaded vehicles can finally be used to recover ashes, i.e. raw materials, from suppliers or be assigned to other tasks.

Before having modeled the cement delivery problem and implemented different solution methods to solve it in an optimization module, an interactive planning system has first been developed in the first part of this thesis. Although this information system has been presented as a software tool that helps the dispatcher in his daily activities, this system can also be seen as a ‘platform’ whose

conception, especially the creation of the entity-relational diagram and its corresponding relational database schema, has allowed to gather and structure information required to model the cement delivery problem. Furthermore, the realization of this platform, i.e. the software tool, has provided not only an automated way to create real life instances of this problem thanks to the properties of daily orders and available resources recorded by the dispatcher, but also a graphical environment and functionalities to represent and analyze solutions obtained for selected instances.

To solve the cement delivery problem, four exact and approached solution methods have been modeled and tested on real life instances in the second part of this thesis. To simplify the resolution of this problem, a three phase method (3P) has first been developed. This one decomposes the problem into three subproblems solved successively related to the allocation of each order to a specific depot, the assignment of deliveries to each vehicle, and the sequencing of the deliveries on each vehicle route. Modeled as mixed integer linear programs and solved with CPLEX, most instances related to these subproblems have been solved in a few seconds and have reached the optimality. Regarding the values of the objective function, solutions obtained with appropriate settings ($3P_{0.3}$) were typically better than those provided by the cement factory. A two phase method (2P) that merges the two first subproblems mentioned before has then been developed in order to use more appropriate local depots and to better estimate the total travel time of vehicles loaded at these depots. Again, and with appropriate settings ($2P_{0.5}$), results given by CPLEX were often better than those implemented by the cement factory and slightly better than those of the three phase method ($3P_{0.3}$). Although most of these instances have been solved in a few minutes and have reached the optimality, the aggregation of optimal solutions related to subproblems solved successively however does not mean that the solution resulting from this aggregation is optimal. For this purpose, a one phase method (1P) that solves the cement delivery problem with only one mixed integer linear program has been developed. Although CPLEX has produced solutions of high quality, no proof of optimality could be obtained. Due to the complexity of this mixed integer linear program, the solver often stopped due to a lack of memory. To face this problematic, a tabu search method (TS) based on this mathematical formulation has finally been implemented. Tested with three different tabu list sizes (TS_8, TS_{15}, TS_{20}), solutions of very high quality, often better than those given by the one phase method, have been produced.

Although all results provided by the solution methods summarized before were always (with one exception) better than those implemented by the cement factory

if the three combined objectives are considered, a better solution does not mean that each of these objectives is strictly better when evaluated separately. While less different vehicles are used to supply a same customer in each solution provided by these methods in order to minimize the corresponding main objective ($\sum_{c \in C} \sum_{k \in V} v_{ck}$), vehicles belonging to third parties are however sometimes used, what penalizes the value of the third objective, i.e. the weighted total travel duration ($\sum_{k \in V} F_k \alpha_k$). Many solutions of the three phase method and the two phase method, as two solutions of the one phase method are concerned by such a situation. Contrariwise, all solutions implemented by the cement factory never use third parties vehicles and are therefore better from this point of view. While results provided by the tabu search method are significantly better than those of the three phase method and the two phase method, they are sometimes worse than those given by the one phase method for half of the instances solved, and whatever the size of the tabu list. Designed to produce as good solutions as the one phase method in less time, the tabu search method has, for a few instances, however required more minutes to find solutions of same quality, even if good solutions were already found quickly. Indeed, many tests due to vehicles that have different capacities or that are preloaded, i.e. that only require a specified cement type for their first delivery, are done during the tabu search process to check the feasibility of each generated solution, what may affect the speed performance of the method. Regarding the modeling of the cement delivery problem, some constraints do not precisely reflect reality. As mentioned before, tasks can be allocated to vehicles. While each task corresponds in reality to a time period, this duration is only subtracted from the daily availability of the vehicle, what does not guarantee the execution of the task in the right time period. While ashes can be recovered from suppliers by unloaded vehicles between two deliveries, this activity is currently not included in the modeling of the problem. Furthermore, time windows possibly specified in the interactive planning system when creating orders have not been taken into account in the cement delivery problem. Note that time windows could contradict the first objective of the objective function. Indeed, this one consists in minimizing the number of different vehicles used to supply a same customer or, in other terms, in compacting all deliveries of a customer on one or a few vehicles.

As mentioned before, many tests done in the tabu search method to check the feasibility of generated solutions may slow down the execution of this method. A possible direction for further research includes the generation of neighboring solutions that can also be unfeasible in order to improve the performance of the tabu search method. Unlike having to check each generated solution of a neigh-

borhood at each iteration to ensure its feasibility, an unfeasible solution could be repaired after a specified number of iterations with a dedicated procedure. Indeed, the surplus of delivered quantities of cement and the correct cement type allocated to each preload could respectively be removed and set by this procedure. Remember that the cement factory has enough vehicles to always generate an initial feasible solution. Therefore, an unfeasible solution could easily be adapted and converted as a feasible solution. In the tabu search method, three different operators have been defined to generate neighboring solutions. A second possible direction for further research consists in implementing additional operators in order to improve the intensification of the tabu search process. However, this could slow down the execution of the metaheuristic due to the addition of new lines code. Note that the creation of a new operator could mutate the tabu search method in a matheuristic, i.e. a solution method that combines metaheuristic and mixed integer linear programming techniques, if implemented as described hereafter. The values of some variables of the mixed integer linear program related to the one phase method would first be fixed with the corresponding values of the solution related to the current iteration. Then, this problem would be solved with a solver like CPLEX to find values for the unfixed variables while the fixed variables would be treated as constants. Note that the fixed variables would be different for each neighbor. Furthermore, the value of variables would be different at each iteration, depending on the position of the current solution in the solutions space. Regarding the tabu list size, this one is currently fixed before the execution of the tabu search method. Although a size of 20 has seemed to provide good results in less time than a size of 8 or 15, the tabu list size could be flexible. A third possible direction for further research includes the modification of the tabu list size directly in the tabu search method to intensify or diversify the search process when needed. As discussed before, the modeling of the cement delivery problem could be improved in the four solution methods to better reflect reality. A fourth possible direction for further research consists in modifying the modeling related to the cement delivery problem and therefore the corresponding solution methods to include time windows, planned tasks and ashes recoveries. Note that the objective function should however be adapted to avoid being in contradiction with the constraints.

Although the solution methods presented in this thesis have been developed to solve a cement delivery problem, these ones can also be used in other similar contexts such as the transportation of fuel, milk or cereals from sites to customers, i.e. the transportation of products sold in bulk. Furthermore, the problem could be adapted to enable simultaneously the transportation of different product types either with the use of partitioned trailers, or with multiple trailers.

Bibliography

[1] J. Albahari and B. Albahari. *C# 4.0 in a Nutshell: The Definitive Reference*. O'Reilly, 4 edition, 2010.

[2] A. S. Alfa, S. S. Heragu, and M. Chen. A 3-opt based simulated annealing algorithm for vehicle routing problems. *Computers & Industrial Engineering*, 21(1–4):635–639, 1991.

[3] C. Archetti, M. W. P. Savelsbergh, and M. G. Speranza. To split or not to split: That is the question. *Transportation Research Part E: Logistics and Transportation Review*, 44(1):114–123, 2008.

[4] C. Archetti, M. G. Speranza, and A. Hertz. A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, 40(1):64–73, 2006.

[5] L. Asbach, U. Dorndorf, and E. Pesch. Analysis, modeling and solution of the concrete delivery problem. *European Journal of Operational Research*, 193(3):820–835, 2009.

[6] M. L. Balinski and R. E. Quandt. On an integer program for a delivery problem. *Operations Research*, 12(2):300–304, 1964.

[7] J. M. Belenguer, M. C. Martinez, and E. Mota. A lower bound for the split delivery vehicle routing problem. *Operations Research*, 48(5):801–810, 2000.

[8] L. D. Bentley and J. L. Whitten. *Systems Analysis and Design for the Global Enterprise*. McGraw-Hill International Edition, 7 edition, 2007.

[9] A. W. Brown, D. J. Carney, E. J. Morris, D. B. Smith, and P. F. Zarrella. *Principles of CASE Tool Integration*. Oxford University Press, 1 edition, 1994.

[10] T. Budd. *An Introduction to Object-Oriented Programming*. Addison-Wesley, 3 edition, 2001.

[11] M. Cantù. *Delphi 2007 Handbook*. M. Cantù, 1 edition, March 2008.

[12] M. Cantù. *Essential Pascal*. M. Cantù, April 2008.

[13] S. Chen, B. Golden, and E. Wasil. The split delivery vehicle routing problem: Applications, algorithms, test problems, and computational results. *Networks*, 49(4):318–329, 2007.

[14] N. Christofides, A. Mingozzi, and P. Toth. Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20(1):255–282, 1981.

[15] N. Christofides, A. Mingozzi, and P. Toth. State-space relaxation procedures for the computation of bounds to routing problems. *Networks*, 11(2):145–164, 1981.

[16] G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12(4):568–581, 1964.

[17] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany. New heuristics for the vehicle routing problem. In *Logistics Systems: Design and Optimization*, pages 279–297. Springer US, 2005.

[18] J.-F. Cordeau and G. Laporte. Tabu search heuristics for the vehicle routing problem. In *Metaheuristic Optimization via Memory and Evolution*, volume 30 of *Operations Research/Computer Science Interfaces Series*, pages 145–163. Springer US, 2005.

[19] G. Desaulniers. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. *Operations Research*, 58(1):179–192, 2010.

[20] K. F. Doerner, G. Fuellerer, R. F. Hartl, M. Gronalt, and M. Iori. Metaheuristics for the vehicle routing problem with loading constraints. *Networks*, 49(4):294–307, 2007.

[21] M. Dror, G. Laporte, and P. Trudeau. Vehicle routing with split deliveries. *Discrete Applied Mathematics*, 50(3):239–254, 1994.

[22] M. Dror and P. Trudeau. Savings by split delivery routing. *Transportation Science*, 23(2):141–145, 1989.

[23] M. L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks*, 11(2):109–124, 1981.

[24] D. Flanagan. *Java in a Nutshell: A Desktop Quick Reference*. O'Reilly, 5 edition, 2005.

[25] M. Fowler. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. The Addison-Wesley Object Technology Series. Addison-Wesley Professional, 3 edition, 2004.

[26] P. W. Frizzell and J. W. Giffin. The split delivery vehicle scheduling problem with time windows and grid network distances. *Computers & Operations Research*, 22(6):655–667, 1995.

[27] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database Systems: The Complete Book*. Prentice Hall, 2 edition, 2009.

[28] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search algorithm for a routing and container loading problem. *Transportation Science*, 40(3):342–350, 2006.

[29] M. Gendreau, M. Iori, G. Laporte, and S. Martello. A tabu search heuristic for the vehicle routing problem with two-dimensional loading constraints. *Networks*, 51(1):4–18, 2008.

[30] M. Gendreau, G. Laporte, and J.-Y. Potvin. Metaheuristics for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 129–154. SIAM, Philadelphia, 2002.

[31] B. E. Gillett and L. R. Miller. A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, 22(2):340–349, 1974.

[32] F. Glover and M. Laguna. Tabu search. In C. R. Reeves, editor, *Modern Heuristic Techniques for Combinatorial Problems*, pages 70–150. John Wiley & Sons, Inc., 1993.

[33] F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[34] I. Gorton. *Essential Software Architecture*. Springer, 2 edition, 2011.

[35] D. Gulczynski, B. L. Golden, and E. A. Wasil. The multi-depot split delivery vehicle routing problem: An integer programming-based heuristic, new test problems, and computational results. *Computers & Industrial Engineering*, 61:794–804, 2011.

[36] E. R. Harold and W. S. Means. *XML in a Nutshell: A Desktop Quick Reference*. O'Reilly, 3 edition, 2004.

[37] A. Hertz, E. Taillard, and D. de Werra. Tabu search. In E. H. L. Aarts and J. K. Lenstra, editors, *Local Search in Combinatorial Optimization*, pages 121–136. John Wiley & Sons, Inc., 1997.

[38] A. Hertz, M. Uldry, and M. Widmer. Integer linear programming models for a cement delivery problem. *European Journal of Operational Research*, 222(3):623–631, 2012.

[39] S. C. Ho and D. Haugland. A tabu search heuristic for the vehicle routing problem with time windows and split deliveries. *Computers & Operations Research*, 31(12):1947–1964, 2004.

[40] F. R. Jacobs, R. B. Chase, and N. J. Aquilano. *Operations and Supply Management*. McGraw-Hill International Edition, 12 edition, 2009.

[41] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59(3):345–358, 1992.

[42] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet. Classical and modern heuristics for the vehicle routing problem. *International Transactions in Operational Research*, 7:285–300, 2000.

[43] G. Laporte, Y. Nobert, and M. Desrochers. Optimal routing under capacity and distance restrictions. *Operations Research*, 33(5):1050–1073, 1985.

[44] G. Laporte and F. Semet. Classical heuristics for the capacitated VRP. In P. Toth and D. Vigo, editors, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, pages 109–128. SIAM, Philadelphia, 2002.

[45] M. Martin. *Borland Delphi 2005/2006*. Le tout en poche. CampusPress, January 2006.

[46] A. Meier and D. H. Nguyen. *Introduction pratique aux bases de données relationnelles*. Springer, 2 edition, 2005.

[47] P. A. Mullaseril, M. Dror, and J. Leung. Split-delivery routeing heuristics in livestock feed distribution. *The Journal of the Operational Research Society*, 48(2):107–116, 1997.

[48] R. Ramakrishnan and J. Gehrke. *Database Management Systems*. McGraw-Hill, 3 edition, 2003.

[49] F. Robusté, C. F. Daganzo, and R. R. Souleyrette. Implementing vehicle routing models. *Transportation Research Part B: Methodological*, 24(4):263–286, 1990.

[50] V. Schmid. *Trucks in Movement: Hybridization of Exact Approaches and Variable Neighborhood Search for the Delivery of Ready-Mixed Concrete*. PhD thesis, Universität Wien, 2007.

[51] V. Schmid, K. F. Doerner, R. F. Hartl, and J.-J. Salazar-González. Hybridization of very large neighborhood search for ready-mixed concrete delivery problems. *Computers & Operations Research*, 37(3):559–574, 2010.

[52] V. Schmid, K. F. Doerner, R. F. Hartl, M. W. P. Savelsbergh, and W. Stoecher. A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43(1):70–85, 2009.

[53] G. Sierksma and G. A. Tijssen. Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research*, 76:261–286, 1998.

[54] A. Silberschatz, P. B. Galvin, and G. Gagne. *Operating System Concepts*. John Wiley & Sons, Inc., 8 edition, 2009.

[55] A. S. Tanenbaum. *Modern Operating Systems*. Prentice Hall, 3 edition, 2008.

[56] M. Uldry. *DailyVery Planner – Manuel d'utilisation*, 2010.

[57] M. Uldry, M. Widmer, and A. Hertz. Two objective functions for a real life split delivery vehicle routing problem. In *Electronic Proceedings of the 'International Conference on Industrial Engineering and Systems Management (IESM 2011)'*, Metz, France, 2011.

[58] M. Widmer. Operational research models and methods in CIM. In H. H. Adelsberger, J. Lazansky, and V. Marik, editors, *Information Management in Computer Integrated Manufacturing*, number 973 in Lecture Notes in Computer Science, pages 179–194. Springer-Verlag Berlin Heidelberg, 1995.

[59] A. Wren and A. Holliday. Computer scheduling of vehicles from one or more depots to a number of delivery points. *Operational Research Quarterly*, 23(3):333–344, 1972.

[60] W. Zhu, H. Qin, A. Lim, and L. Wang. A two-stage tabu search algorithm with enhanced packing heuristics for the 3L-CVRP and M3L-CVRP. *Computers & Operations Research*, 39(9):2178–2195, 2012.

Appendices

Appendix A

Preceding rules

This appendix is an extension of Chapter 6 that presents the 'preceding rules', i.e. the set of the 36 different cases that can be detected when elaborating a transportation planning, using activity diagrams. For this purpose, Figures A.1, A.2 and A.3 respectively illustrate how to detect a case when creating, modifying and deleting a delivery, while Figures A.4 and A.5 respectively illustrate how to detect a case when creating and modifying an ash recovery. No figure exists for the deletion of an ash recovery because this operation concerns only one case.

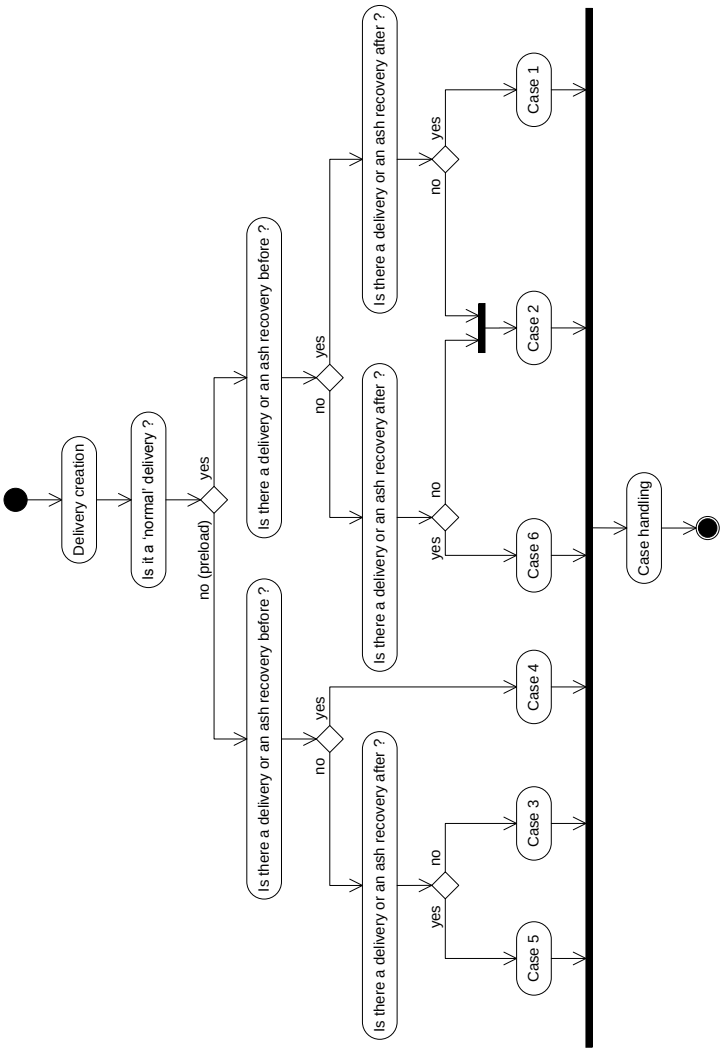

Figure A.1: Delivery creation

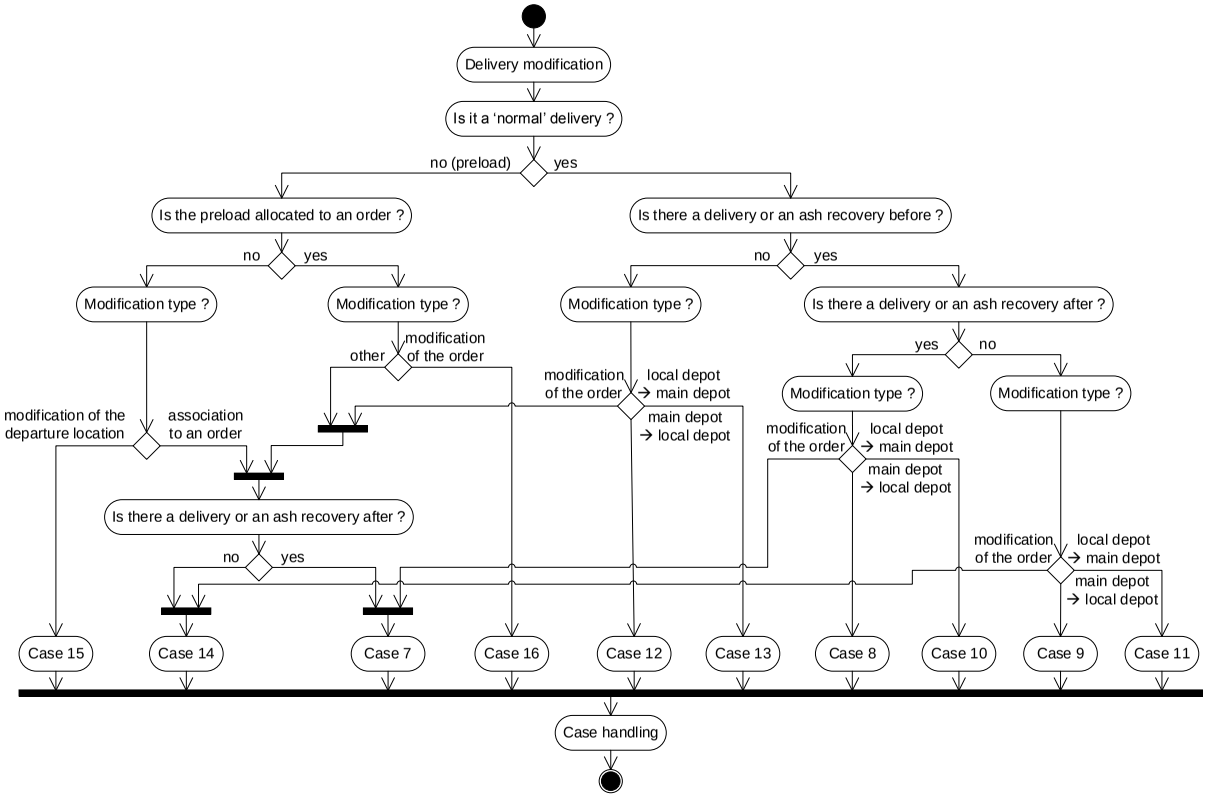

Figure A.2: Delivery modification

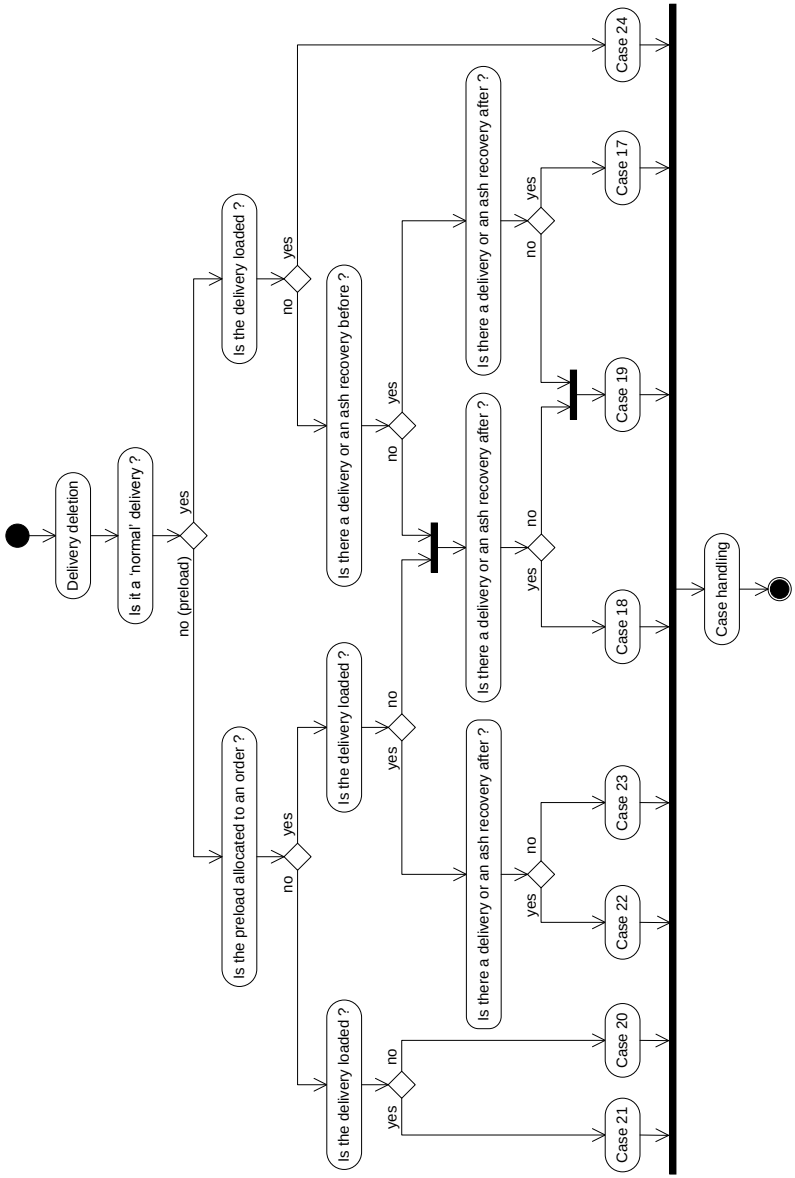

Figure A.3: Delivery deletion

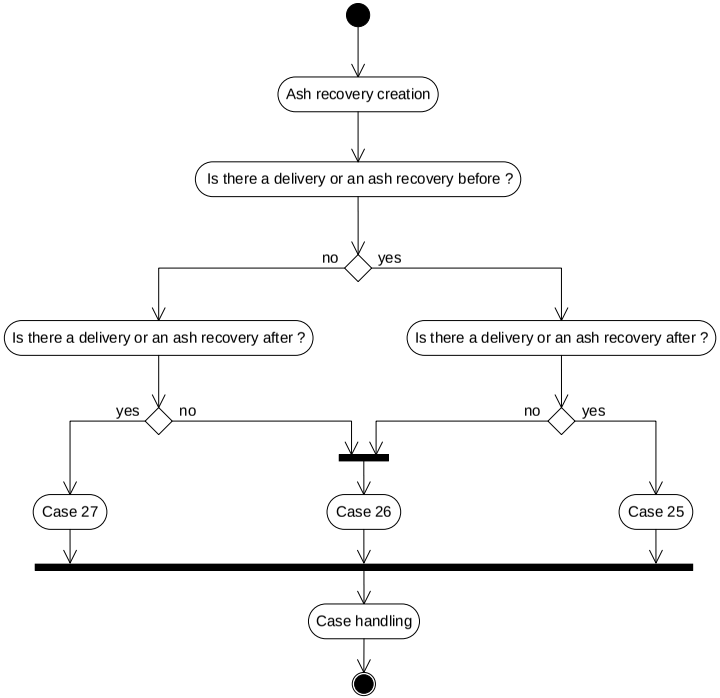

Figure A.4: Ash recovery creation

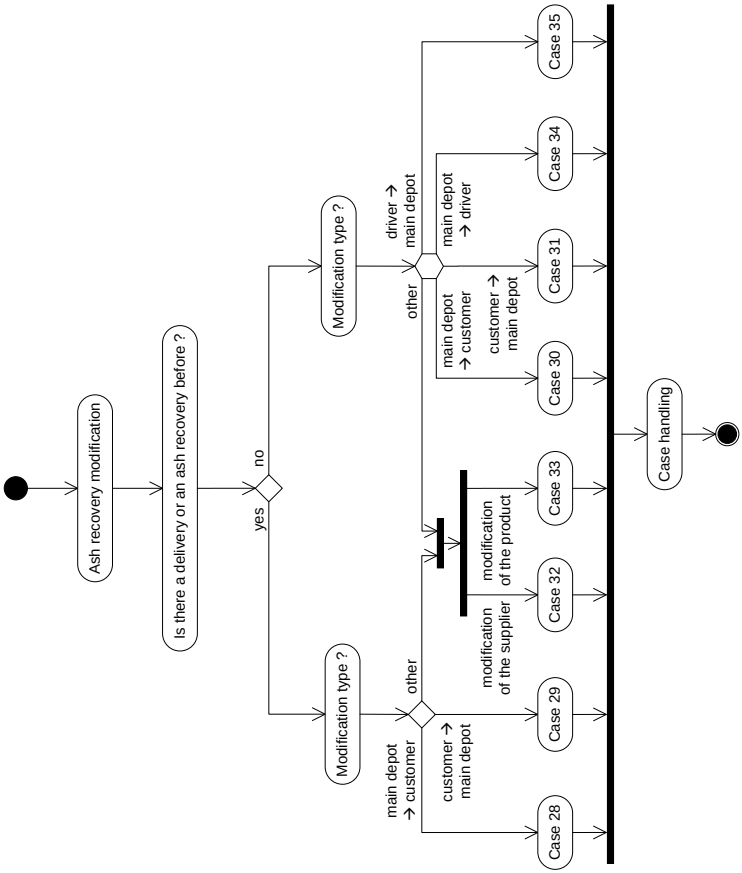

Figure A.5: Ash recovery modification

Appendix B

Cases handling of the preceding rules

This appendix is an extension of Chapter 6 that describes the handling of the 'preceding rules', which ensure the maintenance of a feasible route for each truck whatever the core functionality used by the dispatcher. For ease of reading, the 36 different cases that can be detected are allocated into 6 distinct groups. Indeed, Sections B.1, B.2 and B.3 respectively concern the creation, modification and deletion of a delivery, while Sections B.4, B.5 and B.6 respectively concern the creation, modification and deletion of an ash recovery.

Each case handling described hereafter is illustrated in a figure. While grey planner items indicate either deliveries or ashes recoveries, blue planner items correspond to 'normal' deliveries, green planner items correspond to preloaded deliveries and yellow planner items correspond to ashes recoveries. When a planner item is simultaneously colored in blue and green, it means that this planner item can be either a 'normal' delivery or a preloaded delivery. When the outline of a green planner item is dotted, it means that the preloaded delivery is not yet assigned to an order. Furthermore, when the outline and identifier of a blue or green planner item is colored in red, it means that the delivery is loaded. Note that planner items put in brackets are optional in the case handling. Concerning the modification of a planner item, a white square represents no departure

location, a black square represents the main depot, i.e. the factory, a black triangle represents a local depot, a black house represents the home of a truck-driver, a blue circle represents a 'normal' delivery, a green circle represents a preloaded delivery, and a colored diamond represents an ash recovery. If the outline of a green circle is dotted, it means that the preloaded delivery is not yet assigned to an order. Note that each order and ash supplier is characterized by an identifier. Unlike the color of a circle, the color of a diamond represents the ash type related to the ash recovery.

B.1 Delivery creation

The handling of the 6 different cases related to the creation of a delivery, illustrated by an activity diagram in Figure A.1 of Appendix A, are described in detail hereafter.

Case 1

Illustrated in Figure B.1, this case concerns the creation of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is inserted between two planner items (1 & 3).

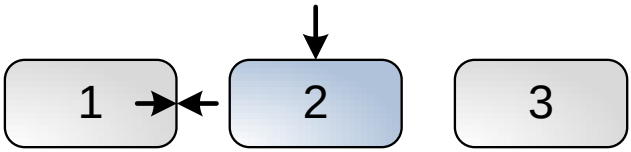

Figure B.1: Case 1

When creating such a delivery (2), the default depot used to load the cement type always corresponds to the main depot, i.e. the factory. This implies eventually a modification of the duration of the planner item that precedes the created delivery (1) if this one is a 'normal' delivery or a preloaded delivery assigned to an order whose truck does not drive back to the main depot after having unloaded its cement type at the customer. To compute the duration of the created delivery (2), it is also necessary to know if the following planner item (3) is a delivery or an ash

recovery. If it is a delivery, the truck drives back to the main or local depot used for this delivery after having unloaded the cement type of the created delivery at the customer. Otherwise, i.e. if it is an ash recovery, the truck directly drives to the ash supplier.

Case 2

Illustrated in Figure B.2, this case concerns the creation of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there is eventually a planner item before (1) but no planner item after.

Figure B.2: Case 2

When creating such a delivery (2), the default depot used to load the cement type always corresponds to the main depot, i.e. the factory. Since there is no planner item after the created delivery, the arrival location of the planner item that eventually precedes the created delivery (1) already corresponds to the main depot. The duration of this planner item (1) is therefore not modified. After having unloaded the cement type of the created delivery (2) at the customer, the truck drives back to the main depot, i.e. the factory, because this is the last planner item of the truck.

Case 3

Illustrated in Figure B.3, this case concerns the creation of a preloaded delivery (2), i.e. a delivery that is not yet assigned to an order, if there is no planner item before and after.

When creating such a delivery (2), no order is assigned and no depot is therefore defined to load the cement type. For this reason, the duration of the delivery cannot be computed and corresponds to a default value, which can be changed in the settings of the interactive planning system.

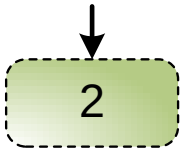

Figure B.3: Case 3

Case 4

Illustrated in Figure B.4, this case concerns the creation of a preloaded delivery (2), i.e. a delivery that is not yet assigned to an order, if there is a planner item before (1) and eventually a planner item after (3).

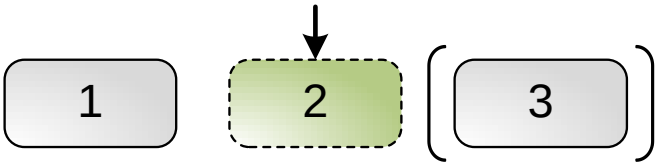

Figure B.4: Case 4

Such a case is forbidden by the interactive planning system because a preloaded delivery has always to correspond to the first delivery of a truck.

Case 5

Illustrated in Figure B.5, this case concerns the creation of a preloaded delivery (2), i.e. a delivery that is not yet assigned to an order, if there is one or two planner items after (3 & 4) but no planner item before.

When creating such a delivery (2), no order is assigned and no depot is therefore defined to load the cement type. For this reason, the duration of the delivery cannot be computed and corresponds to a default value, which can be changed in the settings of the interactive planning system. If the departure location of the planner item that follows the created delivery (3) does not correspond to the main depot, i.e. the factory, the departure location and the duration of this planner item

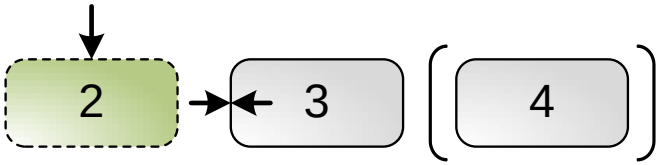

Figure B.5: Case 5

is modified. If the planner item that follows this planner item (4) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that follows the created delivery (3) corresponds to this depot. Otherwise, i.e. if the planner item that follows this planner item (4) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the planner item that follows the created delivery (3) corresponds to the supplier of this ash recovery (4). Note that if no planner item follows the planner item that follows the created delivery (3), the arrival location used to compute the back travel time of this planner item corresponds to the main depot, i.e. the factory, because this is the last planner item of the truck.

Case 6

Illustrated in Figure B.6, this case concerns the creation of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there is one or two planner items after (3 & 4) but no planner item before.

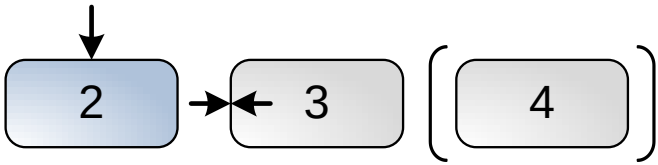

Figure B.6: Case 6

When creating such a delivery (2), the default depot used to load the cement type always corresponds to the main depot, i.e. the factory. If the departure loca-

tion of the planner item that follows the created delivery (3) does not correspond to the main depot, i.e. the factory, the departure location and the duration of this planner item has to be modified. If the planner item that follows this planner item (4) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that follows the created delivery (3) corresponds to this depot. Otherwise, i.e. if the planner item that follows this planner item (4) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the planner item that follows the created delivery (3) corresponds to the supplier of this ash recovery (4). Note that if no planner item follows the planner item that follows the created delivery (3), the arrival location used to compute the back travel time of this planner item corresponds to the main depot, i.e. the factory, because this is the last planner item of the truck. Such a case is only authorized by the interactive planning system if there is beforehand no preloaded delivery allocated to the truck.

B.2 Delivery modification

The handling of the 10 different cases related to the modification of a delivery, illustrated by an activity diagram in Figure A.2 of Appendix A, are described in detail hereafter.

Case 7

Illustrated in Figure B.7, this case concerns either the modification of the customer of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there is a planner item after (3) and eventually a planner item before (1), or the assignment of an order to a preloaded delivery (2), i.e. a delivery that is not yet assigned to an order, or the modification of the order of a preloaded delivery assigned to an order (2), or the modification of the departure location of a preloaded delivery assigned to an order (2), which is changed from the main depot, i.e. the factory, to the home of a truck-driver, or the modification of the departure location of a preloaded delivery assigned to an order (2), which is changed from the home of a truck-driver to the main depot, i.e. the factory, if there is a planner item after (3) but no planner item before.

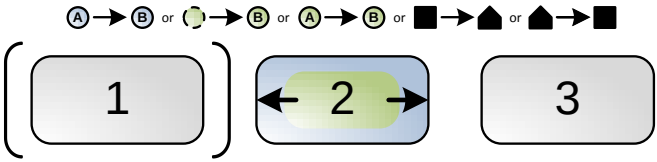

Figure B.7: Case 7

When modifying a delivery (2) in such a way, the departure location used to compute its travel time is unchanged, except if the case concerns the assignment of an order to a preloaded delivery where this location corresponds to the main depot, i.e. the factory, or if the case concerns the modification of the departure location of a preloaded delivery assigned to an order. Due to the assignment or modification of the order, the duration of the delivery (2) is modified. If the planner item that follows the modified delivery (3) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the modified delivery (2) corresponds to this depot. Otherwise, i.e. if the planner item that follows the modified delivery (2) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the modified delivery (2) corresponds to the supplier of this ash recovery (3).

Case 8

Illustrated in Figure B.8, this case concerns the modification of the departure location of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is changed from the main depot, i.e. the factory, to one of the local depots located in the railway stations, if there is a planner item before (1) and after (3).

When modifying a delivery (2) in such a way, the departure location used to compute its travel time is changed to one of the local depots. If the planner item that precedes this modified delivery (1) is a preloaded delivery not assigned to an order or an ash recovery, the modification is forbidden by the interactive planning system. Otherwise, i.e. if this planner item (1) is a 'normal' delivery or a preloaded delivery assigned to an order, the modification of the delivery (2) is authorized, which implies not only a modification of its duration, but also a modification of the duration of the planner item that precedes the modified delivery (1) due to

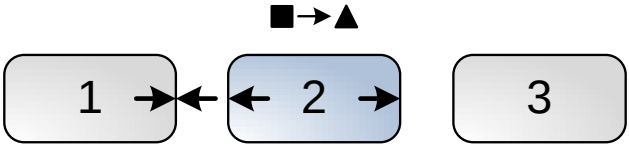

Figure B.8: Case 8

the change of the arrival location used to compute the duration of its back travel time. If the planner item that follows the modified delivery (3) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the modified delivery (2) corresponds to this depot. Otherwise, i.e. if the planner item that follows the modified delivery (2) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the modified delivery (2) corresponds to the supplier of this ash recovery (3).

Case 9

Illustrated in Figure B.9, this case concerns the modification of the departure location of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is changed from the main depot, i.e. the factory, to one of the local depots located in the railway stations, if there is a planner item before (1) but no planner item after.

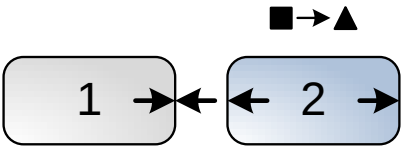

Figure B.9: Case 9

When modifying a delivery (2) in such a way, the departure location used to compute its travel time is changed to one of the local depots. If the planner item that precedes this modified delivery (1) is a preloaded delivery not assigned to an

order or an ash recovery, the modification is forbidden by the interactive planning system. Otherwise, i.e. if this planner item (1) is a 'normal' delivery or a preloaded delivery assigned to an order, the modification of the delivery (2) is authorized, which implies not only a modification of its duration, but also a modification of the duration of the planner item that precedes the modified delivery (1) due to the change of the arrival location used to compute the duration of its back travel time.

Case 10

Illustrated in Figure B.10, this case concerns the modification of the departure location of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is changed from a local depot to the main depot, i.e. the factory, if there is a planner item before (1) and after (3).

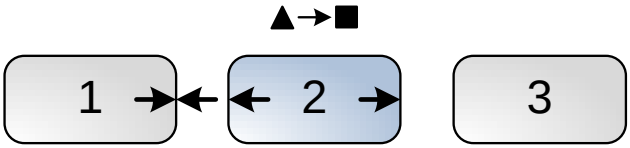

Figure B.10: Case 10

When modifying a delivery (2) in such a way, the departure location used to compute its travel time is changed to the main depot, i.e. the factory. If the planner item that precedes this modified delivery (1) is a preloaded delivery not assigned to an order or an ash recovery, the modification is forbidden by the interactive planning system. Note that such a situation is however not possible due to the handling of the cases presented before. Otherwise, i.e. if this planner item (1) is a 'normal' delivery or a preloaded delivery assigned to an order, the modification of the delivery (2) is authorized, which implies not only a modification of its duration, but also a modification of the duration of the planner item that precedes the modified delivery (1) due to the change of the arrival location used to compute the duration of its back travel time. If the planner item that follows the modified delivery (3) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the modified delivery (2) corresponds to this depot. Otherwise, i.e. if the planner item that follows the modified delivery (2) has no departure location and is therefore an ash recovery with no travel time, the

arrival location needed to compute the back travel time of the modified delivery (2) corresponds to the supplier of this ash recovery (3).

Case 11

Illustrated in Figure B.11, this case concerns the modification of the departure location of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is changed from a local depot to the main depot, i.e. the factory, if there is a planner item before (1) but no planner item after.

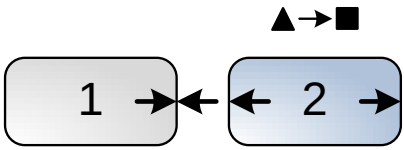

Figure B.11: Case 11

When modifying a delivery (2) in such a way, the departure location used to compute its travel time is changed to the main depot, i.e. the factory. If the planner item that precedes this modified delivery (1) is a preloaded delivery not assigned to an order or an ash recovery, the modification is forbidden by the interactive planning system. Note that such a situation is however not possible due to the handling of the cases presented before. Otherwise, i.e. if this planner item (1) is a 'normal' delivery or a preloaded delivery assigned to an order, the modification of the delivery (2) is authorized, which implies not only a modification of its duration, but also a modification of the duration of the planner item that precedes the modified delivery (1) due to the change of the arrival location used to compute the duration of its back travel time.

Case 12

Illustrated in Figure B.12, this case concerns the modification of the departure location of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is changed from the main depot, i.e. the factory, to one of the local depots located in the railway stations, if there is no planner item before but eventually a planner item after (3).

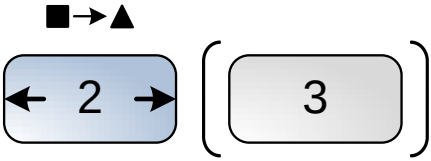

Figure B.12: Case 12

Although this situation is only possible if there is no preloaded delivery allocated to the truck, such a case is forbidden by the interactive planning system.

Case 13

Illustrated in Figure B.13, this case concerns the modification of the departure location of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, which is changed from a local depot to the main depot, i.e. the factory, if there is no planner item before but eventually a planner item after (3).

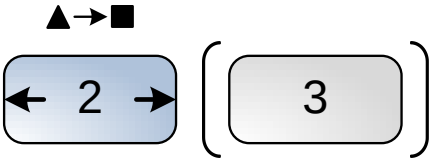

Figure B.13: Case 13

Such a situation is not possible due to the handling of the cases presented before.

Case 14

Illustrated in Figure B.14, this case concerns either the modification of the order of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there is eventually a planner item before (1) but no planner item after, or the assignment of an order to a preloaded delivery (2), i.e. a delivery that is not yet assigned to an order, or the modification of the order of a preloaded delivery

assigned to an order (2), or the modification of the departure location of a preloaded delivery assigned to an order (2), which is changed from the main depot, i.e. the factory, to the home of a truck-driver, or the modification of the departure location of a preloaded delivery assigned to an order (2), which is changed from the home of a truck-driver to the main depot, i.e. the factory, if there is no planner item after.

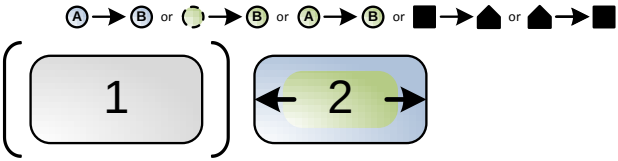

Figure B.14: Case 14

When modifying a delivery (2) in such a way, the departure location used to compute its travel time is unchanged, except if the case concerns the assignment of an order to a preloaded delivery where this location corresponds to the main depot, i.e. the factory, or if the case concerns the modification of the departure location of a preloaded delivery assigned to an order. Due to the assignment or modification of the order, the duration of the delivery (2) is modified. Note that the arrival location of the back travel time is unchanged and corresponds to the main depot, i.e. the factory, because this is the last planner item allocated to the truck.

Case 15

Illustrated in Figure B.15, this case concerns the modification of the departure location of a preloaded delivery (2), i.e. a delivery that is not yet assigned to a customer, if there is no planner item before but eventually a planner item after (3).

Such a modification of the delivery (2) is forbidden by the interactive planning system.

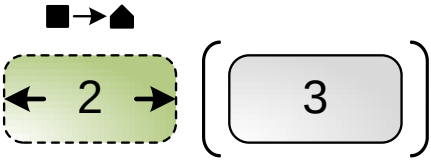

Figure B.15: Case 15

Case 16

Illustrated in Figure B.16, this case concerns the modification of the order of a preloaded delivery (2), i.e. a delivery assigned to an order, if there is no planner item before but eventually a planner item after (3).

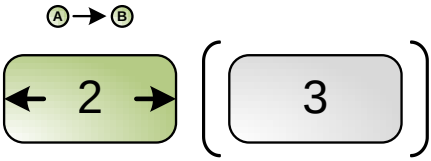

Figure B.16: Case 16

When modifying a delivery (2) in such a way, the departure location used to compute its travel time is unchanged. Due to the modification of the order, the duration of the delivery (2) is modified. If the planner item that follows the modified delivery (3) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the modified delivery (2) corresponds to this depot. Otherwise, i.e. if the planner item that follows the modified delivery (2) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the modified delivery (2) corresponds to the supplier of this ash recovery (3).

B.3 Delivery deletion

The handling of the 8 different cases related to the deletion of a delivery, illustrated by an activity diagram in Figure A.3 of Appendix A, are described in detail hereafter.

Case 17

Illustrated in Figure B.17, this case concerns the deletion of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there is a planner item before (1) and one or two planner items after (3 & 4).

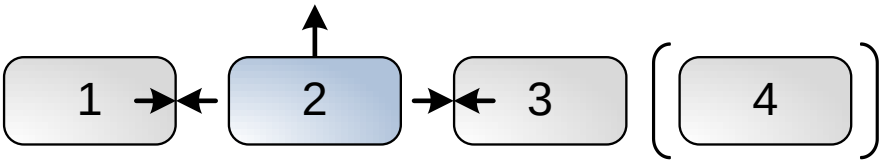

Figure B.17: Case 17

When deleting such a delivery (2), if the planner item that precedes this delivery (1) is an ash recovery or a preloaded delivery that is not yet assigned to an order, the duration of the planner item that follows the deleted delivery is modified, but only if its departure location does not initially correspond to the main depot, i.e. the factory. If the planner item that follows this planner item (4) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that follows the deleted delivery (3) corresponds to this depot. Otherwise, i.e. if the planner item that follows this planner item (4) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the planner item that follows the deleted delivery (3) corresponds to the supplier of this ash recovery (4). Note that if no planner item follows the planner item that follows the deleted delivery (3), the arrival location used to compute the back travel time of this planner item corresponds to the main depot, i.e. the factory, because this is the last planner item of the truck. If the planner item that precedes the deleted delivery (1) is a 'normal' delivery and if the departure location of the planner item that follows

the deleted delivery (3) is not identical to the departure location of the deleted delivery (2) and corresponds to the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that precedes the deleted delivery (1) corresponds to this depot. Otherwise, i.e. if the planner item that follows the deleted delivery (3) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the planner item that precedes the deleted delivery (1) corresponds to the supplier of this ash recovery (3).

Case 18

Illustrated in Figure B.18, this case concerns not only the deletion of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there is one or two planner items after (3 & 4) but no planner item before, but also the deletion of a preloaded delivery, which is not loaded but assigned to an order, if there is one or two planner items after (3 & 4) but no planner item before.

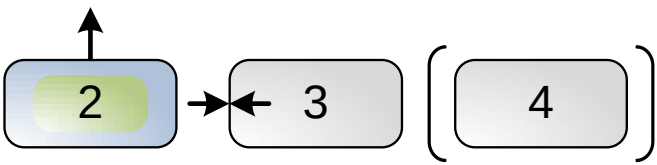

Figure B.18: Case 18

When deleting such a delivery (2), if the departure location of the planner item that follows the deleted delivery (3) does not correspond to the main depot, i.e. the factory, its departure location is changed to the main depot and its duration is modified. If this planner item (3) is not an ash recovery but a delivery, the arrival location needed to compute its back travel time has to be known, except if this planner item (3) is the last planner item of the truck. If the planner item that eventually follows this planner item (4) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that follows the deleted delivery (3) corresponds to this depot. Otherwise, i.e. if the planner item that follows this planner item (4) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to com-

pute the back travel time of the planner item that follows the deleted delivery (3) corresponds to the supplier of this ash recovery (4).

Case 19

Illustrated in Figure B.19, this case concerns not only the deletion of a 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there is eventually a planner item before (1) and no planner item after, but also the deletion of a preloaded delivery, which is not loaded but assigned to an order, if there is no planner item before and after.

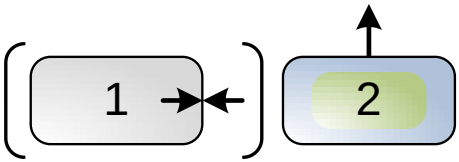

Figure B.19: Case 19

When deleting such a delivery (2), if its departure location does not correspond to the main depot, i.e. the factory, or to the home of a truck-driver, the arrival location of the eventual planner item that precedes this deleted delivery (1) is changed to the main depot, what implies a modification of its duration.

Case 20

Illustrated in Figure B.20, this case concerns the deletion of a preloaded delivery (2), which is not loaded and not assigned to an order, if there is eventually a planner item after (3) but no planner item before.

Such a delivery (B) is deleted without any impact on the eventual following planner item.

Case 21

Illustrated in Figure B.21, this case concerns the deletion of a preloaded delivery (2), which is loaded but not assigned to an order, if there is eventually a planner item after (3) but no planner item before.

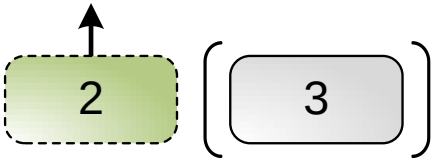

Figure B.20: Case 20

A deletion of such a delivery (2) is forbidden by the interactive planning system.

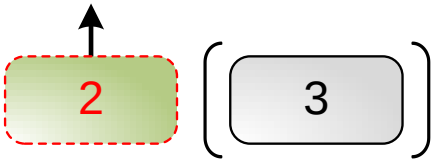

Figure B.21: Case 21

Case 22

Illustrated in Figure B.22, this case concerns the deletion of a preloaded delivery (2), which is loaded and assigned to an order, if there is one or two planner items after (3 & 4) but no planner item before.

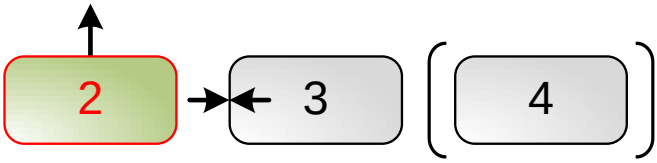

Figure B.22: Case 22

A deletion of such a delivery (2) is forbidden by the interactive planning system. However, the order of the delivery is unassigned and the duration of the delivery is changed to a default value, which can be modified in the settings of

the interactive planning system. If the departure location of the planner item that follows the deleted delivery (3) does not correspond to the main depot, i.e. the factory, its departure location is changed to the main depot and its duration is modified. If this planner item (3) is not an ash recovery but a delivery, the arrival location needed to compute its back travel has to be known, except if this planner item (3) is the last planner item of the truck. If the planner item that eventually follows this planner item (4) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that follows the deleted delivery (3) corresponds to this depot. Otherwise, i.e. if the planner item that follows this planner item (4) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the planner item that follows the deleted delivery (3) corresponds to the supplier of this ash recovery (4).

Case 23

Illustrated in Figure B.23, this case concerns the deletion of a preloaded delivery (2), which is loaded and assigned to an order, if there is no planner item before and after.

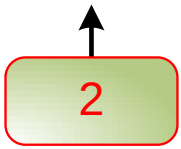

Figure B.23: Case 23

A deletion of such a delivery (2) is forbidden by the interactive planning system. However, the order of the delivery is unassigned and the duration of the delivery is changed to a default value, which can be modified in the settings of the interactive planning system.

Case 24

Illustrated in Figure B.24, this case concerns the deletion of a loaded 'normal' delivery (2), i.e. a delivery that does not correspond to the preload of a truck, if there

is eventually a planner item before (1) and after (3).

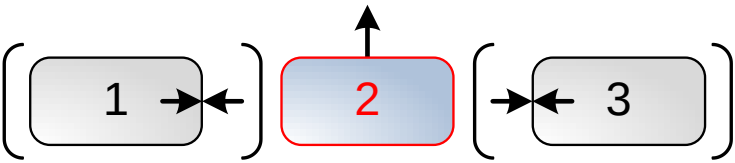

Figure B.24: Case 24

A deletion of such a delivery (2) is forbidden by the interactive planning system.

B.4 Ash recovery creation

The handling of the 3 different cases related to the creation of an ash recovery, illustrated by an activity diagram in Figure A.4 of Appendix A, are described in detail hereafter.

Case 25

Illustrated in Figure B.25, this case concerns the creation of an ash recovery (2) if there is a planner item before (1) and one or two planner items after (3 & 4).

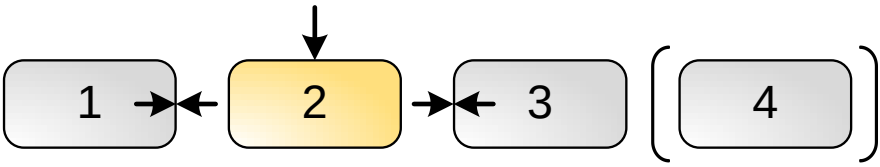

Figure B.25: Case 25

When creating such an ash recovery (2), if the planner item that precedes the created ash recovery (1) is an ash recovery or a preloaded delivery that is not yet assigned to an order, the departure location used to compute the travel time of the created ash recovery (2) corresponds to the main depot, i.e. the factory. If the

planner item that precedes the created ash recovery (1) is a 'normal' delivery or a preloaded delivery assigned to an order, the departure location used to compute the travel time of the created ash recovery (2) also corresponds to the main depot, but only if the planner item that follows the created ash recovery (3) has as departure location the main depot. Otherwise, the created ash recovery (2) has no departure location and the arrival location used to compute the duration of the back travel time of the planner item that precedes the created ash recovery (1) corresponds to the supplier of the created ash recovery (2). If the departure location of the planner item that follows the created ash recovery (3) does not correspond to the main depot, i.e. the factory, the departure location and the duration of this planner item is modified. If the planner item that follows this planner item (4) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that follows the created ash recovery (3) corresponds to this depot. Otherwise, i.e. if the planner item that follows this planner item (4) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the planner item that follows the created ash recovery (3) corresponds to the supplier of this ash recovery (4). Note that if no planner item follows the planner item that follows the created ash recovery (3), the arrival location used to compute the back travel time of this planner item corresponds to the main depot, i.e. the factory, because this is the last planner item of the truck.

Case 26

Illustrated in Figure B.26, this case concerns the creation of an ash recovery (2) if there is eventually a planner item before (1) but no planner item after.

Figure B.26: Case 26

When creating such an ash recovery (2), the departure location used to compute its travel time and the arrival location used to compute its back travel time correspond to the main depot, i.e. the factory.

Case 27

Illustrated in Figure B.27, this case concerns the creation of an ash recovery (2) if there is one or two planner items after (3 & 4) but no planner item before.

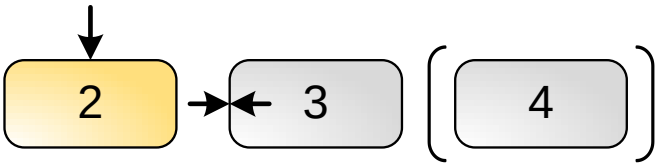

Figure B.27: Case 27

When creating such an ash recovery (2), the departure location used to compute its travel time and the arrival location used to compute its back travel time correspond to the main depot, i.e. the factory. If the departure location of the planner item that follows the created ash recovery (3) does not correspond to the main depot, i.e. the factory, the departure location and the duration of this planner item has to be modified. If the planner item that follows this planner item (4) has as departure location the main depot, i.e. the factory, or one of the local depots located in the railway stations, the arrival location needed to compute the back travel time of the planner item that follows the created ash recovery (3) corresponds to this depot. Otherwise, i.e. if the planner item that follows this planner item (4) has no departure location and is therefore an ash recovery with no travel time, the arrival location needed to compute the back travel time of the planner item that follows the created ash recovery (3) corresponds to the supplier of this ash recovery (4). Note that if no planner item follows the planner item that follows the created ash recovery (3), the arrival location used to compute the back travel time of this planner item corresponds to the main depot, i.e. the factory, because this is the last planner item of the truck.

B.5 Ash recovery modification

The handling of the 8 different cases related to the modification of an ash recovery, illustrated by an activity diagram in Figure A.5 of Appendix A, are described in detail hereafter.

Case 28

Illustrated in Figure B.28, this case concerns the modification of the departure location of an ash recovery (2), which is changed from the main depot, i.e. the factory, to no depot, if there is a planner item before (1) and eventually a planner item after (3).

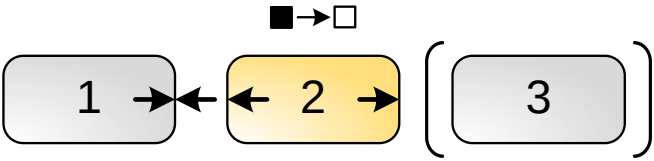

Figure B.28: Case 28

Such a modification of the ash recovery (2) is only authorized by the interactive planning system if the planner item that precedes this ash recovery (1) is a 'normal' delivery or a preloaded delivery assigned to an order. Due to the deletion of the departure location of the ash recovery (2), its duration, which no longer includes a travel time, is modified. The arrival location used to compute the duration of the planner item that precedes this modified ash recovery (1) is changed to the supplier of this ash recovery (2) and the duration of this planner item (1) is modified.

Case 29

Illustrated in Figure B.29, this case concerns the modification of the departure location of an ash recovery (2), which is changed from no depot to the main depot, i.e. the factory, if there is a planner item before (1) and eventually a planner item after (3).

Such a modification of the ash recovery (2) is only authorized by the interactive planning system if the planner item that precedes this ash recovery (1) is a 'normal' delivery or a preloaded delivery assigned to an order. Due to the selection of the main depot as departure location of the ash recovery (2), its duration, which henceforth includes a travel time, is modified. The arrival location used to compute the duration of the planner item that precedes this modified ash recovery (1) is changed to the main depot and the duration of this planner item (1) is modified.

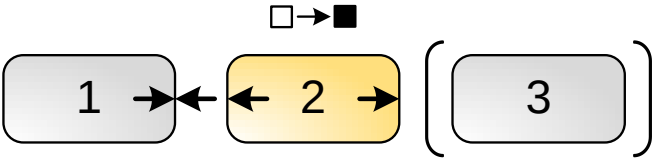

Figure B.29: Case 29

Case 30

Illustrated in Figure B.30, this case concerns the modification of the departure location of an ash recovery (2), which is changed from the main depot, i.e. the factory, to no depot, if there is no planner item before and eventually a planner item after (3).

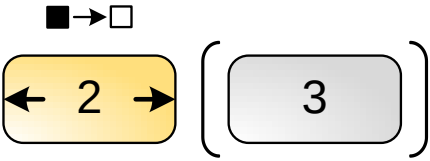

Figure B.30: Case 30

Such a modification of the ash recovery (2) is forbidden by the interactive planning system.

Case 31

Illustrated in Figure B.31, this case concerns the modification of the departure location of an ash recovery (2), which is changed from no depot to the main depot, i.e. the factory, if there is no planner item before and eventually a planner item after (3).

Such a situation is not possible due to the handling of the cases presented before.

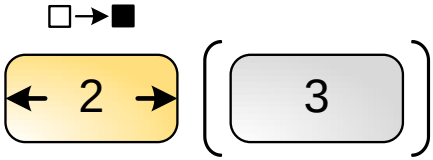

Figure B.31: Case 31

Case 32

Illustrated in Figure B.32, this case concerns the modification of the arrival location of an ash recovery (2), i.e. its supplier, if there is eventually a planner item before (1) and after (3).

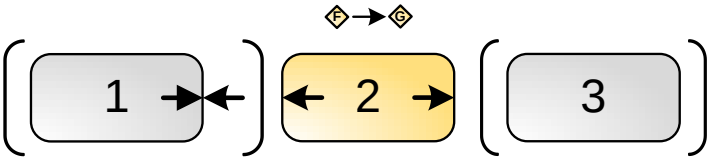

Figure B.32: Case 32

When modifying an ash recovery (2) in such a way, its duration is modified. If the departure location used to compute its travel time does not correspond to the main depot, i.e. if there is no travel time, it is certain that a 'normal' delivery or a preloaded delivery assigned to an order precedes this ash recovery (1). The arrival location used to compute the back travel of this delivery (1) is changed to the new supplier and the duration of this delivery is modified.

Case 33

Illustrated in Figure B.33, this case concerns the modification of the product of an ash recovery (2) if there is eventually a planner item before (1) and after (3).

When modifying an ash recovery (2) in such a way, its duration is modified due to the loading time and unloading time of the product, which can be different.

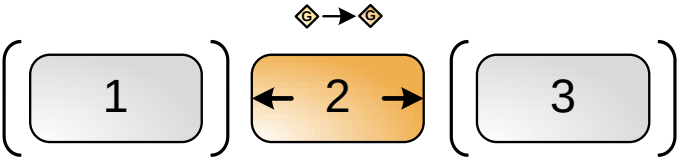

Figure B.33: Case 33

Case 34

Illustrated in Figure B.34, this case concerns the modification of the departure location of an ash recovery (2), which is changed from the main depot, i.e. the factory, to the home of a truck-driver if there is no planner item before and eventually a planner item after (3).

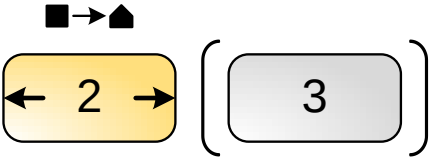

Figure B.34: Case 34

When modifying an ash recovery (2) in such a way, its duration is modified.

Case 35

Illustrated in Figure B.35, this case concerns the modification of the departure location of an ash recovery (2), which is changed from the home of a truck-driver to the main depot, i.e. the factory, if there is no planner item before and eventually a planner item after (3).

When modifying an ash recovery (2) in such a way, its duration is modified.

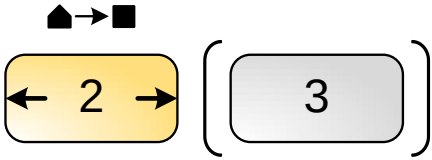

Figure B.35: Case 35

B.6 Ash recovery deletion

The handling of the unique case related to the deletion of an ash recovery is described in detail hereafter.

Case 36

Illustrated in Figure B.36, this case concerns the deletion of an ash recovery (2) if there is eventually a planner item before (1) and after (3).

When deleting an ash recovery (2), if the departure location of this deleted ash recovery (2) does not correspond to the main depot, i.e. the factory, the arrival location of the eventual planner item that precedes this deleted delivery (1) is changed to the main depot, what implies a modification of its duration.

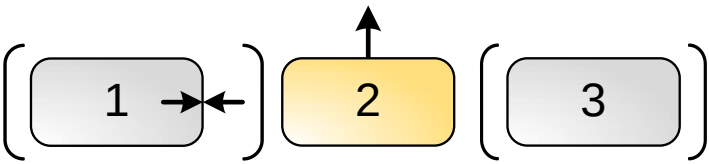

Figure B.36: Case 36

Appendix C

Sets, data and variables of solution methods

This appendix resumes the sets, data and variables used by the solution methods presented in Chapter 8. For this purpose, Table C.1 first defines the sets, Table C.2 then defines the data and Table C.3 finally defines the variables. These tables indicate for each set, data or variable if this one is used by the basic integer linear program (B), the three phase method with MILP (3), the two phase method with MILP (2), the one phase method with MILP (1) and by the one phase method with Tabu Search (T). Note that some variables can be defined several times depending on the solution method that uses them. Furthermore, only the main variables are defined for Tabu Search (T).

Table C.1: Sets of solution methods

Set	Description	Solution methods				
		B	3	2	1	T
V	set of vehicles	✓	✓	✓	✓	✓
C	set of customers	✓	✓	✓	✓	✓
I	set of orders	✓	✓	✓	✓	✓
D	set of depots		✓	✓	✓	✓
$V_i \subset V$	subset of vehicles which can be used to deliver cement for order i	✓	✓	✓	✓	✓
$V^L \subset V$	subset of vehicles which can load cement from local depots		✓	✓	✓	✓
$VP \subset V$	subset of vehicles that are preloaded		✓	✓	✓	✓
$VS \subset V$	subset of vehicles used in a solution					✓
$VC \subset VS$	subset of vehicles whose customer is also visited by another vehicle					✓
$I_c \subset I$	subset of orders placed by customer c	✓	✓	✓	✓	
$I_k \subset I$	subset of orders that vehicle k can handle	✓	✓	✓	✓	✓
$IP^k \subset I$	subset of orders such that preloaded vehicle k can start its daily activity with a delivery for i		✓	✓	✓	✓
$IS^k \subset I$	subset of orders allocated to vehicle k in a solution					✓
$D_i \subset D$	subset of depots where cement can be loaded for order i		✓	✓	✓	✓
TA	tabu list					✓

Table C.2: Data of solution methods

Data	Description	Solution methods				
		B	3	2	1	T
L_{ik}	time needed by vehicle k to load cement for order i	✓				
L_{ijk}	time needed by vehicle k to load cement at depot j for order i		✓	✓	✓	✓
U_{ik}	time needed by vehicle k to unload cement at customer c_i	✓	✓	✓	✓	✓
TL_k	time needed by loaded vehicle k to travel one kilometer	✓	✓	✓	✓	✓
TU_k	time needed by unloaded vehicle k to travel one kilometer	✓	✓	✓	✓	✓
Δ_i	depot j where cement has to be loaded for an order i		✓			
Δ_{ik}	depot j where cement has to be loaded for an order i with vehicle k			✓	✓	✓
δ_c	distance between customer c and the depot	✓				
δ_{cj}	distance between customer c and depot j		✓	✓	✓	✓
θ	weighing factor		✓	✓		
A_k	total availability time of vehicle k	✓	✓	✓	✓	✓
Q_k	capacity of vehicle k	✓	✓	✓	✓	✓
d_i	demand of order i	✓	✓	✓	✓	✓
Nc	maximal number of different vehicles to use to satisfy all orders of a same customer	✓	✓	✓	✓	
Nd	maximal number of different depots that a vehicle can use		✓	✓	✓	
Nk	maximal number of vehicles to use to perform all deliveries	✓	✓	✓	✓	
F_k	Fixed cost for each time unit spent by vehicle k	✓	✓	✓	✓	✓
λ	weighing factor	✓	✓	✓	✓	✓
$iter_{max}$	maximal number of iterations					✓
n_N	size of a neighborhood of solutions					✓
n_{TA}	maximal size of the tabu list					✓
$iter_{mod}$	number of iterations without modification of the current solution					✓
rt_{limit}	run time limit					✓
ct_i	cement type of order i					✓

Table C.3: Variables of solution methods

Variable	Description	Solution methods				
		B	3	2	1	T
n_{ik}	number of deliveries made by vehicle k for order i	✓	✓	✓	✓	✓
x_{ik}	1 if at least one delivery is made for order i by vehicle k, 0 otherwise	✓	✓	✓	✓	✓
v_{ck}	1 if vehicle k makes at least one delivery to customer c, 0 otherwise	✓	✓	✓	✓	✓
w_k	1 if vehicle k makes at least one delivery, 0 otherwise	✓	✓	✓	✓	✓
t_k^i	position of i in the route of vehicle k if $i \in I$ and $x_{ik} = 1$, 0 otherwise	✓	✓	✓	✓	✓
$s_{ii'}^k$	1 if i' is the immediate successor of i on the route of vehicle k, 0 otherwise	✓	✓	✓	✓	✓
u_{ij}	1 if order i is allocated to depot j, 0 otherwise		✓			
y_{ik}	1 if preloaded vehicle k makes its first delivery for order i, 0 otherwise		✓	✓	✓	
z_{jk}	1 if vehicle k makes at least one delivery for an order i with $\Delta_i = j$, 0 otherwise		✓			
z_{jk}	1 if vehicle k makes at least one delivery for an order i with $\Delta_{ik} = j$, 0 otherwise			✓	✓	
e_i	delivered quantity for order i					✓
$iter$	current iteration					✓
$iter_{best}$	best iteration					✓
TD_{first}^k	loading time, travel time and unloading time of the first delivery of vehicle k					✓
TD_{last}^k	back travel time of the last delivery of vehicle k					✓
TD_{inter}^k	all other travel durations of vehicle k					✓

Appendix D

Mixed integer linear programs

This appendix presents the complete mixed integer linear programming models used by the solution methods discussed in Subsections 8.3.1, 8.3.2 and 8.3.3 of Chapter 8. The three mixed integer linear programs of the first solution method, the two mixed integer linear programs of the second solution method and the unique mixed integer linear program of the third solution method are respectively presented in Sections D.1, D.2 and D.3. Note that the sets, data and variables used in these models are defined in Appendix C.

D.1 Solving the problem in three phases

D.1.1 Allocation of each order to a specific depot

Constraints (D.1) and (D.2) impose that each order $i \in I$ is assigned to only one and appropriate depot $j \in D_i$:

$$\sum_{j \in D_i} u_{ij} = 1 \qquad \forall i \in I \tag{D.1}$$

$$\sum_{j \notin D_i} u_{ij} = 0 \qquad \forall i \in I. \tag{D.2}$$

Constraints (D.3) define the domain of the decision variables:

$$u_{ij} \in \{0,1\} \quad \forall i \in I, \forall j \in D_i. \tag{D.3}$$

To minimize the total travel time between the customers and the depot allocated to each of their orders, objective function (D.4) has to be minimized under constraints (D.1)–(D.3):

$$\sum_{i \in I} \sum_{j \in D_i} P_{ij} u_{ij} \tag{D.4}$$

where

$$P_{ij} = \min_{k \in V} \{L_{ijk} + \delta_{c_i j}(TL_k + TU_k)\}.$$

D.1.2 Assignment of deliveries to the vehicles

Constraints (D.5) ensure that no vehicle $k \in V^L$ is used for more that $(1-\theta)A_k$ time units, while constraints (D.6) ensure that no vehicle $k \notin V^L$ is used for more that A_k time units.

$$\sum_{i \in I} R_{ik} n_{ik} \leq (1-\theta) A_k \qquad \forall k \in V^L \tag{D.5}$$

$$\sum_{i \in I} R_{ik} n_{ik} \leq A_k \qquad \forall k \notin V^L \tag{D.6}$$

where

$$R_{ik} = \begin{cases} L_{i\Delta_i k} + U_{ik} + \delta_{c_i \Delta_i}(TL_k + TU_k) & \text{if } k \in V^L \text{ and } \Delta_i \neq 0 \\ L_{i0k} + U_{ik} + \delta_{c_i 0}(TL_k + TU_k) & \text{otherwise.} \end{cases}$$

Constraints (D.7) ensure that all demands are satisfied.

$$\sum_{k \in V_i} Q_k n_{ik} \geq d_i \qquad \forall i \in I. \tag{D.7}$$

Constraints (D.8) impose that each order $i \in I$ is delivered by vehicles that can perform such deliveries.

$$\sum_{k \notin V_i} n_{ik} = 0 \qquad \forall i \in I. \tag{D.8}$$

Constraints (D.9) and (D.10) link variables x_{ik} with variables n_{ik}.

$$n_{ik} \geq x_{ik} \qquad \forall i \in I, \forall k \in V \tag{D.9}$$

$$n_{ik} \leq N_{ik} x_{ik} \qquad \forall i \in I, \forall k \in V \tag{D.10}$$

where

$$N_{ik} = \lceil d_i / Q_k \rceil.$$

Constraints (D.11) link variables x_{ik} with variables v_{ck}.

$$v_{c_i k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V_i. \tag{D.11}$$

Constraints (D.12) ensure that no more than Nc vehicles are used to satisfy the demand of a customer.

$$\sum_{k \in V} v_{ck} \leq Nc \qquad \forall c \in C. \tag{D.12}$$

Constraints (D.13) link variables v_{ck} with variables w_k.

$$w_k \geq v_{ck} \qquad \forall k \in V, \forall c \in C. \tag{D.13}$$

Constraints (D.14) impose that no more than Nk vehicles are used to make all deliveries.

$$\sum_{k \in V} w_k \leq Nk. \tag{D.14}$$

Constraints (D.15) ensure that each preloaded vehicle $k \in VP$ performs its first delivery for an order $i \in IP^k$.

$$\sum_{i \in IP^k} y_{ik} = 1 \qquad \forall k \in VP. \tag{D.15}$$

Constraints (D.16) link variables x_{ik} with variables y_{ik}.

$$y_{ik} \leq x_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{D.16}$$

Constraints (D.17) link variables x_{ik} with variables z_{jk}.

$$z_{\Delta_i k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V^L \cap V_i. \tag{D.17}$$

Constraints (D.18) ensure that no vehicle in V^L loads cement from more than Nd different depots.

$$\sum_{j \in D} z_{jk} \leq Nd \qquad \forall k \in V^L. \tag{D.18}$$

Constraints (D.19)–(D.24) define the domains of the decision variables:

$$n_{ik} \in \mathbb{N} \qquad \forall i \in I, \forall k \in V \tag{D.19}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in I, \forall k \in V \tag{D.20}$$

$$v_{ck} \in \{0,1\} \qquad \forall c \in C, \forall k \in V \tag{D.21}$$

$$w_k \in \{0,1\} \qquad \forall k \in V \tag{D.22}$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in IP^k, \forall k \in VP \tag{D.23}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in D, \forall k \in V^L. \tag{D.24}$$

To minimize the total cost of the vehicle routes, to avoid situations where more than one vehicle are used to supply the demands of a customer and to minimize the number of vehicles used to make the deliveries, objective function (D.25) has to be minimized under constraints (D.5)–(D.24):

$$M(\sum_{c \in C} \sum_{k \in V} v_{ck} + \lambda \sum_{k \in V} w_k) + \sum_{i \in I} \sum_{k \in V_i} F_k R_{ik} n_{ik} \tag{D.25}$$

where

$$M = \sum_{k \in V} F_k A_k.$$

D.1.3 Sequencing the orders on each route

Constraints (D.26) and (D.27) impose that an order i has a successor and a predecessor on the route of vehicle k if and only if vehicle k makes at least one delivery for order i:

$$\sum_{\substack{i' \in S'_k \\ i \neq i'}} s^k_{ii'} = 1 \qquad \forall k \in V, \forall i \in S'_k \tag{D.26}$$

$$\sum_{\substack{i' \in S'_k \\ i \neq i'}} s^k_{i'i} = 1 \qquad \forall k \in V, \forall i \in S'_k. \tag{D.27}$$

Constraints (D.28) impose that every preloaded vehicle $k \in VP$ makes its first delivery for the order $i \in IP^k$ with $y_{ik} = 1$.

$$s^k_{0i} \geq y_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{D.28}$$

Constraints (D.29) and (D.30) eliminate subtours:

$$0 \leq t_i^k \leq |S_k| \qquad \forall k \in V, \forall i \in S_k \tag{D.29}$$

$$t_i^k - t_{i'}^k + |S_k|\, s_{ii'}^k + (|S_k| - 2)\, s_{i'i}^k \leq |S_k| - 1 \;\forall k \in V, \forall i \in S'_k, \forall i' \in S_k, i \neq i'. \tag{D.30}$$

Constraints (D.31)–(D.33) define the domains of the decision variables:

$$t_i^k \in \mathbb{N} \qquad \forall i \in I \cup \{0\}, \forall k \in V \tag{D.31}$$

$$s_{ii'}^k \in \{0,1\} \qquad \forall i \in I \cup \{0\}, \forall i' \in I \cup \{0\}, \forall k \in V \tag{D.32}$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in IP^k, \forall k \in VP. \tag{D.33}$$

To minimize the real delivery time, the estimation error which is defined with objective function (D.34) has to be minimized under constraints (D.26)–(D.33):

$$\sum_{k \in V} \sum_{i \in S'_k} \sum_{\substack{i' \in S'_k \\ i \neq i'}} T_{ii'}^k s_{ii'}^k \tag{D.34}$$

where

$$T_{ii'}^k = \min_{j \in D_{i'}} \{TU_k(\delta_{c_i j} - \delta_{c_i \Delta_i}) + (L_{i'jk} - L_{i'\Delta_{i'}k}) + TL_k(\delta_{c_{i'} j} - \delta_{c_{i'}\Delta_{i'}})\}.$$

D.2 Solving the problem in two phases

D.2.1 Assignment of deliveries to the vehicles

Constraints (D.35) ensure that no vehicle $k \in V^L$ is used for more that $A_k - \theta(E_1^k + E_2^k)$ time units, while constraints (D.36) ensure that no vehicle $k \notin V^L$ is used for more that A_k time units.

$$\sum_{i \in I} R_{ik} n_{ik} \leq A_k - \theta(E_1^k + E_2^k) \qquad \forall k \in V^L \tag{D.35}$$

$$\sum_{i \in I} R_{ik} n_{ik} \leq A_k \qquad \forall k \notin V^L \tag{D.36}$$

where

$$R_{ik} = \begin{cases} L_{i\Delta_{ik}k} + U_{ik} + \delta_{c_i\Delta_{ik}}(TL_k + TU_k) & \text{if } k \in V^L \text{ and } \Delta_{ik} \neq 0 \\ L_{i0k} + U_{ik} + \delta_{c_i 0}(TL_k + TU_k) & \text{otherwise} \end{cases}$$

and

$$E_1^k = w_k(\max_{i \in I_k} T_{0i}^k + \max_{i \in I_k} T_{i0}^k)$$

and

$$T_{0i}^k = \begin{cases} -L_{i\Delta_{ik}k} + TL_k(\delta_{c_i 0} - \delta_{c_i \Delta_{ik}}) & \text{if } k \in VP \\ (L_{i0k} - L_{i\Delta_{ik}k}) + TL_k(\delta_{c_i 0} - \delta_{c_i \Delta_{ik}}) & \text{if } k \notin VP \end{cases}$$

and

$$T_{i0}^k = TU_k(\delta_{c_i 0} - \delta_{c_i \Delta_{ik}})$$

and

$$E_2^k = \begin{cases} (\sum_{i \in I_k} x_{ik} - w_k) \max_{\substack{i,i' \in I_k \\ i \neq i'}} T_{ii'}^k & \text{if } |I_k| > 1 \\ 0 & \text{otherwise} \end{cases}$$

and

$$T_{ii'}^k = \min_{j \in D_{i'}} \{TU_k(\delta_{c_i j} - \delta_{c_i \Delta_{ik}}) + (L_{i'jk} - L_{i'\Delta_{i'k}k}) + TL_k(\delta_{c_{i'} j} - \delta_{c_{i'} \Delta_{i'k}})\}.$$

Constraints (D.37) ensure that all demands are satisfied.

$$\sum_{k \in V_i} Q_k n_{ik} \geq d_i \qquad \forall i \in I. \tag{D.37}$$

Constraints (D.38) impose that each order $i \in I$ is delivered by vehicles that can perform such deliveries.

$$\sum_{k \notin V_i} n_{ik} = 0 \qquad \forall i \in I. \tag{D.38}$$

Constraints (D.39) and (D.40) link variables x_{ik} with variables n_{ik}.

$$n_{ik} \geq x_{ik} \qquad \forall i \in I, \forall k \in V \tag{D.39}$$

$$n_{ik} \leq N_{ik} x_{ik} \qquad \forall i \in I, \forall k \in V \tag{D.40}$$

where

$$N_{ik} = \lceil d_i / Q_k \rceil.$$

Constraints (D.41) link variables x_{ik} with variables v_{ck}.

$$v_{c_i k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V_i. \tag{D.41}$$

Constraints (D.42) ensure that no more than Nc vehicles are used to satisfy the demand of a customer.

$$\sum_{k \in V} v_{ck} \leq Nc \qquad \forall c \in C. \tag{D.42}$$

Constraints (D.43) link variables v_{ck} with variables w_k.

$$w_k \geq v_{ck} \qquad \forall k \in V, \forall c \in C. \tag{D.43}$$

Constraints (D.44) impose that no more than Nk vehicles are used to make all deliveries.

$$\sum_{k \in V} w_k \leq Nk. \tag{D.44}$$

Constraints (D.45) ensure that each preloaded vehicle $k \in VP$ performs its first delivery for an order $i \in IP^k$.

$$\sum_{i \in IP^k} y_{ik} = 1 \qquad \forall k \in VP. \tag{D.45}$$

Constraints (D.46) link variables x_{ik} with variables y_{ik}.

$$y_{ik} \leq x_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{D.46}$$

Constraints (D.47) link variables x_{ik} with variables z_{jk}.

$$z_{\Delta_{ik}k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V^L \cap V_i. \tag{D.47}$$

Constraints (D.48) ensure that no vehicle in V^L loads cement from more than Nd different depots.

$$\sum_{j \in D} z_{jk} \leq Nd \qquad \forall k \in V^L. \tag{D.48}$$

Constraints (D.49)–(D.54) define the domains of the decision variables:

$$n_{ik} \in \mathbb{N} \qquad \forall i \in I, \forall k \in V \tag{D.49}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in I, \forall k \in V \tag{D.50}$$

$$v_{ck} \in \{0,1\} \qquad \forall c \in C, \forall k \in V \tag{D.51}$$

$$w_k \in \{0,1\} \qquad \forall k \in V \tag{D.52}$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in IP^k, \forall k \in VP \tag{D.53}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in D, \forall k \in V^L. \tag{D.54}$$

To minimize the total cost of the vehicle routes, to avoid situations where more than one vehicle are used to supply the demands of a customer and to minimize the number of vehicles used to make the deliveries, objective function (D.55) has to be minimized under constraints (D.35)–(D.54):

$$M(\sum_{c\in C}\sum_{k\in V} v_{ck} + \lambda \sum_{k\in V} w_k) + \sum_{i\in I}\sum_{k\in V_i} F_k R_{ik} n_{ik} \tag{D.55}$$

where

$$M = \sum_{k\in V} F_k A_k.$$

D.2.2 Sequencing the orders on each route

Constraints (D.56) and (D.57) impose that an order i has a successor and a predecessor on the route of vehicle k if and only if vehicle k makes at least one delivery for order i:

$$\sum_{\substack{i'\in S'_k \\ i\neq i'}} s^k_{ii'} = 1 \qquad \forall k \in V, \forall i \in S'_k \tag{D.56}$$

$$\sum_{\substack{i'\in S'_k \\ i\neq i'}} s^k_{i'i} = 1 \qquad \forall k \in V, \forall i \in S'_k. \tag{D.57}$$

Constraints (D.58) impose that every preloaded vehicle $k \in VP$ makes its first delivery for the order $i \in IP^k$ with $y_{ik} = 1$.

$$s^k_{0i} \geq y_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{D.58}$$

Constraints (D.59) and (D.60) eliminate subtours:

$$0 \leq t^k_i \leq |S_k| \qquad \forall k \in V, \forall i \in S_k \tag{D.59}$$

$$t^k_i - t^k_{i'} + |S_k|\, s^k_{ii'} + (|S_k| - 2)\, s^k_{i'i} \leq |S_k| - 1 \;\forall k \in V, \forall i \in S'_k, \forall i' \in S_k, i \neq i'. \tag{D.60}$$

Constraints (D.61)–(D.63) define the domains of the decision variables:

$$t^k_i \in \mathbb{N} \qquad \forall i \in I \cup \{0\}, \forall k \in V \tag{D.61}$$

$$s^k_{ii'} \in \{0,1\} \qquad \forall i \in I \cup \{0\}, \forall i' \in I \cup \{0\}, \forall k \in V \tag{D.62}$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in IP^k, \forall k \in VP. \tag{D.63}$$

To minimize the real delivery time, the estimation error which is defined with objective function (D.64) has to be minimized under constraints (D.56)–(D.63):

$$\sum_{k\in V}\sum_{i\in S'_k}\sum_{\substack{i'\in S'_k\\ i\neq i'}} T^k_{ii'} s^k_{ii'} \tag{D.64}$$

where

$$T^k_{ii'} = \min_{j\in D_{i'}}\{TU_k(\delta_{c_i j} - \delta_{c_i \Delta_{ik}}) + (L_{i'jk} - L_{i'\Delta_{i'k}k}) + TL_k(\delta_{c_{i'} j} - \delta_{c_{i'}\Delta_{i'k}})\}.$$

D.3 Solving the problem in one phase

Constraints (D.65) ensure that no vehicle $k \in V$ is used for more that A_k time units.

$$\alpha_k \leq A_k \qquad \forall k \in V \tag{D.65}$$

where

$$\alpha_k = \sum_{i\in I} R_{ik} n_{ik} + \sum_{i\in I\cup\{0\}}\sum_{\substack{i'\in I\cup\{0\}\\ i\neq i'}} T^k_{ii'} s^k_{ii'}$$

and

$$R_{ik} = \begin{cases} L_{i\Delta_{ik}k} + U_{ik} + \delta_{c_i\Delta_{ik}}(TL_k + TU_k) & \text{if } k \in V^L \text{ and } \Delta_{ik} \neq 0 \\ L_{i0k} + U_{ik} + \delta_{c_i 0}(TL_k + TU_k) & \text{otherwise} \end{cases}$$

and

$$T^k_{ii'} = \min_{j\in D_{i'}}\{TU_k(\delta_{c_i j} - \delta_{c_i \Delta_{ik}}) + (L_{i'jk} - L_{i'\Delta_{i'k}k}) + TL_k(\delta_{c_{i'} j} - \delta_{c_{i'}\Delta_{i'k}})\}.$$

Constraints (D.66) ensure that all demands are satisfied.

$$\sum_{k\in V_i} Q_k n_{ik} \geq d_i \qquad \forall i \in I. \tag{D.66}$$

Constraints (D.67) impose that each order $i \in I$ is delivered by vehicles that can perform such deliveries.

$$\sum_{k\notin V_i} n_{ik} = 0 \qquad \forall i \in I. \tag{D.67}$$

Constraints (D.68) and (D.69) link variables x_{ik} with variables n_{ik}.

$$n_{ik} \geq x_{ik} \qquad \forall i \in I, \forall k \in V \tag{D.68}$$

$$n_{ik} \leq N_{ik} x_{ik} \qquad \forall i \in I, \forall k \in V \tag{D.69}$$

where

$$N_{ik} = \lceil d_i / Q_k \rceil.$$

Constraints (D.70) link variables x_{ik} with variables v_{ck}.

$$v_{c_i k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V_i. \tag{D.70}$$

Constraints (D.71) ensure that no more than Nc vehicles are used to satisfy the demand of a customer.

$$\sum_{k \in V} v_{ck} \leq Nc \qquad \forall c \in C. \tag{D.71}$$

Constraints (D.72) link variables v_{ck} with variables w_k.

$$w_k \geq v_{ck} \qquad \forall k \in V, \forall c \in C. \tag{D.72}$$

Constraints (D.73) impose that no more than Nk vehicles are used to make all deliveries.

$$\sum_{k \in V} w_k \leq Nk. \tag{D.73}$$

Constraints (D.74) and (D.75) impose that an order i has a successor and a predecessor on the route of vehicle k if and only if vehicle k makes at least one delivery for order i.

$$\sum_{\substack{i' \in I \cup \{0\} \\ i \neq i'}} s^k_{ii'} = x_{ik} \qquad \forall k \in V, \forall i \in I \tag{D.74}$$

$$\sum_{\substack{i' \in I \cup \{0\} \\ i \neq i'}} s^k_{i'i} = x_{ik} \qquad \forall k \in V, \forall i \in I. \tag{D.75}$$

Constraints (D.76) and (D.77) impose that a vehicle k leaves the depot and turns back to the depot if only if it makes at least one delivery.

$$\sum_{i \in I} s^k_{0i} = w_k \qquad \forall k \in V \tag{D.76}$$

$$\sum_{i \in I} s_{i0}^k = w_k \qquad \forall k \in V. \tag{D.77}$$

Constraints (D.78) impose that every preloaded vehicle $k \in VP$ makes its first delivery for the order $i \in IP^k$ with $y_{ik} = 1$.

$$s_{0i}^k \geq y_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{D.78}$$

Since each route starts at the main depot, let

$$t_0^k = 0 \qquad \forall k \in V. \tag{D.79}$$

Constraints (D.80) and (D.81) eliminate subtours.

$$0 \leq t_i^k \leq |I| \qquad \forall k \in V, \forall i \in I \tag{D.80}$$

$$t_i^k - t_{i'}^k + |I|\, s_{ii'}^k + (|I| - 2)\, s_{i'i}^k \leq |I| - 1 \; \forall k \in V, \forall i \in I \cup \{0\}, \forall i' \in I, i' \neq i. \tag{D.81}$$

Constraints (D.82) ensure that each preloaded vehicle $k \in VP$ performs its first delivery for an order $i \in IP^k$.

$$\sum_{i \in IP^k} y_{ik} = 1 \qquad \forall k \in VP. \tag{D.82}$$

Constraints (D.83) link variables x_{ik} with variables y_{ik}.

$$y_{ik} \leq x_{ik} \qquad \forall k \in VP, \forall i \in IP^k. \tag{D.83}$$

Constraints (D.84) link variables x_{ik} with variables z_{jk}.

$$z_{\Delta_{ik} k} \geq x_{ik} \qquad \forall i \in I, \forall k \in V^L \cap V_i. \tag{D.84}$$

Constraints (D.85) ensure that no vehicle in V^L loads cement from more than Nd different depots.

$$\sum_{j \in D} z_{jk} \leq Nd \qquad \forall k \in V^L. \tag{D.85}$$

Constraints (D.86)–(D.93) define the domains of the decision variables:

$$n_{ik} \in \mathbb{N} \qquad \forall i \in I, \forall k \in V \tag{D.86}$$

$$x_{ik} \in \{0,1\} \qquad \forall i \in I, \forall k \in V \tag{D.87}$$

$$v_{ck} \in \{0,1\} \qquad \forall c \in C, \forall k \in V \tag{D.88}$$

$$w_k \in \{0,1\} \qquad \forall k \in V \tag{D.89}$$

$$t_i^k \in \mathbb{N} \qquad \forall i \in I \cup \{0\}, \forall k \in V \tag{D.90}$$

$$s_{ii'}^k \in \{0,1\} \qquad \forall i \in I \cup \{0\}, \forall i' \in I \cup \{0\}, \forall k \in V \tag{D.91}$$

$$y_{ik} \in \{0,1\} \qquad \forall i \in IP^k, \forall k \in VP \tag{D.92}$$

$$z_{jk} \in \{0,1\} \qquad \forall j \in D, \forall k \in V^L. \tag{D.93}$$

To minimize the total cost of the vehicle routes, to avoid situations where more than one vehicle are used to supply the demands of a customer and to minimize the number of vehicles used to make the deliveries, objective function (D.94) has to be minimized under constraints (D.65)–(D.93):

$$M(\sum_{c \in C} \sum_{k \in V} v_{ck} + \lambda \sum_{k \in V} w_k) + \sum_{i \in I} \sum_{k \in V_i} F_k \alpha_k \tag{D.94}$$

where

$$M = \sum_{k \in V} F_k A_k.$$

Appendix E

Tabu search algorithm

This appendix presents a detailed pseudocode of the tabu search algorithm discussed in Subsection 8.3.4 of Chapter 8, or more precisely of the procedures used by this solution method. The different phases related to the construction of an initial solution, the improvement of a solution, the generation of the best neighbor (deletion & insertion), the optimization of the insertion, the generation of the best neighbor (delivery swap), the generation of the best neighbor (order swap), and the modification of a solution are respectively presented in Sections E.1, E.2, E.3, E.4, E.5, E.6 and E.7. Note that some sets, data and variables used in these procedures are defined in Appendix C.

E.1 Construction of an initial solution

E.1.1 Initialization phase

This phase is described in detail in Procedure E.1.1.

A boolean B_k for each vehicle $k \in VP$, the number of deliveries n_{ik} done by vehicle $k \in V$ for order $i \in I$, $s^k_{ii'}$ indicating if order $i' \in I$ is the immediate successor of order $i \in I$ on the route of vehicle $k \in V$ and the delivered quantity e_i for order $i \in I$ are set to 0, as TD^k_{first}, TD^k_{inter} and TD^k_{last}.

Procedure E.1.1 Construction of an initial solution (Part 1)

procedure CONSTRUCTINITIALSOLUTION
  $B_k \in \{0,1\} \leftarrow 0 \quad \forall k \in VP$
  $n_{ik} \leftarrow 0 \quad \forall i \in I, \forall k \in V$
  $s^k_{ii'} \leftarrow 0 \quad \forall k \in V, \forall i \in I \cup \{0\}, \forall i' \in I \cup \{0\}, i' \neq i$
  $e_i \leftarrow 0 \quad \forall i \in I$
  $TD^k_{first} \leftarrow 0 \quad \forall k \in V$
  $TD^k_{inter} \leftarrow 0 \quad \forall k \in V$
  $TD^k_{last} \leftarrow 0 \quad \forall k \in V$

E.1.2 First construction phase

This phase is described in detail in Procedure E.1.2. An appropriate order $i \in IP^k$ is allocated to the first delivery of every preloaded vehicle $k \in VP$. If no vehicle $k \in V$ is preloaded, this phase is skipped.

Procedure E.1.2 Construction of an initial solution (Part 2)

  for $k \in VP = 1 \rightarrow |VP|$ **do**
    $i \in \{0\} \cup IP^k \leftarrow 0$
    repeat
      $i \leftarrow i + 1$
      if $e_i < d_i$ **and** $A_k \geq \delta_{c_i 0}(TL_k + TU_k) + U_{ik}$ **then**
        $e_i \leftarrow e_i + Q_k$
        $TD^k_{first} \leftarrow \delta_{c_i 0} TL_k + U_{ik}$
        $TD^k_{last} \leftarrow \delta_{c_i 0} TU_k$
        $n_{ik} \leftarrow n_{ik} + 1$
        $s^k_{0i} \leftarrow 1$
        $s^k_{i0} \leftarrow 1$
        $B_k \leftarrow 1$
      end if
    until $i = |IP^k|$ **or** $B_k = 1$
  end for

For each preloaded vehicle $k \in VP$, an appropriate order $i \in IP^k$ is allocated to its first delivery but only if the order is not entirely satisfied ($e_i < d_i$) and if the vehicle has enough availability to do this delivery ($A_k \geq \delta_{c_i 0}(TL_k + TU_k) + U_{ik}$). For this purpose, orders $i \in IP^k$ are successively explored until these two conditions are satisfied. It is assumed that there is at least one appropriate order i for the first delivery of each preloaded vehicle k.

When the two conditions mentioned before are satisfied, the delivered quantity e_i for order i, TD^k_{first} and TD^k_{last} are updated to take into account the quan-

tity and the duration of the preloaded delivery. Furthermore, the number of deliveries n_{ik} done by vehicle k for order i is updated, as s^k_{0i} and s^k_{i0} to modify the route of vehicle k. Finally, the boolean B_k is set to 1 to stop the exploration of remaining orders.

E.1.3 Second construction phase

This phase is described in detail in Procedure E.1.3. Orders $i' \in I_k$ are split into deliveries and assigned to preloaded vehicles $k \in VP$. If no vehicle $k \in V$ is preloaded, this phase is skipped.

Procedure E.1.3 Construction of an initial solution (Part 3)

for $k \in VP = 1 \rightarrow |VP|$ **do**
  for $i' \in I_k = 1 \rightarrow |I_k|$ **do**
    if $e_{i'} < d_{i'}$ **then**
      $i \in I_k \leftarrow$ last order assigned to vehicle k $\quad (i \neq i')$
      $\eta \leftarrow \lceil \frac{d_{i'} - e_{i'}}{Q_k} \rceil$
      $\Lambda \leftarrow \max\{0, A_k - TD^k_{first} + TD^k_{inter} + \delta_{c_{i'}0} TU_k\}$
      $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_k + L_{i'jk} + \delta_{c_{i'} j} TL_k\} + U_{i'k}$
      if $\Lambda - \Upsilon \geq 0$ **then**
        $e_{i'} \leftarrow e_{i'} + Q_k$
        $TD^k_{inter} \leftarrow TD^k_{inter} + \Upsilon$
        $TD^k_{last} \leftarrow \delta_{c_{i'}0} TU_k$
        $n_{i'k} \leftarrow n_{i'k} + 1$
        $s^k_{ii'} \leftarrow 1$
        $s^k_{i0} \leftarrow 0$
        $s^k_{i'0} \leftarrow 1$
        $\eta \leftarrow \eta - 1$
        $\Lambda \leftarrow \Lambda - \Upsilon$
        $\Upsilon \leftarrow L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k)$
        $\epsilon \leftarrow \min\{\lfloor \frac{\Lambda}{\Upsilon} \rfloor, \eta\}$
        if $\epsilon > 0$ **then**
          $e_{i'} \leftarrow e_{i'} + \epsilon Q_k$
          $TD^k_{inter} \leftarrow TD^k_{inter} + \epsilon \Upsilon$
          $n_{i'k} \leftarrow n_{i'k} + \epsilon$
        end if
      end if
    end if
  end for
end for

For each preloaded vehicle $k \in VP$ and for each order $i' \in I_k$ that is not entirely satisfied ($e_{i'} < d_{i'}$), the last order $i \in I$ assigned to vehicle k, the number η of deliveries needed to entirely satisfy order i' with vehicle k, the remaining availability Λ of vehicle k for the addition of deliveries of order i' and the required availability Υ to do a first delivery of order i' with vehicle k are first determined.

If vehicle k has enough availability to do a first delivery for order i' ($\Lambda - \Upsilon \geq 0$), the delivered quantity $e_{i'}$ for order i', TD^k_{inter} and TD^k_{last} are updated to take into account the quantity and the duration of the delivery. Furthermore, the number of deliveries $n_{i'k}$ done by vehicle k for order i' is updated, as $s^k_{ii'}$, s^k_{i0} and $s^k_{i'0}$ to modify the route of vehicle k. Following the addition of a first delivery for order i', the number η of deliveries needed with vehicle k to entirely satisfy order i', the remaining availability Λ of vehicle k for additional deliveries of order i' and the required availability Υ to do an additional delivery of order i' with vehicle k are then updated.

If additional deliveries can and have to be done with vehicle k for order i', ($\epsilon > 0$), the delivered quantity $e_{i'}$ for order i' and TD^k_{inter} are updated to take into account the quantity and the duration of these deliveries, as the number of deliveries $n_{i'k}$ done by vehicle k for order i'.

E.1.4 Third construction phase

This phase is described in detail in Procedure E.1.4. Orders $i' \in I_k$ are split into deliveries and assigned to vehicles $k \in V \setminus VP$ that are not preloaded.

For each vehicle $k \in V \setminus VP$ and for each order $i' \in I_k$ that is not entirely satisfied ($e_{i'} < d_{i'}$), the number η of deliveries needed to entirely satisfy order i' with vehicle k and the remaining availability Λ of vehicle k for the addition of deliveries of order i' are first determined.

If vehicle k has no deliveries ($\sum_{i \in I_k} s^k_{0i} = 0$), the required availability Υ to do a first delivery of order i' with vehicle k from the main depot is determined. If deliveries of other orders have already been assigned to vehicle k, the last order $i \in I$ assigned to vehicle k and the required availability Υ to do a first delivery of order i' with vehicle k from the appropriate depot are determined.

If vehicle k has enough availability to do a first delivery for order i' ($\Lambda - \Upsilon \geq 0$), the delivered quantity $e_{i'}$ for order i' is updated to take into account the quantity of the delivery. Furthermore, the number of deliveries $n_{i'k}$ done by vehicle k for order i' is updated, as $s^k_{i'0}$ to modify the route of vehicle k.

Procedure E.1.4 Construction of an initial solution (Part 4)

for $k \in V \setminus VP = 1 \rightarrow |V \setminus VP|$ **do**
  for $i' \in I_k = 1 \rightarrow |I_k|$ **do**
    if $e_{i'} < d_{i'}$ **then**
      $\eta \leftarrow \lceil \frac{d_{i'} - e_{i'}}{Q_k} \rceil$
      $\Lambda \leftarrow \max\{0, A_k - TD^k_{first} + TD^k_{inter} + \delta_{c_{i'}0} TU_k\}$
      if $\sum_{i \in I_k} s^k_{0i} = 0$ **then**
        $\Upsilon \leftarrow L_{i'0k} + \delta_{c_{i'}0} TL_k + U_{i'k}$
      else
        $i \in I_k \leftarrow$ last order assigned to vehicle k $\quad (i \neq i')$
        $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_k + L_{i'jk} + \delta_{c_{i'} j} TL_k\} + U_{i'k}$
      end if
      if $\Lambda - \Upsilon \geq 0$ **then**
        $e_{i'} \leftarrow e_{i'} + Q_k$
        $n_{i'k} \leftarrow n_{i'k} + 1$
        $s^k_{i'0} \leftarrow 1$
        if $TD^k_{first} = 0$ **then**
          $TD^k_{first} \leftarrow \Upsilon$
          $s^k_{0i'} \leftarrow 1$
        else
          $TD^k_{inter} \leftarrow TD^k_{inter} + \Upsilon$
          $s^k_{ii'} \leftarrow 1$
          $s^k_{i0} \leftarrow 0$
        end if
        $TD^k_{last} \leftarrow \delta_{c_{i'}0} TU_k$
        $\eta \leftarrow \eta - 1$
        $\Lambda \leftarrow \Lambda - \Upsilon$
        $\Upsilon \leftarrow L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}} (TU_k + TL_k)$
        $\epsilon \leftarrow \min\{\lfloor \frac{\Lambda}{\Upsilon} \rfloor, \eta\}$
        if $\epsilon > 0$ **then**
          $e_{i'} \leftarrow e_{i'} + \epsilon Q_k$
          $TD^k_{inter} \leftarrow TD^k_{inter} + \epsilon \Upsilon$
          $n_{i'k} \leftarrow n_{i'k} + \epsilon$
        end if
      end if
    end if
  end for
end for

If this is the first delivery for vehicle k, TD^k_{first} is updated to take into account the duration of the delivery, as $s^k_{0i'}$ to modify the route of vehicle k. Otherwise, TD^k_{inter} is updated to take into account the duration of the delivery, as $s^k_{ii'}$ and

s^k_{i0} to modify the route of vehicle k. In both cases, TD^k_{last} is updated to take into account the duration of the delivery.

Following the addition of a first delivery for order i', the number η of deliveries needed with vehicle k to entirely satisfy order i', the remaining availability Λ of vehicle k for additional deliveries of order i' and the required availability Υ to do an additional delivery of order i' with vehicle k are then updated.

If additional deliveries can and have to be done with vehicle k for order i', ($\epsilon > 0$), the delivered quantity $e_{i'}$ for order i' and TD^k_{inter} are updated to take into account the quantity and the duration of these deliveries, as the number of deliveries $n_{i'k}$ done by vehicle k for order i'.

It is assumed that there is enough vehicles to construct an initial feasible solution. To have a solution of better quality, it is possible for the *second construction phase* and for the *third construction phase* on the one hand to sort the vehicles $k \in V$ in ascending order of their penalty of use and also to begin with vehicles $k \in V^L$ that can load cement at local depots, and on the other hand to sort the orders $i \in I$ in descending order of the time saving if a local depot is used instead of the main depot. These two combined sorts allow the creation of deliveries of smaller duration.

After having split every order $i \in I$ into deliveries and assigned these deliveries on vehicles $k \in V$, $s^k_{ii'}$ and n_{ik} are returned as solution $\mathfrak{s}$ (cf. Procedure E.1.5).

Procedure E.1.5 Construction of an initial solution (Part 5)

```
89:    𝔰 ← {s^k_{ii'} | k ∈ V, i ∈ I ∪ {0}, i' ∈ I ∪ {0} and i' ≠ i} ∪ {n_{ik} | i ∈ I and k ∈ V}
90:    return 𝔰
91: end procedure
```

E.2 Improvement of a solution

One parameter is needed to run the corresponding procedure: a solution $\mathfrak{s}$. Note that the specification of a vehicle $k^* \in V$ as second parameter is optional and therefore put in brackets.

E.2.1 Initialization phase

This phase is described in detail in Procedure E.2.1.

Procedure E.2.1 Improvement of a solution (Part 1)

1: **procedure** IMPROVESOLUTION($\mathfrak{s}$, $[k^* \in V]$)
2: $VS \leftarrow$ set of vehicles used in solution $\mathfrak{s}$
3: $x_{ik} \leftarrow 1$ if order i is visited by vehicle k in solution $\mathfrak{s}$, 0 otherwise
4: $t_i^k \leftarrow$ position of order i on the route of vehicle k in solution $\mathfrak{s}$
5: $\mathfrak{s}' \leftarrow \emptyset$

The set VS of vehicles used, x_{ik} indicating if order $i \in I$ is visited by vehicle $k \in V$ and the position t_i^k of order $i \in I$ on the route of vehicle $k \in V$ in given solution $\mathfrak{s}$ are first defined as follows:

$$VS = \{k \in V | \sum_{i \in I} n_{ik} \geq 1\}$$

$$x_{ik} = \begin{cases} 1 & \text{if } n_{ik} \geq 1 \quad \forall i \in I, \forall k \in V \\ 0 & \text{otherwise} \end{cases}$$

$$t_0^k = 0 \qquad \forall k \in V$$

$$0 \leq t_i^k \leq |I| \qquad \forall k \in V, \forall i \in I$$

$$t_i^k - t_{i'}^k + |I|\, s_{ii'}^k + (|I| - 2)\, s_{i'i}^k \leq |I| - 1 \;\forall k \in V, \forall i \in I \cup \{0\}, \forall i' \in I, i' \neq i.$$

The improved solution $\mathfrak{s}'$ is then initialized as an empty set $\emptyset$.

E.2.2 Improvement phase

This phase is described in detail in Procedure E.2.2 from line 6 to line 34. The delivery sequence of orders on the route of each vehicle $k \in VS$ is optimized.

For each vehicle $k \in VS$, the partial feasible solution related to its initial route is first set as best partial feasible solution $\mathfrak{s}_k^*$. Note that the optimization process is required only if vehicle k can be loaded at local depots ($k \in V^L$) and, if mentioned, if vehicle k also corresponds to a specified vehicle ($k = k^*$). Indeed, the modification of the position of orders in the delivery sequence of a vehicle that can only be loaded at the main depot has no incidence on its total travel duration.

If the optimization process is required, all the possible permutations $\eta!$ of the position of orders on the route of vehicle k are tested. For each possibility, the positions t_i^k of orders in the sequence are updated, as s_{0i}^k, $s_{ii'}^k$ and s_{i0}^k to modify

Procedure E.2.2 Improvement of a solution (Part 2)

6: **for** $k \in VS = 1 \rightarrow |VS|$ **do**
7: $\mathfrak{s}_k^* \leftarrow \{s_{ii'}^k | i \in I \cup \{0\}, i' \in I \cup \{0\}$ and $i' \neq i\} \cup \{n_{ik} | i \in I\}$
8: **if** $k \in V^L$ [**and** $k = k^*$] **then**
9: $\eta \leftarrow \sum_{i \in I} x_{ik}$
10: **for** $l = 1 \rightarrow \eta!$ **do**
11: $\phi \leftarrow 1$
12: **for** $m = 2 \rightarrow \eta$ **do**
13: $\phi \leftarrow \phi(m-1)$
14: $\omega \leftarrow m - (((l-1) \div \phi) \mod m)$
15: $t_i^k \leftarrow -1$ where $t_i^k = \omega$
16: $t_i^k \leftarrow \omega$ where $t_i^k = m$
17: $t_i^k \leftarrow m$ where $t_i^k = -1$
18: **end for**
19: $s_{ii'}^k \leftarrow 0 \quad \forall i \in I, \forall i' \in I, i' \neq i$
20: $s_{0i}^k \leftarrow 1$ where $t_i^k = 1$
21: **for** $m = 2 \rightarrow \eta - 1$ **do**
22: $s_{ii'}^k \leftarrow 1$ where $t_i^k = m$ and $t_{i'}^k = m + 1$
23: **end for**
24: $s_{i0}^k \leftarrow 1$ where $t_i^k = \eta$
25: $\mathfrak{s}_k \leftarrow \{s_{ii'}^k | i \in I \cup \{0\}, i' \in I \cup \{0\}$ and $i' \neq i\} \cup \{n_{ik} | i \in I\}$
26: **if** $k \notin VP$ **or** ($k \in VP$ **and** $i \in IP^k$ where $t_i^k = 1$) **then**
27: **if** $f(\mathfrak{s}_k) < f(\mathfrak{s}_k^*)$ **then**
28: $\mathfrak{s}_k^* \leftarrow \mathfrak{s}_k$
29: **end if**
30: **end if**
31: **end for**
32: **end if**
33: $\mathfrak{s}' \leftarrow \mathfrak{s}' \cup \mathfrak{s}_k^*$
34: **end for**
35: **return** $\mathfrak{s}'$
36: **end procedure**

the route of vehicle k (cf. lines 11 to 24). A new current solution $\mathfrak{s}_k$ is so obtained. If vehicle k is not preloaded ($k \notin VP$) or if vehicle k is preloaded and if its preload is compatible with the first delivery of current partial solution $\mathfrak{s}_k$ ($k \in VP$ and $i \in IP^k$ where $t_i^k = 1$), this partial solution is set as best partial solution $\mathfrak{s}_k^*$ but only if it is better than the current best partial solution ($f(\mathfrak{s}_k) < f(\mathfrak{s}_k^*)$).

After the optimization process or if no optimization process has been required, the best partial solution $\mathfrak{s}_k^*$ of vehicle k is added to the improved solution ($\mathfrak{s}' \cup \mathfrak{s}_k^*$). Note that a best partial solution is always feasible due to the initial partial solution that is feasible.

After having optimized the delivery sequences of vehicles $k \in V$, the improved solution $\mathfrak{s}'$ is returned (cf. Procedure E.2.2 at line 35).

E.3 Generation of the best neighbor (Deletion & Insertion)

Five parameters are needed to run the corresponding procedure: a solution $\mathfrak{s}$, the tabu list TA presented before, the size of a neighborhood n_N, the size of the tabu list n_{TA} and the current best solution $\mathfrak{s}^*$ of the tabu search.

E.3.1 Initialization phase

This phase is described in detail in Procedure E.3.1.

The set VS of vehicles used, the set IS_k of orders allocated to vehicle $k \in V$, x_{ik} indicating if order $i \in I$ is visited by vehicle $k \in V$ and the set X containing only x_{ik} whose order $i \in I$ is visited by vehicle $k \in V$ in given solution $\mathfrak{s}$ are first defined as follows:

$$VS = \{k \in V | \sum_{i \in I} n_{ik} \geq 1\}$$

$$IS_k = \{i \in I | n_{ik} \geq 1 \quad \forall k \in V\}$$

$$x_{ik} = \begin{cases} 1 & \text{if } n_{ik} \geq 1 \quad \forall i \in I, \forall k \in V \\ 0 & \text{otherwise.} \end{cases}$$

$$X = \{x_{ik} | i \in I, k \in V \text{ and } x_{ik} = 1\}.$$

The solution $\mathfrak{s}'$ related to the best neighbor is then initialized as an empty set $\emptyset$ and the current solution $\mathfrak{s}$ is saved in $\mathfrak{s}^\circ$.

Finally, TD^k_{first}, TD^k_{inter} and TD^k_{last} presented before are initialized to 0 and updated (cf. lines 11 to 31) for each vehicle $k \in VS$. Indeed, the duration related to the loading time, the travel time and the unloading time of the first delivery TD^k_{first} is first computed. Note that the loading time is not taken into account if the vehicle is preloaded ($k \in VP$). Then, the intermediate duration TD^k_{inter} is determined by browsing each order $i' \in IS_k$ allocated to the vehicle. If the first order requires multiple deliveries ($i' = 1$ and $n_{i'k} > 1$), the duration of additional

Procedure E.3.1 Generation of the best neighbor (Deletion & Insertion) (Part 1)

procedure GENERATEBESTNEIGHBOR($\mathfrak{s}, TA, n_N, n_{TA}, \mathfrak{s}^*$)
  $VS \leftarrow$ set of vehicles used in solution $\mathfrak{s}$
  $IS_k \leftarrow$ set of orders allocated to vehicle k in solution $\mathfrak{s}$
  $x_{ik} \leftarrow 1$ if order i is visited by vehicle k in solution $\mathfrak{s}$, 0 otherwise
  $X \leftarrow \{x_{ik} | i \in I, k \in V \text{ and } x_{ik} = 1\}$
  $\mathfrak{s}' \leftarrow \emptyset$
  $\mathfrak{s}^\circ \leftarrow \mathfrak{s}$
  $TD^k_{first} \leftarrow 0 \quad \forall k \in V$
  $TD^k_{inter} \leftarrow 0 \quad \forall k \in V$
  $TD^k_{last} \leftarrow 0 \quad \forall k \in V$
  for $k \in VS = 1 \rightarrow |VS|$ **do**
    if $k \in VP$ **then**
      $TD^k_{first} \leftarrow \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
    else
      $TD^k_{first} \leftarrow L_{i0k} + \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
    end if
    for $i' \in IS_k = 1 \rightarrow |IS_k|$ **do**
      $\Upsilon \leftarrow 0$
      if $i' = 1$ **and** $n_{i'k} > 1$ **then**
        $\Upsilon \leftarrow (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
      else if $i' > 1$ **then**
        $i \in IS_k \leftarrow$ order preceding i' for vehicle $k \quad (i \neq i')$
        $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_k + L_{i'jk} + \delta_{c_{i'} j} TL_k\} + U_{i'k}$ where $s^k_{ii'} = 1$
        if $n_{i'k} > 1$ **then**
          $\Upsilon \leftarrow \Upsilon + (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
        end if
      end if
      $TD^k_{inter} \leftarrow TD^k_{inter} + \Upsilon$
    end for
    $TD^k_{last} \leftarrow \delta_{c_i 0} TU_k$ where $s^k_{i0} = 1$
  end for

deliveries is taken into account. For the following orders of the vehicle ($i' > 1$), the duration of a first delivery is considered and, if additional deliveries are required for this order ($n_{i'k} > 1$), the duration of these ones are also taken into account. Finally, the duration related to the back travel of the last delivery TD^k_{last} is computed.

E.3.2 Deletion phase

This phase is described in detail in Procedure E.3.2.

Procedure E.3.2 Generation of the best neighbor (Deletion & Insertion) (Part 2)

```
if n_N > 0 then
    n_N ← min{|X|, n_N}
else
    n_N ← |X|
end if
for l = 1 → n_N do
    𝔰 ← 𝔰°
    if n_N ≠ |X| then
        ϵ ← RANDOM(1, |X|)
    else
        ϵ ← l
    end if
    i' ← order related to the ϵth element of set X
    k^del ← vehicle related to the ϵth element of set X
    if not (k^del ∈ VP and s^{k^del}_{0i'} = 1 and n_{i'k^del} = 1 and |IP^{k^del}| = 1) then
        n_{i'k^del} ← n_{i'k^del} − 1
        if n_{i'k^del} = 0 then
            if s^{k^del}_{0i'} = 1 then
                i ← 0
            else
                i ∈ I ← order preceding i' for vehicle k^del    (i ≠ i')
            end if
            if s^{k^del}_{i'0} = 1 then
                i'' ← 0
            else
                i'' ∈ I ← order following i' for vehicle k^del    (i' ≠ i'')
            end if
            s^{k^del}_{ii'} ← 0
            s^{k^del}_{i'i''} ← 0
            if i ≠ i'' then
                s^{k^del}_{ii''} ← 1
            end if
        end if
        𝔰 ← IMPROVESOLUTION(𝔰, k^del)
        Θ ← f(𝔰)
        e_{i'} ← Σ_{k∈V} (n_{i'k} Q_k)
```

The size of a neighborhood n_N or more precisely the number of orders $i \in I$ allocated to vehicles $k \in V$ whose delivery has to be deleted is first defined, although mentioned as parameter. Indeed, the mentioned size n_N can be greater than the maximal size $|X|$ of a neighborhood and is therefore adapted ($\min\{|X|, n_N\}$) if needed. If no size is mentioned ($n_N = 0$), the entire neighborhood of size $|X|$ has to be generated.

For each deletion l, solution $\mathfrak{s}$ is beforehand reinitialized with the content of saved solution $\mathfrak{s}^\circ$. If the size of a neighborhood is not maximal ($n_N \neq |X|$), an element $x_{i'k^{del}}$ of set X is selected randomly. Otherwise, the lth element of this set is selected. If the deletion of a delivery of order i' from the route of vehicle k^{del} is allowed, the number of deliveries $n_{i'k^{del}}$ done by vehicle k^{del} for order i' is updated. Note that the deletion of a delivery is forbidden if this one is the first and unique delivery of an order allocated to a preloaded vehicle and if no other delivery of the vehicle can be set as first delivery ($k^{del} \in VP$ and $s_{0i'}^{k^{del}} = 1$ and $n_{i'k^{del}} = 1$ and $|IP^{k^{del}}| = 1$). If following this deletion there is no more delivery for order i' with vehicle k^{del}, $s_{0i'}^{k^{del}}$, $s_{i'0}^{k^{del}}$ and $s_{ii'}^{k^{del}}$ are also updated to modify the route of vehicle k^{del}.

Before reinserting one or many deliveries of order i' on other vehicles $k \in V \setminus \{k^{del}\}$ in the *insertion phase*, the partial solution related to vehicle k^{del} may be improved, the value of the objective function related to current solution $\mathfrak{s}$ is saved as Θ and the delivered quantity $e_{i'}$ for order i' is adapted.

E.3.3 Insertion phase

This phase is described in detail in Procedure E.3.3 from line 68 to line 85. The deleted delivery of order $i' \in I$ is reinserted on the route of other vehicles.

For each vehicle $k \in V \setminus \{k^{del}\}$ that can handle this order ($i' \in I_k$), the number η of required deliveries is first defined. Indeed, the capacity of vehicle k can differ from the capacity of vehicle k^{del} and more than one delivery may be required to keep order i' entirely satisfied.

Described in detail later in procedure OPTIMIZEINSERTION($\mathfrak{s}, i', k, \eta, TD_{first}^k$, $TD_{inter}^k, TD_{last}^k, 1$), the insertion process first deletes possible unnecessary deliveries of order i' from the route of other vehicles $k' \in V \setminus \{k\}$ due to the future insertion of one or more deliveries of order i' on the route of vehicle k and then inserts these deliveries at the most appropriate position on the route of vehicle k.

If vehicle k has enough availability to insert the required number of deliveries of order i' ($TD_{first}^k + TD_{inter}^k + TD_{last}^k + TD_\theta^k \leq A_k$), if no neighbor has been generated before ($\mathfrak{s}' = \emptyset$) or if the current solution is better than the best solution ($\Theta + \theta \leq f(\mathfrak{s}')$), if the insertion of order i' on the route of vehicle k is not tabu ($(i', k) \notin TA$) or if the current solution is better than the best solution of the tabu search ($\Theta + \theta < f(\mathfrak{s}^*)$), i.e. satisfies the aspiration criterion, current vehicle k^{del}

Procedure E.3.3 Generation of the best neighbor (Deletion & Insertion) (Part 3)

```
68:             for k ∈ V = 1 → |V| do
69:                 if k ≠ k^del and i' ∈ I_k then
70:                     η ← ⌈(d_{i'} − e_{i'}) / Q_k⌉
71:                     S ← OptimizeInsertion(𝔰, i', k, η, TD^k_first, TD^k_inter, TD^k_last, 1)
72:                     {𝔰, θ, TD^k_θ} ← S
73:                     if TD^k_first + TD^k_inter + TD^k_last + TD^k_θ ≤ A_k then
74:                         if 𝔰' = ∅ or Θ + θ ≤ f(𝔰') then
75:                             if (i', k) ∉ TA or Θ + θ < f(𝔰*) then
76:                                 k^del_TA ← k^del
77:                                 i'_TA ← i'
78:                                 𝔰' ← 𝔰
79:                             end if
80:                         end if
81:                     end if
82:                 end if
83:             end for
84:         end if
85:     end for
86:     if |TA| = n_TA then
87:         TA ← TA \ {(i_1, k_1)} where (i_1 ∈ I, k_1 ∈ V) is the first tuple of TA
88:     end if
89:     TA ← TA ∪ {(i'_TA, k^del_TA)}
90:     return {𝔰', TA}
91: end procedure
```

and current order i' are respectively saved as k_{TA}^{del} and i'_{TA}. Furthermore, current neighboring solution $\mathfrak{s}$ is set as best neighbor $\mathfrak{s}'$.

After having generated the neighboring solutions and selected among them the best neighbor, the tabu list TA is updated (cf. lines 86 to 89). Before inserting a new element in this list, the first element is deleted but only if the list has beforehand reached its maximal size n_{TA}. A tuple (i'_{TA}, k_{TA}^{del}) related to vehicle k^{del} and order i' is set as tabu for the insertion. Indeed, the reinsertion of a delivery of order i' on the route of vehicle k^{del} is forbidden for n_{TA} iterations.

The solution $\mathfrak{s}'$ related to the best neighbor is finally returned, as the updated tabu list TA (cf. Procedure E.3.3 at line 90).

E.4 Optimization of the insertion

Eight parameters are needed to run the corresponding procedure: a solution $\mathfrak{s}$, an order $i^* \in I$, a vehicle $k^* \in V$, a number of deliveries to insert η, the total travel duration of vehicle k^* before the insertion split in $TD^{k^*}_{first}$, $TD^{k^*}_{inter}$ and $TD^{k^*}_{last}$, and a boolean CP indicating if the *cleaning phase* is required or not.

E.4.1 Initialization phase

This phase is described in detail in Procedure E.4.1.

Procedure E.4.1 Optimization of the insertion (Part 1)

procedure OPTIMIZEINSERTION($\mathfrak{s}, i^*, k^*, \eta, TD^{k^*}_{first}, TD^{k^*}_{inter}, TD^{k^*}_{last}, CP$)
  $VS \leftarrow$ set of vehicles used in solution $\mathfrak{s}$
  $IS_k \leftarrow$ set of orders allocated to vehicle k in solution $\mathfrak{s}$
  $\varepsilon \leftarrow e_{i^*} + \eta Q_{k^*} - d_{i^*}$ where $e_{i^*} \leftarrow \sum_{k \in V}(n_{i^*k} Q_k)$
  $\mathfrak{s}^\circ \leftarrow \mathfrak{s}$
  $TD^{k^\circ}_{first} \leftarrow TD^{k^*}_{first}$
  $TD^{k^\circ}_{inter} \leftarrow TD^{k^*}_{inter}$
  $TD^{k^\circ}_{last} \leftarrow TD^{k^*}_{last}$
  $\theta \leftarrow 0$

The set VS of vehicles used, the set IS_k of orders allocated to vehicle $k \in V$ and the delivered surplus ε following the insertion of additional deliveries for order i^* on the route of vehicle k^* are first defined as follows:

$$VS = \{k \in V | \sum_{i \in I} n_{ik} \geq 1\}$$

$$IS_k = \{i \in I | n_{ik} \geq 1 \quad \forall k \in V\}$$

$$\varepsilon = e_{i^*} + \eta Q_{k^*} - d_{i^*}$$

where

$$e_{i^*} = \sum_{k \in V} (n_{i^*k} Q_k).$$

The current solution $\mathfrak{s}$ and the current total travel duration $TD^{k^*}_{first}$, $TD^{k^*}_{inter}$ and $TD^{k^*}_{last}$ of vehicle k^* are respectively saved in $\mathfrak{s}^\circ$, $TD^{k^\circ}_{first}$, $TD^{k^\circ}_{inter}$ and $TD^{k^\circ}_{last}$. Finally, the variation of the objective function θ following the insertion of additional deliveries of order i^* on vehicle k^* is set to 0.

E.4.2 Cleaning phase

This phase is described in detail in Procedures E.4.2 and E.4.3.

Procedure E.4.2 Optimization of the insertion (Part 2)

```
if CP = 1 then
    repeat
        VE ← ∅
        for k ∈ VS = 1 → |VS| do
            if x_{i*k} = 1 and ε > Q_k then
                if not (k ∈ VP and s^k_{0i*} = 1 and n_{i*k} = 1 and |IP^k| = 1) then
                    VE ← VE ∪ {k}
                end if
            end if
        end for
        DEL ← ∅
        if VE ≠ ∅ then
            for k ∈ VE = 1 → |VE| do
                𝔰 ← 𝔰°
                n_{i*k} ← n_{i*k} − 1
                if n_{i*k} = 0 then
                    if k ∈ VP and s^k_{0i*} = 1 and |IP^k| > 1 then
                        i'_∘ ∈ IP^k ← order usable for preloaded vehicle k   (i'_∘ ≠ i*)
                        i_∘ ∈ I ← order preceding i'_∘ for vehicle k   (i_∘ ≠ i'_∘)
                        if s^k_{i'_∘0} = 1 then
                            s^k_{i'_∘0} ← 0
                            s^k_{i_∘0} ← 1
                        else
                            i''_∘ ∈ I ← order following i'_∘ for vehicle k   (i'_∘ ≠ i''_∘)
                            s^k_{i_∘ i'_∘} ← 0
                            s^k_{i'_∘ i''_∘} ← 0
                            s^k_{i_∘ i''_∘} ← 1
                        end if
```

If required ($CP = 1$), unnecessary deliveries of order i^* are deleted from the route of vehicles $k \in V \setminus \{k^*\}$ due to the future insertion of one or many deliveries of this order on the route of vehicle k^*.

For this purpose, a set VE of vehicles $k \in VS$ whose one or many deliveries of order i^* can be deleted from their route is first created. Note that only vehicles where the surplus of delivered quantity is greater than their capacity ($\varepsilon > Q_k$) are considered. Furthermore, the deletion of a delivery is forbidden if this one is the

Procedure E.4.3 Optimization of the insertion (Part 3)

```
                    i'' ∈ I ← order following i* for vehicle k   (i* ≠ i'')
                    s^k_{i* i''} ← 0
                    s^k_{0 i*} ← 0
                    s^k_{i'_∘ i''} ← 1
                    s^k_{0 i'_∘} ← 1
                else
                    if s^k_{0 i*} = 1 then
                        i ← 0
                    else
                        i ∈ I ← order preceding i* for vehicle k   (i ≠ i*)
                    end if
                    if s^k_{i* 0} = 1 then
                        i'' ← 0
                    else
                        i'' ∈ I ← order following i* for vehicle k   (i* ≠ i'')
                    end if
                    s^k_{i i*} ← 0
                    s^k_{i* i''} ← 0
                    if i ≠ i'' then
                        s^k_{i i''} ← 1
                    end if
                end if
                𝔰_k ← 𝔰
                θ_k ← f(𝔰_k) − f(𝔰°)
                DEL ← DEL ∪ {(k, 𝔰_k, θ_k)}
            end if
        end for
    end if
    if DEL ≠ ∅ then
        k_μ ∈ VE where θ_{k_μ} = min_{k∈VE}{θ_k ∈ (k, 𝔰_k, θ_k) ∈ DEL}
        θ ← θ + θ_{k_μ}
        𝔰° ← 𝔰 ← 𝔰_{k_μ}
        ε ← ε − Q_{k_μ}
    end if
  until VE = ∅ or DEL = ∅
end if
```

first and unique delivery of an order allocated to a preloaded vehicle and if no other delivery of the vehicle can be set as first delivery ($k \in VP$ and $s^k_{0i^*} = 1$ and $n_{i^*k} = 1$ and $|IP^k| = 1$).

For each vehicle $k \in VE$, solution $\mathfrak{s}$ is first reinitialized with saved solution $\mathfrak{s}^\circ$ and the number of deliveries n_{i^*k} done by vehicle k for order i^* is updated.

If the deletion concerns the first delivery of a preloaded vehicle and if another delivery $i'_\circ$ of this vehicle can be set as preloaded delivery ($k \in VP$ and $s^k_{0i^*} = 1$ and $|IP^k| > 1$), $s^k_{i'_\circ 0}$ and $s^k_{i_\circ 0}$ or $s^k_{i_\circ i'_\circ}$, $s^k_{i'_\circ i''_\circ}$ and $s^k_{i_\circ i''_\circ}$, depending if order $i'_\circ$ is the last order on the route of vehicle k, are first updated to modify the route of vehicle k, as $s^k_{i^* i''}$, $s^k_{0i^*}$, $s^k_{i'_\circ i''}$ and $s^k_{0i'_\circ}$. Note that $i_\circ$ and $i''_\circ$ respectively corresponds to the order that precedes and follows order $i'_\circ$ on the route of vehicle k and that i'' corresponds to the order that follows order i^* on the route of vehicle k.

If the deleted delivery does not concern a preloaded delivery, $s^k_{ii^*}$, $s^k_{i^* i''}$ and eventually $s^k_{ii''}$ are updated to modify the route of vehicle k. Then, the modified solution $\mathfrak{s}$ and the variation of the objective function between the modified solution and the initial solution ($f(\mathfrak{s}_k) - f(\mathfrak{s}^\circ)$) are saved as a tuple $(k, \mathfrak{s}_k, \theta_k)$ in a set DEL intended to contain the possible deletions of deliveries related to the delivered surplus ε.

After having explored all the deletion possibilities, vehicle $k_\mu \in VE$ whose variation of the objective function compared to the initial solution is the smallest one is selected. The variation of the objective function θ is then updated, as the solutions $\mathfrak{s}$ and $\mathfrak{s}^\circ$, and the delivered surplus ε. This process is repeated until there is no more unnecessary delivery of order i^* to delete for vehicles $k \in VE$.

E.4.3 Insertion phase

This phase is described in detail in Procedures E.4.4, E.4.5 and in Procedure E.4.6 from line 119 to line 144. One or more deliveries of order i^* are reinserted at the most appropriate position on the route of vehicle k^*.

If vehicle k^* has no delivery ($\sum_{i \in I} x_{ik^*} = 0$), the number of deliveries $n_{i^*k^*}$ done by vehicle k^* for order i^* is updated, as $s^{k^*}_{0i^*}$ and $s^{k^*}_{i^*0}$ to modify the route of vehicle k^*. Furthermore, the initial values of $TD^{k^*}_{last}$, $TD^{k^*}_{inter}$ and $TD^{k^*}_{last}$ are replaced to take into account the total travel duration of vehicle k^*.

If vehicle k^* does at least a delivery for order i^* ($x_{i^*k^*} = 1$), the number of deliveries $n_{i^*k^*}$ done by vehicle k^* for order i^* is updated, as $TD^{k^*}_{inter}$ to take into account the duration of additional deliveries of order i^*.

If vehicle k^* can only be loaded at the main depot ($k^* \notin V^L$), the order i^* is positioned at the end of the route of vehicle k^*, the number of deliveries $n_{i^*k^*}$

Procedure E.4.4 Optimization of the insertion (Part 4)

if $\sum_{i \in I} x_{ik^*} = 0$ **then**
  $s_{0i^*}^{k^*} \leftarrow 1$
  $s_{i^*0}^{k^*} \leftarrow 1$
  $n_{i^*k^*} \leftarrow \eta$
  $TD_{first}^{k^*} \leftarrow L_{i^*0k^*} + \delta_{c_{i^*}0} TL_{k^*} + U_{i^*k^*}$
  $TD_{last}^{k^*} \leftarrow \delta_{c_{i^*}0} TU_{k^*}$
  $TD_{inter}^{k^*} \leftarrow (\eta - 1)(L_{i^*\Delta_{i^*k^*}k^*} + U_{i^*k^*} + \delta_{c_{i^*}\Delta_{i^*k^*}}(TU_{k^*} + TL_{k^*}))$
else if $x_{i^*k^*} = 1$ **then**
  $n_{i^*k^*} \leftarrow n_{i^*k^*} + \eta$
  $TD_{inter}^{k^*} \leftarrow TD_{inter}^{k^*} + \eta(L_{i^*\Delta_{i^*k^*}k^*} + U_{i^*k^*} + \delta_{c_{i^*}\Delta_{i^*k^*}}(TU_{k^*} + TL_{k^*}))$
else if $k^* \notin V^L$ **then**
  $i \in I_{k^*} \leftarrow$ last order allocated to vehicle k^* $\quad (i \neq i^*)$
  $s_{i0}^{k^*} \leftarrow 0$
  $s_{ii^*}^{k^*} \leftarrow 1$
  $s_{i^*0}^{k^*} \leftarrow 1$
  $n_{i^*k^*} \leftarrow \eta$
  $TD_{last}^{k^*} \leftarrow \delta_{c_{i^*}0} TU_{k^*}$
  $TD_{inter}^{k^*} \leftarrow TD_{inter}^{k^*} + \min_{j \in D_{i^*}} \{\delta_{c_{ij}} TU_{k^*} + L_{i^*jk^*} + \delta_{c_{i^*}j} TL_{k^*}\} + U_{i^*k^*}$
  $TD_{inter}^{k^*} \leftarrow TD_{inter}^{k^*} + (\eta - 1)(L_{i^*\Delta_{i^*k^*}k^*} + U_{i^*k^*} + \delta_{c_{i^*}\Delta_{i^*k^*}}(TU_{k^*} + TL_{k^*}))$
else
  $INS \leftarrow \emptyset$
  $\mathfrak{s}^\circ \leftarrow \mathfrak{s}$

done by vehicle k^* for order i^* is updated, as $s_{i0}^{k^*}$, $s_{ii^*}^{k^*}$ and $s_{i^*0}^{k^*}$ to modify the route of vehicle k^*. Furthermore, the initial value of $TD_{last}^{k^*}$ is replaced and $TD_{inter}^{k^*}$ is updated to take into account the duration of the deliveries of order i^*.

Otherwise, i.e. if there is initially no delivery of order i^* on the route of vehicle k and if this vehicle can be loaded at local depots, the insertion of deliveries of order i^* has first to be tested at every position on the route of vehicle k^* to find the most appropriate position. Note that current solution $\mathfrak{s}$ is initially saved as solution $\mathfrak{s}^\circ$.

For each order $i \in IS_{k^*}$ in the initial route of vehicle k^*, the solution $\mathfrak{s}$ is beforehand reinitialized with solution $\mathfrak{s}^\circ$. The cement type of the order to insert ct_{i^*} and of the current order ct_i are also defined.

If order i is the first order of vehicle k^* $(i = 1)$, if vehicle k^* is not a preloaded vehicle or if vehicle k is preloaded with a same cement type as the one of delivery i^* $((k^* \in V \setminus VP)$ or $(k^* \in VP$ and $ct_{i^*} = ct_i))$, $s_{0i}^{k^*}$, $s_{0i^*}^{k^*}$ and $s_{i^*i}^{k^*}$ are updated to

Procedure E.4.5 Optimization of the insertion (Part 5)

```
for i ∈ IS_{k*} = 1 → |IS_{k*}| do
    𝔰 ← 𝔰°
    ct_{i*} ← the cement type of order i*
    ct_i ← the cement type of order i
    if i = 1 and ((k* ∈ V \ VP) or (k* ∈ VP and ct_{i*} = ct_i)) then
        s^{k*}_{0i} ← 0
        s^{k*}_{0i*} ← 1
        s^{k*}_{i*i} ← 1
    else if i = |IS_{k*}| then
        s^{k*}_{i0} ← 0
        s^{k*}_{ii*} ← 1
        s^{k*}_{i*0} ← 1
    else
        i' ∈ I ← order following i for vehicle k*    (i ≠ i')
        s^{k*}_{ii'} ← 0
        s^{k*}_{ii*} ← 1
        s^{k*}_{i*i'} ← 1
    end if
    n_{i*k*} ← η
    𝔰_i ← 𝔰
    θ_i ← f(𝔰_i) − f(𝔰°)
    INS ← INS ∪ {(i, 𝔰_i, θ_i)}
end for
```

modify the route of vehicle k^*. If order i is the last order of vehicle k^* ($i = |IS_{k^*}|$), $s^{k^*}_{i0}$, $s^{k^*}_{ii^*}$ and $s^{k^*}_{i^*0}$ are updated to modify the route of vehicle k^*. Otherwise, $s^{k^*}_{ii'}$, $s^{k^*}_{ii^*}$ and $s^{k^*}_{i^*i'}$ are updated to modify the route of vehicle k^*. Note that i' corresponds to the order that follows i on the route of vehicle k^*.

After having inserted order i^* on the route of vehicle k^*, the number of deliveries $n_{i^*k^*}$ done by vehicle k^* for order i^* is updated, the modified solution $\mathfrak{s}$ and the variation of the objective function between the modified solution and the initial solution ($f(\mathfrak{s}_i) - f(\mathfrak{s}^\circ)$) are saved as a tuple $(i, \mathfrak{s}_i, \theta_i)$ in a set INS intended to contain the possible insertions of deliveries on the route of vehicle k^*.

After having explored all the possible insertions of order $i \in IS_{k^*}$ on the route of vehicle $k^* \in V^L$ ($INS \neq \emptyset$), the order $i_\mu \in IS_{k^*}$ related to the most appropriate insertion is selected and the corresponding solution $\mathfrak{s}_{i_\mu}$ is saved as current solution $\mathfrak{s}$. If order i^* has been inserted at the first position on the route of vehicle k^*, the value of $TD^{k^*}_{first}$ is replaced and $TD^{k^*}_{inter}$ is updated. If order i^* has been

Procedure E.4.6 Optimization of the insertion (Part 6)

119: **if** $INS \neq \emptyset$ **then**
120: $i_\mu \in IS_{k^*}$ where $\theta_{i_\mu} = \min_{i \in IS_{k^*}} \{\theta_i \in (i, \mathfrak{s}_i, \theta_i) \in INS\}$
121: $\mathfrak{s} \leftarrow \mathfrak{s}_{i_\mu}$
122: **if** $s^{k^*}_{0i^*} = 1$ **then**
123: $TD^{k^*}_{first} \leftarrow \delta_{c_{i^*}0} TL_{k^*} + U_{i^*k^*}$
124: **if** $k^* \notin VP$ **then**
125: $TD^{k^*}_{first} \leftarrow TD^{k^*}_{first} + L_{i^*0k^*}$
126: **end if**
127: $\Upsilon \leftarrow (\eta - 1)(L_{i^*\Delta_{i^*k^*}k^*} + U_{i^*k^*} + \delta_{c_{i^*}\Delta_{i^*k^*}}(TU_{k^*} + TL_{k^*}))$
128: $\Upsilon \leftarrow \Upsilon + \min_{j \in D_{i_\mu}} \{\delta_{c_{i^*}j} TU_{k^*} + L_{i_\mu jk^*} + \delta_{c_{i_\mu}j} TL_{k^*}\} + U_{i_\mu k^*}$
129: $TD^{k^*}_{inter} \leftarrow TD^{k^*}_{inter} + \Upsilon$
130: **else if** $s^{k^*}_{i^*0} = 1$ **then**
131: $\Upsilon \leftarrow \min_{j \in D_{i^*}} \{\delta_{c_{i_\mu}j} TU_{k^*} + L_{i^*jk^*} + \delta_{c_{i^*}j} TL_{k^*}\} + U_{i^*k^*}$
132: $\Upsilon \leftarrow \Upsilon + (\eta - 1)(L_{i^*\Delta_{i^*k^*}k^*} + U_{i^*k^*} + \delta_{c_{i^*}\Delta_{i^*k^*}}(TU_{k^*} + TL_{k^*}))$
133: $TD^{k^*}_{inter} \leftarrow TD^{k^*}_{inter} + \Upsilon$
134: $TD^{k^*}_{last} \leftarrow \delta_{c_{i^*}0} TU_{k^*}$
135: **else if** $s^{k^*}_{i_\mu i^*} = 1$ **then**
136: $i' \in I \leftarrow$ order following i^* for vehicle k^* $\quad (i^* \neq i')$
137: $\Upsilon \leftarrow \min_{j \in D_{i^*}} \{\delta_{c_{i_\mu}j} TU_{k^*} + L_{i^*jk^*} + \delta_{c_{i^*}j} TL_{k^*}\} + U_{i^*k^*}$
138: $\Upsilon \leftarrow \Upsilon + (\eta - 1)(L_{i^*\Delta_{i^*k^*}k^*} + U_{i^*k^*} + \delta_{c_{i^*}\Delta_{i^*k^*}}(TU_{k^*} + TL_{k^*}))$
139: $\Upsilon \leftarrow \Upsilon + \min_{j \in D_{i'}} \{\delta_{c_{i^*}j} TU_{k^*} + L_{i'jk^*} + \delta_{c_{i'}j} TL_{k^*}\}$
140: $\Upsilon \leftarrow \Upsilon - \min_{j \in D_{i'}} \{\delta_{c_{i_\mu}j} TU_{k^*} + L_{i'jk^*} + \delta_{c_{i'}j} TL_{k^*}\}$
141: $TD^{k^*}_{inter} \leftarrow TD^{k^*}_{inter} + \Upsilon$
142: **end if**
143: **end if**
144: **end if**
145: $\theta \leftarrow \theta + f(\mathfrak{s}) - f(\mathfrak{s}^\circ)$
146: $TD^k_\theta \leftarrow (TD^{k^*}_{first} + TD^{k^*}_{inter} + TD^{k^*}_{last}) - (TD^{k^\circ}_{first} + TD^{k^\circ}_{inter} + TD^{k^\circ}_{last})$
147: **return** $\{\mathfrak{s}, \theta, TD^k_\theta\}$
148: **end procedure**

inserted at the last position on the route of vehicle k^*, $TD^{k^*}_{inter}$ is updated and the value of $TD^{k^*}_{last}$ is replaced. Otherwise, if order i^* has been inserted after order i_μ on the route of vehicle k^*, $TD^{k^*}_{inter}$ is updated.

Following the insertion of the required number of deliveries of order i^* on the route of vehicle k^*, the variation of the objective function $(f(\mathfrak{s}) - f(\mathfrak{s}^\circ))$ due to this insertion is added to θ and the variation of the total travel duration of vehicle k^* following this insertion is saved in TD^k_θ (cf. Procedure E.4.6 from line 145 to line

146). The solution $\mathfrak{s}$ related to the most appropriate insertion is finally returned, as the variation of the objective function θ and of the total travel duration TD_{θ}^{k} (cf. Procedure E.4.6 at line 147).

E.5 Generation of the best neighbor (Delivery swap)

Five parameters are needed to run the corresponding procedure: a solution $\mathfrak{s}$, the tabu list TA presented before, the size of a neighborhood n_N, the size of the tabu list n_{TA} and the current best solution $\mathfrak{s}^*$ of the tabu search.

E.5.1 Initialization phase

This phase is described in detail in Procedure E.5.1.

The set VS of vehicles used, the set IS_k of orders allocated to vehicle $k \in V$, x_{ik} indicating if order $i \in I$ is visited by vehicle $k \in V$ and the set X containing only x_{ik} whose order $i \in I$ is visited by vehicle $k \in V$ in given solution $\mathfrak{s}$ are first defined as follows:

$$VS = \{k \in V | \sum_{i \in I} n_{ik} \geq 1\}$$

$$IS_k = \{i \in I | n_{ik} \geq 1 \quad \forall k \in V\}$$

$$x_{ik} = \begin{cases} 1 & \text{if } n_{ik} \geq 1 \quad \forall i \in I, \forall k \in V \\ 0 & \text{otherwise} \end{cases}$$

$$X = \{x_{ik} | i \in I, k \in V \text{ and } x_{ik} = 1\}.$$

The set W intended to contain orders and vehicles implied in each neighboring solution is then initialized as an empty set $\emptyset$, as the solution $\mathfrak{s}'$ related to the best neighbor. The current solution $\mathfrak{s}$ is saved in $\mathfrak{s}^\circ$.

Finally, TD_{first}^{k}, TD_{inter}^{k} and TD_{last}^{k} presented before are initialized to 0 and updated (cf. lines 12 to 32) for each vehicle $k \in VS$. Indeed, the duration related to the loading time, the travel time and the unloading time of the first delivery TD_{first}^{k} is first computed. Note that the loading time is not taken into account if the vehicle is preloaded ($k \in VP$). Then, the intermediate duration TD_{inter}^{k} is determined by browsing each order $i' \in IS_k$ allocated to the vehicle. If the first order requires multiple deliveries ($i' = 1$ and $n_{i'k} > 1$), the duration of additional

Procedure E.5.1 Generation of the best neighbor (Delivery swap) (Part 1)

procedure GENERATEBESTNEIGHBOR($\mathfrak{s}, TA, n_N, n_{TA}, \mathfrak{s}^*$)
  $VS \leftarrow$ set of vehicles used in solution $\mathfrak{s}$
  $IS_k \leftarrow$ set of orders allocated to vehicle k in solution $\mathfrak{s}$
  $x_{ik} \leftarrow 1$ if order i is visited by vehicle k in solution $\mathfrak{s}$, 0 otherwise
  $X \leftarrow \{x_{ik} | i \in I, k \in V \text{ and } x_{ik} = 1\}$
  $W \leftarrow \emptyset$
  $\mathfrak{s}' \leftarrow \emptyset$
  $\mathfrak{s}^\circ \leftarrow \mathfrak{s}$
  $TD^k_{first} \leftarrow 0 \quad \forall k \in V$
  $TD^k_{inter} \leftarrow 0 \quad \forall k \in V$
  $TD^k_{last} \leftarrow 0 \quad \forall k \in V$
  for $k \in VS = 1 \rightarrow |VS|$ **do**
    if $k \in VP$ **then**
      $TD^k_{first} \leftarrow \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
    else
      $TD^k_{first} \leftarrow L_{i0k} + \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
    end if
    for $i' \in IS_k = 1 \rightarrow |IS_k|$ **do**
      $\Upsilon \leftarrow 0$
      if $i' = 1$ **and** $n_{i'k} > 1$ **then**
        $\Upsilon \leftarrow (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
      else if $i' > 1$ **then**
        $i \in IS_k \leftarrow$ order preceding i' for vehicle $k \quad (i \neq i')$
        $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_k + L_{i'jk} + \delta_{c_{i'} j} TL_k\} + U_{i'k}$ where $s^k_{ii'} = 1$
        if $n_{i'k} > 1$ **then**
          $\Upsilon \leftarrow \Upsilon + (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
        end if
      end if
      $TD^k_{inter} \leftarrow TD^k_{inter} + \Upsilon$
    end for
    $TD^k_{last} \leftarrow \delta_{c_i 0} TU_k$ where $s^k_{i0} = 1$
  end for

deliveries is taken into account. For the following orders of the vehicle ($i' > 1$), the duration of a first delivery is considered and, if additional deliveries are required for this order ($n_{i'k} > 1$), the duration of these ones are also taken into account. Finally, the duration related to the back travel of the last delivery TD^k_{last} is computed.

E.5.2 First deletion phase

This phase is described in detail in Procedures E.5.2 and E.5.3.

Procedure E.5.2 Generation of the best neighbor (Delivery swap) (Part 2)

```
if n_N > 0 then
    n_N ← min{|X|, n_N}
else
    n_N ← |X|
end if
for l = 1 → n_N do
    𝔰 ← 𝔰°
    if n_N ≠ |X| then
        ϵ ← RANDOM(1, |X|)
    else
        ϵ ← l
    end if
    i^a ← order related to the ϵth element of set X
    k^a ← vehicle related to the ϵth element of set X
    if not (k^a ∈ VP and s^{k^a}_{0 i^a} = 1 and n_{i^a k^a} = 1 and |IP^{k^a}| = 1) then
        n_{i^a k^a} ← n_{i^a k^a} − 1
        if n_{i^a k^a} = 0 then
            if s^{k^a}_{0 i^a} = 1 then
                i ← 0
            else
                i ∈ I ← order preceding i^a for vehicle k^a    (i ≠ i^a)
            end if
            if s^{k^a}_{i^a 0} = 1 then
                i'' ← 0
            else
                i'' ∈ I ← order following i^a for vehicle k^a    (i^a ≠ i'')
            end if
            s^{k^a}_{i i^a} ← 0
            s^{k^a}_{i^a i''} ← 0
            if i ≠ i'' then
                s^{k^a}_{i i''} ← 1
            end if
        end if
        𝔰 ← IMPROVESOLUTION(𝔰, k^a)
```

The size of a neighborhood n_N or more precisely the number of orders $i \in I$ allocated to vehicles $k \in V$ whose delivery has to be deleted is first defined, although mentioned as parameter. Indeed, the mentioned size n_N can be greater than the maximal size $|X|$ of a neighborhood and is therefore adapted ($\min\{|X|, n_N\}$) if needed. If no size is mentioned ($n_N = 0$), the entire neighborhood of size $|X|$ has to be generated.

Procedure E.5.3 Generation of the best neighbor (Delivery swap) (Part 3)

$TD^{k^a}_{first} \leftarrow 0$
$TD^{k^a}_{inter} \leftarrow 0$
$TD^{k^a}_{last} \leftarrow 0$
if $k^a \in VP$ **then**
  $TD^{k^a}_{first} \leftarrow \delta_{c_i 0} TL_{k^a} + U_{ik^a}$ where $s^{k^a}_{0i} = 1$
else
  $TD^{k^a}_{first} \leftarrow L_{i0k^a} + \delta_{c_i 0} TL_{k^a} + U_{ik^a}$ where $s^{k^a}_{0i} = 1$
end if
for $i' \in IS_{k^a} = 1 \rightarrow |IS_{k^a}|$ **do**
  $\Upsilon \leftarrow 0$
  if $i' = 1$ **and** $n_{i'k^a} > 1$ **then**
    $\Upsilon \leftarrow (n_{i'k^a} - 1)(L_{i'\Delta_{i'k^a}k^a} + U_{i'k^a} + \delta_{c_{i'}\Delta_{i'k^a}}(TU_{k^a} + TL_{k^a}))$
  else if $i' > 1$ **then**
    $i \in IS_{k^a} \leftarrow$ order preceding i' for vehicle k^a $\quad (i \neq i')$
    $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_{k^a} + L_{i'jk^a} + \delta_{c_{i'} j} TL_{k^a}\} + U_{i'k^a}$ where $s^{k^a}_{ii'} = 1$
    if $n_{i'k^a} > 1$ **then**
      $\Upsilon \leftarrow \Upsilon + (n_{i'k^a} - 1)(L_{i'\Delta_{i'k^a}k^a} + U_{i'k^a} + \delta_{c_{i'}\Delta_{i'k^a}}(TU_{k^a} + TL_{k^a}))$
    end if
  end if
  $TD^{k^a}_{inter} \leftarrow TD^{k^a}_{inter} + \Upsilon$
end for
$TD^{k^a}_{last} \leftarrow \delta_{c_i 0} TU_{k^a}$ where $s^{k^a}_{i0} = 1$
$e_{i^a} \leftarrow \sum_{k \in V}(n_{i^a k} Q_k)$

For each deletion l, solution $\mathfrak{s}$ is beforehand reinitialized with the content of saved solution $\mathfrak{s}^\circ$. If the size of a neighborhood is not maximal ($n_N \neq |X|$), an element $x_{i^a k^a}$ of set X is selected randomly. Otherwise, the lth element of this set is selected. If the deletion of a delivery of order i^a from the route of vehicle k^a is allowed, the number of deliveries $n_{i^a k^a}$ done by vehicle k^a for order i^a is updated. Note that the deletion of a delivery is forbidden if this one is the first and unique delivery of an order allocated to a preloaded vehicle and if no other delivery of the vehicle can be set as first delivery ($k^a \in VP$ and $s^{k^a}_{0i^a} = 1$ and $n_{i^a k^a} = 1$ and $|IP^{k^a}| = 1$). If following this deletion there is no more delivery for order i^a with vehicle k^a, $s^{k^a}_{0i^a}$, $s^{k^a}_{i^a 0}$ and $s^{k^a}_{ii^a}$ are also updated to modify the route of vehicle k^a.

Before deleting a delivery of other orders $i \in I \setminus \{i^a\}$ from the route of other vehicles $k \in V \setminus \{k^a\}$ in the *second deletion phase*, the first partial solution related to vehicle k^a may be improved, $TD^{k^a}_{first}$, $TD^{k^a}_{inter}$ and $TD^{k^a}_{last}$ whose aggregation corresponds to the total travel duration of vehicle k^a are updated, and the delivered quantity e_{i^a} for order i^a is adapted.

E.5.3 Second deletion phase

This phase is described in detail in Procedure E.5.4.

Procedure E.5.4 Generation of the best neighbor (Delivery swap) (Part 4)

```
C^a ← {c|c ∈ C, i ∈ I, x_{ik^a} = 1 and c_i ≠ c_{i^a}}
X' ← {x_{ik}|i ∈ I, k ∈ V, x_{ik} = 1, i ≠ i^a, k ≠ k^a and c_i ∈ C^a}
𝔰^◇ ← 𝔰
for m = 1 → |X'| do
    𝔰 ← 𝔰^◇
    i^b ← order related to the mth element of set X'
    k^b ← vehicle related to the mth element of set X'
    if (k^b, i^a, k^a, i^b) ∉ W then
        W ← W ∪ {(k^a, i^b, k^b, i^a)}
        if not (k^b ∈ VP and s^{k^b}_{0i^b} = 1 and n_{i^b k^b} = 1 and |IP^{k^b}| = 1) then
            n_{i^b k^b} ← n_{i^b k^b} − 1
            if n_{i^b k^b} = 0 then
                if s^{k^b}_{0i^b} = 1 then
                    i ← 0
                else
                    i ∈ I ← order preceding i^b for vehicle k^b    (i ≠ i^b)
                end if
                if s^{k^b}_{i^b 0} = 1 then
                    i'' ← 0
                else
                    i'' ∈ I ← order following i^b for vehicle k^b    (i^b ≠ i'')
                end if
                s^{k^b}_{ii^b} ← 0
                s^{k^b}_{i^b i''} ← 0
                if i ≠ i'' then
                    s^{k^b}_{ii''} ← 1
                end if
            end if
            𝔰 ← ImproveSolution(𝔰, k^b)
            Θ ← f(𝔰)
            e_{i^b} ← Σ_{k∈V} (n_{i^b k} Q_k)
```

The number of other orders $i \in I \setminus \{i^a\}$ allocated to other vehicles $k \in V \setminus \{k^a\}$ whose delivery has to be deleted is first defined. This number corresponds to the size of set $|X'|$ which only considers vehicles $k \neq k^a$ and orders $i \neq i^a$ whose customer is visited by vehicle k^a ($c_i \in C^a$). The current solution $\mathfrak{s}$ is saved as $\mathfrak{s}^\diamond$.

For each deletion m, solution $\mathfrak{s}$ is beforehand reinitialized with the content of saved solution $\mathfrak{s}^{\diamond}$. If the deletion of a delivery of order i^b from the route of vehicle k^b is allowed, the number of deliveries $n_{i^b k^b}$ done by vehicle k^b for order i^b is updated. Only partial neighboring solutions that do not have already been created before ($(k^b, i^a, k^a, i^b) \notin W$) may be generated. To take this into consideration, orders and vehicles related to each eventual partial neighboring solution are saved as a tuple (k^a, i^b, k^b, i^a) in set W. Note that the deletion of a delivery is forbidden if this one is the first and unique delivery of an order allocated to a preloaded vehicle and if no other delivery of the vehicle can be set as first delivery ($k^b \in VP$ and $s^{k^b}_{0i^b} = 1$ and $n_{i^b k^b} = 1$ and $|IP^{k^b}| = 1$). If following this deletion there is no more delivery for order i^b with vehicle k^b, $s^{k^b}_{0i^b}$, $s^{k^b}_{i^b 0}$ and $s^{k^b}_{ii^b}$ are also updated to modify the route of vehicle k^b.

Before reinserting one or more deliveries of order i^b on the route of vehicle k^a and one or more deliveries of order i^a on the route of vehicle k^b in the *swap phase*, the second partial solution related to vehicle k^b may be improved, the value of the objective function related to current solution $\mathfrak{s}$ is saved as Θ and the delivered quantity e_{i^b} for order i^b is adapted.

E.5.4 Swap phase

This phase is described in detail in Procedure E.5.5 and in Procedure E.5.6 from line 153 to line 171.

If vehicle k^a can handle order i^b ($i^b \in I_{k^a}$) and if vehicle k^b can handle order i^a ($i^a \in I_{k^b}$), the deleted delivery of order i^b is first reinserted on the route of vehicle k^a and the deleted delivery of order i^a is then reinserted on the route of vehicle k^b after having updated for each vehicle $k \in VS$ its total travel duration, i.e. TD^k_{first}, TD^k_{inter} and TD^k_{last}.

For vehicles k^a and k^b, the number η^a and η^b of required deliveries is first defined. Indeed, the capacity of vehicle k^a can differ from the capacity of vehicle k^b and more than one delivery may be required to keep orders i^a and i^b entirely satisfied.

Described in detail before in procedure OPTIMIZEINSERTION($\mathfrak{s}, i^b, k^a, \eta^b, TD^{k^a}_{first}, TD^{k^a}_{inter}, TD^{k^a}_{last}, 1$), the insertion process related to vehicle k^a and order i^b first deletes possible unnecessary deliveries of order i^b from the route of other vehicles $k \in V \setminus \{k^a\}$ due to the future insertion of one or more deliveries of order i^b

Procedure E.5.5 Generation of the best neighbor (Delivery swap) (Part 5)

if $i^b \in I_{k^a}$ **and** $i^a \in I_{k^b}$ **then**
  $\eta^b \leftarrow \lceil \frac{d_{i^b} - e_{i^b}}{Q_{k^a}} \rceil$
  $S^a \leftarrow$ OPTIMIZEINSERTION$(\mathfrak{s}, i^b, k^a, \eta^b, TD^{k^a}_{first}, TD^{k^a}_{inter}, TD^{k^a}_{last}, 1)$
  $\{\mathfrak{s}, \theta^a, TD^{k^a}_{\theta^a}\} \leftarrow S^a$
  for $k \in VS = 1 \rightarrow |VS|$ **do**
    $TD^k_{first} \leftarrow 0$
    $TD^k_{inter} \leftarrow 0$
    $TD^k_{last} \leftarrow 0$
    if $k \in VP$ **then**
      $TD^k_{first} \leftarrow \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
    else
      $TD^k_{first} \leftarrow L_{i0k} + \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
    end if
    for $i' \in IS_k = 1 \rightarrow |IS_k|$ **do**
      $\Upsilon \leftarrow 0$
      if $i' = 1$ **and** $n_{i'k} > 1$ **then**
        $\Upsilon \leftarrow (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
      else if $i' > 1$ **then**
        $i \in IS_k \leftarrow$ order preceding i' for vehicle k $\quad (i \neq i')$
        $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_k + L_{i'jk} + \delta_{c_{i'} j} TL_k\}$ where $s^k_{ii'} = 1$
        $\Upsilon \leftarrow \Upsilon + U_{i'k}$ where $s^k_{ii'} = 1$
        if $n_{i'k} > 1$ **then**
          $\Upsilon \leftarrow \Upsilon + (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
        end if
      end if
      $TD^k_{inter} \leftarrow TD^k_{inter} + \Upsilon$
    end for
    $TD^k_{last} \leftarrow \delta_{c_i 0} TU_k$ where $s^k_{i0} = 1$
  end for
  $\eta^a \leftarrow \lceil \frac{d_{i^a} - e_{i^a}}{Q_{k^b}} \rceil$
  $S^b \leftarrow$ OPTIMIZEINSERTION$(\mathfrak{s}, i^a, k^b, \eta^a, TD^{k^b}_{first}, TD^{k^b}_{inter}, TD^{k^b}_{last}, 1)$
  $\{\mathfrak{s}, \theta^b, TD^{k^b}_{\theta^b}\} \leftarrow S^b$

on the route of vehicle k^a and then inserts these deliveries at the most appropriate position on the route of vehicle k^a. Described in detail before in procedure OPTIMIZEINSERTION$(\mathfrak{s}, i^a, k^b, \eta^a, TD^{k^b}_{first}, TD^{k^b}_{inter}, TD^{k^b}_{last}, 1)$, the insertion process related to vehicle k^b and order i^a first deletes possible unnecessary deliveries of order i^a from the route of other vehicles $k \in V \setminus \{k^b\}$ due to the future insertion of

Procedure E.5.6 Generation of the best neighbor (Delivery swap) (Part 6)

```
                    if TD^{k^a}_{first} + TD^{k^a}_{inter} + TD^{k^a}_{last} + TD^{k^a}_{θ^a} ≤ A_{k^a} and
                      TD^{k^b}_{first} + TD^{k^b}_{inter} + TD^{k^b}_{last} + TD^{k^b}_{θ^b} ≤ A_{k^b} then
                      if 𝔰′ = ∅ or Θ + θ^a + θ^b ≤ f(𝔰′) then
                        if ((i^a, k^b) ∉ TA and (i^b, k^a) ∉ TA) or
                                                Θ + θ^a + θ^b ≤ f(𝔰*) then
                          k^a_{TA} ← k^a
                          k^b_{TA} ← k^b
                          i^a_{TA} ← i^a
                          i^b_{TA} ← i^b
                          𝔰′ ← 𝔰
                        end if
                      end if
                    end if
                  end if
                end if
              end for
            end if
        end for
        if |TA| = n_{TA} then
            TA ← TA \ {(i_1, k_1)} where (i_1 ∈ I, k_1 ∈ V) is the first tuple of TA
        end if
        if |TA| = (n_{TA} − 1) then
            TA ← TA \ {(i_1, k_1)} where (i_1 ∈ I, k_1 ∈ V) is the first tuple of TA
        end if
        TA ← TA ∪ {(i^a_{TA}, k^a_{TA}), (i^b_{TA}, k^b_{TA})}
        return {𝔰′, TA}
    end procedure
```

one or more deliveries of order i^a on the route of vehicle k^b and then inserts these deliveries at the most appropriate position on the route of vehicle k^b.

If vehicles k^a and k^b have enough availability to insert the required number of deliveries of orders i^b and i^a ($TD^{k^a}_{first} + TD^{k^a}_{inter} + TD^{k^a}_{last} + TD^{k^a}_{\theta^a} \leq A_{k^a}$) and ($TD^{k^b}_{first} + TD^{k^b}_{inter} + TD^{k^b}_{last} + TD^{k^b}_{\theta^b} \leq A_{k^b}$), if no neighbor has been generated before ($\mathfrak{s}' = \emptyset$) or if the current solution is better than the best solution ($\Theta + \theta^a + \theta^b \leq f(\mathfrak{s}')$), if the insertion of order i^a on the route of vehicle k^b and the insertion of order i^b on the route of vehicle k^a are not tabu ($(i^a, k^b) \notin TA$, $(i^b, k^a) \notin TA$) or if the current solution is better than the best solution of the tabu search ($\Theta + \theta^a + \theta^b \leq f(\mathfrak{s}^*)$), i.e. satisfies the aspiration criterion, current vehicles k^a and k^b, and current orders i^a and i^b are respectively saved as k^a_{TA}, k^b_{TA}, i^a_{TA}

and i^b_{TA}. Furthermore, current neighboring solution $\mathfrak{s}$ is set as best neighbor $\mathfrak{s}'$.

After having generated the neighboring solutions and selected among them the best neighbor, the tabu list TA is updated (cf. lines 172 to 178). Before inserting a new element in this list, the first and eventually the second element are deleted but only if the list has beforehand reached its maximal size n_{TA}. 2 tuples (i^a_{TA}, k^a_{TA}) and (i^b_{TA}, k^b_{TA}) related to vehicles k^a and k^b and orders i^a and i^b are set as tabu for the insertion. Indeed, the reinsertion of a delivery of order i^a on the route of vehicle k^a and the reinsertion of a delivery of order i^b on the route of vehicle k^b are forbidden for n_{TA} iterations.

The solution $\mathfrak{s}'$ related to the best neighbor is finally returned, as the updated tabu list TA (cf. Procedure E.5.6 at line 179).

E.6 Generation of the best neighbor (Order swap)

Five parameters are needed to run the corresponding procedure: a solution $\mathfrak{s}$, the tabu list TA presented before, the size of a neighborhood n_N, the size of the tabu list n_{TA} and the current best solution $\mathfrak{s}^*$ of the tabu search.

E.6.1 Initialization phase

This phase is described in detail in Procedure E.6.1.

The set VS of vehicles used, the set IS_k of orders allocated to vehicle $k \in V$, x_{ik} indicating if order $i \in I$ is visited by vehicle $k \in V$ and the set X containing only x_{ik} whose vehicles $k \in V$ have the same capacity as another vehicle $k' \in VS$ and that do not concern the first order s^k_{0i} of a preloaded vehicle $k \in VP$ in given solution $\mathfrak{s}$ are first defined as follows:

$$VS = \{k \in V | \sum_{i \in I} n_{ik} \geq 1\}$$

$$IS_k = \{i \in I | n_{ik} \geq 1 \quad \forall k \in V\}$$

$$x_{ik} = \begin{cases} 1 & \text{if } n_{ik} \geq 1 \quad \forall i \in I, \forall k \in V \\ 0 & \text{otherwise} \end{cases}$$

$$X = \{x_{ik} | i \in I, k \in V, x_{ik} = 1, (\exists k' \in VS | Q_{k'} = Q_k \text{ and } k' \neq k) \text{ and not } (k \in VP \text{ and } s^k_{0i} = 1)\}.$$

Procedure E.6.1 Generation of the best neighbor (Order swap) (Part 1)

1: **procedure** GENERATEBESTNEIGHBOR($\mathfrak{s}, TA, n_N, n_{TA}, \mathfrak{s}^*$)
2: $VS \leftarrow$ set of vehicles used in solution $\mathfrak{s}$
3: $IS_k \leftarrow$ set of orders allocated to vehicle k in solution $\mathfrak{s}$
4: $x_{ik} \leftarrow 1$ if order i is visited by vehicle k in solution $\mathfrak{s}$, 0 otherwise
5: $X = \{x_{ik} | i \in I, k \in V, x_{ik} = 1, (\exists k' \in VS | Q_{k'} = Q_k \text{ and } k' \neq k) \text{ and not } (k \in VP \text{ and } s^k_{0i} = 1)\}$
6: $W \leftarrow \emptyset$
7: $\mathfrak{s}' \leftarrow \emptyset$
8: $\mathfrak{s}^\circ \leftarrow \mathfrak{s}$
9: $TD^k_{first} \leftarrow 0 \quad \forall k \in V$
10: $TD^k_{inter} \leftarrow 0 \quad \forall k \in V$
11: $TD^k_{last} \leftarrow 0 \quad \forall k \in V$
12: **for** $k \in VS = 1 \rightarrow |VS|$ **do**
13: **if** $k \in VP$ **then**
14: $TD^k_{first} \leftarrow \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
15: **else**
16: $TD^k_{first} \leftarrow L_{i0k} + \delta_{c_i 0} TL_k + U_{ik}$ where $s^k_{0i} = 1$
17: **end if**
18: **for** $i' \in IS_k = 1 \rightarrow |IS_k|$ **do**
19: $\Upsilon \leftarrow 0$
20: **if** $i' = 1$ **and** $n_{i'k} > 1$ **then**
21: $\Upsilon \leftarrow (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
22: **else if** $i' > 1$ **then**
23: $i \in IS_k \leftarrow$ order preceding i' for vehicle $k \quad (i \neq i')$
24: $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_k + L_{i'jk} + \delta_{c_{i'} j} TL_k\} + U_{i'k}$ where $s^k_{ii'} = 1$
25: **if** $n_{i'k} > 1$ **then**
26: $\Upsilon \leftarrow \Upsilon + (n_{i'k} - 1)(L_{i'\Delta_{i'k}k} + U_{i'k} + \delta_{c_{i'}\Delta_{i'k}}(TU_k + TL_k))$
27: **end if**
28: **end if**
29: $TD^k_{inter} \leftarrow TD^k_{inter} + \Upsilon$
30: **end for**
31: $TD^k_{last} \leftarrow \delta_{c_i 0} TU_k$ where $s^k_{i0} = 1$
32: **end for**

The set W intended to contain orders and vehicles implied in each neighboring solution is then initialized as an empty set $\emptyset$, as the solution $\mathfrak{s}'$ related to the best neighbor. The current solution $\mathfrak{s}$ is saved in $\mathfrak{s}^\circ$.

Finally, TD^k_{first}, TD^k_{inter} and TD^k_{last} presented before are initialized to 0 and updated (cf. lines 12 to 32) for each vehicle $k \in VS$. Indeed, the duration related to the loading time, the travel time and the unloading time of the first delivery TD^k_{first} is first computed. Note that the loading time is not taken into account if the vehicle is preloaded ($k \in VP$). Then, the intermediate duration TD^k_{inter} is

determined by browsing each order $i' \in IS_k$ allocated to the vehicle. If the first order requires multiple deliveries ($i' = 1$ and $n_{i'k} > 1$), the duration of additional deliveries is taken into account. For the following orders of the vehicle ($i' > 1$), the duration of a first delivery is considered and, if additional deliveries are required for this order ($n_{i'k} > 1$), the duration of these ones are also taken into account. Finally, the duration related to the back travel of the last delivery TD^k_{last} is computed.

E.6.2 First deletion phase

This phase is described in detail in Procedures E.6.2 and E.6.3.

The size of a neighborhood n_N or more precisely the number of orders $i \in I$ allocated to vehicles $k \in V$ whose all deliveries have to be deleted is first defined, although mentioned as parameter. Indeed, the mentioned size n_N can be greater than the maximal size $|X|$ of a neighborhood and is therefore adapted ($\min\{|X|, n_N\}$) if needed. If no size is mentioned ($n_N = 0$), the entire neighborhood of size $|X|$ has to be generated.

For each deletion l, solution $\mathfrak{s}$ is beforehand reinitialized with the content of saved solution $\mathfrak{s}^\circ$. If the size of a neighborhood is not maximal ($n_N \neq |X|$), an element $x_{i^a k^a}$ of set X is selected randomly. Otherwise, the lth element of this set is selected. The number of deliveries $n_{i^a k^a}$ done by vehicle k^a for order i^a is set to 0. Following this deletion, $s^{k^a}_{0i^a}$, $s^{k^a}_{i^a 0}$ and $s^{k^a}_{ii^a}$ are also updated to modify the route of vehicle k^a.

Before deleting all deliveries of other orders $i \in I \setminus \{i^a\}$ from the route of other vehicles $k \in V \setminus \{k^a\}$ in the *second deletion phase*, the first partial solution related to vehicle k^a may be improved, and $TD^{k^a}_{first}$, $TD^{k^a}_{inter}$ and $TD^{k^a}_{last}$ whose aggregation corresponds to the total travel duration of vehicle k^a are updated.

E.6.3 Second deletion phase

This phase is described in detail in Procedures E.6.4 and E.6.5.

The number of other orders $i \in I \setminus \{i^a\}$ allocated to other vehicles $k \in V \setminus \{k^a\}$ whose all deliveries have to be deleted is first defined. This number corresponds to the size of set $|X'|$ which only considers vehicles $k \neq k^a$ whose capacity corresponds to Q_{k^a} and orders $i \neq i^a$ which are not allocated to a preload (not ($k \in VP$ and $s^k_{0i} = 1$)).

Procedure E.6.2 Generation of the best neighbor (Order swap) (Part 2)

```
    if n_N > 0 then
        n_N ← min{|X|, n_N}
    else
        n_N ← |X|
    end if
    for l = 1 → n_N do
        𝔰 ← 𝔰°
        if n_N ≠ |X| then
            ϵ ← RANDOM(1, |X|)
        else
            ϵ ← l
        end if
        i^a ← order related to the ϵth element of set X
        k^a ← vehicle related to the ϵth element of set X
        η^a ← number of deliveries related to the ϵth element of set X
        n_{i^a k^a} ← 0
        if s^{k^a}_{0 i^a} = 1 then
            i ← 0
        else
            i ∈ I ← order preceding i^a for vehicle k^a    (i ≠ i^a)
        end if
        if s^{k^a}_{i^a 0} = 1 then
            i'' ← 0
        else
            i'' ∈ I ← order following i^a for vehicle k^a    (i^a ≠ i'')
        end if
        s^{k^a}_{i i^a} ← 0
        s^{k^a}_{i^a i''} ← 0
        if i ≠ i'' then
            s^{k^a}_{i i''} ← 1
        end if
        𝔰 ← IMPROVESOLUTION(𝔰, k^a)
```

For each deletion m, solution $\mathfrak{s}$ is beforehand reinitialized with the content of saved solution $\mathfrak{s}^{\diamond}$. The number of deliveries $n_{i^b k^b}$ done by vehicle k^b for order i^b is set to 0. Only partial neighboring solutions that do not have already been created before ($(k^b, i^a, k^a, i^b) \notin W$) may be generated. To take this into consideration, orders and vehicles related to each eventual partial neighboring solution are saved as a tuple (k^a, i^b, k^b, i^a) in set W. Following this deletion, $s^{k^b}_{0 i^b}$, $s^{k^b}_{i^b 0}$ and $s^{k^b}_{i i^b}$ are also updated to modify the route of vehicle k^b.

Procedure E.6.3 Generation of the best neighbor (Order swap) (Part 3)

65: $TD^{k^a}_{first} \leftarrow 0$
66: $TD^{k^a}_{inter} \leftarrow 0$
67: $TD^{k^a}_{last} \leftarrow 0$
68: **if** $k^a \in VP$ **then**
69: $TD^{k^a}_{first} \leftarrow \delta_{c_i 0} TL_{k^a} + U_{ik^a}$ where $s^{k^a}_{0i} = 1$
70: **else**
71: $TD^{k^a}_{first} \leftarrow L_{i0k^a} + \delta_{c_i 0} TL_{k^a} + U_{ik^a}$ where $s^{k^a}_{0i} = 1$
72: **end if**
73: **for** $i' \in IS_{k^a} = 1 \rightarrow |IS_{k^a}|$ **do**
74: $\Upsilon \leftarrow 0$
75: **if** $i' = 1$ **and** $n_{i'k^a} > 1$ **then**
76: $\Upsilon \leftarrow (n_{i'k^a} - 1)(L_{i'\Delta_{i'k^a}k^a} + U_{i'k^a} + \delta_{c_{i'}\Delta_{i'k^a}}(TU_{k^a} + TL_{k^a}))$
77: **else if** $i' > 1$ **then**
78: $i \in IS_{k^a} \leftarrow$ order preceding i' for vehicle k^a $\quad (i \neq i')$
79: $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_{k^a} + L_{i'jk^a} + \delta_{c_{i'} j} TL_{k^a}\} + U_{i'k^a}$ where $s^{k^a}_{ii'} = 1$
80: **if** $n_{i'k^a} > 1$ **then**
81: $\Upsilon \leftarrow \Upsilon + (n_{i'k^a} - 1)(L_{i'\Delta_{i'k^a}k^a} + U_{i'k^a} + \delta_{c_{i'}\Delta_{i'k^a}}(TU_{k^a} + TL_{k^a}))$
82: **end if**
83: **end if**
84: $TD^{k^a}_{inter} \leftarrow TD^{k^a}_{inter} + \Upsilon$
85: **end for**
86: $TD^{k^a}_{last} \leftarrow \delta_{c_i 0} TU_{k^a}$ where $s^{k^a}_{i0} = 1$

Before reinserting all deleted deliveries of order i^b on vehicle k^a and all deleted deliveries of order i^a on vehicle k^b in the *swap phase*, the second partial solution related to vehicle k^b may be improved, $TD^{k^b}_{first}$, $TD^{k^b}_{inter}$ and $TD^{k^b}_{last}$ whose aggregation corresponds to the total travel duration of vehicle k^b are updated, and the value of the objective function related to current solution $\mathfrak{s}$ is saved as Θ.

E.6.4 Swap phase

This phase is described in detail in Procedure E.6.6 from line 137 to line 157.

If vehicle k^a can handle order i^b ($i^b \in I_{k^a}$) and if vehicle k^b can handle order i^a ($i^a \in I_{k^b}$), all deleted deliveries of order i^b are first reinserted on the route of vehicle k^a and all deleted deliveries of order i^a are then reinserted on the route of vehicle k^b.

Procedure E.6.4 Generation of the best neighbor (Order swap) (Part 4)

```
    X' ← {x_ik | i ∈ I, k ∈ V, x_ik = 1, i ≠ i^a, k ≠ k^a, Q_k = Q_{k^a} and not (k ∈ VP and s^k_{0i} = 1)}
    𝔰^◇ ← 𝔰
    for m = 1 → |X'| do
        𝔰 ← 𝔰^◇
        i^b ← order related to the mth element of set X'
        k^b ← vehicle related to the mth element of set X'
        η^b ← number of deliveries related to the mth element of set X'
        n_{i^b k^b} ← 0
        if (k^b, i^a, k^a, i^b) ∉ W then
            W ← W ∪ {(k^a, i^b, k^b, i^a)}
            if s^{k^b}_{0 i^b} = 1 then
                i ← 0
            else
                i ∈ I ← order preceding i^b for vehicle k^b   (i ≠ i^b)
            end if
            if s^{k^b}_{i^b 0} = 1 then
                i'' ← 0
            else
                i'' ∈ I ← order following i^b for vehicle k^b   (i^b ≠ i'')
            end if
            s^{k^b}_{i i^b} ← 0
            s^{k^b}_{i^b i''} ← 0
            if i ≠ i'' then
                s^{k^b}_{i i''} ← 1
            end if
            𝔰 ← ImproveSolution(𝔰, k^b)
```

Described in detail before in procedure OptimizeInsertion($\mathfrak{s}, i^b, k^a, \eta^b, TD^{k^a}_{first}, TD^{k^a}_{inter}, TD^{k^a}_{last}, 0$), the insertion process related to vehicle k^a and order i^b inserts all deleted deliveries of order i^b at the most appropriate position on the route of vehicle k^a. Described in detail before in procedure OptimizeInsertion($\mathfrak{s}, i^a, k^b, \eta^a, TD^{k^b}_{first}, TD^{k^b}_{inter}, TD^{k^b}_{last}, 0$), the insertion process related to vehicle k^b and order i^a inserts all deleted deliveries of order i^a at the most appropriate position on the route of vehicle k^b.

If vehicles k^a and k^b have enough availability to insert all deleted deliveries of orders i^b and i^a ($TD^{k^a}_{first} + TD^{k^a}_{inter} + TD^{k^a}_{last} + TD^{k^a}_{\theta^a} \leq A_{k^a}$) and ($TD^{k^b}_{first} + TD^{k^b}_{inter} + TD^{k^b}_{last} + TD^{k^b}_{\theta^b} \leq A_{k^b}$), if no neighbor has been generated before ($\mathfrak{s}' = \emptyset$) or if the current solution is better than the best solution ($\Theta + \theta^a + \theta^b \leq f(\mathfrak{s}')$), if the

Procedure E.6.5 Generation of the best neighbor (Order swap) (Part 5)

113: $TD^{k^b}_{first} \leftarrow 0$
114: $TD^{k^b}_{inter} \leftarrow 0$
115: $TD^{k^b}_{last} \leftarrow 0$
116: **if** $k^b \in VP$ **then**
117: $TD^{k^b}_{first} \leftarrow \delta_{c_i 0} TL_{k^b} + U_{ik^b}$ where $s^{k^b}_{0i} = 1$
118: **else**
119: $TD^{k^b}_{first} \leftarrow L_{i0k^b} + \delta_{c_i 0} TL_{k^b} + U_{ik^b}$ where $s^{k^b}_{0i} = 1$
120: **end if**
121: **for** $i' \in IS_{k^b} = 1 \rightarrow |IS_{k^b}|$ **do**
122: $\Upsilon \leftarrow 0$
123: **if** $i' = 1$ **and** $n_{i'k^b} > 1$ **then**
124: $\Upsilon \leftarrow (n_{i'k^b} - 1)(L_{i'\Delta_{i'k^b}k^b} + U_{i'k^b} + \delta_{c_{i'}\Delta_{i'k^b}}(TU_{k^b} + TL_{k^b}))$
125: **else if** $i' > 1$ **then**
126: $i \in IS_{k^b} \leftarrow$ order preceding i' for vehicle k^b $\quad (i \neq i')$
127: $\Upsilon \leftarrow \min_{j \in D_{i'}} \{\delta_{c_i j} TU_{k^b} + L_{i'jk^b} + \delta_{c_{i'} j} TL_{k^b}\}$ where $s^{k^b}_{ii'} = 1$
128: $\Upsilon \leftarrow \Upsilon + U_{i'k^b}$ where $s^{k^b}_{ii'} = 1$
129: **if** $n_{i'k^b} > 1$ **then**
130: $\Upsilon \leftarrow \Upsilon + (n_{i'k^b} - 1)(L_{i'\Delta_{i'k^b}k^b} + U_{i'k^b} + \delta_{c_{i'}\Delta_{i'k^b}}(TU_{k^b} + TL_{k^b}))$
131: **end if**
132: **end if**
133: $TD^{k^b}_{inter} \leftarrow TD^{k^b}_{inter} + \Upsilon$
134: **end for**
135: $TD^{k^b}_{last} \leftarrow \delta_{c_i 0} TU_{k^b}$ where $s^{k^b}_{i0} = 1$
136: $\Theta \leftarrow f(\mathfrak{s})$

insertion of order i^a on the route of vehicle k^b and the insertion of order i^b on the route of vehicle k^a are not tabu ($(i^a, k^b) \notin TA$, $(i^b, k^a) \notin TA$) or if the current solution is better than the best solution of the tabu search ($\Theta + \theta^a + \theta^b \leq f(\mathfrak{s}^*)$), i.e. satisfies the aspiration criterion, current vehicles k^a and k^b, and current orders i^a and i^b are respectively saved as k^a_{TA}, k^b_{TA}, i^a_{TA} and i^b_{TA}. Furthermore, current neighboring solution $\mathfrak{s}$ is set as best neighbor $\mathfrak{s}'$.

After having generated the neighboring solutions and selected among them the best neighbor, the tabu list TA is updated (cf. lines 158 to 164). Before inserting a new element in this list, the first and eventually the second element are deleted but only if the list has beforehand reached its maximal size n_{TA}. Two tuples (i^a_{TA}, k^a_{TA}) and (i^b_{TA}, k^b_{TA}) related to vehicles k^a and k^b and orders i^a and i^b are set as tabu for the insertion. Indeed, the reinsertion of deliveries of order i^a

Procedure E.6.6 Generation of the best neighbor (Order swap) (Part 6)

```
137:                if i^b ∈ I_{k^a} and i^a ∈ I_{k^b} then
138:                    S^a ← OptimizeInsertion(𝔰, i^b, k^a, η^b, TD^{k^a}_{first}, TD^{k^a}_{inter}, TD^{k^a}_{last}, 0)
139:                    {𝔰, θ^a, TD^{k^a}_{θ^a}} ← S^a
140:                    S^b ← OptimizeInsertion(𝔰, i^a, k^b, η^a, TD^{k^b}_{first}, TD^{k^b}_{inter}, TD^{k^b}_{last}, 0)
141:                    {𝔰, θ^b, TD^{k^b}_{θ^b}} ← S^b
142:                    if TD^{k^a}_{first} + TD^{k^a}_{inter} + TD^{k^a}_{last} + TD^{k^a}_{θ^a} ≤ A_{k^a} and
143:                            TD^{k^b}_{first} + TD^{k^b}_{inter} + TD^{k^b}_{last} + TD^{k^b}_{θ^b} ≤ A_{k^b} then
144:                        if 𝔰′ = ∅ or Θ + θ^a + θ^b ≤ f(𝔰′) then
145:                            if ((i^a, k^b) ∉ TA and (i^b, k^a) ∉ TA) or Θ + θ^a + θ^b ≤ f(𝔰*) then
146:                                k^a_{TA} ← k^a
147:                                k^b_{TA} ← k^b
148:                                i^a_{TA} ← i^a
149:                                i^b_{TA} ← i^b
150:                                𝔰′ ← 𝔰
151:                            end if
152:                        end if
153:                    end if
154:                end if
155:            end if
156:        end for
157:    end for
158:    if |TA| = n_{TA} then
159:        TA ← TA \ {(i_1, k_1)} where (i_1 ∈ I, k_1 ∈ V) is the first tuple of TA
160:    end if
161:    if |TA| = (n_{TA} − 1) then
162:        TA ← TA \ {(i_1, k_1)} where (i_1 ∈ I, k_1 ∈ V) is the first tuple of TA
163:    end if
164:    TA ← TA ∪ {(i^a_{TA}, k^a_{TA}), (i^b_{TA}, k^b_{TA})}
165:    return {𝔰′, TA}
166: end procedure
```

on the route of vehicle k^a and the reinsertion of deliveries of order i^b on the route of vehicle k^b are forbidden for n_{TA} iterations.

The solution $\mathfrak{s}'$ related to the best neighbor is finally returned, as the updated tabu list TA (cf. Procedure E.6.6 at line 165).

E.7 Modification of a solution

One parameter is needed to run the corresponding procedure: a solution $\mathfrak{s}$.

E.7.1 Initialization phase

This phase is described in detail in Procedure E.7.1.

Procedure E.7.1 Modification of a solution (Part 1)

procedure MODIFYSOLUTION($\mathfrak{s}$)
  $VS \leftarrow$ set of vehicles used in solution $\mathfrak{s}$
  $VC \subseteq VS \leftarrow$ set of vehicles whose customer is also visited by another vehicle
  $n_V \leftarrow \left\lceil \frac{|VS|}{2} \right\rceil$
  $n_V^a \leftarrow \min\{|VC|, \left\lceil \frac{n_V}{2} \right\rceil\}$
  $n_V^b \leftarrow n_V - n_V^a$
  $\mathfrak{s}^\circ \leftarrow \mathfrak{s}$
  $d_i^\circ \leftarrow d_i \quad \forall i \in I$
  $V^\circ \leftarrow V$
  $VP^\circ \leftarrow VP$
  repeat

The set VS of vehicles used, the set $VC \subseteq VS$ of vehicles whose customer is also visited by another vehicle in given solution $\mathfrak{s}$ are first defined as follows:

$$VS = \{k \in V | \sum_{i \in I} n_{ik} \geq 1\}$$

$$VC = \{k \in VS | \exists i \in I \text{ where } x_{ik} = 1 \text{ and } \sum_{k' \in V} v_{c_i k'} > 1\}$$

where

$$x_{ik} = \begin{cases} 1 & \text{if } n_{ik} \geq 1 \quad \forall i \in I, \forall k \in V \\ 0 & \text{otherwise.} \end{cases}$$

The number n_V of vehicles whose allocated deliveries have to be deleted is set as half of the vehicles used in the current solution $\mathfrak{s}$. To force the reduction of the number of customers visited by different vehicles, half of the vehicles implied by the deletion n_V^a concerns vehicles whose customer is also visited by another vehicle. This number may however be adapted if there is not enough such vehicles. The remainder n_V^b concerns the remaining vehicles implied by the deletion.

Finally, the current solution $\mathfrak{s}$, the demand of each order $d_i \in I$, the available vehicles V and the available preloaded vehicles VP are respectively saved in $\mathfrak{s}^\circ$, d_i°, V° and VP°.

E.7.2 Deletion phase

This phase is described in detail in Procedure E.7.2.

Procedure E.7.2 Modification of a solution (Part 2)

$\mathfrak{s} \leftarrow \mathfrak{s}^\circ$
$d_i \leftarrow d_i^\circ \quad \forall i \in I$
$V \leftarrow V^\circ$
$VP \leftarrow VP^\circ$
for $l = 1 \rightarrow n_V^a$ **do**
  $k \in VC \leftarrow \text{RANDOM}(1, |VC|)$
  $n_{ik} \leftarrow 0 \quad \forall i \in I$
  $s_{ii'}^k \leftarrow 0 \quad \forall i \in I \cup \{0\}, \forall i' \neq i \in I \cup \{0\}$
end for
for $l = 1 \rightarrow n_V^b$ **do**
  $k \in VS \leftarrow \text{RANDOM}(1, |VS|)$
  $n_{ik} \leftarrow 0 \quad \forall i \in I$
  $s_{ii'}^k \leftarrow 0 \quad \forall i \in I \cup \{0\}, \forall i' \neq i \in I \cup \{0\}$
end for
$\mathfrak{s}^a \leftarrow \mathfrak{s}$

Before trying to modify the current solution, solution $\mathfrak{s}$, the demand of each order $d_i \in I$, the available vehicles V and the available preloaded vehicles VP are respectively reinitialized with the content of $\mathfrak{s}^\circ$, d_i°, V° and VP°. Indeed, more than one test may be required to generate a modified solution that is feasible.

First of all, n_V^a vehicles $k \in VC$ whose customer is also visited by another vehicle are selected randomly. The number of deliveries n_{ik} done by each of these selected vehicles is set to 0, as all $s_{ii'}^k$ indicating if order $i' \in I$ is the immediate successor of order $i \in I$ on the route of the vehicle.

Then, n_V^b remaining vehicles $k \in VS$ are selected randomly. The number of deliveries n_{ik} done by each of these selected vehicles is set to 0, as all $s_{ii'}^k$ indicating if order $i' \in I$ is the immediate successor of order $i \in I$ on the route of the vehicle.

The partial solution $\mathfrak{s}$ resulting of the deletion of the deliveries is finally saved as first partial solution $\mathfrak{s}^a$.

E.7.3 Construction phase

This phase is described in detail in Procedure E.7.3 from line 27 to line 33.

Procedure E.7.3 Modification of a solution (Part 3)

```
27:         for i ∈ I = 1 → |I| do
28:             e_i ← Σ_{k∈V} (n_ik Q_k)
29:             d_i ← max{0, d_i − e_i}
30:         end for
31:         V ← V \ VS
32:         VP ← VP \ {VP ⊆ VS}
33:         𝔰^b ← ConstructInitialSolution
34:     until e_i ≥ d_i ∀i ∈ I
35:     𝔰′ ← 𝔰^a ∪ 𝔰^b
36:     return 𝔰′
37: end procedure
```

The allocation of new deliveries to unused vehicles, i.e. the construction of a second partial solution $\mathfrak{s}^b$, can be done with the procedure CONSTRUCTINITIALSOLUTION. For this purpose, the demand d_i of each order $i \in I$ is adapted by subtracting the already delivered quantities e_i. Furthermore, the set of vehicles V and the set of preloaded vehicles VP are updated to not take into account vehicles that are already used in the first partial solution $\mathfrak{s}^a$.

The *deletion phase* and *construction phase* are rerun until the second partial solution $\mathfrak{s}^b$ is feasible, i.e. until the demand d_i of each order $i \in I$ is entirely satisfied. After these two phases, the two partial solutions are merged ($\mathfrak{s}^a \cup \mathfrak{s}^b$) and returned as the modified solution $\mathfrak{s}'$ (cf. Procedure E.7.3 at line 36).

Appendix F

Tabu search method performance

This appendix presents the performance of the tabu search method discussed in Subsection 9.2.2 of Chapter 9, or more precisely of TS_8, TS_{15} and TS_{20}. For this purpose, statistics of this subsection are illustrated in a figure and commented for each of the ten instances from Section F.1 to Section F.10.

As mentioned before, a plot first displays in each of the following figures the progression of the results provided by TS_8, TS_{15} and TS_{20} in time. The x-axis shows the computational time in seconds (CPU) while the y-axis displays the value of $f(\mathfrak{s})$, i.e. the objective function. The progression of a solution is displayed in orange for TS_8, in blue for TS_{15} and in green for TS_{20}. Note that for each method, 10 progressions related to 10 resolutions of an instance are drawn. The progression of the solution provided by 1P is also displayed, in red, in order to be compared with the results of the tabu search method. Furthermore, the value of the objective function related to the solution given by the cement factory (CF) and the estimated lower bound (LB) computed by CPLEX for 1P are finally mentioned in the plot with dotted lines. Box-and-whisker plots are then used to present for the final results of TS_8, TS_{15} and TS_{20} the value of the objective function and related to the three different objectives, i.e. the total value of the variables v_{ck}, the total number $\sum_{k \in V} w_k$ of vehicles used, and the total weighted delivery time $\sum_{k \in V} F_k \alpha_k$. Note that a red line indicates the value for the result obtained with 1P.

F.1 Statistics related to instance 1

Solutions provided by TS_8, TS_{15}, TS_{20} and 1P for instance 1 are considered in Figure F.1. For TS_8, 3, 5 and 2 solutions are from 1% of their final value respectively before 900 seconds, between 901 and 3'600 seconds, and between 3'601 and 4'700 seconds. For TS_{15}, 7, 2 and 1 solutions are from 1% of their final value respectively before 900 seconds, between 901 and 3'600 seconds, and between 3'601 and 4'200 seconds. For TS_{20}, 8 and 2 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 2'500 seconds. Note that the solution provided by 1P reaches 1% of its final value at 508 seconds while all solutions provided by the tabu search method are lower than 4.6% of their final value at 900 seconds.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that the final results provided by TS_8, TS_{15} and TS_{20} are close to the one computed by CPLEX for 1P. While the box-and-whisker plots related to the first ($\sum v_{ck}$) and second ($\sum w_k$) objective show that solutions of TS_8, TS_{15} and TS_{20} are similar to the solution given by 1P from this point of view, the box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 4'520 and 4'675 minutes, between 4'530 and 4'540 minutes, and between 4'520 and 4'540 minutes (while 1P gives 4'535 minutes).

F.2 Statistics related to instance 2

Solutions obtained with TS_8, TS_{15}, TS_{20} and 1P for instance 2 are evaluated in Figure F.2. For TS_8, 1, 5, 2 and 2 solutions are from 1% of their final value respectively before 900 seconds, between 901 and 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'201 and 13'000 seconds. For TS_{15}, 4, 1 and 5 solutions are from 1% of their final value respectively between 901 and 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'201 and 19'200 seconds. For TS_{20}, 4, 5 and 1 solutions are from 1% of their final value respectively between 901 and 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'201 and 7'400 seconds. Note that the solution provided by 1P reaches 1% of its final value at 1'557 seconds while all solutions provided by the tabu search method are respectively lower than 9.6%, 8.8% and 9.6% of their final value at 900 seconds for TS_8, TS_{15} and TS_{20}.

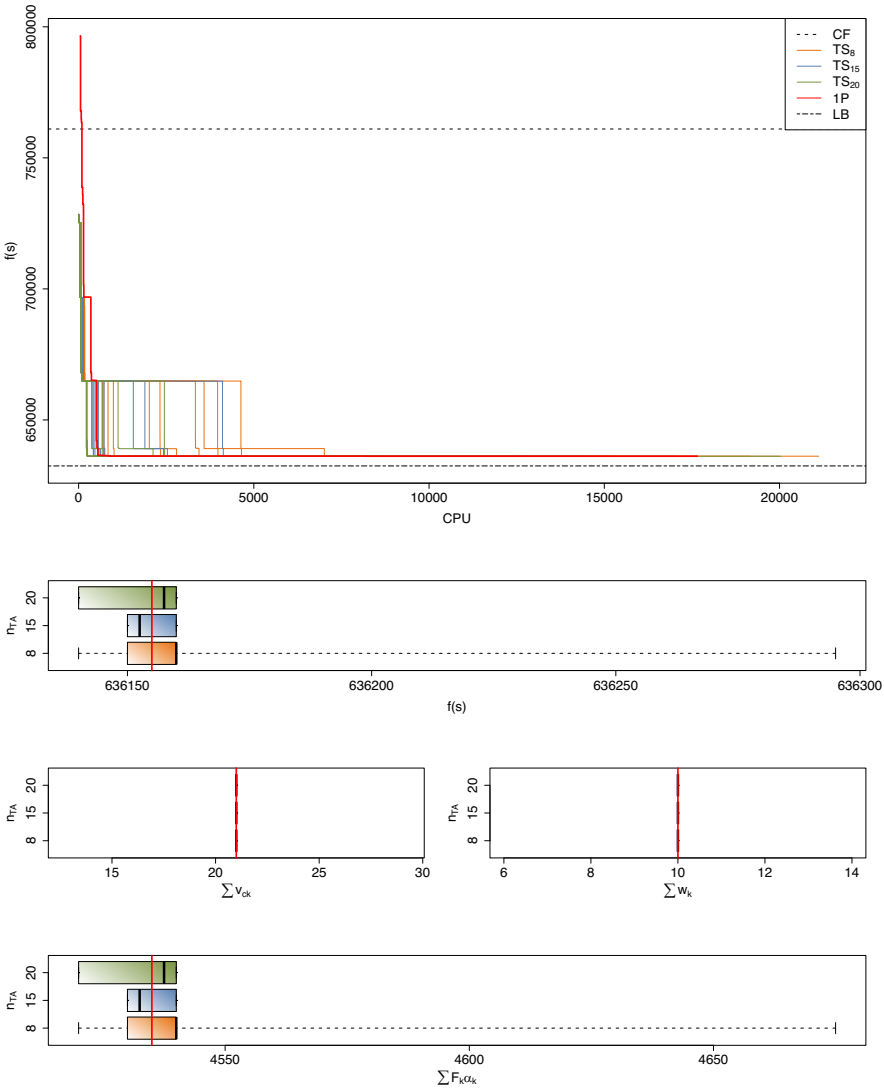

Figure F.1: Statistics related to instance 1

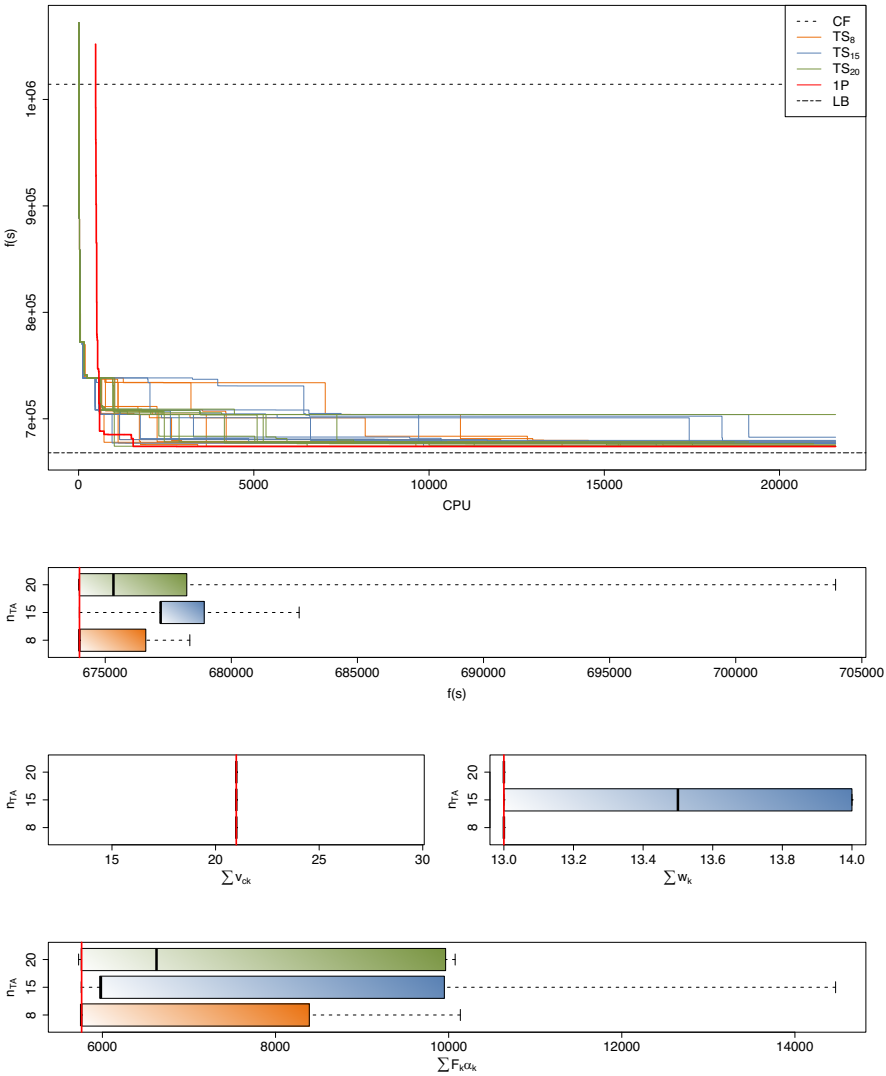

Figure F.2: Statistics related to instance 2

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are similar or slightly worse than the one computed by CPLEX for 1P. The box-and-whisker plots related to the first ($\sum v_{ck}$) and second ($\sum w_k$) objective show that the solutions of TS_8, TS_{15}, TS_{20} are similar to the solution given by 1P from this point of view, except for TS_{15} which uses 1 vehicle more for 5 solutions. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'755 and 10'135 minutes due to the use of third party vehicles for 1 solution, between 5'755 and 14'470 minutes due to the use of third party vehicles for 4 solutions, and between 5'725 and 10'075 minutes due to the use of third party vehicles for 5 solutions (while 1P gives 5'760 minutes).

F.3 Statistics related to instance 3

Solutions provided by TS_8, TS_{15}, TS_{20} and 1P for instance 3 are studied in Figure F.3. For TS_8, 9 and 1 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 2'700 seconds. For TS_{15}, 7 and 3 solutions are from 1% of their final value respectively between 901 and 3'600 seconds, and between 3'601 and 18'500 seconds. Note that 2 of the 3 solutions obtained between 3'601 and 18'500 seconds are better than all solutions provided by TS_8 and other solutions of TS_{15}. For TS_{20}, 4, 1 and 5 solutions are from 1% of their final value respectively between 901 and 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'201 and 20'100 seconds. It can be observed that all solutions obtained between 7'201 and 20'100 seconds are better than 9 solutions of TS_8, 7 solutions of TS_{15} and than the 4 solutions of TS_{20} found before 900 seconds. The solution provided by 1P reaches 1% of its final value at 1'518 seconds and is always worse than solutions provided by TS_8, TS_{15}, TS_{20} along the time progression. Furthermore, all solutions provided by the tabu search method are respectively lower than 3.3%, 3.8% and 3.8% of their final value at 900 seconds for TS_8, TS_{15} and TS_{20}.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are always better than the one computed by CPLEX for 1P. The box-and-whisker plots related to the first objective ($\sum v_{ck}$) show that the solutions of TS_8 are similar to the solution given by 1P from this point of view, while TS_{15} and TS_{20} use 1 vehicle less to deliver a customer respectively for 3 and 6 instances (most of these instances use third party vehicles). The box-and-whisker plots related to the second objective ($\sum w_k$) shows that solutions of TS_8, TS_{15}, TS_{20}

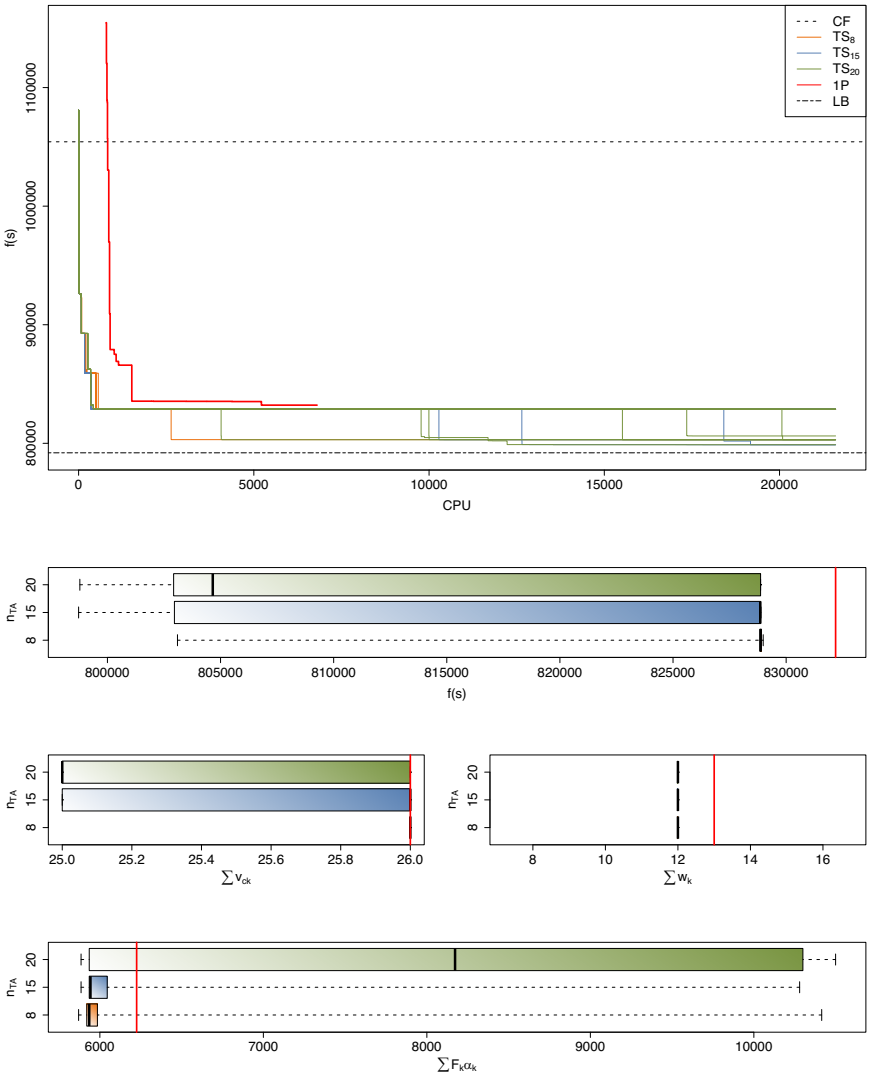

Figure F.3: Statistics related to instance 3

always use one vehicle less than the solution given by 1P. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'870 and 10'415 minutes due to the use of third party vehicles for 1 solution, between 5'885 and 10'280 minutes due to the use of third party vehicles for 1 solution, and between 5'885 and 10'500 minutes due to the use of third party vehicles for 5 solutions (while 1P gives 6'225 minutes).

F.4 Statistics related to instance 4

Solutions obtained with TS_8, TS_{15}, TS_{20} and 1P for instance 4 are considered in Figure F.4. For TS_8, TS_{15} and TS_{20}, all solutions are from 1% of their final value respectively before 60, 40 and 60 seconds. Note that the solution provided by 1P reaches 1% of its final value at 1'304 seconds and is always worse than solutions provided by TS_8, TS_{15} and TS_{20} along the time progression.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are always better than the one computed by CPLEX for 1P. While the box-and-whisker plots related to the first objective ($\sum v_{ck}$) show that solutions of TS_8, TS_{15} and TS_{20} are similar from this point of view, the box-and-whisker plots related to the second objective ($\sum w_k$) indicate that solutions of TS_8, TS_{15}, TS_{20} use between 1 and 2 vehicles less than the solution given by 1P. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'560 and 6'080 minutes, between 5'560 and 6'120 minutes, and between 5'540 and 5'815 minutes (while 1P, which uses third party vehicles, gives 9'855 minutes).

F.5 Statistics related to instance 5

Solutions provided by TS_8, TS_{15}, TS_{20} and 1P for instance 5 are evaluated in Figure F.5. For TS_8, 1, 3 and 6 solutions are from 1% of their final value respectively before 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'200 and 21'400 seconds. For TS_{15}, 2, 2 and 6 solutions are from 1% of their final value before 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'201 and 17'300 seconds. Although final solutions of TS_8 and TS_{15} are relatively similar, TS_{15} has

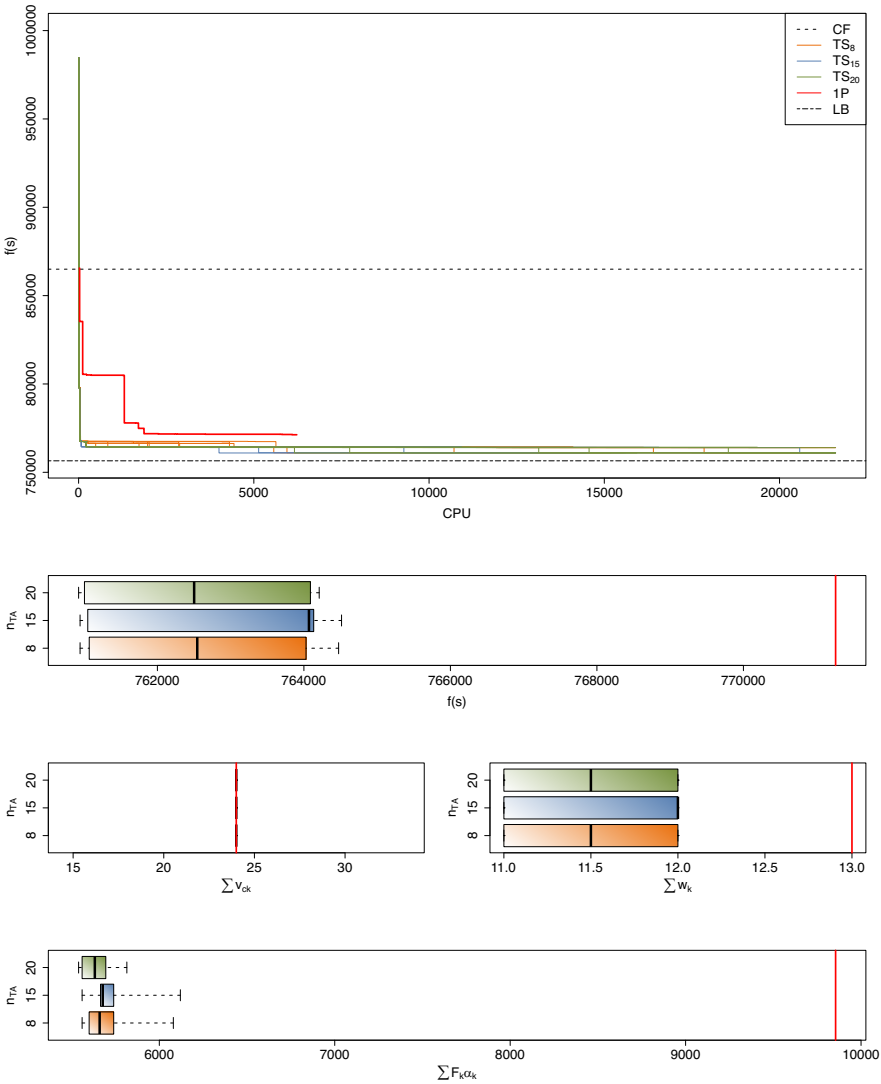

Figure F.4: Statistics related to instance 4

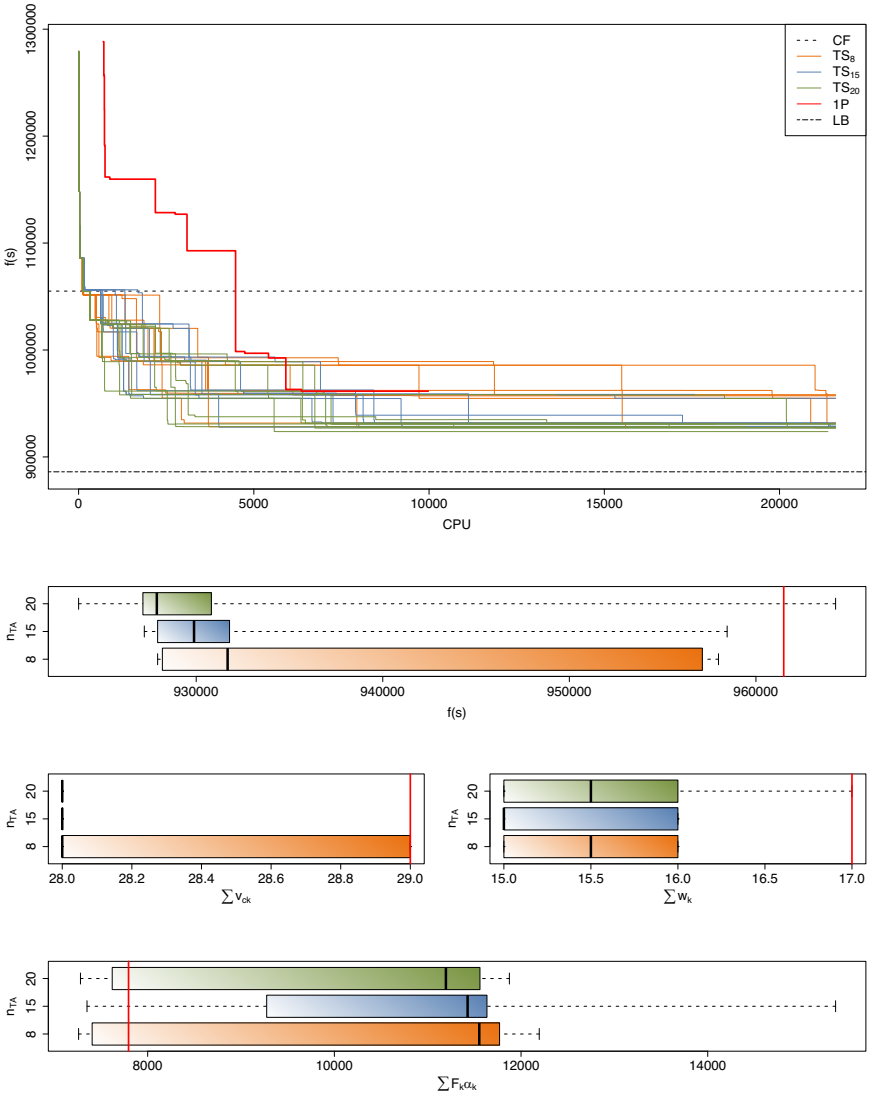

Figure F.5: Statistics related to instance 5

found them faster than TS_8. For TS_{20}, 2, 5 and 3 solutions are from 1% of their final value before 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'201 and 20'200 seconds. Even if final solutions obtained with TS_{15} and TS_{20} are close, TS_{20} has found them faster than TS_{15}, and therefore faster than TS_8. Note that all solutions provided by the tabu search method are respectively lower than 13.3%, 13.9% and 11.3% of their final value at 900 seconds for TS_8, TS_{15}, TS_{20}. Note that the solution provided by 1P reaches 1% of its final value at 5'913 seconds and is always worse than 4, 8 and 8 solutions respectively provided by TS_8, TS_{15} and TS_{20} along the time progression.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are always better than the one computed by CPLEX for 1P, except for 1 solution provided by TS_{20}. The box-and-whisker plots related to the first objective ($\sum v_{ck}$) show that 6, 8 and 9 solutions respectively provided by TS_8, TS_{15} and TS_{20} use 1 vehicle less than the solution given by 1P to deliver a customer. The box-and-whisker plots related to the third objective ($\sum w_k$) indicate that solutions of TS_8, TS_{15}, TS_{20} use 1 or 2 vehicles less than the solution given by 1P, except for 1 solution of TS_{20}. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 7'260 and 12'195 minutes due to the use of third party vehicles for 6 solutions, between 7'350 and 15'370 minutes due to the use of third party vehicles for 7 solutions, and between 7'280 and 11'875 minutes due to the use of third party vehicles for 7 solutions (while 1P gives 7'795 minutes).

F.6 Statistics related to instance 6

Solutions obtained with TS_8, TS_{15}, TS_{20} and 1P for instance 6 are studied in Figure F.6. For TS_8, 3 and 7 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 3'300 seconds. For TS_{15} and TS_{20}, all solutions are from 1% of their final value respectively before 200 and 170 seconds. Note that all solutions provided by the tabu search method are respectively lower than 3.9%, 0.5% and 0.4% of their final value at 900 seconds for TS_8, TS_{15}, TS_{20}. Furthermore, the solution provided by 1P reaches 1% of its final value at 7'066 seconds and is always worse than solutions provided by TS_8, TS_{15}, TS_{20} along the time progression, except for 1 solution of TS_8 and TS_{15}.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are always better than the one computed by CPLEX for 1P.

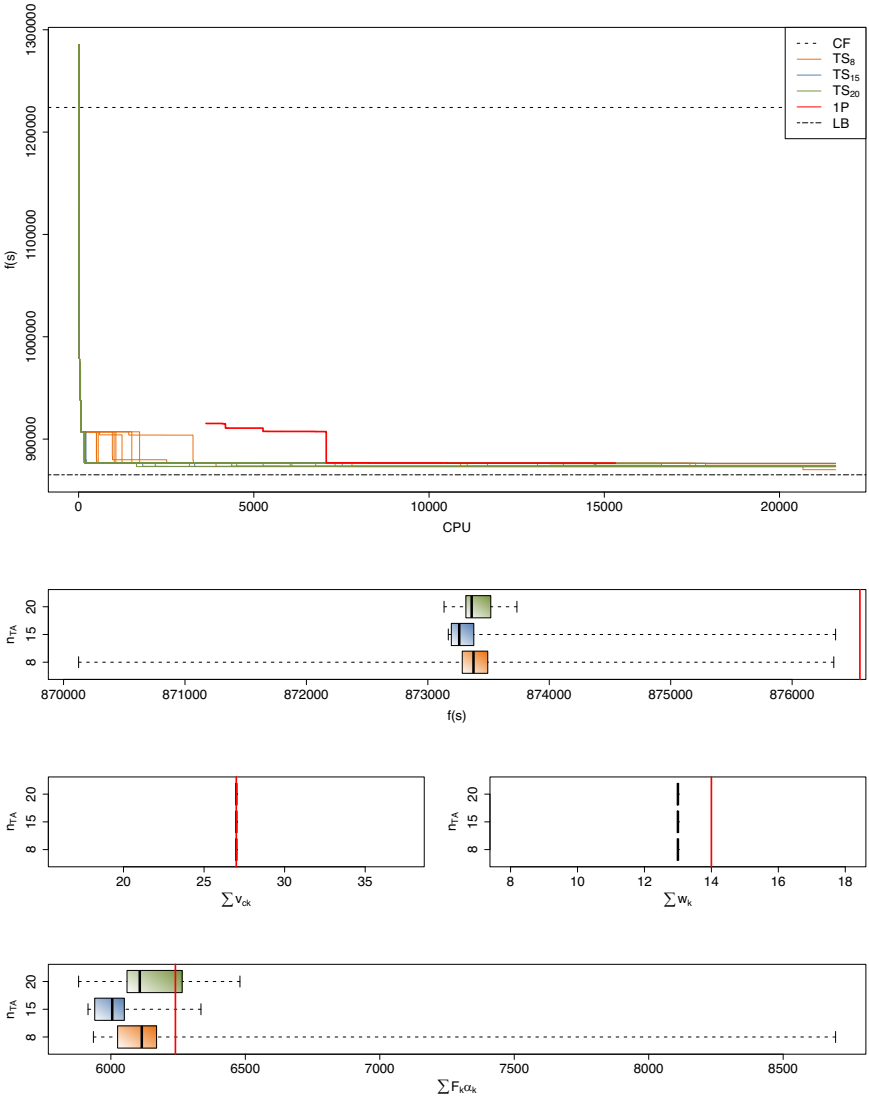

Figure F.6: Statistics related to instance 6

While the box-and-whisker plots related to the first objective ($\sum v_{ck}$) show that solutions of TS_8, TS_{15} and TS_{20} are similar to the solution given by 1P from this point of view, the box-and-whisker plots related to the second objective ($\sum w_k$) indicate that solutions of TS_8, TS_{15} and TS_{20} mostly use one vehicle less than the solution given by 1P. Note that 1 solution provided by TS_8 and TS_{15} uses the same number of vehicles than the solution of 1P, while 1 solution of TS_8 uses two vehicles less than this solution. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'935 and 8'695 minutes due to the use of third party vehicles for 1 solution, between 5'915 and 6'335 minutes, and between 5'880 and 6'480 minutes (while 1P gives 6'240 minutes).

F.7 Statistics related to instance 7

Solutions provided by TS_8, TS_{15}, TS_{20} and 1P for instance 7 are considered in Figure F.7. For TS_8, 5 and 5 solutions are from 1% of their final value respectively before 900 seconds, and between 901 seconds and 2'700 seconds. For TS_{15} and TS_{20}, all solutions are from 1% of their final value respectively before 900 and 700 seconds. Note that all solutions provided by the tabu search method are respectively lower than 4.6%, 0.8% and 0.9% of their final value at 900 seconds for TS_8, TS_{15} and TS_{20} while the solution provided by 1P reaches 1% of its final value at 3'542 seconds.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are either slightly better or worse than the one computed by CPLEX for 1P. While the box-and-whisker plots related to the first objective ($\sum v_{ck}$) show that solutions of TS_8, TS_{15}, TS_{20} are similar from this point of view, the box-and-whisker plots related to the second objective ($\sum w_k$) indicate that solutions of TS_8, TS_{15} and TS_{20} often use 1 vehicle more than the solution given by 1P. Indeed, 6 solutions of TS_8, 7 solutions of TS_{15} and TS_{20} are concerned. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'675 and 5'825 minutes, between 5'560 and 5'790 minutes, and between 5'555 and 5'730 minutes (while 1P gives 5'790 minutes).

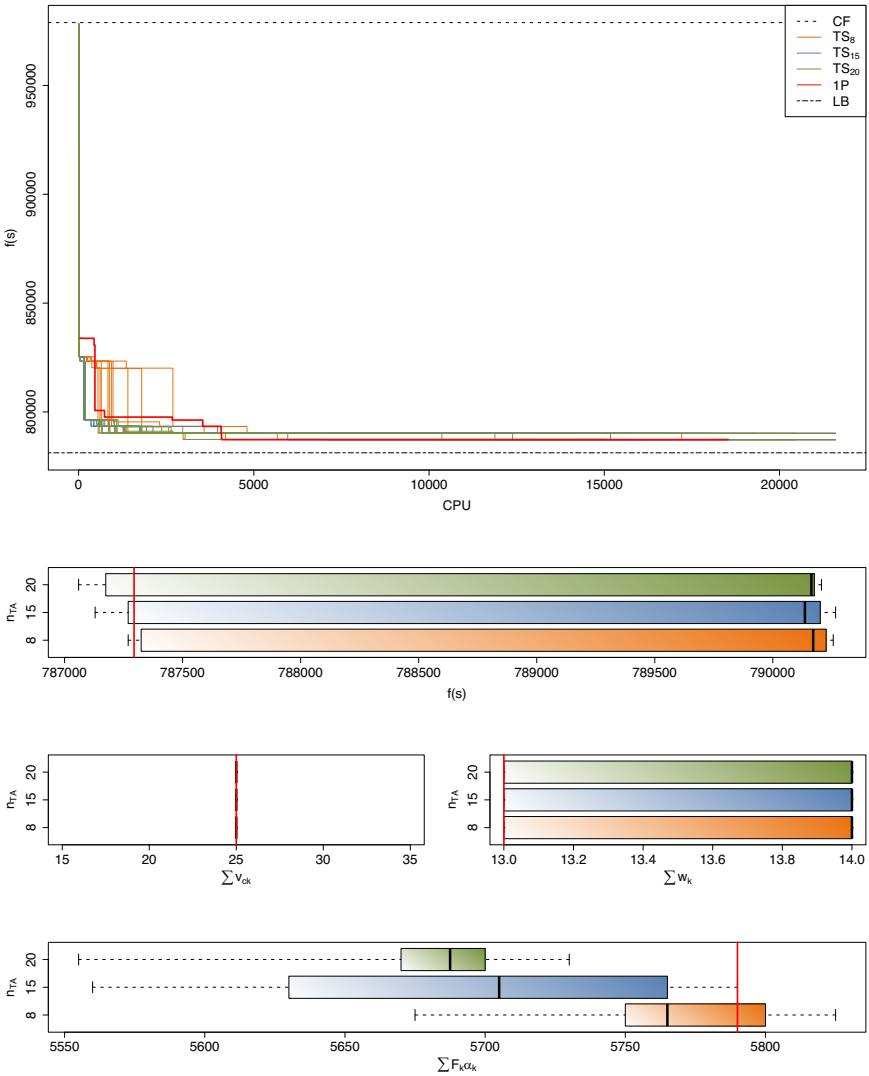

Figure F.7: Statistics related to instance 7

F.8 Statistics related to instance 8

Solutions obtained with TS_8, TS_{15}, TS_{20} and 1P for instance 8 are evaluated in Figure F.8. For TS_8, 5 and 5 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 2'400 seconds. For TS_{15}, 9 and 1 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 1'300 seconds. For TS_{20}, all solutions are from 1% of their final value before 120 seconds. Note that all solutions provided by the tabu search method are respectively lower than 4.4%, 4.4% and 0.5% of their final value at 900 seconds for TS_8, TS_{15} and TS_{20} while the solution provided by 1P reaches 1% of its final value at 535 seconds and is always worse along the time progression than 2, 7 and all solutions respectively provided by TS_8, TS_{15} and TS_{20}.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are always better than the one computed by CPLEX for 1P. The box-and-whisker plots related to the first objective ($\sum v_{ck}$) show that solutions of TS_8, TS_{15}, TS_{20} are similar than the solution given by 1P from this point of view, while the box-and-whisker plots related to the second objective ($\sum w_k$) indicate that solutions of TS_8, TS_{15}, TS_{20} always use one vehicle less than the solution given by 1P, and even two vehicles less for 1 solution of TS_{20}. The box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'240 and 5'320 minutes, between 5'245 and 5'320 minutes, and between 5'230 and 5'350 minutes (while 1P gives 5'475 minutes).

F.9 Statistics related to instance 9

Solutions provided by TS_8, TS_{15}, TS_{20} and 1P for instance 9 are studied in Figure F.9. For TS_8, 2, 5 and 3 solutions are from 1% of their final value respectively before 3'600 seconds, between 3'601 and 7'200 seconds, and between 7'201 and 13'300 seconds. For TS_{15}, 2 and 8 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 3'600 seconds. For TS_{20}, 1, 7 and 2 solutions are from 1% of their final value respectively before 900 seconds, between 901 and 3'600 seconds, and between 3'601 and 6'300 seconds. Note that all solutions provided by the tabu search method are respectively lower than 4.1%, 4.2% and 4.2% of their final value at 900 seconds for TS_8, TS_{15} and TS_{20}. Furthermore, the solution provided by 1P reaches 1% of its final value at 11'532 seconds and is

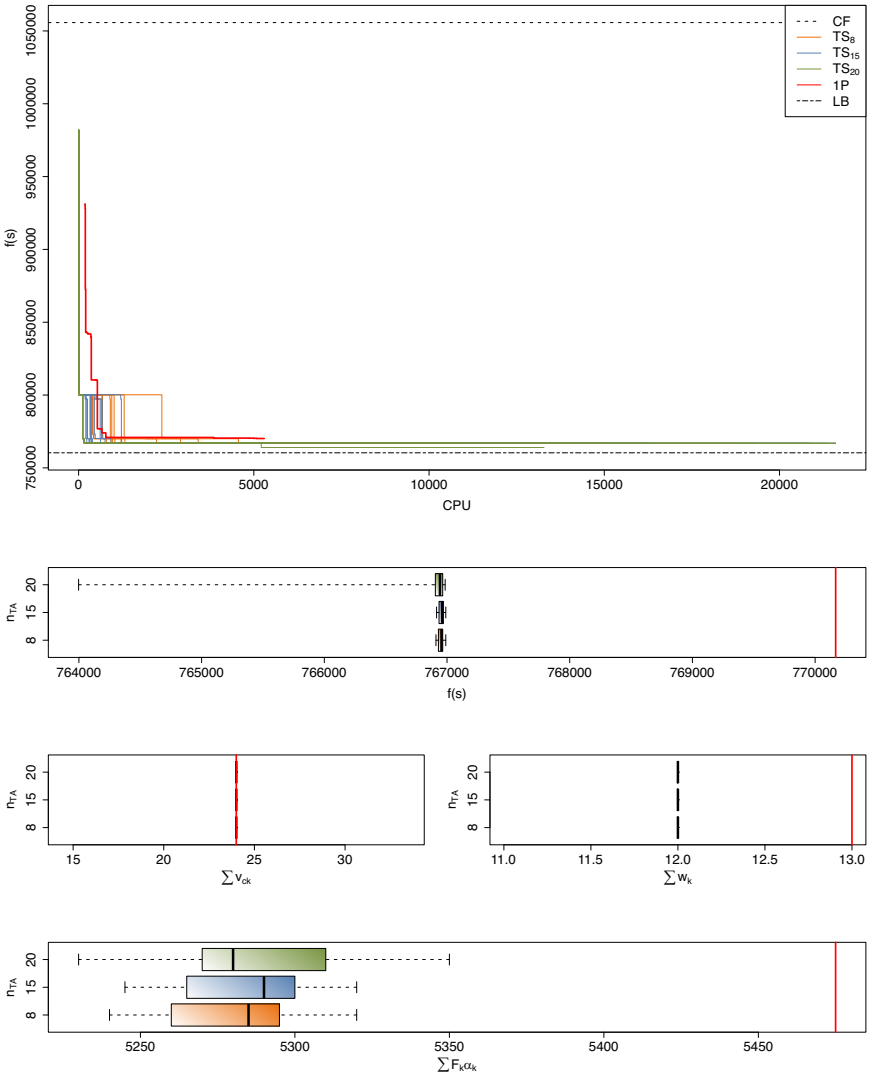

Figure F.8: Statistics related to instance 8

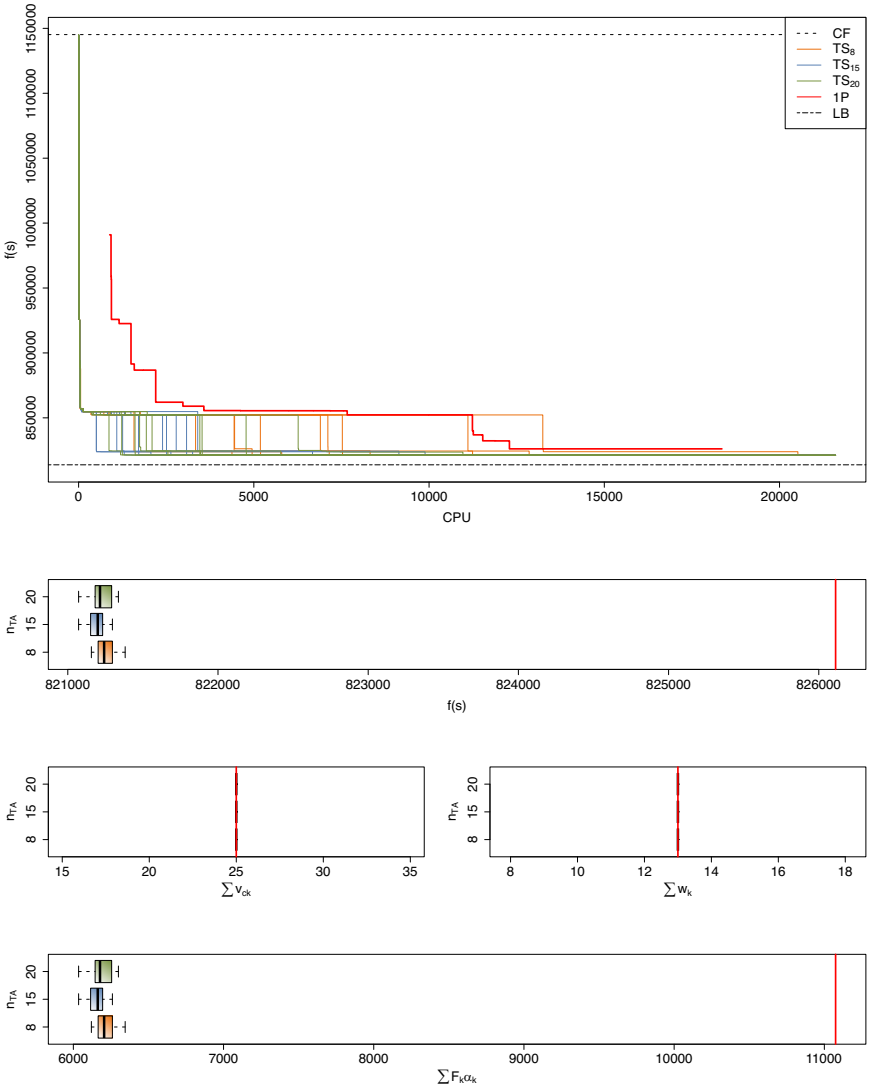

Figure F.9: Statistics related to instance 9

always worse than solutions provided by TS_8, TS_{15} and TS_{20} along the time progression, except for 2 solutions provided by TS_8.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are always better than the one computed by CPLEX for 1P. While the box-and-whisker plots related to the first ($\sum v_{ck}$) and second ($\sum w_k$) objective show that solutions of TS_8, TS_{15}, TS_{20} are similar to the solution given by 1P from this point of view, the box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 6'120 and 6'345 minutes, between 6'035 and 6'260 minutes, and between 6'035 and 6'300 minutes (while 1P, which uses third party vehicles, gives 11'075 minutes).

F.10 Statistics related to instance 10

Solutions obtained with TS_8, TS_{15}, TS_{20} and 1P for instance 10 are considered in Figure F.10. For TS_8, all solutions are from 1% of their final value before 900 seconds. For TS_{15}, 6 and 4 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 1'400 seconds. For TS_{20}, 9 and 1 solutions are from 1% of their final value respectively before 900 seconds, and between 901 and 1'300 seconds. Note that all solutions provided by the tabu search method are respectively lower than 0.5%, 5.7% and 4.7% of their final value at 900 seconds for TS_8, TS_{15} and TS_{20} while the solution provided by 1P reaches 1% of its final value at 489 seconds.

The box-and-whisker plots displayed for $f(\mathfrak{s})$ indicate that solutions provided by TS_8, TS_{15} and TS_{20} are similar or slightly worse than the solution computed by CPLEX for 1P, except for 1 solution that is strictly better with TS_8 than with 1P. While the box-and-whisker plots related to the first ($\sum v_{ck}$) and second ($\sum w_k$) objective show that solutions of TS_8, TS_{15} and TS_{20} are similar to the solution given by 1P from this point of view, the box-and-whisker plots affiliated to the third objective ($\sum F_k \alpha_k$) indicate that the total weighted delivery time of the solutions provided by TS_8, TS_{15} and TS_{20} is respectively between 5'720 and 6'120 minutes, between 5'765 and 6'070 minutes, and between 5'765 and 5'965 minutes (while 1P gives 5'765).

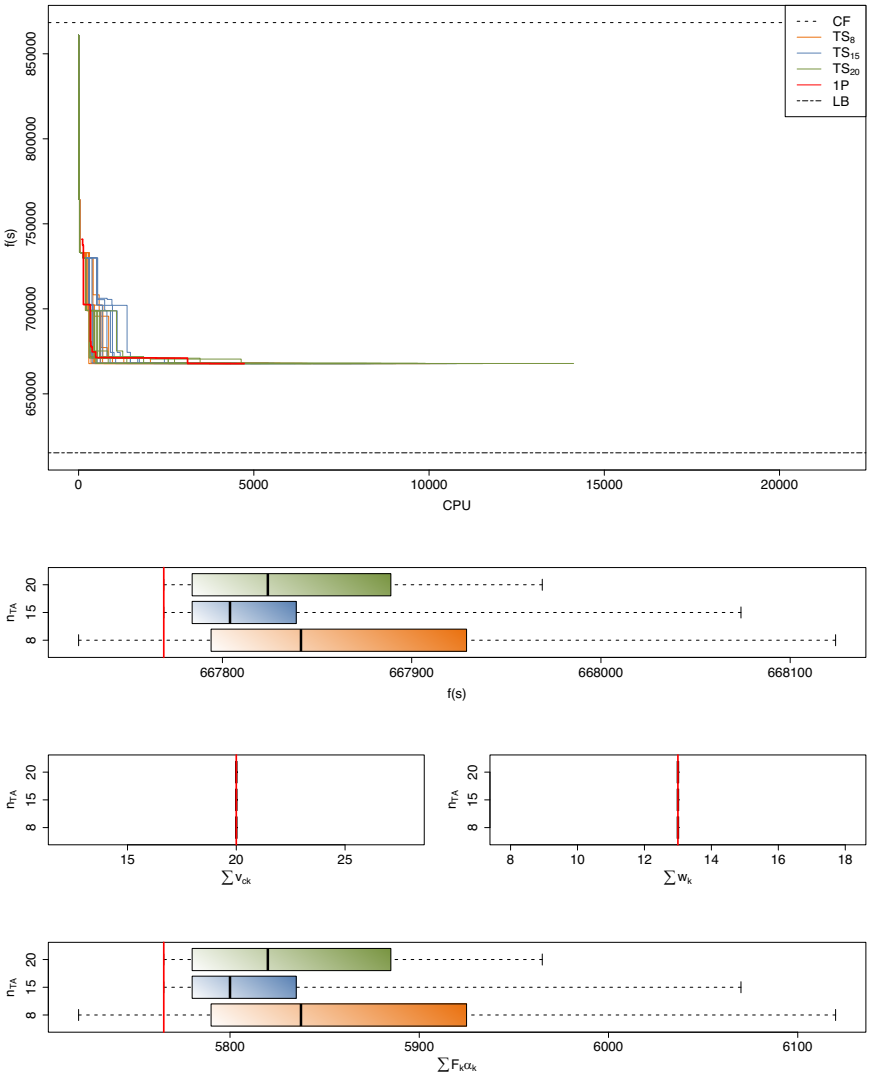

Figure F.10: Statistics related to instance 10

Appendix G

Solutions of computational experiments

This appendix contains detailed information about the results of the computational experiments discussed in Chapter 9. Table G.1 first displays data related to the solutions implemented by the cement factory. For each instance, are presented the value of the objective function $f(\mathfrak{s})$, the lower bound estimated by CPLEX when solving the problem in one phase, the gap between this lower bound and the value of the objective function (in absolute and relative value), the total delivery time $\sum \alpha_k$, the total weighted delivery time $\sum F_k \alpha_k$, the total number of different vehicles used to satisfy all orders of a same customer $\sum v_{ck}$, and the number of vehicles used $\sum w_k$ to do the deliveries. Asterisks (*) denote a value related or compared with a solution obtained when solving the problem with CPLEX in one phase.

While Tables G.2 and G.3 show information about the results provided by the three phase method with $\theta = 0.60$ and $\theta = 0.30$, Tables G.4 and G.5 display data related to the solutions of the two phase method with $\theta = 1.00$ and $\theta = 0.50$. Compared to Table G.1, these tables contain for each instance additional columns related to the second (first) subproblem of the three phase method (two phase method) indicating the number of rows, columns and nonzeros coefficients of the associated integer linear program, the time units (in second) required to solve

the problem (CPU) and to find the best feasible solution ($\text{CPU}_{\mathfrak{s}}$), and the relative value of the gap between the best feasible solution found and the estimated lower bound (Gap) for the MILP. The same columns are then displayed for Table G.6, which shows information about the one phase method. However, there is one column less for the gap to avoid data redundancy.

Finally, Tables G.7, G.8 and G.9 present information about the solutions of the tabu search method respectively with n_{TA} = 8, 15 and 20. Note that 10 results are displayed for each instance. Compared to Table G.1, additional columns indicate the time units (in second) and the number of iterations needed to solve the problem (respectively CPU and $iter$) and to find the best feasible solution (respectively $\text{CPU}_{\mathfrak{s}}$ and $iter_{\mathfrak{s}}$).

Table G.1: Solutions of the cement factory

Instance	$f(\mathfrak{s})$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	761'013.00	632'442.98	128'570.02	20.33%	5'940	5'940	25	13
2	1'014'254.00	668'029.03	346'224.97	51.83%	7'430	7'430	32	16
3	1'054'268.00	791'983.15	262'284.85	33.12%	7'445	7'445	33	16
4	864'967.50	756'504.31	108'463.19	14.34%	7'260	7'260	27	15
5	1'055'070.50	886'068.38	169'002.12	19.07%	8'180	8'180	32	17
6	1'224'091.50	865'115.58	358'975.92	41.49%	7'485	7'485	38	17
7	978'875.50	781'221.77	197'653.73	25.30%	7'195	7'195	31	17
8	1'055'727.50	760'340.69	295'386.81	38.85%	6'920	6'920	33	17
9	1'145'193.00	813'818.84	331'374.16	40.72%	7'860	7'860	35	17
10	868'356.00	615'317.77	253'038.23	41.12%	7'440	7'440	26	17

Table G.2: Solutions with MILP in three phases ($\theta = 0.60$)

Instance	Rows	Columns	Non zeros	CPU	$\text{CPU}_{\mathfrak{s}}$	$f(\mathfrak{s})$	Lower bound*	Gap*	Gap* (%)	Gap (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	765	522	2'408	3	3	718'175.00	632'442.98	85'732.02	13.56%	0.00%	5'600	14'780	23	15
2	1'172	817	3'500	1'963	22	793'087.00	668'029.03	125'057.97	18.72%	0.00%	6'985	19'990	24	18
3	1'302	850	3'892	2	2	859'013.50	791'983.15	67'030.35	8.46%	0.00%	7'360	20'950	26	17
4	1'051	690	3'170	301	164	886'286.00	756'504.31	129'781.69	17.16%	0.00%	7'085	19'550	27	18
5	1'289	932	4'021	64	29	1'052'105.00	886'068.38	166'036.62	18.74%	0.00%	9'005	26'960	31	20
6	1'453	1'018	4'465	23	14	904'460.50	865'115.58	39'344.92	4.55%	0.00%	7'345	18'820	27	19
7	1'107	785	3'516	6	3	878'623.50	781'221.77	97'401.73	12.47%	0.00%	7'080	19'860	27	19
8	1'002	778	3'245	158	21	873'820.00	760'340.69	113'479.31	14.92%	0.00%	6'235	9'385	27	16
9	1'222	900	3'896	4'861	349	1'007'876.00	813'818.84	194'057.16	23.85%	0.00%	7'955	19'295	30	19
10	1'019	733	3'050	14	6	758'207.00	615'317.77	142'889.23	23.22%	0.00%	7'295	15'395	22	19

Table G.3: Solutions with MILP in three phases ($\theta = 0.30$)

Instance	Rows	Columns	Non zeros	CPU	$CPU_{\mathfrak{s}}$	$f(\mathfrak{s})$	Lower bound*	Gap*	Gap* (%)	Gap (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	793	549	2'526	299	3	696'921.00	632'442.98	64'478.02	10.20%	0.00%	5'010	5'010	23	11
2	1'241	843	3'662	3'717	70	734'429.50	668'029.03	66'400.47	9.94%	0.00%	6'280	6'280	23	13
3	1'364	881	4'016	121	41	803'706.00	791'983.15	11'722.85	1.48%	0.00%	6'345	11'025	25	12
4	1'085	707	3'238	5	5	767'888.50	756'504.31	11'384.19	1.50%	0.00%	6'485	6'485	24	13
5	1'428	981	4'325	78	20	958'234.00	886'068.38	72'165.62	8.14%	0.00%	7'645	7'645	29	16
6	1'552	1'069	4'694	100	8	876'778.00	865'115.58	11'662.42	1.35%	0.00%	6'460	6'460	27	14
7	1'254	862	3'870	21'600	5	793'512.50	781'221.77	12'290.73	1.57%	0.00%	6'065	6'065	25	15
8	1'146	838	3'575	21'600	18	797'785.00	760'340.69	37'444.31	4.92%	0.02%	5'890	5'890	25	12
9	1'366	981	4'244	4'725	408	917'656.00	813'818.84	103'837.16	12.76%	0.32%	6'550	6'550	28	14
10	1'202	811	3'517	48	42	675'447.00	615'317.77	60'129.23	9.77%	0.00%	6'150	10'335	20	14

Table G.4: Solutions with MILP in two phases ($\theta = 1.00$)

Instance	Rows	Columns	Non zeros	CPU	$CPU_{\mathfrak{s}}$	$f(\mathfrak{s})$	Lower bound*	Gap*	Gap* (%)	Gap (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	780	519	2'441	17	8	706'313.00	632'442.98	73'870.02	11.68%	0.00%	5'060	8'660	23	13
2	1'223	842	3'703	21'600	29	784'405.50	668'029.03	116'376.47	17.42%	0.16%	6'790	14'305	24	17
3	1'364	881	4'120	5	5	818'808.00	791'983.15	26'824.85	3.39%	0.00%	7'095	14'025	25	16
4	1'085	707	3'308	16	15	809'127.50	756'504.31	52'623.19	6.96%	0.00%	6'795	11'610	25	15
5	1'408	971	4'385	21'600	109	1'006'653.50	886'068.38	120'585.12	13.61%	0.03%	8'615	15'680	30	19
6	1'550	1'068	4'792	12	11	900'385.50	865'115.58	35'269.92	4.08%	0.00%	7'320	14'745	27	19
7	1'228	844	3'913	5	5	836'835.50	781'221.77	55'613.73	7.12%	0.00%	6'620	13'730	26	17
8	1'127	828	3'596	409	26	867'627.50	760'340.69	107'286.81	14.11%	0.00%	6'215	6'215	27	15
9	1'352	971	4'307	22'103	21'634	960'933.00	813'818.84	147'114.16	18.08%	0.13%	7'335	9'540	29	17
10	1'110	794	3'420	258	45	677'943.00	615'317.77	62'625.23	10.18%	0.00%	6'615	6'615	20	16

Table G.5: Solutions with MILP in two phases ($\theta = 0.50$)

Instance	Rows	Columns	Non zeros	CPU	$\text{CPU}_{\mathfrak{s}}$	$f(\mathfrak{s})$	Lower bound*	Gap*	Gap* (%)	Gap (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	788	547	2'538	295	48	696'961.00	632'442.98	64'518.02	10.20%	0.00%	5'050	5'050	23	11
2	1'223	842	3'703	787	172	737'501.00	668'029.03	69'471.97	10.40%	0.00%	6'355	6'355	23	14
3	1'364	881	4'120	1'022	72	806'071.50	791'983.15	14'088.35	1.78%	0.00%	6'540	10'365	25	13
4	1'085	707	3'308	10'789	7'130	767'548.50	756'504.31	11'044.19	1.46%	0.00%	6'145	6'145	24	13
5	1'428	981	4'437	113	8	927'384.00	886'068.38	41'315.62	4.66%	0.00%	7'860	7'860	28	16
6	1'552	1'069	4'797	15'153	14'975	880'067.50	865'115.58	14'951.92	1.73%	0.00%	6'685	6'685	27	15
7	1'253	861	3'964	5	6	793'497.50	781'221.77	12'275.73	1.57%	0.00%	6'050	6'050	25	15
8	1'160	846	3'689	115	31	797'615.00	760'340.69	37'274.31	4.90%	0.00%	5'720	5'720	25	12
9	1'368	982	4'336	2'536	31	855'586.00	813'818.84	41'767.16	5.13%	0.00%	6'460	6'460	26	14
10	1'152	806	3'496	598	598	667'844.00	615'317.77	52'526.23	8.54%	0.00%	5'840	5'840	20	13

Table G.6: Solutions with MILP in one phase

Instance	Rows	Columns	Non zeros	CPU	$\text{CPU}_{\mathfrak{s}}$	$f(\mathfrak{s})$	Lower bound	Gap	Gap (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	8'525	7'933	47'390	17'655	8'181	636'155.00	632'442.98	3'712.02	0.59%	4'535	4'535	21	10
2	14'169	13'393	80'320	21'600	6'988	673'979.50	668'029.03	5'950.47	0.89%	5'760	5'760	21	13
3	16'776	16'055	97'260	6'805	5'216	832'186.50	791'983.15	40'203.35	5.08%	6'225	6'225	26	13
4	11'298	10'780	63'678	6'226	6'097	771'258.50	756'504.31	14'754.19	1.95%	5'940	9'855	24	13
5	16'559	15'558	93'478	9'978	6'357	961'490.50	886'068.38	75'422.12	8.51%	7'795	7'795	29	17
6	20'520	19'416	117'846	15'316	10'513	876'558.00	865'115.58	11'442.42	1.32%	6'240	6'240	27	14
7	14'399	13'187	80'026	18'535	4'079	787'294.50	781'221.77	6'072.73	0.78%	5'790	5'790	25	13
8	13'491	12'895	77'444	5'296	5'075	770'167.50	760'340.69	9'826.81	1.29%	5'475	5'475	24	13
9	16'819	15'690	94'823	18'356	12'436	826'112.00	813'818.84	12'293.16	1.51%	6'215	11'075	25	13
10	8'855	8'328	47'964	4'726	4'332	667'769.00	615'317.77	52'451.23	8.52%	5'765	5'765	20	13

Table G.7: Solutions with tabu search method ($TA = 8$)

Instance	CPU	CPU_s	$iter$	$iter_s$	$f(s)$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	11'895	2'792	12'873	2'873	636'140.00	632'442.98	3'697.02	0.58%	4'520	4'520	21	10
1	19'165	11'072	22'378	12'378	636'140.00	632'442.98	3'697.02	0.58%	4'520	4'520	21	10
1	10'395	2'521	13'128	3'128	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	15'829	8'448	20'367	10'367	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	8'632	830	11'094	1'094	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	8'205	1'011	11'168	1'168	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	14'334	6'365	17'665	7'656	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	21'117	13'395	26'523	16'523	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	15'281	7'179	18'794	8'794	636'250.00	632'442.98	3'807.02	0.60%	4'630	4'630	21	10
1	9'274	865	11'159	1'159	636'295.00	632'442.98	3'852.02	0.61%	4'675	4'675	21	10
2	20'919	3'402	11'862	1'862	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	21'600	9'702	13'278	5'786	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	21'600	13'796	13'061	8'467	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	21'600	15'402	12'274	8'676	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	21'600	15'458	13'971	9'876	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	21'600	18'316	13'008	10'933	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	21'600	16'675	12'386	9'446	674'658.00	668'029.03	6'628.97	0.99%	5'700	9'435	21	12
2	21'600	11'035	13'097	6'586	676'609.50	668'029.03	8'580.47	1.28%	5'780	8'390	21	13
2	16'291	1'127	10'542	542	677'241.00	668'029.03	9'211.97	1.38%	6'025	6'025	21	14
2	21'600	14'547	13'302	8'917	678'354.50	668'029.03	10'325.47	1.55%	5'815	10'135	21	13
3	21'600	2'647	10'034	1'250	803'096.00	791'983.15	11'112.85	1.40%	6'140	10'415	25	12
3	21'600	18'656	10'035	8'735	828'806.00	791'983.15	36'822.85	4.65%	5'870	5'870	26	12
3	21'600	5'271	8'755	2'253	828'841.00	791'983.15	36'857.85	4.65%	5'905	5'905	26	12
3	21'600	14'729	10'054	7'074	828'856.00	791'983.15	36'872.85	4.66%	5'920	5'920	26	12
3	21'600	17'388	10'330	8'214	828'866.00	791'983.15	36'882.85	4.66%	5'930	5'930	26	12
3	21'600	2'795	10'138	1'267	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
3	21'600	15'179	9'890	7'037	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
3	21'600	4'700	9'910	2'158	828'906.00	791'983.15	36'922.85	4.66%	5'970	5'970	26	12
3	21'600	6'727	9'970	3'031	828'921.00	791'983.15	36'937.85	4.66%	5'985	5'985	26	12
3	21'600	1'822	9'958	824	828'986.00	791'983.15	37'002.85	4.67%	6'050	6'050	26	12
4	21'600	17'848	20'789	16'954	760'944.50	756'504.31	4'440.19	0.59%	5'560	5'560	24	11
4	15'191	5'764	15'617	5'617	761'004.50	756'504.31	4'500.19	0.59%	5'620	5'620	24	11

continued on next page

Table G.7: Solutions with tabu search method ($TA = 8$), continued from previous page

Instance	CPU	CPU_5	$iter$	$iter_5$	$f(s)$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
4	15'549	5'945	15'821	5'821	761'069.50	756'504.31	4'565.19	0.60%	5'685	5'685	24	11
4	21'600	16'401	21'764	16'371	761'069.50	756'504.31	4'565.19	0.60%	5'685	5'685	24	11
4	21'587	10'712	20'103	10'103	761'124.50	756'504.31	4'620.19	0.61%	5'740	5'740	24	11
4	21'600	11'913	21'322	11'464	763'964.00	756'504.31	7'459.69	0.99%	5'570	5'570	24	12
4	21'600	13'726	18'919	11'917	763'994.00	756'504.31	7'489.69	0.99%	5'600	5'600	24	12
4	21'088	11'589	20'860	10'860	764'029.00	756'504.31	7'524.69	0.99%	5'635	5'635	24	12
4	16'117	5'576	15'189	5'189	764'139.00	756'504.31	7'634.69	1.01%	5'745	5'745	24	12
4	14'105	2'883	12'681	2'681	764'474.00	756'504.31	7'969.69	1.05%	6'080	6'080	24	12
5	21'600	7'967	13'140	5'030	927'932.50	886'068.38	41'864.12	4.72%	7'240	11'515	28	15
5	21'600	15'746	13'331	9'652	928'007.50	886'068.38	41'939.12	4.73%	7'315	11'590	28	15
5	19'021	3'714	12'296	2'296	928'187.50	886'068.38	42'119.12	4.75%	7'495	11'770	28	15
5	21'600	21'362	13'182	12'995	931'244.00	886'068.38	45'175.62	5.10%	7'445	11'720	28	16
5	20'081	3'091	12'024	2'024	931'649.00	886'068.38	45'580.62	5.14%	7'850	12'125	28	16
5	21'600	20'994	13'446	13'035	931'719.00	886'068.38	45'650.62	5.15%	7'470	12'195	28	16
5	21'600	9'717	12'848	5'976	954'742.50	886'068.38	68'674.12	7.75%	7'260	7'260	29	15
5	21'600	9'569	12'004	5'441	957'127.50	886'068.38	71'059.12	8.02%	7'035	9'645	29	15
5	21'600	19'853	12'627	11'573	957'989.00	886'068.38	71'920.62	8.12%	7'400	7'400	29	16
5	21'600	6'113	13'133	3'805	957'994.00	886'068.38	71'925.62	8.12%	7'405	7'405	29	16
6	21'600	20'846	8'489	8'197	870'124.00	865'115.58	5'008.42	0.58%	5'935	5'935	27	12
6	21'600	17'954	8'407	7'006	873'193.50	865'115.58	8'077.92	0.93%	5'940	5'940	27	13
6	21'600	3'173	8'846	1'387	873'283.50	865'115.58	8'167.92	0.94%	6'030	6'030	27	13
6	21'595	18'062	7'240	6'082	873'363.50	865'115.58	8'247.92	0.95%	6'110	6'110	27	13
6	21'600	3'974	8'745	1'648	873'373.50	865'115.58	8'257.92	0.95%	6'120	6'120	27	13
6	21'600	11'087	8'575	4'208	873'378.50	865'115.58	8'262.92	0.96%	6'125	6'125	27	13
6	21'600	10'899	8'570	4'346	873'423.50	865'115.58	8'307.92	0.96%	6'170	6'170	27	13
6	21'600	6'766	8'481	2'768	873'493.50	865'115.58	8'377.92	0.97%	6'240	6'240	27	13
6	21'600	17'883	8'640	7'278	875'948.50	865'115.58	10'832.92	1.25%	6'085	8'695	27	13
6	21'600	6'723	8'527	2'827	876'343.00	865'115.58	11'227.42	1.30%	6'025	6'025	27	14
7	14'444	4'192	13'895	3'895	787'269.50	781'221.77	6'047.73	0.77%	5'765	5'765	25	13
7	21'600	17'214	20'365	16'167	787'304.50	781'221.77	6'082.73	0.78%	5'800	5'800	25	13
7	21'600	15'073	20'754	14'473	787'324.50	781'221.77	6'102.73	0.78%	5'820	5'820	25	13
7	13'646	3'038	12'945	2'945	787'329.50	781'221.77	6'107.73	0.78%	5'825	5'825	25	13
7	21'600	11'759	20'554	10'969	790'151.00	781'221.77	8'929.23	1.14%	5'675	5'675	25	14
7	11'826	1'158	11'133	1'133	790'191.00	781'221.77	8'969.23	1.15%	5'715	5'715	25	14
7	17'832	7'185	16'824	6'824	790'226.00	781'221.77	9'004.23	1.15%	5'750	5'750	25	14

continued on next page

Table G.7: Solutions with tabu search method (TA = 8), continued from previous page

Instance	CPU	$CPU_{\mathfrak{s}}$	$iter$	$iter_{\mathfrak{s}}$	$f(\mathfrak{s})$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
7	21'600	18'278	18'336	15'459	790'226.00	781'221.77	9'004.23	1.15%	5'750	5'750	25	14
7	16'822	5'975	15'577	5'577	790'241.00	781'221.77	9'019.23	1.15%	5'765	5'765	25	14
7	11'799	1'274	11'268	1'268	790'256.00	781'221.77	9'034.23	1.16%	5'780	5'780	25	14
8	18'712	10'590	22'575	12'575	766'910.00	760'340.69	6'569.31	0.86%	5'240	5'240	24	12
8	12'576	4'246	15'113	5'113	766'915.00	760'340.69	6'574.31	0.86%	5'245	5'245	24	12
8	15'121	6'933	18'226	8'226	766'930.00	760'340.69	6'589.31	0.87%	5'260	5'260	24	12
8	14'695	5'656	15'984	5'984	766'945.00	760'340.69	6'604.31	0.87%	5'275	5'275	24	12
8	10'543	2'467	13'009	3'009	766'955.00	760'340.69	6'614.31	0.87%	5'285	5'285	24	12
8	13'638	5'800	17'079	7'079	766'955.00	760'340.69	6'614.31	0.87%	5'285	5'285	24	12
8	12'503	3'818	14'598	4'598	766'965.00	760'340.69	6'624.31	0.87%	5'295	5'295	24	12
8	14'763	6'446	17'780	7'780	766'965.00	760'340.69	6'624.31	0.87%	5'295	5'295	24	12
8	9'965	1'626	12'095	2'095	766'990.00	760'340.69	6'649.31	0.87%	5'320	5'320	24	12
8	13'307	5'170	16'335	6'335	766'990.00	760'340.69	6'649.31	0.87%	5'320	5'320	24	12
9	21'600	11'657	15'248	8'203	821'157.00	813'818.84	7'338.16	0.90%	6'120	6'120	25	13
9	21'600	13'243	13'895	8'641	821'182.00	813'818.84	7'363.16	0.90%	6'145	6'145	25	13
9	21'600	10'369	15'468	7'377	821'202.00	813'818.84	7'383.16	0.91%	6'165	6'165	25	13
9	21'600	15'115	15'702	11'068	821'207.00	813'818.84	7'388.16	0.91%	6'170	6'170	25	13
9	18'249	4'376	13'386	3'386	821'212.00	813'818.84	7'393.16	0.91%	6'175	6'175	25	13
9	21'600	20'590	15'191	14'726	821'272.00	813'818.84	7'453.16	0.92%	6'235	6'235	25	13
9	21'600	17'660	16'041	13'202	821'277.00	813'818.84	7'458.16	0.92%	6'240	6'240	25	13
9	21'600	8'488	15'523	6'067	821'297.00	813'818.84	7'478.16	0.92%	6'260	6'260	25	13
9	21'347	7'549	15'458	5'458	821'372.00	813'818.84	7'553.16	0.93%	6'335	6'335	25	13
9	21'600	20'591	15'647	14'935	821'382.00	813'818.84	7'563.16	0.93%	6'345	6'345	25	13
10	4'720	719	11'732	1'732	667'724.00	615'317.77	52'406.23	8.52%	5'720	5'720	20	13
10	4'281	292	10'672	672	667'769.00	615'317.77	52'451.23	8.52%	5'765	5'765	20	13
10	10'034	5'946	24'591	14'591	667'794.00	615'317.77	52'476.23	8.53%	5'790	5'790	20	13
10	10'508	6'384	25'852	15'852	667'799.00	615'317.77	52'481.23	8.53%	5'795	5'795	20	13
10	6'058	2'050	14'931	4'931	667'839.00	615'317.77	52'521.23	8.54%	5'835	5'835	20	13
10	10'744	6'528	25'858	15'858	667'844.00	615'317.77	52'526.23	8.54%	5'840	5'840	20	13
10	4'928	831	12'042	2'042	667'859.00	615'317.77	52'541.23	8.54%	5'855	5'855	20	13
10	5'082	965	12'238	2'238	667'929.00	615'317.77	52'611.23	8.55%	5'925	5'925	20	13
10	5'753	1'849	14'654	4'654	668'004.00	615'317.77	52'686.23	8.56%	6'000	6'000	20	13
10	5'414	1'290	13'196	3'196	668'124.00	615'317.77	52'806.23	8.58%	6'120	6'120	20	13

Table G.8: Solutions with tabu search method ($TA = 15$)

Instance	CPU	CPU_s	$iter$	$iter_s$	$f(s)$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	7'224	465	10'641	641	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	7'583	467	10'641	641	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	7'945	488	10'665	665	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	7'669	650	10'957	957	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	11'082	3'796	15'382	5'382	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	8'736	831	11'141	1'141	636'155.00	632'442.98	3'712.02	0.59%	4'535	4'535	21	10
1	12'453	5'160	17'634	7'634	636'155.00	632'442.98	3'712.02	0.59%	4'535	4'535	21	10
1	7'708	748	11'060	1'060	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	12'842	5'934	17'881	7'881	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	14'529	7'296	21'085	11'085	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
2	21'600	20'839	14'495	13'923	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	21'600	16'641	14'563	11'503	674'174.50	668'029.03	6'145.47	0.92%	5'955	5'955	21	13
2	21'600	8'710	13'628	5'616	677'186.00	668'029.03	9'156.97	1.37%	5'970	5'970	21	14
2	21'600	11'218	14'119	7'513	677'186.00	668'029.03	9'156.97	1.37%	5'970	5'970	21	14
2	21'600	12'112	13'907	7'782	677'191.00	668'029.03	9'161.97	1.37%	5'975	5'975	21	14
2	21'600	17'469	13'370	10'799	677'206.00	668'029.03	9'176.97	1.37%	5'990	5'990	21	14
2	21'600	12'323	13'938	7'980	678'169.50	668'029.03	10'140.47	1.52%	5'810	9'950	21	13
2	21'600	18'376	12'712	10'781	678'924.50	668'029.03	10'895.47	1.63%	5'890	10'705	21	13
2	21'600	6'620	13'268	3'993	679'691.00	668'029.03	11'661.97	1.75%	5'865	8'475	21	14
2	21'600	19'123	13'326	11'835	682'689.50	668'029.03	14'660.47	2.19%	5'830	14'470	21	13
3	21'600	17'811	11'224	9'144	798'726.00	791'983.15	6'742.85	0.85%	6'045	6'045	25	12
3	21'600	20'129	10'969	10'078	798'766.00	791'983.15	6'782.85	0.86%	6'085	6'085	25	12
3	21'600	10'429	11'047	5'206	802'961.00	791'983.15	10'977.85	1.39%	6'005	10'280	25	12
3	21'600	17'307	10'914	8'677	828'821.00	791'983.15	36'837.85	4.65%	5'885	5'885	26	12
3	21'600	2'149	10'528	1'080	828'841.00	791'983.15	36'857.85	4.65%	5'905	5'905	26	12
3	21'600	1'646	10'588	808	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
3	21'600	1'713	10'182	835	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
3	21'600	3'506	10'417	1'720	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
3	20'649	575	10'268	268	828'886.00	791'983.15	36'902.85	4.66%	5'950	5'950	26	12
3	21'545	576	10'268	298	828'886.00	791'983.15	36'902.85	4.66%	5'950	5'950	26	12
4	14'481	4'006	13'741	3'741	760'944.50	756'504.31	4'440.19	0.59%	5'560	5'560	24	11
4	21'600	21'409	22'506	22'299	761'004.50	756'504.31	4'500.19	0.59%	5'620	5'620	24	11

continued on next page

Table G.8: Solutions with tabu search method ($TA = 15$), continued from previous page

Instance	CPU	CPU_5	$iter$	$iter_5$	$f(s)$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
4	21'600	12'840	22'309	13'121	761'049.50	756'504.31	4'545.19	0.60%	5'665	5'665	24	11
4	15'612	5'137	14'937	4'937	761'094.50	756'504.31	4'590.19	0.61%	5'710	5'710	24	11
4	9'987	335	10'332	332	764'069.00	756'504.31	7'564.69	1.00%	5'675	5'675	24	12
4	10'390	399	10'396	396	764'069.00	756'504.31	7'564.69	1.00%	5'675	5'675	24	12
4	14'777	4'330	14'338	4'338	764'074.00	756'504.31	7'569.69	1.00%	5'680	5'680	24	12
4	10'909	1'162	11'209	1'209	764'134.00	756'504.31	7'629.69	1.01%	5'740	5'740	24	12
4	16'554	6'606	16'267	6'267	764'174.00	756'504.31	7'669.69	1.01%	5'780	5'780	24	12
4	10'927	884	10'902	902	764'514.00	756'504.31	8'009.69	1.06%	6'120	6'120	24	12
5	21'600	9'390	13'898	6'074	927'219.00	886'068.38	41'150.62	4.64%	7'695	7'695	28	16
5	19'808	4'020	12'521	2'521	927'827.50	886'068.38	41'759.12	4.71%	7'090	11'410	28	15
5	21'600	21'568	13'814	13'791	927'932.50	886'068.38	41'864.12	4.72%	7'240	11'515	28	15
5	21'600	7'994	13'505	5'274	928'567.50	886'068.38	42'499.12	4.80%	7'470	12'150	28	15
5	21'600	21'440	13'855	13'714	928'799.00	886'068.38	42'730.62	4.82%	7'565	9'275	28	16
5	21'600	8'447	13'717	5'170	930'969.00	886'068.38	44'900.62	5.07%	7'620	11'445	28	16
5	21'600	11'301	13'795	6'994	931'159.00	886'068.38	45'090.62	5.09%	7'360	11'635	28	16
5	21'600	11'496	13'466	7'270	931'787.50	886'068.38	45'719.12	5.16%	7'270	15'370	28	15
5	21'600	15'307	14'215	10'026	954'832.50	886'068.38	68'764.12	7.76%	7'350	7'350	29	15
5	17'580	1'357	10'837	837	958'457.50	886'068.38	72'389.12	8.17%	7'330	10'975	29	15
6	21'600	7'954	8'685	3'184	873'168.50	865'115.58	8'052.92	0.93%	5'915	5'915	27	13
6	21'600	7'421	9'388	3'095	873'183.50	865'115.58	8'067.92	0.93%	5'930	5'930	27	13
6	21'600	4'372	8'866	1'905	873'193.50	865'115.58	8'077.92	0.93%	5'940	5'940	27	13
6	21'600	3'423	8'963	1'525	873'228.50	865'115.58	8'112.92	0.94%	5'975	5'975	27	13
6	21'600	14'008	8'935	5'698	873'233.50	865'115.58	8'117.92	0.94%	5'980	5'980	27	13
6	21'600	15'045	9'235	6'469	873'283.50	865'115.58	8'167.92	0.94%	6'030	6'030	27	13
6	21'600	9'590	8'491	3'770	873'303.50	865'115.58	8'187.92	0.95%	6'050	6'050	27	13
6	21'600	5'470	9'196	2'382	873'378.50	865'115.58	8'262.92	0.96%	6'125	6'125	27	13
6	21'600	14'863	8'701	6'041	873'588.50	865'115.58	8'472.92	0.98%	6'335	6'335	27	13
6	21'600	14'502	8'708	5'875	876'358.00	865'115.58	11'242.42	1.30%	6'040	6'040	27	14
7	16'122	5'968	15'848	5'848	787'129.50	781'221.77	5'907.73	0.76%	5'625	5'625	25	13
7	21'600	15'175	21'053	14'855	787'224.50	781'221.77	6'002.73	0.77%	5'720	5'720	25	13
7	21'600	12'385	21'112	12'186	787'269.50	781'221.77	6'047.73	0.77%	5'765	5'765	25	13
7	21'600	21'522	21'147	21'065	790'036.00	781'221.77	8'814.23	1.13%	5'560	5'560	25	14
7	19'957	9'869	19'860	9'860	790'106.00	781'221.77	8'884.23	1.14%	5'630	5'630	25	14
7	19'321	9'076	18'859	8'859	790'166.00	781'221.77	8'944.23	1.14%	5'690	5'690	25	14
7	21'600	12'548	21'076	12'371	790'166.00	781'221.77	8'944.23	1.14%	5'690	5'690	25	14

continued on next page

Table G.8: Solutions with tabu search method ($TA = 15$), continued from previous page

Instance	CPU	$CPU_{\mathfrak{s}}$	$iter$	$iter_{\mathfrak{s}}$	$f(\mathfrak{s})$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
7	13'115	2'931	12'929	2'929	790'201.00	781'221.77	8'979.23	1.15%	5'725	5'725	25	14
7	15'876	5'709	15'726	5'726	790'261.00	781'221.77	9'039.23	1.16%	5'785	5'785	25	14
7	12'479	2'427	12'460	2'460	790'266.00	781'221.77	9'044.23	1.16%	5'790	5'790	25	14
8	20'869	12'724	25'482	15'482	766'915.00	760'340.69	6'574.31	0.86%	5'245	5'245	24	12
8	9'025	1'087	11'433	1'433	766'925.00	760'340.69	6'584.31	0.87%	5'255	5'255	24	12
8	10'537	2'498	13'129	3'129	766'935.00	760'340.69	6'594.31	0.87%	5'265	5'265	24	12
8	13'038	4'386	15'321	5'321	766'955.00	760'340.69	6'614.31	0.87%	5'285	5'285	24	12
8	16'306	8'244	20'373	10'373	766'955.00	760'340.69	6'614.31	0.87%	5'285	5'285	24	12
8	9'202	1'228	11'504	1'504	766'965.00	760'340.69	6'624.31	0.87%	5'295	5'295	24	12
8	12'566	4'009	15'073	5'073	766'965.00	760'340.69	6'624.31	0.87%	5'295	5'295	24	12
8	10'396	2'585	13'472	3'472	766'970.00	760'340.69	6'629.31	0.87%	5'300	5'300	24	12
8	21'600	16'877	26'638	20'844	766'975.00	760'340.69	6'634.31	0.87%	5'305	5'305	24	12
8	11'582	3'324	14'200	4'200	766'990.00	760'340.69	6'649.31	0.87%	5'320	5'320	24	12
9	21'600	9'958	16'466	7'780	821'072.00	813'818.84	7'253.16	0.89%	6'035	6'035	25	13
9	21'600	11'617	16'605	9'023	821'127.00	813'818.84	7'308.16	0.90%	6'090	6'090	25	13
9	21'600	15'622	15'925	11'354	821'152.00	813'818.84	7'333.16	0.90%	6'115	6'115	25	13
9	21'600	11'014	16'528	8'551	821'162.00	813'818.84	7'343.16	0.90%	6'125	6'125	25	13
9	21'600	10'110	15'840	7'543	821'187.00	813'818.84	7'368.16	0.91%	6'150	6'150	25	13
9	21'600	15'732	16'370	11'906	821'212.00	813'818.84	7'393.16	0.91%	6'175	6'175	25	13
9	21'600	21'503	16'006	15'929	821'217.00	813'818.84	7'398.16	0.91%	6'180	6'180	25	13
9	16'209	2'736	12'084	2'084	821'232.00	813'818.84	7'413.16	0.91%	6'195	6'195	25	13
9	16'601	3'254	12'313	2'313	821'282.00	813'818.84	7'463.16	0.92%	6'245	6'245	25	13
9	20'337	6'681	15'175	5'175	821'297.00	813'818.84	7'478.16	0.92%	6'260	6'260	25	13
10	6'177	2'142	15'360	5'360	667'769.00	615'317.77	52'451.23	8.52%	5'765	5'765	20	13
10	10'787	6'694	26'583	16'583	667'769.00	615'317.77	52'451.23	8.52%	5'765	5'765	20	13
10	5'188	1'057	12'445	2'445	667'784.00	615'317.77	52'466.23	8.53%	5'780	5'780	20	13
10	8'328	4'191	20'483	10'483	667'794.00	615'317.77	52'476.23	8.53%	5'790	5'790	20	13
10	8'179	4'110	19'942	9'942	667'799.00	615'317.77	52'481.23	8.53%	5'795	5'795	20	13
10	8'427	4'445	20'973	10'973	667'809.00	615'317.77	52'491.23	8.53%	5'805	5'805	20	13
10	8'976	4'890	22'248	12'248	667'809.00	615'317.77	52'491.23	8.53%	5'805	5'805	20	13
10	6'734	2'695	16'468	6'468	667'839.00	615'317.77	52'521.23	8.54%	5'835	5'835	20	13
10	5'580	1'477	13'619	3'619	668'004.00	615'317.77	52'686.23	8.56%	6'000	6'000	20	13
10	4'505	386	10'886	886	668'074.00	615'317.77	52'756.23	8.57%	6'070	6'070	20	13

Table G.9: Solutions with tabu search method ($TA = 20$)

Instance	CPU	CPU_s	$iter$	$iter_s$	$f(s)$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
1	9'455	2'148	13'522	3'522	636'140.00	632'442.98	3'697.02	0.58%	4'520	4'520	21	10
1	10'998	3'958	16'159	6'159	636'140.00	632'442.98	3'697.02	0.58%	4'520	4'520	21	10
1	20'026	12'842	27'879	17'879	636'140.00	632'442.98	3'697.02	0.58%	4'520	4'520	21	10
1	7'123	245	10'392	392	636'150.00	632'442.98	3'707.02	0.59%	4'530	4'530	21	10
1	17'058	9'545	23'840	13'840	636'155.00	632'442.98	3'712.02	0.59%	4'535	4'535	21	10
1	7'834	678	11'010	1'010	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	8'537	1'358	11'956	1'956	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	9'773	2'762	14'328	4'328	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	11'526	4'064	15'438	5'438	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
1	14'144	7'422	20'134	10'134	636'160.00	632'442.98	3'717.02	0.59%	4'540	4'540	21	10
2	21'600	11'446	14'404	7'812	673'944.50	668'029.03	5'915.47	0.89%	5'725	5'725	21	13
2	21'600	17'047	14'684	11'196	673'944.50	668'029.03	5'915.47	0.89%	5'725	5'725	21	13
2	21'600	17'515	14'272	11'634	673'974.50	668'029.03	5'945.47	0.89%	5'755	5'755	21	13
2	16'282	1'018	10'563	563	674'224.50	668'029.03	6'195.47	0.93%	6'005	6'005	21	13
2	21'600	9'998	13'510	6'605	675'188.00	668'029.03	7'158.97	1.07%	5'780	9'965	21	12
2	21'600	15'063	13'895	9'767	675'464.50	668'029.03	7'435.47	1.11%	5'715	7'245	21	13
2	20'974	7'155	14'752	4'752	676'629.50	668'029.03	8'600.47	1.29%	5'800	8'410	21	13
2	19'124	2'483	11'495	1'495	678'234.50	668'029.03	10'205.47	1.53%	5'830	10'015	21	13
2	18'707	2'920	11'850	1'850	678'294.50	668'029.03	10'265.47	1.54%	5'890	10'075	21	13
2	21'600	16'245	13'509	9'941	703'959.50	668'029.03	35'930.47	5.38%	5'775	5'775	22	13
3	21'600	13'542	11'830	7'023	798'781.00	791'983.15	6'797.85	0.86%	6'100	6'100	25	12
3	21'600	15'518	10'568	7'538	802'926.00	791'983.15	10'942.85	1.38%	5'970	10'245	25	12
3	21'600	16'746	10'972	8'692	802'926.00	791'983.15	10'942.85	1.38%	5'970	10'245	25	12
3	21'600	21'314	10'694	10'525	802'981.00	791'983.15	10'997.85	1.39%	6'025	10'300	25	12
3	21'600	10'001	10'845	4'954	803'106.00	791'983.15	11'122.85	1.40%	6'150	10'425	25	12
3	21'600	17'508	10'763	8'702	806'206.50	791'983.15	14'223.35	1.80%	6'225	10'500	25	13
3	21'600	18'812	10'573	9'257	828'821.00	791'983.15	36'837.85	4.65%	5'885	5'885	26	12
3	21'600	3'743	10'941	1'841	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
3	21'600	13'982	10'452	6'750	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
3	21'600	15'292	10'380	7'430	828'871.00	791'983.15	36'887.85	4.66%	5'935	5'935	26	12
4	21'600	14'352	21'722	14'357	760'924.50	756'504.31	4'420.19	0.58%	5'540	5'540	24	11
4	15'929	6'160	16'192	6'192	760'944.50	756'504.31	4'440.19	0.59%	5'560	5'560	24	11

continued on next page

Table G.9: Solutions with tabu search method ($TA = 20$), continued from previous page

Instance	CPU	CPU_5	$iter$	$iter_5$	$f(s)$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
4	21'600	15'576	22'482	16'439	761'004.50	756'504.31	4'500.19	0.59%	5'620	5'620	24	11
4	21'600	17'498	22'458	18'051	761'014.50	756'504.31	4'510.19	0.60%	5'630	5'630	24	11
4	21'600	18'539	22'463	19'348	761'049.50	756'504.31	4'545.19	0.60%	5'665	5'665	24	11
4	21'600	19'117	21'870	19'329	763'954.00	756'504.31	7'449.69	0.98%	5'560	5'560	24	12
4	14'692	4'611	14'383	4'383	764'029.00	756'504.31	7'524.69	0.99%	5'635	5'635	24	12
4	13'884	4'362	14'710	4'710	764'089.00	756'504.31	7'584.69	1.00%	5'695	5'695	24	12
4	11'150	894	10'892	892	764'194.00	756'504.31	7'689.69	1.02%	5'800	5'800	24	12
4	19'365	8'877	19'076	9'076	764'209.00	756'504.31	7'704.69	1.02%	5'815	5'815	24	12
5	21'382	5'629	13'633	3'633	923'697.50	886'068.38	37'629.12	4.25%	7'280	7'280	28	15
5	21'600	6'795	13'956	4'176	926'984.00	886'068.38	40'915.62	4.62%	7'460	7'460	28	16
5	21'600	21'018	13'993	13'593	927'144.00	886'068.38	41'075.62	4.64%	7'620	7'620	28	16
5	21'600	19'219	14'043	12'408	927'517.50	886'068.38	41'449.12	4.68%	7'275	11'100	28	15
5	21'600	20'232	14'141	13'309	927'802.50	886'068.38	41'734.12	4.71%	7'110	11'385	28	15
5	21'600	12'593	14'320	8'485	927'977.50	886'068.38	41'909.12	4.73%	7'285	11'560	28	15
5	18'165	2'990	11'948	1'948	928'292.50	886'068.38	42'224.12	4.77%	7'195	11'875	28	15
5	18'150	2'765	11'749	1'749	930'814.00	886'068.38	44'745.62	5.05%	7'465	11'290	28	16
5	21'600	20'200	13'843	12'887	931'284.00	886'068.38	45'215.62	5.10%	7'485	11'760	28	16
5	21'600	11'090	13'945	7'190	964'270.50	886'068.38	78'202.12	8.83%	7'605	10'575	29	17
6	21'600	1'653	9'587	814	873'133.50	865'115.58	8'017.92	0.93%	5'880	5'880	27	13
6	21'600	5'907	8'821	2'430	873'288.50	865'115.58	8'172.92	0.94%	6'035	6'035	27	13
6	21'600	20'348	9'025	8'517	873'313.50	865'115.58	8'197.92	0.95%	6'060	6'060	27	13
6	21'600	18'048	9'068	7'639	873'323.50	865'115.58	8'207.92	0.95%	6'070	6'070	27	13
6	21'600	6'143	9'501	2'725	873'338.50	865'115.58	8'222.92	0.95%	6'085	6'085	27	13
6	21'600	16'784	8'954	7'036	873'383.50	865'115.58	8'267.92	0.96%	6'130	6'130	27	13
6	21'600	12'170	8'893	4'869	873'468.50	865'115.58	8'352.92	0.97%	6'215	6'215	27	13
6	21'600	4'770	9'365	2'134	873'518.50	865'115.58	8'402.92	0.97%	6'265	6'265	27	13
6	21'600	14'845	9'273	6'284	873'548.50	865'115.58	8'432.92	0.97%	6'295	6'295	27	13
6	21'600	13'093	8'633	5'332	873'733.50	865'115.58	8'617.92	1.00%	6'480	6'480	27	13
7	16'921	7'104	17'290	7'290	787'059.50	781'221.77	5'837.73	0.75%	5'555	5'555	25	13
7	20'427	10'367	20'328	10'328	787'139.50	781'221.77	5'917.73	0.76%	5'635	5'635	25	13
7	21'600	11'880	21'549	11'949	787'174.50	781'221.77	5'952.73	0.76%	5'670	5'670	25	13
7	19'050	8'926	18'940	8'940	790'151.00	781'221.77	8'929.23	1.14%	5'675	5'675	25	14
7	15'193	5'272	15'164	5'164	790'161.00	781'221.77	8'939.23	1.14%	5'685	5'685	25	14
7	12'023	1'862	11'913	1'913	790'166.00	781'221.77	8'944.23	1.14%	5'690	5'690	25	14
7	14'188	3'960	13'921	3'921	790'166.00	781'221.77	8'944.23	1.14%	5'690	5'690	25	14

continued on next page

Table G.9: Solutions with tabu search method ($TA = 20$), continued from previous page

Instance	CPU	CPU_5	$iter$	$iter_5$	$f(s)$	Lower bound*	Gap*	Gap* (%)	$\sum \alpha_k$	$\sum F_k \alpha_k$	$\sum v_{ck}$	$\sum w_k$
7	21'600	11'532	21'461	11'472	790'176.00	781'221.77	8'954.23	1.15%	5'700	5'700	25	14
7	16'748	6'887	16'865	6'865	790'191.00	781'221.77	8'969.23	1.15%	5'715	5'715	25	14
7	20'501	10'378	20'281	10'281	790'206.00	781'221.77	8'984.23	1.15%	5'730	5'730	25	14
8	13'281	5'212	16'661	6'661	763'997.50	760'340.69	3'656.81	0.48%	5'350	5'350	24	11
8	21'600	13'959	25'896	17'192	766'900.00	760'340.69	6'559.31	0.86%	5'230	5'230	24	12
8	21'335	13'433	27'004	17'004	766'905.00	760'340.69	6'564.31	0.86%	5'235	5'235	24	12
8	13'423	5'386	16'792	6'792	766'940.00	760'340.69	6'599.31	0.87%	5'270	5'270	24	12
8	21'600	14'237	26'460	17'348	766'940.00	760'340.69	6'599.31	0.87%	5'270	5'270	24	12
8	10'913	2'904	13'526	3'526	766'945.00	760'340.69	6'604.31	0.87%	5'275	5'275	24	12
8	15'209	7'031	19'309	9'309	766'955.00	760'340.69	6'614.31	0.87%	5'285	5'285	24	12
8	19'332	11'116	24'010	14'010	766'965.00	760'340.69	6'624.31	0.87%	5'295	5'295	24	12
8	10'166	2'085	12'671	2'671	766'980.00	760'340.69	6'639.31	0.87%	5'310	5'310	24	12
8	9'517	1'517	11'883	1'883	766'985.00	760'340.69	6'644.31	0.87%	5'315	5'315	24	12
9	21'600	16'348	16'386	12'323	821'072.00	813'818.84	7'253.16	0.89%	6'035	6'035	25	13
9	21'600	20'233	15'860	14'883	821'157.00	813'818.84	7'338.16	0.90%	6'120	6'120	25	13
9	21'600	13'112	16'051	9'657	821'182.00	813'818.84	7'363.16	0.90%	6'145	6'145	25	13
9	21'600	20'420	16'156	15'207	821'182.00	813'818.84	7'363.16	0.90%	6'145	6'145	25	13
9	18'799	4'934	13'788	3'788	821'212.00	813'818.84	7'393.16	0.91%	6'175	6'175	25	13
9	21'600	16'492	16'121	12'052	821'217.00	813'818.84	7'398.16	0.91%	6'180	6'180	25	13
9	18'584	4'656	13'567	3'567	821'262.00	813'818.84	7'443.16	0.91%	6'225	6'225	25	13
9	21'600	8'667	15'254	6'400	821'292.00	813'818.84	7'473.16	0.92%	6'255	6'255	25	13
9	18'735	4'988	13'801	3'801	821'297.00	813'818.84	7'478.16	0.92%	6'260	6'260	25	13
9	17'641	3'577	12'688	2'688	821'337.00	813'818.84	7'518.16	0.92%	6'300	6'300	25	13
10	9'056	4'902	21'918	11'918	667'769.00	615'317.77	52'451.23	8.52%	5'765	5'765	20	13
10	6'632	2'559	16'279	6'279	667'774.00	615'317.77	52'456.23	8.53%	5'770	5'770	20	13
10	7'780	3'660	19'131	9'131	667'784.00	615'317.77	52'466.23	8.53%	5'780	5'780	20	13
10	11'531	7'439	28'193	18'193	667'789.00	615'317.77	52'471.23	8.53%	5'785	5'785	20	13
10	5'450	1'376	13'300	3'300	667'809.00	615'317.77	52'491.23	8.53%	5'805	5'805	20	13
10	8'330	4'170	20'186	10'186	667'839.00	615'317.77	52'521.23	8.54%	5'835	5'835	20	13
10	14'125	9'896	34'049	24'049	667'859.00	615'317.77	52'541.23	8.54%	5'855	5'855	20	13
10	4'671	517	11'245	1'245	667'889.00	615'317.77	52'571.23	8.54%	5'885	5'885	20	13
10	9'669	5'571	23'630	13'630	667'964.00	615'317.77	52'646.23	8.56%	5'960	5'960	20	13
10	8'628	4'474	20'898	10'898	667'969.00	615'317.77	52'651.23	8.56%	5'965	5'965	20	13