A New Solution Approach for the Inventory Routing Problem: Using Vehicle Routing Problem Constructive Heuristics

Henri Thierry TOUTOUNJI

A THESIS SUBMITTED

FOR THE DEGREE MASTER OF ENGINEERING

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgements

I would like to thank Dr. Wikrom Jaruphongsa for his expertise, help and support throughout the writing of this thesis. His kindness and optimism created a very motivating work environment that made this thesis possible. I also thank Prof. Chew Ek Peng for the time and energy he devoted to my work, and for his regular feedbacks.

Warm thanks to Yurou Zhou, who helped me finalize this work by reading and commenting it.

I would like to express my gratitude to all my friends here who supported me during this work, especially Philippe Briat who accompanied me throughout this project.

Finally, I would like to thank my family, in Lebanon, Brussels and Paris, for providing the love and encouragement I needed to complete this Master.

Abstract

The Inventory Routing Problem (IRP) is an extension of the vehicle routing problem (VRP) that couples inventory control and routing decisions. This thesis studies an IRP where a warehouse replenishes several customers using a finite fleet of capacitated vehicles. Each customer faces a deterministic demand over a finite planning horizon, and has a finite capacity to keep local inventory. The goal is to minimize system-wide transportation costs over the planning horizon. Our main contribution lies in transforming this problem into an *equivalent* VRP with *fixed size orders*, in which split deliveries are allowed and orders must reach the customer between specified days. The transformation allows us to design a constructive heuristic inspired by the VRP literature. This heuristic was run on small instances, and provided solutions with a cost no more than 5.33% above optimum on average. On bigger instances, where no information is available on the optimum, our heuristic outperformed a myopic heuristic by 13% in average cost.

Contents

List of Figures

Chapter 1

Introduction

1.1 Description of the IRP

The inventory routing problem (IRP) is a very challenging problem that arises in various distribution systems. It involves managing simultaneously inventory control and vehicle routing in organizations where one or several warehouses are responsible for the replenishment of a set of geographically dispersed customers. These customers face a demand for products spread over time, and are entitled to keep local inventory. Deliveries are made using a fleet of capacitated trucks.

The IRP is a much more complex problem than the usual capacitated vehicle routing problem (CVRP). In the VRP, routing decisions are made to fulfill, by the end of the day, fixed orders placed by the customers. In the IRP, there are no customer orders, and the routing decisions are dictated by the inventory behavior of the customers, which is itself driven by their daily demand patterns. Given the customers' inventory data and information on the customers' demand, the manager must consequently make several decisions over a given planning horizon:

- Which customers to visit on each day of the planning horizon

- What quantities to deliver to each customer
- How to combine these deliveries into routes.

The objective is to minimize the distribution costs in the system, over the planning horizon. The costs considered vary from one study to another. For example, transportation costs are always taken into account, but inventory holding costs are not often considered. In studies where the customer's demand is stochastic, expected shortage costs are included as well.

It is important to note that the IRP is NP-Hard. Indeed, with a planning horizon of one day, infinite truck capacities and infinite customer capacities, this problem reduces to a Traveling Salesman Problem(TSP). The TSP was shown to be NP-hard in Karp (1972).

1.2 Industrial motivation

In the industry, the IRP can be applied to various distribution systems. Traditionally, researchers and practitioners have focused on the distribution of industrial gases. The reason for that lies in the business practices of these industries. Indeed, in the sector of heating oil or industrial gases, replenishment and inventory control were managed by the supplier very early. Note, however, that the generalization of vendor-managed inventory (VMI) policies as a business practice drove the need to extend the study of the IRP to different distribution structures. Several studies found in the literature are reviewed in Chapter 2.

1.3 Focus, motivation and contribution

In this paper, we focus on the finite-horizon case, where the customer's demand is known (deterministic) and day-dependent(dynamic). Only transportation costs will be considered, and the customer's inventory will have a finite capacity. These assumptions are mainly motivated by the above-mentioned industrial gas industry. This model can however find application in various sectors, such as the soft drinks industry, supermarket chains, and department stores.

As previously highlighted, the complexity of the IRP comes from the absence of fixed customer orders, which prevents us from building good feasible solutions to the IRP with VRP heuristics. This shows in the IRP literature, as no simple constructive heuristic can be found, with the noticeable exception of Bertazzi *et al.* (2002). Our motivation is therefore to fill this gap and try to propose a procedure that will allow us to design an efficient construction-improvement heuristic for the IRP, using the tools available from the VRP literature.

Our contribution can be summed up as follows: we show how we can transform our IRP, with the previously mentioned characteristics, into an *equivalent* problem. This latter problem will be referred to as the "Multi period VRP with due dates and split demand" (MVRPD), and is much closer to the classical VRP than our IRP. The goal of this transformation is inherent to the nature of the MVRPD: this problem explicitly considers independent, fixed, volumes of products that must be delivered to customer locations between two given days. This transformation will therefore allow us to use classic constructive heuristics found in the VRP and the split-delivery VRP literatures to build a good feasible solution to our original problem.

This thesis is organized in the following way: Chapter 2 is a literature review composed of two distinct parts: an extensive review of IRP approaches is first conducted, followed by a description of several existing VRP solution methods that will

be useful to our study. Chapter 3 gives a formal description of the IRP that we wish to tackle, and proposes an IP formulation. The core of our contribution is found in Chapter 4, which describes how the IRP can be transformed into an equivalent Multi period VRP with due dates. This result is then exploited in Chapter 5, to design a constructive heuristic for our IRP. The computational results obtained with this heuristic are finally presented and analyzed in Chapter 6. An overall conclusion of our study is given in Chapter 7.

Chapter 2

Literature review

This literature review will tackle two different topics: Firstly, we will review existing studies of Inventory Routing Problems, which will allow us to understand that this designation can encompass a wide panel of situations, that call for various solution methods. Secondly, we will give an overview of the wide VRP and Split-Delivery VRP literature. We will more specifically focus on studies proposing constructive heuristic algorithms that will help us design our own solution method.

2.1 Inventory Routing Problem studies

2.1.1 Classifications

We will start by introducing two articles which aim to define the IRP and classify the different approaches undertaken.

First of all, Federgruen & Simchi-Levi (1995) discussed the motivations for the IRP and introduced a framework that distinguishes two variants of the IRP: the single period model, with stochastic demand, and the infinite horizon model, with deterministic demand rate. Two articles, namely Federgruen & Zipkin (1984) and

Anily & Federgruen (1990) illustrated these categories. Though this classification gave an initial overview of the different aspects of the IRP, it overlooked several approaches that did not fit this description, such as single period models with deterministic demand, multi period models, and infinite horizon models with stochastic demand.

A second attempt to classify the IRP can be found in Baita *et al.* (1998). In this review, the authors started by defining the IRP as a class of problems having the following aspects in common: routing (necessity to organize a movement of goods between different sites), inventory (relevance of the volume and value of the goods moved), and dynamic behavior(repeated decisions have to be made). Within this class of problems, a classification framework was proposed that took into account all the characteristics of the different approaches encountered in the literature: topology of the problem, number of items considered, type of demand considered, type of decision to be taken, constraints considered, objective sought, costs considered and solution approach proposed. Different articles were then presented, regrouped by the type of decision to be taken: frequency-based or time-domain based.

2.1.2 Infinite horizon, deterministic demand approaches

The following is a review of infinite horizon deterministic demand approaches. All the papers described in this section consider the same type of systems: a warehouse replenishes geographically dispersed customers. These customers face a constant, deterministic demand rate. The objective is to find long-term replenishment strategies that minimize system-wide costs. A strategy consists of the construction of delivery routes, and the computation of the optimal replenishment frequency for each route. Note however that, though all these papers represent a very important part of the IRP literature, the problem they tackle and the tools they use are very different from the problem we focus on.

Anily & Federgruen (1990) considered only a specific class of strategies: fixed partition policies (FPP). This class can be described as follows: the customers are partitioned into regions and their demands are allowed to be split between several regions. The FPP is then a set of replenishment strategies where, whenever a customer is visited in a region, all the customers of this region are visited as well. This allowed the authors to transform this problem into a general partitioning problem, and to obtain several interesting results: two lower bounds over all the policies were proposed, as well as an asymptotically optimal heuristic, using a modified circular partitioning scheme. A discussion of this approach can be found in Hall (1991) and Anily & Federgruen (1991).

Several studies following similar ideas can be found in the literature. Anily & Federgruen (1993) extended the above model to a system where the central warehouse is explicitly considered as a stock-keeping location: holding costs are charged, and the warehouse has a limited capacity. The warehouse must therefore be periodically replenished, and fixed ordering costs are incurred. Here also, lower bounds were computed, and an upper bound falling within 6% of the lower bound was proposed.

Gallego & Simchi-Levi (1990) characterized the effectiveness of direct shipping strategies in these one-warehouse multiple retailers systems. The authors started by computing a lower bound of the system-wide cost over all inventory-routing strategies. Using this bound, they showed that, when the Economic Lot Size of all the retailers is at least 71% of the truck capacity, the effectiveness of the direct shipping strategies is at least 94%.

Using the same fixed-partition-policy as in Anily & Federgruen (1990), Bramel & Simchi-Levi (1995) developed a location-based heuristic that splits the customers into replenishment regions. This partition was found by solving a capacity-concentrator problem (CCP) derived from the original IRP. Indeed, even though the CCP is NP-

hard, existing techniques are known to be able to find good solutions within a reasonable time frame.

Chan *et al.* (1998) studied Zero Inventory Policies and Fixed Partition Policies in one-warehouse, multiple-retailers systems. They computed a lower bound, built a FPP solution and gave a probabilistic analysis of the optimality gap for this solution.

Finally, we find it necessary to mention here the approach developed by Bertazzi *et al.* (1997), which tackled the same issue, but with additional characteristics: a warehouse supplies *several* products to geographically dispersed customers who face a constant demand rate for each product. The specificity of this article was that replenishment is made using a finite set of replenishment frequencies. The authors proposed a heuristic construction to decide the replenishment strategies. Computational results were shown.

2.1.3 Finite horizon, stochastic IRP

We now describe a series of articles that share several characteristics. They are all motivated by the air products industry. Traditionally, in this industry, a plant supplies a region of customers who keep local inventory in a tank with finite capacity, and the supplier is responsible for designing the schedule and the routes of the deliveries. The objective is to minimize the operating costs, while avoiding customers' stockouts. Moreover, in all those studies, the demand is generally considered unknown or stochastic, and is often equated with the available capacity in the customer's tank. This means that many of these studies implicitly choose a delivery policy where the customers are replenished to full capacity whenever they are visited.

Golden *et al.* (1984) described an empirical solution approach to this problem. They developed a heuristic that aimed to minimize the daily operational costs,

while attempting to ensure a sufficient level of product at each customer location. Their approach was as follows: for each customer, an "emergency level" equal to the ratio of his current inventory level to his tank capacity is computed. All the customers whose emergency levels are higher than a chosen critical level are chosen as "potential" customers. Customers are then ranked using the ratio of emergency to delivery cost, and a TSP is then iteratively built: the ranked customers are added one at a time to the itinerary, until the total tour duration exceeds a pre-established maximum duration T_{MAX}. The tour is then split into routes. If no feasible solution is found, T_{MAX} is decreased, and the procedure is repeated.

Dror & Ball (1987) built replenishment routes for a similar system in a more sophisticated way, by taking into account the probability distribution function(PDF) of the customers' demands. In this study, the authors used results from a one-customer, deterministic demand system to compute "incremental costs" incurred on the year-long planning whenever a customer is replenished in the coming week before his inventory drops to zero. Using these incremental costs, as well as the costs charged for stockouts and the demand PDF of each customer the authors computed the expected cost $E_i(t)$ for replenishing a specific customer i on any day t. Under some assumptions, they showed the existence of t^*, the optimal replenishment day, that minimizes $E_i(t)$. A four-step heuristic was then developed: firstly, customers to be included in the coming week's schedule are selected based on their t^*. Secondly, a generalized assignment problem is solved to assign these customers to delivery days. Thirdly, efficient routes are built using a Clarke and Wright algorithm. Finally, local improvements are made on the obtained solution. Computational details of this approach can be found in Dror *et al.* (1985).

Trudeau & Dror (1992) developed several improvements to this approach. First of all, they refined the computation of $E_i(t)$ using conditional probabilities which enabled them to obtain a more accurate value of t^*. Then, they modified the cus-

tomer selection in the first step of the algorithm, thus adding more flexibility to the assignment procedure. Finally, they computed a costing procedure that takes into account the route failures. Bard *et al.* (1998) discussed a similar approach, but combined with a rolling horizon framework: a 2-week schedule was computed, but only the first week was actually implemented. The authors adapted the customer selection, customer assignment and route designing steps to the case where several satellites allow the truck to refill during his tour. The incremental costs used in the computation of the best replenishment day differed from the ones proposed in Dror & Ball (1987) and can be found in Jaillet *et al.* (2002).

2.1.4 Finite Horizon, mixed-integer programming models

In the following section, we will describe another category of articles tackling the IRP. The situations dealt with here are similar to the ones in the previous section: a central warehouse replenishes several customers, and seeks to design a replenishment schedule for the next planning period. Demand is generally deterministic, but can be stochastic. Unlike the studies presented in the previous section, the following articles present mixed-integer programming (MIP) optimization models that describe the system considered, and design solutions based on this optimization model.

Federgruen & Zipkin (1984) considered a system where the supply at the central warehouse is limited, and the demand at the different customers is considered as a random variable. The objective was therefore to minimize the total transportation, and expected inventory and shortage costs. This problem was modeled as a nonlinear integer program. Capitalizing many ideas from the Vehicle Routing problem, the authors then proposed two solution methods. First of all, they developed a modified interchange heuristic based on the "r-opt" methods of the VRP. Then, they described an exact algorithm, using a general Bender's decomposition inspired by the method of Fisher & Jaikumar (1978) on deterministic VRP's.

Chien *et al.* (1989) also tackled the problem of limited supply, in a study taking into consideration a deterministic customer demand. Their objective was therefore to distribute this limited amount of products so as to maximize profits. They considered a single day approach, but, by passing information from one day to another, their model simulated multiple periods. A MIP model was proposed to optimally allocate the inventory among the customers. This MIP was solved, using a Lagrangian-based heuristic and computational results were exhibited.

Bertazzi *et al.* (2002) studied a problem similar to the one we will focus on in this paper, in which customers face a deterministic and dynamic demand, and have a finite capacity for holding local inventory. Holding costs are however considered at the central warehouse, as well as at the different retailers. The authors investigated a replenishment policy where, whenever a customer is visited, it is replenished to full capacity. A heuristic was presented, that makes good use of a graph representing the delivery schedules to build feasible solutions. Exhaustive computational results were exhibited, where different combinations of costs accounted for different distribution structures.

Campbell *et al.* (2002) developed a finite horizon, deterministic demand model of the IRP. They considered a system where the customers face a constant demand rate, and have a finite local inventory capacity. The first phase of the solution method is an interesting IP model that aims to optimize the deliveries over a two-week rolling horizon. In this model, the complexity of the routing computations is reduced. Indeed, only a given set of routes with known characteristics (such as duration or cost) are considered for the deliveries. Trucks are allowed to serve multiple trips per day, and maximum route duration is enforced. Several techniques are proposed to allow tractability of the model, such as considering only a given set of allowed routes, aggregating several time periods at the end of the planning horizon, or relaxing some integrality constraints. The solution to this IP indicates

quantities to be delivered to each customer on each day. Using these quantities as indications, the second phase then builds an actual delivery schedule for the next two days, using more accurate demand information, and taking into account the proper timing of this demand. The computational results were interesting, as they showed different performance measures of the solution method.

2.1.5 Related studies

We find it necessary to describe two approaches that deal with different aspects of the IRP.

Firstly, Webb & Larson (1995) studied an IRP at the strategic level: their goal was to determine the size of the fleet needed to operate a one-warehouse, multiple-retailers distribution system. In this prospect, all possible realizations of the tactical IRP need to be considered. A heuristic was proposed, that estimates the fleet size by dividing customers into a set of clusters.

Secondly, Berman & Larson (2001) considered the IRP at an operational level, by trying to optimize the deliveries within a given, fixed route, where the driver has the responsibility to decide the quantities delivered to each customer visited. Incremental costs for early and late deliveries were computed, on the basis of the customer's inventory level (real or estimated). These costs were then used in a dynamic programming framework to compute the optimal delivery policy.

2.2 Vehicle Routing Problem studies

The following section will give an overview of the VRP and split-delivery VRP literature, and will highlight some solution approaches that inspired us when designing the heuristic described in Chapter 5.

2.2.1 VRP solution methods

A wide range of studies of the VRP can be found in the literature, but we will restrict ourselves to the description of classic VRP approaches that do not consider additional constraints or specific features. Readers wishing a broader description of the VRP can refer to the early work of Bodin *et al.* (1983). A more recent review of exact and approximate solution methods can be found in Laporte (1992). Finally, the book by Toth & Vigo (2002) studies extensively the different aspects of the VRP: a complete overview of the different formulations of the problem, and a wide spectrum of solution methods are detailed. In order to familiarize the reader with the different alternatives, we will list the mainstream approaches encountered in the literature.

2.2.1.1 Exact solution methods

In the VRP literature, the most widely described *exact* solution method is based on *branch-and-bound* algorithms. An extensive description of these branch-and-bound techniques can be found in Laporte & Nobert (1987). It is shown that basic lower bounds can be obtained by relaxing some VRP constraints, which amounts to replacing the VRP by simpler problems, such as assignment problem or finding spanning trees. Better lower bounds can be obtained with more elaborate methods: for example, Fisher (1994) proposed a strengthened VRP relaxation obtained by including some of the relaxed constraints in the objective function in a Lagrangian way, while E. Hadjiconstantinou (1995) used a lower bound computed by finding a feasible solution to the dual of a set-partitioning VRP formulation.

Branch-and-cut is another less investigated exact solution method. In this approach, the linear relaxation of the VRP is considered. Because of the non-polynomial number of constraints, this relaxation cannot be fed into an LP solver.

A great number of constraints are therefore dropped, and valid inequalities(cutting planes) are progressively added. The amount of research published in that area is more limited than in branch-and bound techniques, and publications are often focused on specific aspects of the procedure, such as finding valid inequalities. The reader can however refer to Ralphs *et al.* (2003) for a complete implementation of this approach.

Finally, we found several studies that consider the set-covering formulation of the VRP: all the feasible routes are implicitly included in the IP formulation, which therefore contains a great number of columns. Exact algorithms using this formulation have been described by Agarwal *et al.* (1989) or the more recent papers by E. Hadjiconstantinou (1995) and Desrochers *et al.* (1992).

2.2.1.2 Constructive heuristics

The methods discussed in the previous paragraphs have a high theoretical value. However, they are seldom used in practice, as they can only solve instances of modest size, and require a lot of computing time. This highlights the need for simple, fast-running and robust heuristics that produce solutions of a reasonable quality.

The most commonly used heuristic is the Clarke & Wright (1964) algorithm. This constructive method is based on the notion of *savings*. The savings obtained by merging routes $(0, ..., i, 0)$ and$(0, j, ..., 0)$ is $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. A first routing plan is initiated with n $(0, i, 0)$ routes and the routes are progressively merged, starting from the highest feasible savings. The procedure stops when no positive savings can be achieved. Several enhancements to this savings algorithms can be found in the literature. Baskell (1967) and Yellow (1970) for example, included a route shape parameter λ in the savings computation $s_{ij} = c_{i0} + c_{0j} - \lambda c_{ij}$, while Desrochers

& Verhoog (1989) or Altinkemer & Gavish (1989) implemented a matching-based approach using the savings computation.

Another constructive method to obtain a feasible routing plan is the iterative insertion. Starting from an empty plan, routes are grown by iteratively inserting visits that will incur the smallest additional cost. Mole & Jameson (1976) implemented a sequential version of this algorithm, while Christofides *et al.* (1979) developed a more sophisticated method using both sequential and parallel route constructions.

2.2.1.3 Improvement heuristics

In the approaches presented in the previous paragraphs, the method described gives an initial feasible solution. Routing plans with lower costs can then be obtained using *improvement heuristics* that try to apply elementary modifications to the current solution.

The most common improvement heuristic is the λ-*opt* technique, initiated in the TSP literature (See Lin (1965)). This method removes λ arcs from the current solution and examines the ways to reconnect them. If a cost-saving combination is found, it is implemented. The procedure is repeated until no improvement is found. Or (1976) described a method, the *Or-Opt*, commonly used in practice: 3, 2 or 1 consecutive arcs are displaced to a cheaper location, until no improvement is found.

Breedam (1994) described 3 other multi-route improvements: the crossing, the exchange, and the relocation. These operators will be described later on, as we will be using them in our heuristic.

2.2.1.4 Metaheuristics

The previous paragraph described basic neighborhood operators that defined a range of "neighbors" of a given solution, by modifying one or several of its arcs. This opens the way to a deeper exploration of the search space using metaheuristics. These techniques, whose general framework is common to several combinatorial optimization problems, have produced several best-so-far results to "hard" VRP instances. In these metaheuristics, the search is conducted by going from one feasible solution to another in its neighborhood, while allowing cost increasing moves, which increases the chances for the algorithm to escape from local minimum. This in general requires more computational time than the classic heuristics. Here are, very briefly, some successful studies: good implementations of simulated annealing can be found in Osman (1993) and in Golden *et al.* (1998), while Gendreau & Laporte (1994) and Taillard (1993) developed interesting tabu search frameworks. We do not wish to elaborate on the details of their approaches, as it is beyond the scope of our study here.

2.2.2 The Split-delivery VRP

The split-delivery VRP (SDVRP) is a problem very similar to the VRP: a fleet of capacitated vehicles has to deliver orders from a depot to a set of customers over a single period. The goal is to minimize the distance traveled. The specificity of the SDVRP is that no restriction is made on the number of visits a customer may receive: a customer's order can be *split* between two or more deliveries.

The studies available in the literature on the SDVRP are not as numerous as the ones focusing on the VRP.

The problem was first introduced by Dror & Trudeau (1989). In that article,

the SDVRP is formally described, and formulated. Using an example, the authors showed how cost savings can be achieved when split-deliveries are allowed. Several properties of the optimal solution were exhibited, and a heuristic was proposed: it starts from a feasible VRP solution, and looks for cost-saving opportunities by splitting deliveries. The computational experiments showed that, when average demand is at least 10% of vehicle capacity, the savings achieved by splitting deliveries are significant.

Dror *et al.* (1994) refined the previous formulation and proposed several valid inequalities. These inequalities were derived from the analysis of the subtour elimination constraints, as well as from other observations made on the model. The inequalities were used as cuts in a constraint-relaxation algorithm: a lower bound is first obtained using the inequalities in a LP relaxation of the problem; then a branch-and bound procedure seeks the optimum. A 10-customer instance was solved to optimality, and the lower bound obtained showed that the heuristic proposed by Dror & Trudeau (1990) produced solutions within 9% of optimum.

A more technical set of valid inequalities can be found in Belenguer *et al.* (2000). The authors showed that the convex hull of the set of feasible solutions is a polyhedron. Several facet-defining inequalities were derived, and were used in a cutting-plane algorithm, as in Dror *et al.* (1994). The algorithm was successful in finding the optimal solution for a 50-customer SDVRP.

Frizzel & Giffing (1992) studied a different SDVRP where the nodes are located on a grid network distance, which does not guarantee the triangular inequality. Additionally, the authors investigated the possibility of limiting the size and the number of splits allowed. A heuristic and computational results were presented. In Frizzel & Giffing (1995), time windows were added to the model.

Mullaseril *et al.* (1997) gave a real-world application to the SDVRP. They fo-

cused on an arc-routing problem (Capacitated Rural Postman Problem, CRPP) encountered in a cattle-feeding ranch. The heuristic proposed in Dror & Trudeau (1990) was adapted to their model, and good computational results were presented.

Another real-life application can be found in Sierksma & Tijssen (1998). This article dealt with routing helicopters between off-shore platforms for crew exchanges. The problem is identical to the SDVRP but imposes a maximum route length. Several new properties of the optimal solution were derived and two different heuristics were proposed. The first one is a column generation procedure that starts from a fractional lower bound, and rounds it up to a feasible solution. The second one is a two-step constructive algorithm entitled "cluster-and-route procedure". Several improvement heuristics were also proposed, and the computational results were compared with traditional VRP heuristics.

Several interesting theoretical results on the SDVRP were derived in Archetti *et al.* (2004). Some properties of the optimal solution were exhibited, and a bound on the savings that can be done by allowing split deliveries was given: the value of the optimal solution of the SDVRP cannot be less than half the optimal value obtained in the corresponding VRP. This bound is tight, as examples were exhibited.

Finally, Archetti *et al.* (2003) developed a tabu search, where a simple initial solution is first built, then the tabu procedure is run, and the final solution is improved by deleting cycles and re-optimizing each route. The tabu search provided solutions of better quality than the algorithm proposed by Dror & Trudeau (1989)(5% on average), but required more computational time.

Chapter 3

Problem description and model formulation

As shown in the previous chapter, the designation "Inventory Routing Problem" can refer to a wide variety of situations, depending on the assumptions made and the industrial problem considered. This chapter will therefore formally describe the problem this thesis will focus on. A definition of our problem will first be given in Section 3.1, together with the motivations behind this definition. Section 3.2 then gives an exhaustive list of our assumptions, which will allow the reader to get a clear picture of the problem tackled. A mathematical program is proposed in Section 3.3. Finally, Section 3.4 refines the latter formulation by tightening some constraints.

3.1 Problem definition and motivations

The problem we will focus on is a finite-horizon inventory-routing problem, where the demand of the customers is both deterministic and dynamic, and where transportation costs are solely considered, that is holding costs are overlooked: the depot manages the replenishment of the customers using a fixed fleet over the planning horizon, and aims to minimize total transportation costs while preventing stockouts.

Note that, among the reviewed IRP approaches in Section 2.1.4 , Campbell *et al.* (2002) considered a problem very similar to ours, but did not consider dynamic demand. Bertazzi *et al.* (2002) considered dynamic demand, but their study considered a cost structure where holding costs represented a large part of the total costs. Furthermore the industrial problem tackled in our approach is similar to the studies focusing on industrial gases described in Section 2.1.3. The latter however considered stochastic demand, included the expected stockout cost, and generally allowed customers to be visited only once in the planning horizon.

We chose to model the demand as deterministic because we feel that, even though uncertainty is present, customers' usage or demand can be quite predictable for planning horizon of medium size (one or two weeks). Furthermore, by embedding our study in a rolling horizon framework, where, for example, two weeks are computed and only one is implemented, we are able to update the demand data regularly.

3.2 Assumptions

Below are the different assumptions our model is based on.

Planning Horizon We will consider a finite planning horizon (Typically 14 days).

Customers The location of the customers are known. They have a finite capacity for keeping local inventory.

Costs No holding costs will be considered in this study. The costs will therefore only consist of the *transportation costs* which are deterministic and known.

Deliveries The products will be shipped from the central warehouse to the different customers using a *homogeneous* fleet of vehicles of known capacity. No additional constraints will be imposed on the route, such as maximum duration.

Furthermore we will *not* restrict a customer to be visited at most once during a given day which means that split deliveries are allowed.

Inventory Stockouts will not be allowed, and both demand and deliveries of the current day will be taken into account when computing the inventory.

Demand The amount of product consumed at each customer location is known, and day-dependent. We will assume that the highest demand is smaller than the vehicle capacity.

Supply No limit will be imposed on the amount of supply available at the central warehouse.

Operating modes and delivery times We will make the hypothesis that a delivery made on a given day can be used to meet the demand required on that day. This means that we will not consider any operating modes and delivery times within a given day.

Objective The objective is to build a delivery schedule so as to

Minimize transportation costs while

- avoiding stockouts
- not exceeding vehicle capacity
- not exceeding customers' capacity

3.3 Model

We can now state our model as a mathematical program, using the following notations:

A set N of customers is served by a depot denoted as 0. During each day $t \in \mathcal{H} = \{1, 2, ..., T\}$ of the planning period, a known quantity d_i^t, called *demand of*

customer i on day t is absorbed at the local inventory of customer i. The level of this inventory, denoted as I_i^t cannot exceed the customer's *capacity* C_i. The quantity I_i^0 will denote the *initial inventory*, and we will assume it is known. The traveling costs c_{ij}, with $i, j \in N \bigcup \{0\}$ are all known as well. On each day of the planning period, we dispose of a fleet of K vehicles indexed by $k \in \mathcal{K} = \{1, ..., K\}$. Each vehicle can carry up to a volume W of products to accomplish the deliveries.

Our decision variables are the following:

$$x_{ij}^{kt} = \begin{cases} \text{undefined}, & \text{if } i = j \\ 1, & \text{if arc } (i,j) \text{ is visited on day } t \text{ by vehicle } k \\ 0, & \text{otherwise} \end{cases}$$

q_i^{kt} is the quantity delivered on day t to customer i by vehicle k

$$y_i^{kt} = \begin{cases} 1, & \text{if customer } i \text{ is visited on day } t \text{ by vehicle } k \\ 0, & \text{otherwise} \end{cases}$$

With these notations, the inventory of a customer i at the *end* of a given day t is given by:

$$I_i^t = I_i^{t-1} + \sum_{k \in \mathcal{K}} q_i^{kt} - d_i^t$$

that is

$$I_i^t = I_i^0 + \sum_{t'=1}^{t} \sum_{k \in \mathcal{K}} q_i^{kt'} - \sum_{t'=1}^{t} d_i^{t'}$$

Let us define the *cumulated demand of customer i*:

$$D^t(i) = \sum_{t'=1}^{t} d_i^{t'}$$

With this notation we can write:

$$I_i^t = I_i^0 - D^t(i) + \sum_{t'=1}^{t} \sum_{k \in \mathcal{K}} q_i^{kt'} \tag{3.1}$$

Our problem can now be stated as follows:

$$\text{Minimize} \sum_{t=1}^{T} \sum_{k \in \mathcal{K}} \sum_{i,j \in N \cup \{0\}} c_{ij} x_{ij}^{kt} \tag{3.2}$$

Subject to:

$$\sum_{i \in N \cup \{0\}} x_{il}^{kt} - \sum_{j \in N \cup \{0\}} x_{lj}^{kt} = 0 \qquad \forall l \in N \cup \{0\}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.3}$$

$$\sum_{i \in N \cup \{0\}} q_i^{kt} \leq W \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.4}$$

$$\sum_{j \in N \cup \{0\}} x_{ij}^{kt} = y_i^{kt} \qquad \forall i \in N, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.5}$$

$$q_i^{kt} \leq y_i^{kt} W \qquad \forall i \in N, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.6}$$

$$\sum_{i,j \in S} x_{ij}^{kt} \leq |S| - 1 \qquad \forall S \subseteq N / |S| \geq 2, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.7}$$

$$D^t(i) - I_i^0 \leq \sum_{t'=1}^{t} \sum_{k \in \mathcal{K}} q_i^{kt'} \qquad \forall i \in N, \forall t \in \mathcal{H} \tag{3.8}$$

$$\sum_{t'=1}^{t} \sum_{k \in \mathcal{K}} q_i^{kt'} \leq D^t(i) + C_i - I_i^0, \qquad \forall i \in N, \forall t \in \mathcal{H} \tag{3.9}$$

$$y_i^{kt} \in \{0, 1\} \qquad \forall i \in N, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.10}$$

$$x_{ij}^{kt} \in \{0, 1\} \qquad \forall i, j \in N \cup \{0\}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.11}$$

$$q_i^{kt} \geq 0 \qquad \forall i \in N, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{3.12}$$

The objective function (3.2) aims at minimizing the total transportation costs. Constraints (3.3) to (3.7) guarantee the consistency of the routing plan: constraint set (3.3) ensures flow conservation, constraint set (3.4) imposes the total load on each vehicle not to exceed the vehicle capacity and constraint sets (3.5) and (3.6) guarantee a customer is visited when receiving a positive delivery volume. Finally, constraints (3.7) forbid subtours disconnected from the depot. The constraints that will need special attention in our study are the constraints defined by (3.8) and (3.9). They ensure that:

- No stockouts occur
- The capacities of all the customers are not exceeded

To understand the formulation of (3.8) and (3.9), note that it is obtained by ensuring that the inventory defined in (3.1) stays non-negative(no stockouts) and below customer capacity. The left-hand side of (3.8) is then obtained by noticing that all the delivered quantities must be positive.

In our formulation, there is no limitation on the number of deliveries a customer can receive on a given day: split deliveries are therefore allowed.

3.4 Property of the optimal solution

It is important for our study to point out the following property of the optimal solution:

Lemma 1. *There exists an optimal solution to the IRP where the final inventory of all the customers is empty, that is:*

$$I_i^T = 0, \forall i \in N$$

Proof. The proof is quite straightforward. Consider an optimal solution of the IRP where a set of customers $F \subset N$ has inventory left on day T. An optimal solution with empty final inventory can be obtained by simply removing excess delivery volumes to this set of customers *without changing the routing sets and itineraries.*

No constraints are violated, since vehicle occupation is *reduced* and the routing is not modified. Moreover, only the final inventory level is affected and drops to zero in the new solution. Applying this treatment to all customers in F will leave the value of the objective function unchanged, and all the customers will have an empty final inventory. □

This property will help us tighten the constraint of our IRP formulation, by setting the total delivery quantity $\sum_{t'=1}^{T}\sum_{k\in\mathcal{K}} q_i^{kt'}$ to a value that leaves the final inventory empty.

Let us define, for notational convenience the *cumulated deliveries made to customer i*:

$$Q^t(i) = \sum_{t'=1}^{t}\sum_{k\in\mathcal{K}} q_i^{kt'}$$

With these notations, $Q^t(i)$ is the total volume that reaches the customer up to day t, and we can reformulate the inventory constraints:

Corrolary 1. *The set of constraints (3.8) and (3.9) can be replaced by the set*

$$D^t(i) - I_i^0 \leq Q^t(i) \qquad \forall i \in N, \forall t \in \{1, ..., T-1\} \tag{3.13}$$

$$Q^t(i) \leq D^t(i) + C_i - I_i^0 \qquad \forall i \in N, \forall t \in \{1, ..., T-1\} \tag{3.14}$$

$$Q^T(i) = D^T(i) - I_i^0 \tag{3.15}$$

This modified formulation will be used in the next chapter to show that, when forcing the final inventory to be empty, our IRP is equivalent to a multiple-period VRP with fixed-size order.

Chapter 4

Transposition of the IRP into a rich VRP

In this section, we will show how the above-defined IRP can be transformed into a multiple-period VRP with due dates, and with split-deliveries allowed. This new, transformed problem will be referred to as the MVRPD.

4.1 Motivation

The main problem with the formulation of the IRP written in Section 3.3 lies in the very nature of the constraint sets (3.8)-(3.9). These constraints, which ensure that no stockouts occur and that capacity is not exceeded, proceed by always considering *cumulated demand* $D^t(i)$ and *cumulated delivered quantities* $Q^t(i)$. Although the formulation is quite compact, it has major drawbacks that prevent us from using it to design constructive heuristic. Indeed, the high level of interaction between the different constraints

- makes it difficult to identify and individualize constrained deliveries.

- induces a heavy constraint propagation in constructive heuristics.

A good illustration of these issues can be found in the heuristic proposed by Bertazzi *et al.* (2002). In the article, the authors used a constructive insertion algorithm to build a feasible solution. Their construction required **all the deliveries** of a given customer to be inserted in the current solution **together**, to ensure that all the inventory constraints are satisfied.

Our reformulation of the problem will therefore aim to individualize, as much as possible, all the constraints induced by the inventory behavior of a given client. This will be attained by shifting from the current *time*-driven formulation of the IRP, where each day of the planning period induces two inventory constraints, to a *demand*-driven formulation(MVRPD), where the demand is split into *loads* that have to be delivered to the customer location between two specific days.

Before detailing our procedure, we will provide a formal description and a linear IP formulation of the MVRPD:

4.2 Description of the MVRPD

The MVRPD can be described as follows. During a horizon of T days, a set $\mathcal{R}$ of geographically dispersed demand points needs to be replenished in a given product by a depot denoted as 0. Each demand point $r \in \mathcal{R}$ needs a volume δ_r of product, that has to reach the customer between an earliest and a latest delivery date $E_r, L_r \in \mathcal{H}$. The costs c_{rs} of traveling between any two nodes $r, s \in \mathcal{R} \cup \{0\}$ are known. There is infinite supply at the depot. On each day of the horizon, the deliveries are made using a fleet of K vehicles of capacity W, indexed by $k \in \mathcal{K} = \{1, ..., K\}$. We **allow the demand of a customer to be split among vehicles or days**, as long as

the totality of the loads is delivered within the specified days. An important feature for our problem is that several demand points will share the **same geographical location**, but will have different demand volumes required between different delivery dates. The objective is to minimize the transportation costs.

We can formulate a mathematical program for our problem, starting with the decision variables:

$$x_{rs}^{kt} = \begin{cases} \text{undefined}, & \text{if } r = s \\ 1, & \text{if arc } (r,s) \text{ is visited on day } t \text{ by vehicle } k \\ 0, & \text{otherwise} \end{cases}$$

φ_r^{kt} is the quantity delivered by vehicle k on day t to demand point r

$$y_r^{kt} = \begin{cases} 1, & \text{if demand point } r \text{ is visited by vehicle } k \text{ on day } t \\ 0, & \text{otherwise} \end{cases}$$

The MVRPD can therefore be stated as

$$\text{Minimize} \sum_{t=1}^{T} \sum_{k \in \mathcal{K}} \sum_{r,s \in \mathcal{R} \cup \{0\}} c_{rs} x_{rs}^{kt} \tag{4.1}$$

Subject to

$$\sum_{r\in\mathcal{R}\cup\{0\}} x_{rl}^{kt} - \sum_{s\in\mathcal{R}\cup\{0\}} x_{ls}^{kt} = 0 \qquad \forall l \in \mathcal{R}\cup\{0\}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.2}$$

$$\sum_{r\in\mathcal{R}\cup\{0\}} \varphi_r^{kt} \leq W \qquad \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.3}$$

$$\sum_{s\in\mathcal{R}\cup\{0\}} x_{rs}^{kt} = y_r^{kt} \qquad \forall r \in \mathcal{R}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.4}$$

$$\varphi_r^{kt} \leq y_r^{kt} W \qquad \forall r \in \mathcal{R}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.5}$$

$$\sum_{r,s\in S} x_{rs}^{kt} \leq |S| - 1 \qquad \forall S \subseteq \mathcal{R}/|S| \geq 2, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.6}$$

$$\sum_{k\in\mathcal{K}} \sum_{t=E_r}^{L_r} \varphi_r^{kt} = \delta_r \qquad \forall r \in \mathcal{R} \tag{4.7}$$

$$y_r^{kt} \in \{0,1\} \qquad \forall r \in \mathcal{R}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.8}$$

$$x_{rs}^{kt} \in \{0,1\} \qquad \forall r,s \in \mathcal{R}\cup\{0\}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.9}$$

$$\varphi_r^{kt} \geq 0 \qquad \forall r \in \mathcal{R}, \forall k \in \mathcal{K}, \forall t \in \mathcal{H} \tag{4.10}$$

The sets of constraints (4.3) to (4.6) guarantee, like their IRP equivalents (3.4) to (3.7) in Section 3.3, the consistency of the routing plan. The demand fulfillment constraints (4.7) replace the IRP set of constraints driving the inventory behavior (3.8) and (3.9). They ensure the loads are fully delivered within the specified window $[E_r, L_r]$.

4.3 Underlying concept, Procedure undertaken

Our goal is now to show that, given a valid IRP (as described in section 3.3) one can build a MVRPD with the characteristics above, such that the two problems will be equivalent.

Note that the **time horizon** $\mathcal{H}$, the **number of available vehicles** K and the **vehicle capacity** W are **common to the two formulations**, and will be used without alteration when transposing our problems.

The MVRPD will therefore be completely defined once the set $\mathcal{R}$ is known, together with the delivery dates E_r, L_r corresponding to every demand point.

The main idea behind our procedure subsequently lies in establishing a relationship between the distinct demand fulfilment constraints in the two problems. In the IRP, constraint set (3.8) ensures no stockouts occur, while constraints (3.9) forbid the inventory level to exceed capacity C_i. We will show that with a proper definition of the MVRPD data, as well as a consistent transposition of the feasible solutions from one problem to another, the constraints can be parallelized with, respectively, the latest and earliest delivery dates constraint (4.7).

Indeed, each IRP customer i will be replaced with a set of MVRPD demand points $\mathcal{R}_i \subseteq \mathcal{R}$ located at the same geographical location. Therefore, from now on, our focus will be on **one fixed IRP customer** i.

We will proceed as follows:

- Set up, in Section 4.4 a consistent panel of notations that will help us describe the two problems and interrelate them in the rest of the study.
- Explain the procedure that builds up a valid MVRPD, using the data of an existing IRP, in Section 4.5. We will highlight in Theorem 1 two properties of the MVRPD generated in that procedure, which will allow us to relate our two problems.
- Section 4.6 will then discuss how a feasible IRP solution should be handled to build a feasible MVRPD solution, and vice versa. Again, compact properties will be derived in Theorem 2, to account for the relationship between the feasible solutions.

- These two theorems will allow us to formally show in Section 4.7 that, when building up the MVRPD as explained in Section 4.5 and transposing the decisions of feasible solutions as described in Section 4.6, any feasible solution of one problem is *indeed* feasible for another.
- Finally, Section 4.8 will illustrate our procedure with a simple example.

4.4 Notations

Before starting any formal description of our approach, it is important to properly introduce some notations used in the following sections.

Omission of customer index i

As explained in Section 4.3, the focus of the next few sections will be on a given customer i. For clarity, the index i will subsequently be omitted when referring to the characteristics of an IRP customer. Only the set $\mathcal{R}_i \subseteq \mathcal{R}$ will keep that index in order to remind the reader that it refers to a set of MVRPD demand points that emanate from the same IRP customer i and that all share the same geographical location.

Cumulated quantities

Given the nature of the inventory constraints (3.8) and (3.9), cumulated quantities such as demand, delivered quantities or loads will be often referred to. For clarity, these quantities will be denoted with the CAPITAL letter of the original variable, and a superscript indicating the last term on the summation. For example, the cumulated demand of the customer will be denoted as $D^t = \sum_{t'=1}^{t} d^{t'}$ (Note that,

	IRP:One customer $i \in N$	
	Designation	Notation
Data	Capacity	C
	Initial inventory	I^0
	Daily demand	d^t
	Cumulated demand	$D^t = \sum_{t'=1}^{t} d^{t'}$
Decisions	Vol. delivered on day t by a vehicle k	q^{kt}
	Cumulated deliveries	$Q^t = \sum_{k \in \mathcal{K}} \sum_{t'=1}^{t} q^{kt'}$

Table 4.1: Notations relative to the original IRP

	MVRPD:Set of loads $\mathcal{R}_i \subset \mathcal{R}$	
	Designation	Notation
Data	Earliest, latest delivery dates of $r \in \mathcal{R}_i$	L_r and E_r
	Volume of a load r	δ_r
	Cumulated loads	$\Delta^r = \sum_{r'=1}^{r} \delta_{r'}$
Decisions	Quantity delivered by a vehicle k on day t to demand point r	φ_r^{kt}

Table 4.2: Notations relative to the destination problem MVRPD

as already indicated, the index i is omitted). Likewise, the cumulated deliveries will be referred to as $Q^t = \sum_{k \in \mathcal{K}} \sum_{t'=1}^{t} q^{kt'}$. Moreover, once the set $\mathcal{R}_i$ is defined, we will consider the cumulated loads themselves: $\Delta^r = \sum_{r'=1}^{r} \delta_{r'}$, where the sum is conducted on the ordered set. Tables 4.1 and 4.2 sum up all the notations that will be used subsequently and should be used as reference, while Figure 4.1 describes our approach using these notations.

4.5 Formal transposition of the data

The following description will formalize our transposition procedure. This procedure considers an IRP customer with a finite capacity C, any demand pattern $(d^t)_{t \in \{1,\dots,T\}}$,

Problem	IRP:One customer $i \in N$	$\rightsquigarrow$	MVRPD: Set $\mathcal{R}_i \subseteq \mathcal{R}$
Data	I_0, C, and $(d^t$ and $D^t)_{t\in\mathcal{H}}$	$\rightsquigarrow$	$\mathcal{R}_i$ and $\{(E_r, L_r, \delta_r), r \in \mathcal{R}_i\}$
Decisions	q^{kt} and Q^t	$\rightsquigarrow$	φ_r^{kt}
Constraints	$D^t - I_0 \leq Q^t \leq D^t - I_0 + C$ $(\forall t < T)$ and $Q^T = D^T - I_0$	$\Longleftrightarrow$	$\sum_k \sum_{t=E_r}^{L_r} \varphi_r^{kt} = \delta_r$ $\forall r \in \mathcal{R}_i$

Figure 4.1: Description of the transposing approach

where $max\{d^t, t \in \{1, ..., T\}\} \leq C$ and an initial inventory $I^0 \leq C$, and outputs a set $\mathcal{R}_i$ of MVRPD loads. We will assume that all the notations introduced in Section 4.4 and summed up in Tables 4.1 and 4.2 are known.

The idea behind our transposition procedure is very simple. It consists of breaking up the total delivery volume $Q^T = D^T - I_0$ into an *ordered* set of loads. Each load r in this set will have a volume δ_r, a latest delivery date L_r reflecting the "no stockouts" constraint, and an earliest delivery date, accounting for the customer capacity restriction. As explained in the next section 4.6, the validity of this procedure will rely on the fact that the ordered loads will be delivered in *that specific order* to the IRP customer considered.

The procedure makes great use of the cumulated demand and is therefore very graphical in description. We will consequently make great use of figures to describe it, and derive analytical properties later on, that will be summed up in Theorem 1 on page 38.

4.5.1 Step 1 - Total delivery volume $D^T - I_0$

To visualize the cumulated demand$(D^t)_{t\leq T}$ on a unified scale, the daily demand volumes (d^t) are arranged one after another. Initial inventory is accounted for on this scale by removing I_0 units from the lower part of the diagram. Figure 4.2

illustrates how, once this is done, one can clearly visualize the total delivery volume $Q^T = D^T - I_0$.

Moreover, this figure shows that we can identify graphically the first day whose demand is not totally covered by the initial inventory: $t_0 = min\{t \in \mathcal{H} : D^t > I^0\}$. That means that stockouts will not occur if no deliveries are made before t_0. Our

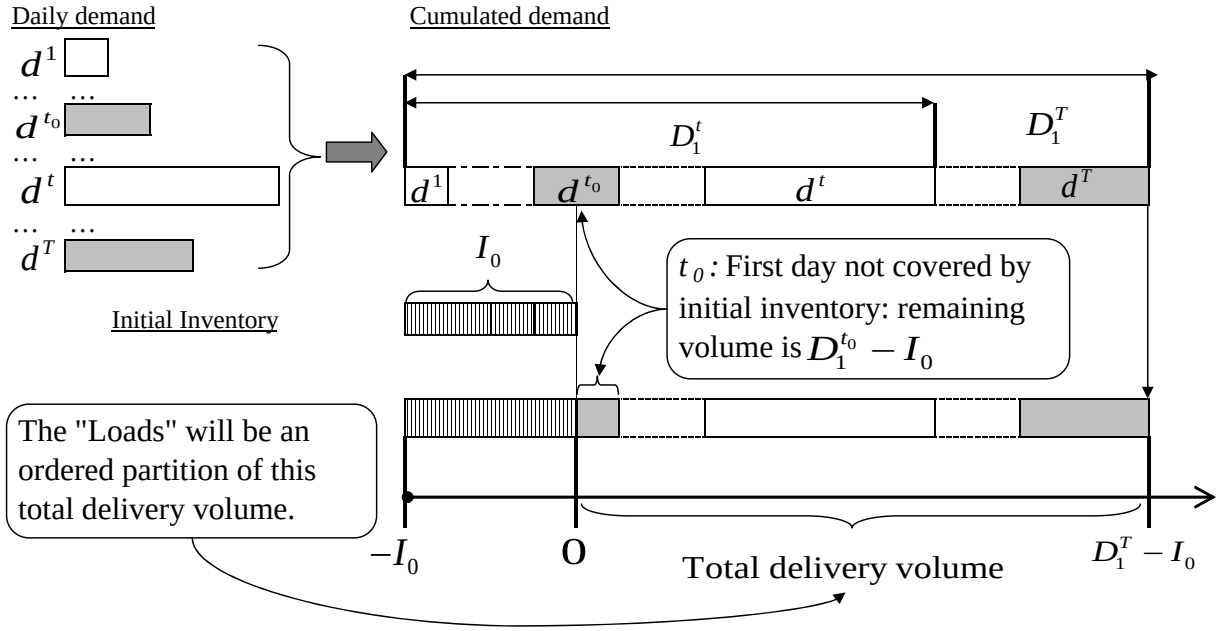

Figure 4.2: Defining t_0 and the total delivery volume

task is now to extract "loads", by partitioning this total delivery volume $D^T - I^0$. This volume can be represented by the interval $[0, D^t - I^0]$ of an x axis. This axis will be used to define the latest delivery date $L(x)$ and the earliest delivery date $E(x)$ of a given part of the delivery volume.

The partition will be realized in two steps. Firstly, the "no stockouts" constraint will impose latest delivery dates and therefore give a first division of the total delivery volume. Secondly, the customer capacity constraint will impose earliest delivery dates, that will further split this volume. The loads will then be simply obtained by merging these two partitions.

4.5.2 Step 2 - Latest delivery dates

The first partition is the most straightforward. The "no stockouts" constraints will require that the volume of a given day's demand d^t is delivered before or on that specific day t, unless the initial inventory can cover this demand volume. Hence, any load corresponding to the volume of d^t will be assigned a latest delivery date of t, as illustrated in Figure 4.3. More formally, we have created "blocks" in the total delivery volume such that:

$$t = t_0 : \quad \{x \in (0, D^{t_0} - I_0]\} \Leftrightarrow \{L(x) = t_0\}$$
$$\forall t > t_0 \quad \{x \in (D^{t-1} - I_0, D^t - I_0]\} \Leftrightarrow \{L(x) = t\}$$

which *implies*

$$max\{x : L(x) \leq t\} = D^t - I_0 \quad \forall t \geq t_0 \tag{4.11}$$

The next section shows how the capacity constraint may break these blocks into pieces that do not have the same earliest delivery date.

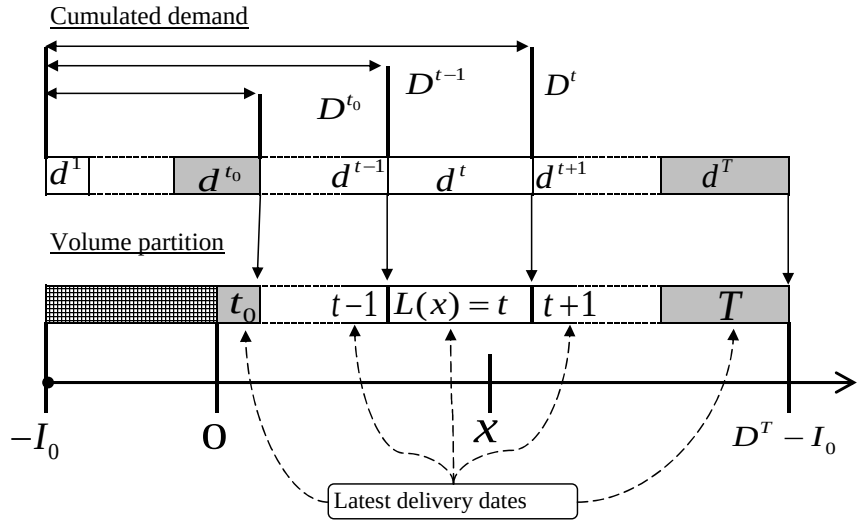

Figure 4.3: Finding the latest delivery dates

4.5.3 Step 3 - Earliest delivery date

In the same fashion, we will now identify "blocks" of the total delivery volume that share the same earliest delivery date E. In this prospect, the capacity constraint $Q^t \leq D^t + C - I_0$ is visualized on the same scale by shifting the cumulated demand by C units and reporting it in the total delivery volume. In that way, one can graphically identify the volume that can fit in the customer's capacity on a given day. Thus, for any day $\theta \in \mathcal{H}$, any part of $[0, D^T - I_0]$ facing d^θ on the "shifted" scale will be assigned an earliest delivery date $E = \theta$, as shown in Figure 4.4.

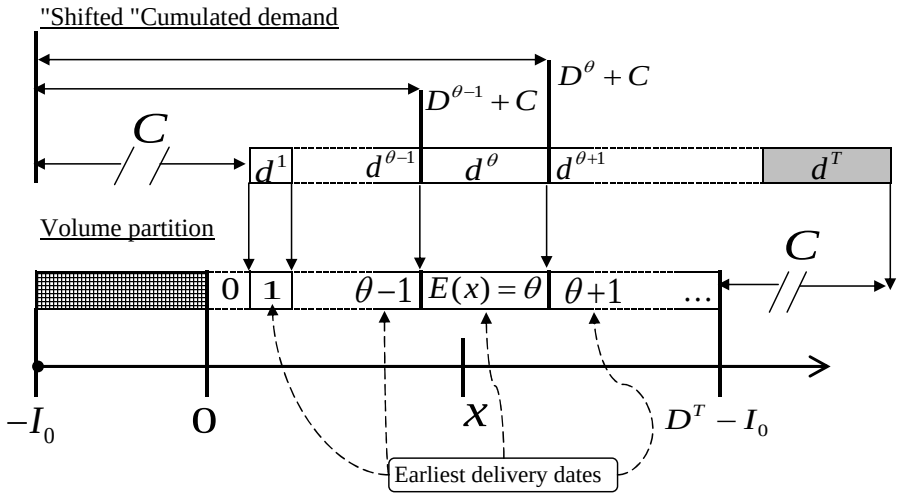

Figure 4.4: Finding the earliest delivery dates

We have created here a different set of "blocks" such that:

$$\forall \theta \in \mathcal{H} \quad \{x \in (D^{\theta-1} + C - I_0, D^\theta + C - I_0]\} \Leftrightarrow \{E(x) = \theta\}$$

which *implies*:

$$\forall x \in [0, D^T - I_0], \quad \forall \theta \in \mathcal{H}, \quad \{E(x) > \theta\} \Leftrightarrow \{x > D^\theta + C - I_0\} \tag{4.12}$$

4.5.4 Step 4 - Merging of the two partitions

The loads are now obtained by merging the two partitions defined in the previous two sections, and by noting that a given load r must have a *unique* latest delivery L_r and a *unique* earliest delivery date E_r as shown in Figure 4.5. The loads are then simply numbered from 1 to R=$|\mathcal{R}_i|$, starting from the left side of the axis, as shown in Figure 4.6. The set $\mathcal{R}_i$ is now fully defined, and we can therefore use the load volumes δ_r as well as the cumulative loads $\Delta^r = \sum_{r' \leq r} \delta_{r'}$.

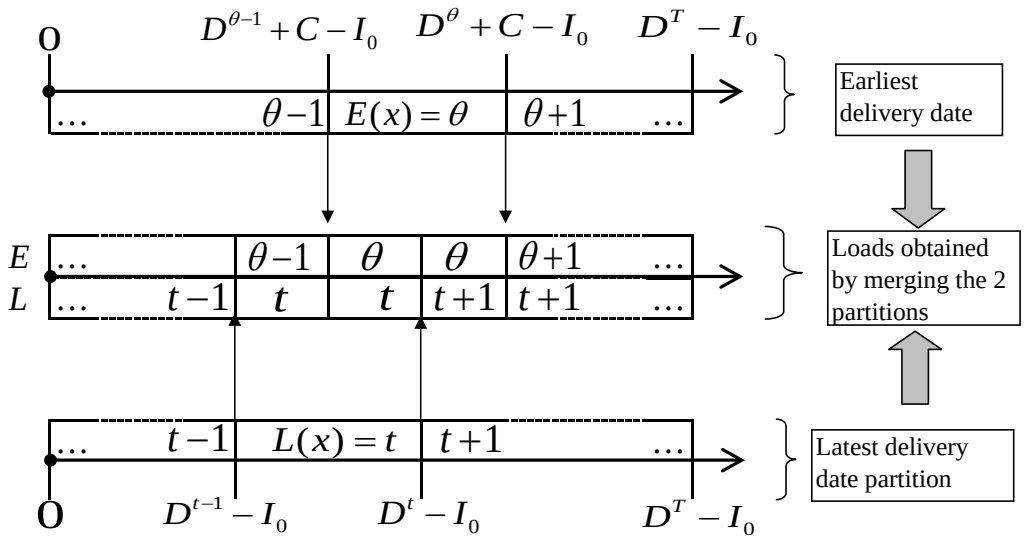

Figure 4.5: Creating the loads by combining the two partitions

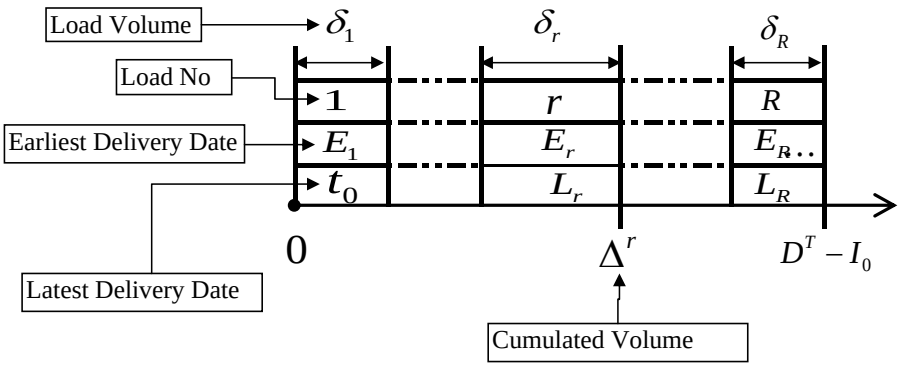

Figure 4.6: The set of loads obtained

4.5.5 Analytical properties of the loads

It is now possible to derive analytical properties on the loads defined. We will first identify $r_{max}(t)$ as the last load with its latest delivery date on day t

$$\forall t > t_0, \quad r_{max}(t) = max\{r \in \mathcal{R}_i : L_r = t\}$$

We therefore have

$$r_{max(t)} = max\{x : L(x) \leq t\}$$

And relation (4.11) allows us to conclude that

$$\forall t \geq t_0, \quad r_{max(t)} = D^t - I_0 \tag{4.13}$$

Another property, relative to the latest delivery dates, can be derived by simply applying relation (4.12) to $x = \Delta^r$:

$$\forall r \in \mathcal{R}_i, \quad (E_r > t) \Leftrightarrow (\Delta^r > D^t + C - I_0) \tag{4.14}$$

Finally, it is important to note that, by applying the two partitionings, we cannot generate more than $T + T = 2 \times T$ loads.

These three properties can be grouped together for further use:

Theorem 1. *At the end of our constructive procedure, the loads built have the following properties:*

$$\Delta^{r_{max}(t)} = D^t - I^0 \qquad \forall t \geq t_0 \tag{4.15}$$

$$\forall r \in \mathcal{R}_i, (E_r > t) \Leftrightarrow (\Delta^r > D^t + C - I_0) \qquad \forall t \in \mathcal{H} \tag{4.16}$$

$$|\mathcal{R}_i| = R \leq 2 \times T \tag{4.17}$$

4.6 Formal transposition of the decisions

Our proof of the equivalence of the two formulations will be based on the following simple concept: we will consider a feasible IRP solution, and show how it can be used

to generate a feasible MVRPD solution with the same cost, and vice versa. Before actually showing the equivalence, we will explain how feasible solutions should be transposed from one problem to another.

a - MVRPD to IRP The transposition of the MVRPD decisions φ_r^{kt} to IRP decisions q^{kt} is simple and straightforward : the decisions are obtained by simply cumulating all the loads delivered on a given day on a given truck: $q^{kt} = \sum_{r \in \mathcal{R}_i} \varphi_r^{kt}$. Transposing the solution from the IRP to the MVRPD will require a more technical procedure.

b - IRP to MVRPD A feasible IRP solution will be described by the set of delivery quantities q^{kt} that satisfy the inventory constraints(3.8) and (3.9). We therefore wish to distribute those quantities among loads to obtain the MVRPD deliveries φ_r^{kt}.

For clarity, this will be done by *aggregating all the trucks* first. Our goal is then to assign the volume distributed in the IRP during a given *day* $\sum_k q^{kt}$ to loads, to obtain the volumes distributed on a given day to a given load $\sum_k \varphi_r^{kt}$. This is done by serving the loads in the *order defined in section 4.5*.

The idea is quite similar to the work done on cumulated deliveries in the previous section, and is therefore very graphical in description. The IRP delivery volumes q^{kt} are first aggregated by day $\sum_{k \in \mathcal{K}} q^{kt}$. These volumes are then accumulated one after another to visualize the cumulated deliveries $Q^t = \sum_{t' \leq t} \sum_{k \in \mathcal{K}} q^{kt'}$. Given the IRP constraint, the total delivery volume is $Q^T = D^T - I_0$, and the cumulated deliveries consequently define a *partition* of $[0, D^t - I_0]$, as shown in Figure 4.7.

We have seen in the previous paragraphs that the loads themselves $(r)_{1 \leq r \leq R}$ induced another partition of $[0, D^t - I_0]$. We will merge these two partitions (load

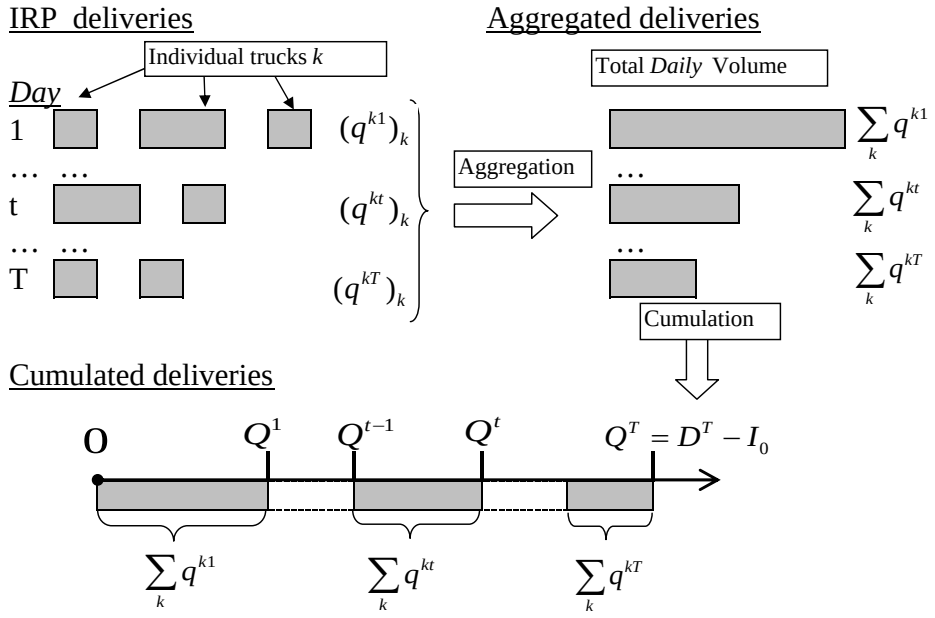

Figure 4.7: Aggregating and cumulating the IRP deliveries to partition $[0, D^T - I_0]$

definition and IRP deliveries) as shown in Figure 4.8, to define the daily MVRPD delivery volumes $\sum_k \varphi_r^{kt}$. The merging can be formally described as follows:

$$\sum_{k\in\mathcal{K}} \varphi_r^{kt} \neq 0 \Rightarrow \begin{cases} \Delta^r > Q^{t-1} & \text{The load did not "fit" in the previous day} \\ \quad \text{and} & \\ \Delta^{r-1} < Q^t & \text{The previous load did not "fill" the current day} \end{cases}$$

and

$$\sum_k \sum_{r\in\mathcal{R}_i} \varphi_r^{kt} = \sum_k q^{kt}$$

$$\sum_{k,t} \varphi_r^{kt} = \delta_r$$

Assigning loads to trucks In the previous paragraph, we have assigned the different loads to delivery days, such that $\sum_k \sum_{r\in\mathcal{R}_i} \varphi_r^{kt} = \sum_k q^{kt}$ is verified. We have not explained, however, how these loads should be dispatched to different trucks. This can be done in any way satisfying $\sum_r \varphi_r^{kt} = q^{kt}$. All the loads of a given IRP customer being located at the same location, this assignment will have

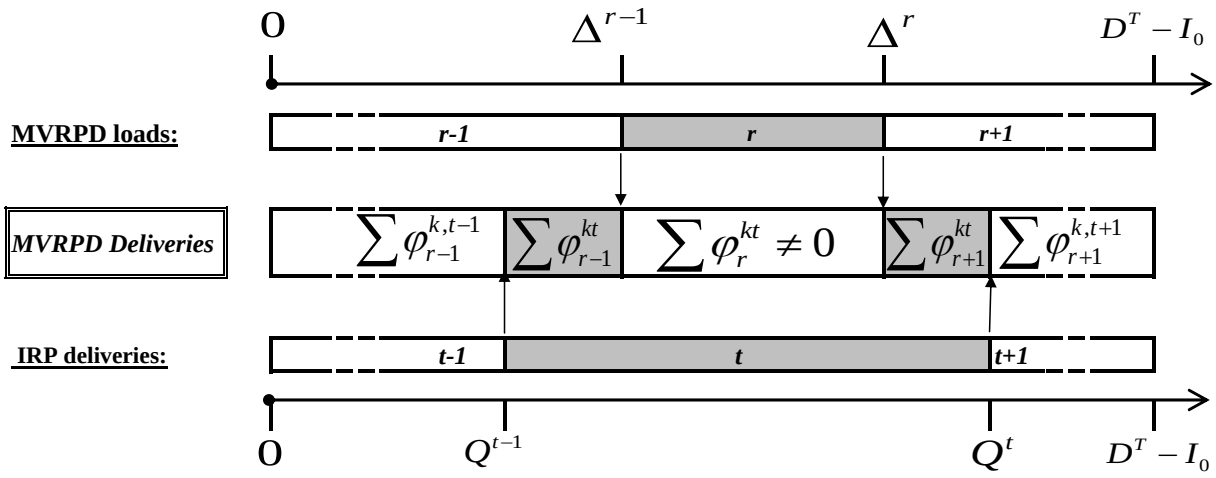

Figure 4.8: Creating the MVRPD deliveries

no impact on the routing itself. We will therefore have additionally:

$$\sum_r \varphi_r^{kt} = q^{kt}, \quad \forall k, t$$

All these properties can be grouped in a theorem for later use:

Theorem 2.

$$\sum_{k \in \mathcal{K}} \varphi_r^{kt} \neq 0 \Rightarrow \begin{cases} \Delta^r > Q^{t-1} & \textit{The load did not "fit" in the previous day} \\ \quad \textit{and} & \\ \Delta^{r-1} < Q^t & \textit{The previous load did not "fill" the current day} \end{cases} \tag{4.18}$$

$$\sum_r \varphi_r^{kt} = q^{kt}, \quad \forall k, t \tag{4.19}$$

$$\sum_{k,t} \varphi_r^{kt} = \delta_r \tag{4.20}$$

We have shown, in Section 4.5, how to build a MVRPD problem from the data of an IRP. Theorem 1 summed up the properties of the MVRPD data obtained. Then we have described the procedure undertaken to transpose a feasible solution of one problem to the other. Theorem 2 gave the properties of the MVRPD decisions obtained when applying this transposition. We will now show that, with these properties, any solution feasible for one problem is also feasible for the other.

4.7 Equivalence of the formulations

In this subsection, we will show that, given any IRP problem, the MVRPD defined by

$$\mathcal{R} = \bigcup_{i \in N} \mathcal{R}_i$$

is fully equivalent to the IRP. We therefore consider a given IRP, and the MVRPD defined by the procedures described in the previous sections.

4.7.1 IRP to MVRPD

Consider a feasible IRP solution, and consider the set of MVRPD decisions φ_r^{kt} defined in section 4.6. We want to check that

- The two solutions have the same cost
- The vehicle capacity restriction (4.3) is respected
- Constraint (4.7), that requires the loads to be delivered within $[E_r, L_r]$ is respected in the MVRPD solution

Note, first of all, that the routing plan defined in the two problems are the same. Indeed property (4.19) ensures that the visits to demand points are the same, given that we located all the loads generated from a given IRP customer at the same geographical location. Therefore, the two feasible solutions will have the ***same cost***. Furthermore, given property (4.19), and since the IRP solution respects vehicle capacity, the capacity restriction is respected in the MVRPD.

Consequently, we only have to check whether the constraints (4.7) are all respected.

$$\sum_{k\in\mathcal{K}}\sum_{t=E_r}^{L_r}\varphi_r^{kt}=\delta_r \qquad \forall r\in\mathcal{R}$$

Suppose this constraint is violated for a given load r. Given property (4.20), that ensures that the whole volume of a load is delivered during the planning period, this means that either

1. $(\exists t<E_r:\varphi_r^{kt}\neq 0)$: a delivery reaches load r before its earliest day or
2. $(\exists t>L_r:\varphi_r^{kt}\neq 0)$: a delivery reaches load r after its last day

Case 1: $(\exists t<E_r:\varphi_r^{kt}\neq 0)$

$$(t<E_r)\Rightarrow D^t+C-I^0<\Delta^r \qquad \text{(Given } t<E_r \text{ and Theorem 1)}$$

Suppose now that we have $\Delta^{r-1}<D^t+C-I^0$, that is $\Delta^{r-1}<D^t+C-I^0<\Delta^r$. This would imply that there is a load r' between $r-1$ and r, with an earliest delivery date $E_{r'}>E_{r-1}$, as described in Paragraph 4.5.3. This contradicts the fact that, by construction, load $r-1$ precedes load r. We therefore have $\Delta^{r-1}\geq D^t+C-I^0$.

Additionally:

$$\begin{aligned}(\varphi_r^{kt}\neq 0)&\Rightarrow \Delta^{r-1}<Q^t && \text{(See Theorem 2)}\\ &\Rightarrow D^t+C-I^0<Q^t && \text{(Given } \Delta^{r-1}\geq D^t+C-I^0)\end{aligned}$$

This is impossible, given that our feasible IRP solution respects the customer capacity constraint: Case 1 is impossible.

Case 2: $(\exists t > L_r : \varphi_r^{kt} \neq 0)$

$$
\begin{aligned}
(\exists t > L_r : \varphi_r^{kt} \neq 0) \Rightarrow & Q^{t-1} < \Delta^r && \text{(See Theorem 2)} \\
\Rightarrow & Q^{t-1} < \Delta^{r_{max}(L_r)} && (r_{max}(L_r) \geq r \text{ by definition of } r_{max}) \\
\Rightarrow & Q^{t-1} < D^{L_r} - I_0 && \text{(See Theorem 1)} \\
\Rightarrow & Q^{t-1} < D^{t-1} - I_0 && \text{Given}(L_r < t) \Rightarrow (L_r \leq t-1)
\end{aligned}
$$

This is impossible, given that our feasible IRP solution respects the "no stockouts" constraint: Case 2 is impossible.

We have therefore shown that any feasible IRP solution satisfies the MVRPD constraints.

4.7.2 MVRPD to IRP

Consider a feasible MVRPD solution, and consider the IRP data obtained by aggregating the deliveries to different loads $q^{kt} = \sum_{r \in \mathcal{R}_i} \varphi_r^{kt}$. For the same reasons explained in the previous paragraph, this IRP data does not violate the vehicle capacity constraint, and has the same total cost as that of the MVRPD solution.
Let us first show that no stockouts will occur,$\forall t \in \mathcal{H}$:

$$
\begin{aligned}
Q^t = \sum_{k \in \mathcal{K}} \sum_{t'=1}^{t} \sum_{r \in \mathcal{R}_i} \varphi_r^{kt} &\geq \sum_{k \in \mathcal{K}} \sum_{t'=1}^{t} \sum_{r : L_r \leq t} \varphi_r^{kt} \\
&= \sum_{r : L_r \leq t} \delta_r && \text{given } \sum_{k \in \mathcal{K}} \sum_{t'=E_r}^{L_r} \varphi_r^{kt} = \delta_r \\
&= \Delta^{r_{max}(t)} && \text{by definition} \\
&= D^t - I^0 && \text{(Given Theorem 1)}
\end{aligned}
$$

And the "no stockouts" constraint is therefore respected every day.

Now, we have to check that customer capacity is not exceeded.

$$
\begin{aligned}
Q^t = \sum_{k\in\mathcal{K}}\sum_{t'=1}^{t}\sum_{r\in\mathcal{R}_i}\varphi_r^{kt} &= \sum_{k\in\mathcal{K}}\sum_{t'=1}^{t}\sum_{r:E_r\le t}\varphi_r^{kt} && \text{Because } \varphi_r^{kt}=0 \text{ if } t<E_r\\
&\le \sum_{r\in\mathcal{R}_i:E_r\le t}\delta_r\\
&= \Delta^{r_1} && \text{Where } r_1 \text{ is a load such that } E_{r_1}\le t\\
&\le D^t + C - I_0 && \text{(Given Theorem 1 and given } E_{r_1}\le t)
\end{aligned}
$$

Consequently, the customer capacity constraint is respected.

We have shown that any feasible IRP solution is feasible for the MVRPD, and vice-versa: these two formulations are thus EQUIVALENT.

An example illustrating this equivalence is given in the next section, and will help the reader familiarize with our approach.

4.8 Working on an example

Section 4.8.1 will start by describing the data of a small IRP and we will apply the different steps explained in the previous sections to this example. In Section 4.8.2, the latest delivery dates will apply a first partition of the total delivery volume. Then, in Section 4.8.3, the earliest delivery dates will define another partition that accounts for the customer's limited capacity and Section 4.8.4 will merge these two partitions to obtain loads. Finally in Section 4.8.5, we will consider an IRP solution and show how we build a MVRPD solution from it.

4.8.1 Data of an IRP example

For clarity, we chose to apply our procedure to an example of small size: we will consider a planning horizon of **4 days** and an IRP customer with the following

demand pattern, capacity and initial inventory level:

Capacity	$C = 4$			
Initital inventory	$I^0 = 2$			
Demand	$d^1 = 1$	$d^2 = 2$	$d^3 = 3$	$d^4 = 4$
Cumulated	$D^1 = 1$	$D^2 = 3$	$D^3 = 6$	$D^4 = 10$

With this data, the "no stockouts" inventory constraints (3.8) can be written:

	$0 \leq Q^1$	$3 - 2 \leq Q^2$	$6 - 2 \leq Q^3$	$Q^4 = 10 - 2$
That is:	$0 \leq Q^1$	$1 \leq Q^2$	$4 \leq Q^3$	$Q^4 = 8$

And the constraints enforcing customer capacity (3.9) are simply:

	$Q^1 \leq 5 - 2$	$Q^2 \leq 7 - 2$	$Q^3 \leq 10 - 2$	$Q^4 = 10 - 2$
That is:	$Q^1 \leq 3$	$Q^2 \leq 5$	$Q^3 \leq 8$	$Q^4 = 8$

This numerical IRP instance will be now transposed to a MVRPD

4.8.2 Latest delivery dates

We will proceed here with the first steps of the method explained in Section 4.5, that define the total delivery volume $D_T - I_0$, and apply latest delivery dates to some segment of this volume.

The first day which demand is not covered by the initial inventory $t_0 = min\{t : D^t > I_0\} = 2$ here, since $D^2 = 3 > 2 = I_0$, and $D^1 = 1 \leq 2 = I_0$. Furthermore, the total delivery volume is $D^4 - I_0 = 10 - 2 = 8$.

As illustrated in Figure 4.9, this volume will be partitioned in 3 segments with distinct latest delivery dates.

$L = t_0 = 2$	$\forall x \in [0, D^{t_0} - I_0] = [0, 1]$
$L = 3$	$\forall x \in (D^2 - I_0, D^3 - I_0] = (1, 4]$
$L = 4$	$\forall x \in (D^3 - I_0, D^4 - I_0] = (4, 8]$

We have now to define a different partition of the volume with earliest delivery dates.

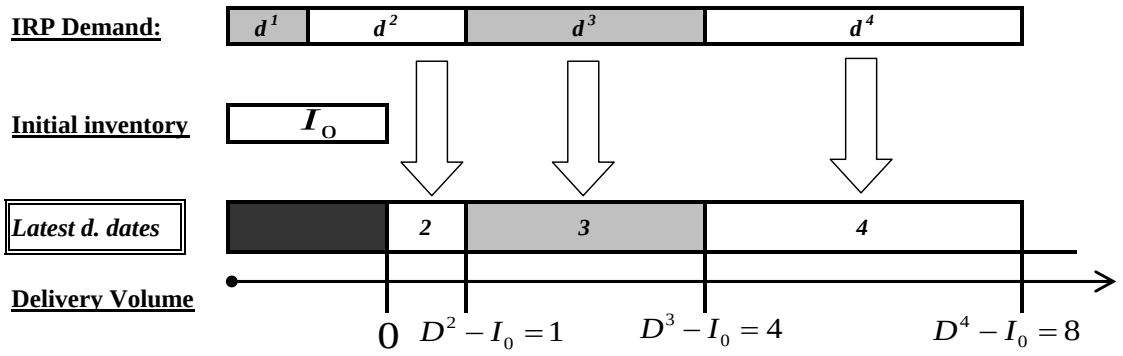

Figure 4.9: Latest delivery dates of the example

4.8.3 Earliest delivery dates

In this section, the total delivery volume is partitioned differently to account for customer capacity. This is done graphically, as described in Figure 4.10, by using the "shifted" cumulated demand to visualize which parts of the delivery volume fit in a given day's inventory. The total delivery volume was partitioned into 4 zones

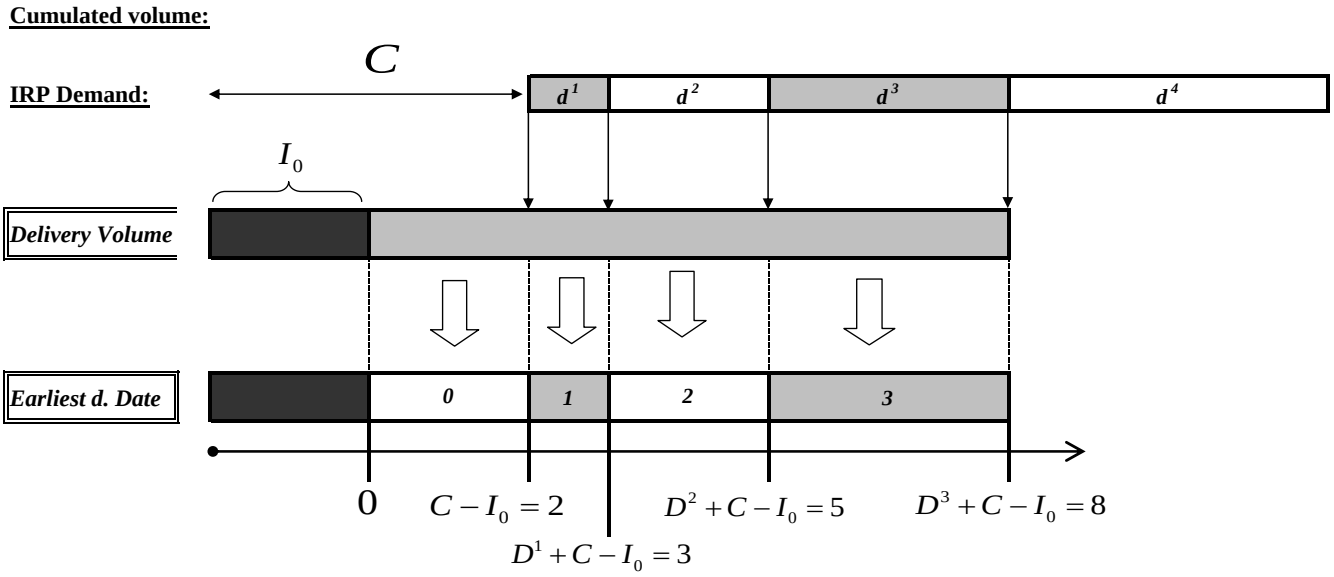

Figure 4.10: Earliest delivery dates of the example

with distinct earliest delivery dates:

$$
\begin{aligned}
E = 0 \quad & \forall x \in [0, C - I_0] = (0, 2] \\
E = 1 \quad & \forall x \in (C - I_0, D^1 + C - I_0] = (2, 3] \\
E = 2 \quad & \forall x \in (D^1 + C - I_0, D^2 + C - I_0] = (3, 5] \\
E = 3 \quad & \forall x \in (D^2 + C - I_0, D^3 + C - I_0] = (5, 8]
\end{aligned}
$$

4.8.4 Merging of the partitions

The loads will now be defined by merging the two partitions previously presented, and by numbering the loads, as shown on Figure 4.11. As the figure shows, a total

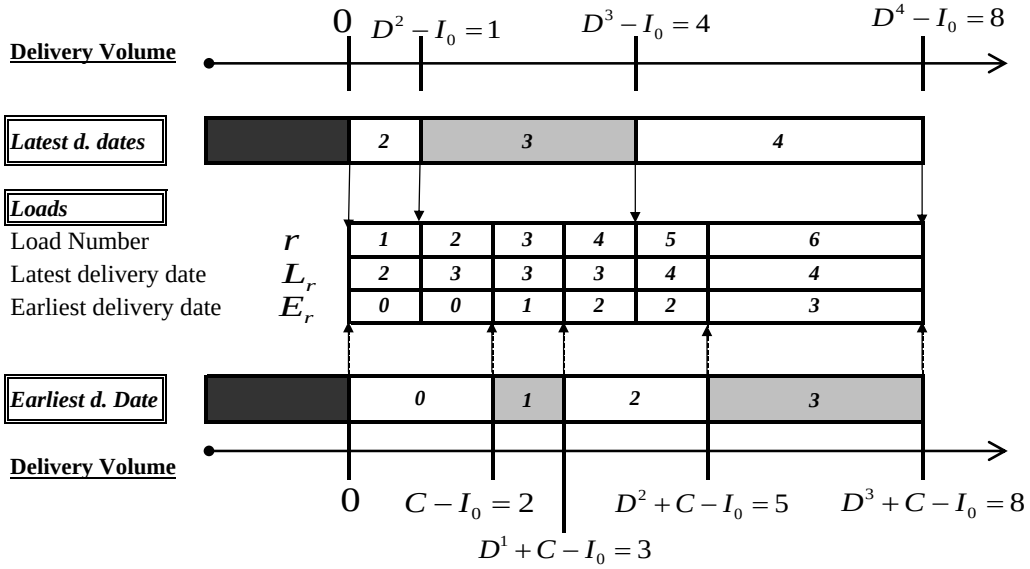

Figure 4.11: Merging of the partition and load numbering

of 6 loads were obtained:

$$
\begin{aligned}
&\text{Load } 1 \quad \delta_1 = 1 \quad E_1 = 0 \quad L_1 = 2 \\
&\text{Load } 2 \quad \delta_2 = 1 \quad E_2 = 0 \quad L_2 = 3 \\
&\text{Load } 3 \quad \delta_3 = 1 \quad E_3 = 1 \quad L_3 = 3 \\
&\text{Load } 4 \quad \delta_4 = 1 \quad E_4 = 2 \quad L_4 = 3 \\
&\text{Load } 5 \quad \delta_5 = 1 \quad E_5 = 2 \quad L_5 = 4 \\
&\text{Load } 6 \quad \delta_6 = 3 \quad E_6 = 3 \quad L_6 = 4
\end{aligned}
$$

So far, we have used our example to illustrate how the IRP data of a given customer should be used to generate a consistent set of MVRPD loads. We will now

show how, starting with a feasible IRP solution, one is supposed to build a feasible MVRPD solution.

4.8.5 Transposing the decisions

IRP to MVRPD To illustrate how decisions are transposed from the IRP to the MVRPD, we will pick a *given* delivery policy that is feasible for the demand parameters of the IRP customer we study in this example. The daily delivery quantities are listed hereafter, and one can verify, using the constraints listed in Section 4.8.1, that such a delivery schedule respects all the IRP constraints. Without loss of generality, and for simplicity, we assumed that only one vehicle was available, and therefore omit the index k of the vehicle.

Deliveries	$q^1 = 2$	$q^2 = 2$	$q^3 = 2$	$q^4 = 2$
Cumulated	$Q^1 = 2$	$, Q^2 = 4$	$Q^3 = 6$	$Q^4 = 8$

We wish to build a feasible MVRPD solution using these IRP decisions. This is done, as described in Section 4.6, by going through the deliveries in the order they appear, and by assigning the delivery volumes to the different loads, in the order that these loads appear in. Figure 4.12 illustrates the result obtained for our numerical example. This gives a total of seven deliveries:

Day 1:	1 unit goes to load 1 and 1 unit to load 2	$\rightarrow$	total of $2 = q^1$ units
Day 2:	1 unit goes to load 3 and 1 unit to load 4	$\rightarrow$	total of $2 = q^2$ units
Day 3:	1 unit goes to load 5 and 1 unit to load 6	$\rightarrow$	total of $2 = q^3$ units
Day 4:	2 units go to load 6	$\rightarrow$	total of $2 = q^4$ units

The deliveries φ_r^{kt} defined in this way obviously respect the MVRPD constraints (4.7), that is $\sum_t \varphi_r^t = \delta_r$.

MVRPD to IRP The inverse transformation is straightforward: a feasible MVRPD solution $(\varphi_r^{kt})_{r,k,t}$ can easily be transformed into a feasible IRP solution, by simply choosing $q^{kt} = \sum_r \varphi_r^{kt}$, that is by aggregating the deliveries made on day t to all demand points r. Given the definition of the loads, no stockouts will occur, and the customer capacity won't be exceeded.

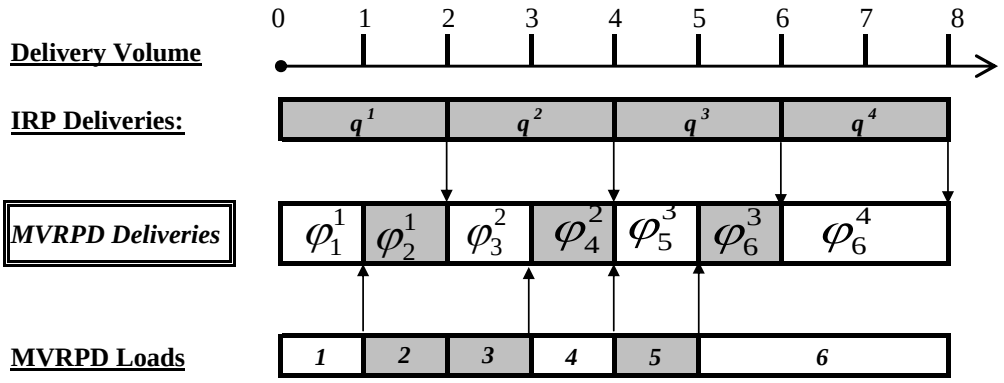

Figure 4.12: Assigning delivery volumes to specific loads

This was the last step we wanted to illustrate in our example. Starting from the data of an IRP customer, we generated MVRPD loads with latest and earliest delivery dates. Then, we showed how to transpose a feasible IRP solution (set of deliveries) to a feasible MVRPD and vice versa.

4.9 Conclusion

We can now summarize and conclude this chapter, that represented the core of our contribution. We started by formulating a variant of the VRP, the multiple-period vehicle routing problem with delivery dates and split demand (MVRPD), where a set of demand points must receive a given delivery volume between two specified dates $[E_r, L_r]$, and where split-deliveries are allowed. We then showed a relationship of equivalence between the IRP and the MVRPD: we explained how a given IRP customer would generate several MVRPD demand points, and how the

latest and earliest delivery dates of those demand points were dictated by the "no stockouts" constraint and customer capacity constraint respectively. A formal proof of the equivalence of the two problems was given in Section 4.7. Finally, Section 4.8 exhibited an example that illustrated how one should build the equivalent problem, and how feasible solutions of the problems should be transposed. The next chapter will make use of this transformation to design a simple, constructive heuristic using techniques from the VRP literature.

Chapter 5

A constructive heuristic

5.1 Introduction

In the previous chapter, we showed how, for any IRP, we could build an equivalent MVRPD. The goal of this transformation is, as already highlighted, to reduce the interactions between the different inventory constraints, and to obtain a problem that is easier to manipulate. Indeed, starting from a *time-driven* IRP, we are now considering a rich, but still *order-driven* VRP: the MVRPD with split deliveries. This chapter will focus on proposing a heuristic to find good solutions to this equivalent problem.

The transposition allows us to use a great number of tools that can be found in the exhaustive VRP literature available. Our MVRPD has however, two specificities that require us to develop a customized procedure. Firstly, allowing split-deliveries restricts us to a less-studied category of VRP solutions: the Split-Delivery VRP (SDVRP). Secondly, we will have to include specific constraints to account for the due-dates and earliest delivery dates requirements.

5.2 Description of the heuristic

We will now describe the constructive heuristic we designed to fit the structure of our problem, the MVRPD. We will start by some general facts about our heuristic, followed by a description of the different modules that compose it.

5.2.1 Generalities

The MVRPD It is useful to quickly remind the reader of the description of the MVRPD given in Section 4.2 : during a horizon of T days, a set N' of demand points are served by a depot 0. Each demand point $r \in N'$ has a demand δ_r of product, that must be delivered between days E_r and L_r. The travel costs between 2 nodes c_{rs} are known.

We will assume from now on that these costs are proportional to the euclidian distance, with a proportionality factor of one: $c_{rs} = \sqrt{(x_r - x_s)^2 + (y_r - y_s)^2}$. Our problem is therefore, from now on, a **symmetrical** routing problem.

K vehicles of capacity W are available everyday. We **allow the demand of a customer to be split among vehicles or days**. The objective is to minimize the transportation costs.

Motivation As already developed in the previous chapter, an IRP problem with a set of n customers and a planning horizon of T days will become a MVRPD problem with $n \times T$ to $2 \times n \times T$ demand points after transformation. Hence, with as little as 10 nodes and a planning horizon of 10 days, we could end up considering a MVRPD with up to 200 demand points. Consequently, it is difficult to try to find an optimal solution to our instances. Indeed, one must keep in mind that the very elaborate

cutting-plane technique developed by Belenguer *et al.* (2000) was able to find an optimal solution to *one* instance of a SDVRP with 50 nodes, and that our problem has a structure very similar to this SDVRP, but with additional constraints.

Heuristic structure: 5 independent modules We will therefore focus on designing a fast-running constructive heuristic inspired by earlier studies in the literature. This very simple algorithm works with 5 different "modules".

- *Module 0 - REDUCE*: Reduces the problem to consider only demand points with a volume that fits into the vehicle $\delta_i \leq W$
- *Module 1 - INITIAL*: Construction of a first, feasible solution using a customized savings algorithm.
- *Module 2 - IMPROVE*: Improvement of the solution using the 2-Opt, Or-Opt, exchange, cross and relocate operators.
- *Module 3 - IMPROVE SPLIT*: Improvement of the solution using k-split interchange and route addition.
- *Module 4 - VOLUME OPT*: Optimizes the vehicle occupation by adding extra volume to some deliveries.

INITIAL and *IMPROVE* are run once, then a loop iterates *K SPLIT,ROUTE ADD* and *IMPROVE* until no improvement is found. *VOLUME OPT* is run on the final routing plan.

Notations We will now set up a few simple notations that will help us describe the different modules contained in our heuristic.

In a routing plan, a route r will be a tour starting from and ending at the depot, executed by a vehicle k_r on day t_r, and serving a set S_r of customers. The total load on that route will be $W_r = \sum_{i \in S_r} q_i^{k_r t_r}$. The spare capacity will then be $s_r = W - W_r$. On a partial or complete routing plan, the set of unused vehicles will be denoted as V. A "vehicle" in V will be denoted by (k, t), k being the index of the physical vehicle, and t being its operating day.

Now that these general facts have been explained, we can describe the different modules of our heuristic.

5.2.2 *Module 0* – REDUCE: Reduction of the problem

As no restricting hypotheses have been made on this matter, we have to explicitly consider the situation where a demand point can have a demand volume exceeding the vehicle capacity. This is done, as in Archetti *et al.* (2003), by creating $\lfloor \frac{\delta_i}{W} \rfloor$ direct trips with fully loaded vehicles to any such demand point. The vehicle and day used to serve a direct delivery are *randomly* chosen in the set $V_i = \{(k, t)/E_i \leq t \leq L_i\}$, that is the set of vehicles with operating days compatible with the delivery dates of load i. Once all the direct deliveries are created for a given customer, only the remaining demand $\delta_i - \lfloor \frac{\delta_i}{W} \rfloor$ is considered when examining the demand point for routing. This procedure is repeated for all the customers, which gives us a *reduced* instance. We will work with this reduced instance from now on.

It has been shown in Archetti *et al.* (2004) that, when solving to optimality, considering the above-described reduced instance gives a worst-case error of 3/2 when distances in the network satisfy the triangle inequality.

5.2.3 *Module 1*– INITIAL

The first module, *INITIAL*, builds up a feasible solution using a custom made sequential savings algorithm, described hereafter. If the savings algorithm could find a feasible solution, a greedy algorithm is then applied to the problem to build a feasible initial plan, regardless of the cost.

We say that an arc (i, j) is *compatible* with a partially scheduled vehicle k on day t if it satisfies the delivery dates constraints: $E_i \leq t \leq L_i$ and $E_j \leq t \leq L_j$. The module can be written as follows:

1. *Initialization*: Compute the savings $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ for $i, j = 1, ..., n$ and $i \neq j$ and rank them in a non-increasing fashion in a list L. Initialize the set of available vehicles V, by removing the vehicles used in the previous module REDUCE.

2. *Choice of the vehicle*: If V is empty go to step 5. Else **randomly** pick up a vehicle (k, t) in V.

3. *Route initialization*: Starting from the top, scan L for an arc (i, j) compatible with vehicle (k, t), and that satisfies $\delta_i + \delta_j \leq W$. If no such arc is found, remove the vehicle from V and go to step 2. Else, create the route $(0, i, j, 0)$ with the chosen vehicle, and remove arc (i, j) from L.

4. *Route Merge*: Scan L to find an arc (i, j) compatible with vehicle (k, t) on day t, that can be feasibly added to the start or end of the vehicle route. If no such arc is found, remove the vehicle (k, t) from V, and go to step 2. If such an arc is found, append the arc to the route and repeat step 4.

5. *Termination of savings procedure*: If all demand points are on a route, the algorithm is successful: stop. Otherwise, it could not build a feasible solution: go to step 6.

6. *Greedy algorithm*: assigns loads to vehicles regardless of cost considerations.

   (a) Initialize the set of unassigned loads as $U = \{\delta_r\}_{1 \leq r \leq N'}$ and the current day as $t = 1$.

   (b) If $t = T + 1$ terminate: a feasible assignment of the loads was found only if $U = \{\}$. Else initialize the current vehicle as $k = 1$.

   (c) If $k = K + 1$ set $t = t + 1$ and go to (b). Else initialize the available capacity of the current vehicle to $W_{kt}^{free} = W$. Let $U_{kt} = \{r \in U : E_r \leq t\}$ the set of unassigned loads that can be delivered before t.

   (d) If $U_{kt} = \{\}$ then set k=k+1 and go to (c). Else let $r_{kt} = arg(min\{L_r : r \in U_{kt}\})$ the load with the most constrained delivery date that has not been considered for current vehicle yet and set $U_{kt} = U_{kt} - \{r_{kt}\}$. If $L_{r_{kt}} < t$ then terminate: no feasible solution could be built. Else if $\delta_r > Q_{kt}^{free}$ then go to (d): the load does not fit in the current vehicle. Else *assign load* r_{kt} *to vehicle* k *on day* t, set $U = U - \{r_{kt}\}$, set $W_{kt}^{free} = W_{kt}^{free} - \delta_{r_{kt}}$ and go to 4.

   If the greedy procedure finds a feasible vehicle-load assignment, route each truck using a classic savings algorithm.

Note that, in step 2 of INITIAL, a vehicle (k, t) is chosen **randomly**. Several other methods(start from the first day, start from the last day, start from the busiest day, start from the least busy day, iterate through the planning period etc.) were experimented but none gave significantly better results. We therefore adopted the randomization method. Moreover, we believe that this randomization allows us to overcome cases of infeasibility, by building the schedule in a different order. The reader might have noticed that, in the solution built by INITIAL, no split deliveries were performed. This will be examined later.

5.2.4 *Module 2*–IMPROVE

This section describes a procedure that takes as input a feasible solution and tries to improve it by applying elementary modifications to the routing plan. Our local improvement procedure uses 5 different operators that are described hereafter:

Intra-Route improvement: the 2-Opt Operator In a 2-Opt neighborhood, 2 arcs on the same route are removed and reconnected to improve the total cost of the route, as illustrated in Figure 5.1:

1. If all routes have been examined, go to step 3. Choose a route r.
2. If all combinations of 2 arcs have been examined on r, go to step 1. Else, choose two arcs from the vehicle route, and try the other possible reconnection of the remaining parts of the route. If the move reduces total cost, implement it. Repeat step 2.
3. End.

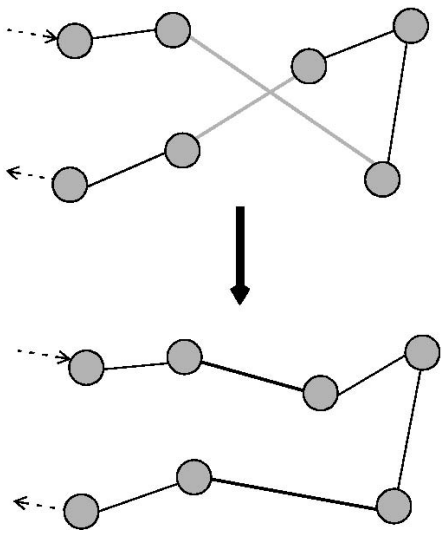

Figure 5.1: A basic arc interchange in the 2-opt procedure

Intra-Route improvement: the Or-Opt Operators This procedure, suggested by Or (1976) tries to relocate consecutive visits within a given route(see Figure 5.2):

1. If all routes have been examined, go to step 5. Choose an existing route r.
2. If all **single** visits have been examined go to step 3. Else, choose **one** visit from the vehicle route, and try to relocate it elsewhere in the route. If the move reduces total cost, implement it. Repeat step 2.
3. If all of **two consecutive** visits have been examined go to step 4. Else, choose **two consecutive** visits from the vehicle route, and try to relocate them elsewhere in the route. If the move reduces total cost, implement it. Repeat step 3.
4. If all **three consecutive** visits have been examined go to step 1. Else, choose **three consecutive** visits from the vehicle route, and try to relocate them elsewhere in the route. If the move reduces total cost, implement it. Repeat step 4.
5. End.

Note that, when implementing the 2-Opt and Or-Opt operators, there is no need to check for feasibility, since the arcs are moved within given routes, and both capacity constraints and delivery dates are known to be compatible.

Inter-route improvement: the relocate operator This operator removes a visit from a given route and tries to reinsert it in *another* route, as illustrated in Figure 5.3:

1. If all routes have been examined, go to step 3. Else, choose a route r.

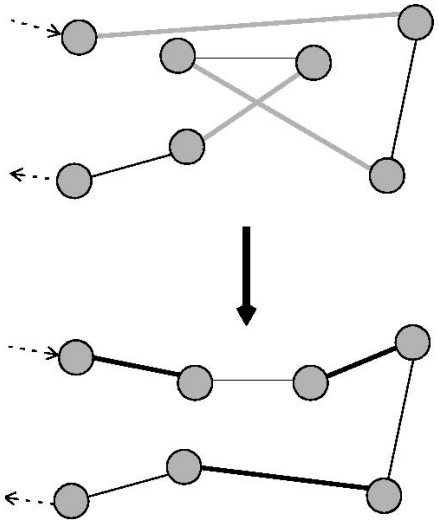

Figure 5.2: Relocation of 2 consecutive visits in the Or-Opt procedure

2. If all visits have been examined go to step 1. Else, choose a visit from r, and try to relocate it in another route. If the move reduces total cost, and if it is feasible, implement it. Repeat step 2.

3. End.

Note that, for this operator, feasibility has to be checked when transferring a visit from one vehicle to another. We have to check whether the delivery dates of demand point i are compatible with the new vehicle route r' executed on day $t_{r'}$ ($E_i \leq t_{r'} \leq L_i$) and whether the new vehicle has enough free capacity to carry the new load ($\delta_i \leq s_{r'}$).

Inter-route improvement: the exchange operator This operator swaps visits belonging to different routes, as shown in Figure 5.4:

1. If all pairs of existing routes have been examined, go to step 3. Choose **two** routes r and r'.

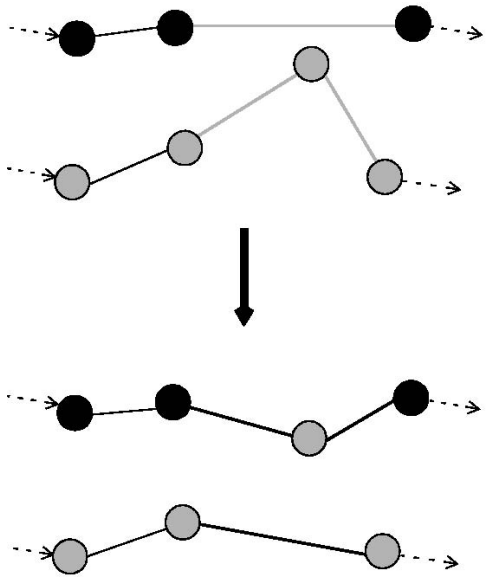

Figure 5.3: Relocate operator: Relocation of a visit to another vehicle

2. If all pairs of visits have been examined go to step 1. Else, choose **one visit in each of the vehicle route**, and try to swap them. If the move reduces total cost, and if it is feasible, implement it. Repeat step 2.

3. End.

Again, for this operator, feasibility has to be checked when swapping visits. When exchanging visit i on route r with visit i' on route r', delivery dates must be compatible with the new vehicle: $E_i \leq t_{r'} \leq L_i$ and $E_{i'} \leq t_r \leq L_{i'}$. Capacity has to be checked as well: $\delta_{i'} \leq s_r + \delta_i$ and $\delta_i \leq s_{r'} + \delta_{i'}$.

Inter-route improvement: the cross operator The cross operator looks for cost improvements by exchanging the end-parts of two routes. See the illustration in Figure 5.5.

1. If all pairs of routes have been examined, go to step 3. Choose **two** routes r and r'

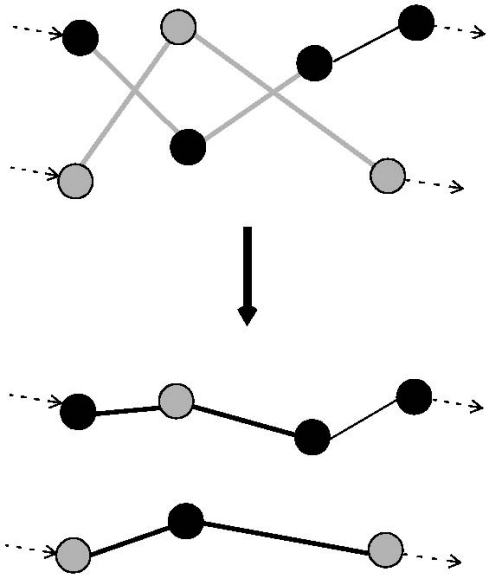

Figure 5.4: The exchange operator

2. If all pairs of visits have been examined go to step 1. Else, choose two visits: i in r and i' in r'. Try to swap *all* the visits following i with *all* the visits following i'. If the move reduces total cost, and if it is feasible, implement it. Repeat step 2.

3. End.

Feasibility check is more complicated in this case. If $S_{r,i}$ is the set of visits following i in r and $S_{r',i}$ is the set of visits following i' in r', we must have:

$$\begin{cases} \text{Delivery dates:} & E_j \leq t_{r'} \leq L_j & \forall j \in S_{r,i} \\ & E_{j'} \leq t_r \leq L_{j'} & \forall j' \in S_{r',i'} \\ \text{Capacity constraint:} & \sum_{j \in S_{r,i}} \delta_j \leq s_{r'} + \sum_{j' \in S_{r',i'}} \delta_{j'} & \\ & \sum_{j' \in S_{r',i'}} \delta_{j'} \leq s_r + \sum_{j \in S_{r,i}} \delta_j & \end{cases}$$

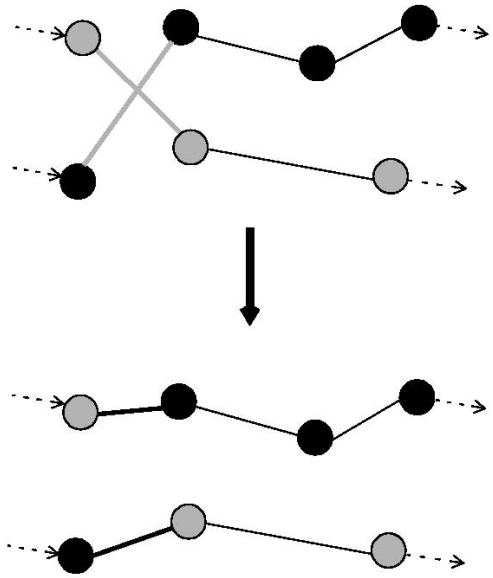

Figure 5.5: The cross operator

Structure of Module 2 - IMPROVE Now that all the operators are defined clearly, the description of our module IMPROVE is straightforward: it consists in executing the 2-opt, Or-opt, relocate, exchange and cross procedures, in that order. If any improvement is found, the 5 procedures are repeated, until no more improvements are found.

The reader should take note that, for the five operators introduced in the previous paragraphs, a *first-accept* strategy was adopted. That is, whenever a move reduces total cost, it is implemented. We tested another strategy, the *best-accept* strategy, that looks for the move that will produce the **biggest cost reduction** to implement it, in each operator. This strategy turned out to be very time-consuming, and did not deliver significantly better results.

5.2.5 *Module 3*– IMPROVE SPLIT

Both the INITIAL and IMPROVE modules, inherited from the classic CVRP literature deal with a routing problem where demand points are allowed to be visited

only once. In other words, the demand volume cannot be *split* between several trucks. We will now develop how, starting from such a feasible solution, we can seek cost improvement by allowing split demand. This procedure is an adaptation of the work of Dror & Trudeau (1989) on the SDVRP, that simply takes into account the multiple-period aspect of our problem.

We therefore consider a feasible routing plan where, again, each route r, served by a vehicle k on day t has a spare capacity of $s_r = W - \sum_{i \in N'} q_i^{kt} y_i^{kt}$

Two subroutines are going to be applied to this solution to reduce cost by considering split deliveries: the **k-split interchange** and the **route addition**.

5.2.5.1 The k-split interchange

This first subroutine examines the current solution for potential split deliveries. We will start by describing a specific case of the k-split interchange, the 2-split interchange, that will be extended later for $k \geq 3$.

Consider 3 routes that are numbered 1, 2 and 3, and served respectively by vehicles (k_1, t_1), (k_2, t_2) and (k_3, t_3). Suppose we have s_1,s_2>0, and let p be a node served by route 3 satisfying:

$$\begin{cases} \text{Delivery dates:} & E_p \leq t_1, t_2 \leq L_p \\ \text{Capacity constraint:} & s_1 + s_2 \geq q_p^{k_3 t_3} \end{cases}$$

Let i_1, j_1 and i_2, j_2 be consecutive points on routes 1 and 2 respectively. Let b_p and a_p be the nodes served immediately *before* p and *after* p on route 3. The cost savings obtained by splitting the delivery volume $q_p^{k_3 t_3}$ between routes 1 and 2, and by removing the delivery to node p from route 3 can therefore be written as:

$$SAV_2(p) = c_{i_1 j_1} + c_{i_2 j_2} + c_{b_p p} + c_{p a_p} - c_{p j_1} - c_{i_1 p} - c_{i_2 p} - c_{p j_2} - c_{b_p a_p}$$

This formulation can easily be understood by noting that, in order to split demand of p between routes 1 and 2, 4 arcs needed to be removed, and 5 other added, as shown in Figure 5.6.

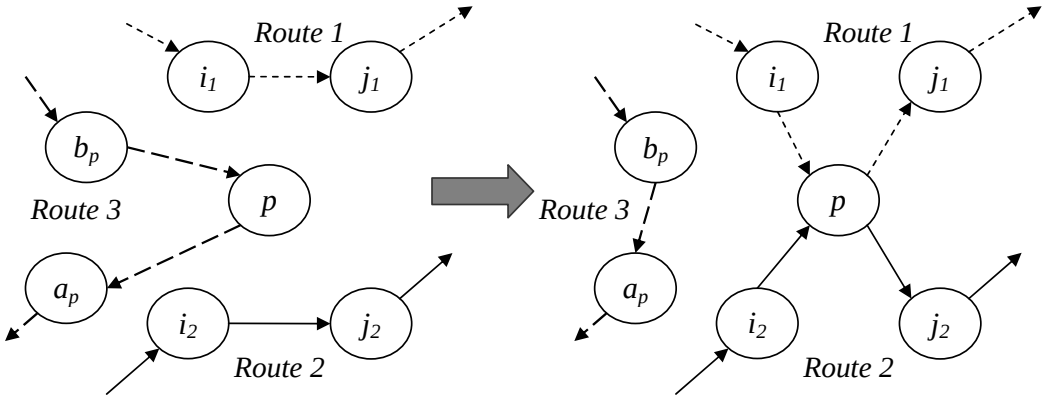

Figure 5.6: Splitting a delivery across two routes

It is straightforward to extend the 2-split interchange to a r-split interchange. A demand point p served by a vehicle (k, t) can be split among r routes $1, \cdots, r$ if :

$$\text{Delivery dates:} \qquad E_p \leq t_l \leq L_p, \forall l \in \{1, \cdots, r\}$$

$$\text{Capacity constraint:} \qquad \sum_{l=1}^{r} s_l \geq q_p^{kt}$$

And the cost savings induced is:

$$SAV_k(p) = \sum_{l=1}^{r} (c_{i_t j_t} - c_{p j_t} - c_{i_t p}) + c_{b_p p} + c_{p a_p} - c_{b_p a_p}$$

Three technical issues need to be raised to fully describe the k-split interchange: firstly, when examining a potential k-split where the customer p is already split between 2 or more routes, the procedure examines all split configurations for demand point p, and not only the additional splits that leave existing splits unchanged. Point p is therefore removed from *all* the routes serving it, before examining all the combinations involving vehicles with sufficient spare capacity and compatible delivery days.

Secondly, it is necessary to explain how the demand volume of node p is shared among the k routes. This is done by ordering those routes in a list, in non-decreasing order of their spare capacity s_r. Starting at the top of the list, each route is "filled up", which means it will deliver a volume s_r to p. This is repeated until the whole volume is dispatched.

Thirdly, our k-split interchange procedure was implemented in a *best-accept* framework: *all* potential splits are examined, and the move leading to the highest cost savings is implemented. While this strategy may require more computational time than the *first-accept* strategy, it provided significantly better computational results.

5.2.5.2 Route addition

This second subroutine was also adapted from the work of Dror & Trudeau (1989), to take into account the delivery date constraints of our problem. It arises from the fact that in some cases, the addition of a new route that regroups split deliveries may reduce total cost. An example is shown in Figure 5.7, where, when demand point 2 is split between two routes, the total cost is $c_{01} + c_{12} + c_{20} + c_{02} + c_{23} + c_{30}$. When node 2 is routed alone, the total cost is $c_{01} + c_{10} + c_{02} + c_{20} + c_{03} + c_{30}$, which makes a difference of $c_{12} + c_{23} - c_{10} - c_{30}$.

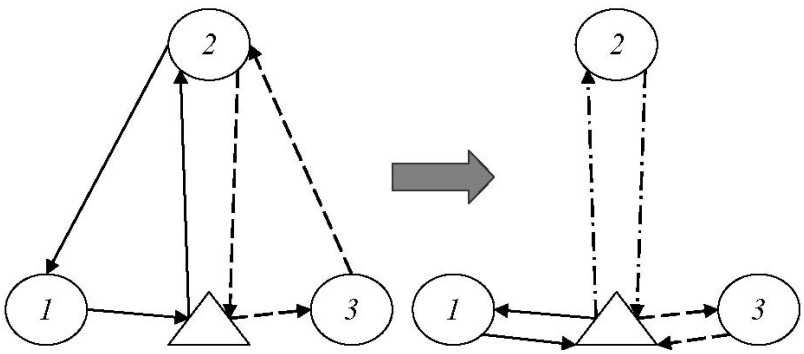

Figure 5.7: The route addition procedure

In their procedure, Dror & Trudeau (1989) described how to implement the route addition procedure to regroup k-splits, with $k \geq 2$. In our experiments, we did not find any route addition achieving positive savings with $k > 2$. To save computational time we restricted ourselves to regrouping 2-splits when implementing this route addition procedure.

When considering a node p in which demand is split among two routes 1 and 2, we have to check nine (9) possible configurations for the composition of the new route: these are all the possible combinations that can result from adding a new route containing the node p, and possibly one or two other nodes, taken among the immediate successor and predecessor of p in routes 1 and 2. All the combinations are listed in Table 5.1. In that table, a_{ip} and b_{ip} are used to name, respectively, the visits immediately *after* and *before* p on route $i, i = 1, 2$.

Feasibility has to be checked as follows: for any combination, let S_{new} be the set of nodes served by the route newly added, as listed in the column *New Route* of Table 5.1. Define $L_{S_{new}} = min\{L_i, i \in S_{new}\}$ and $E_{S_{new}} = max\{E_i, i \in S_{new}\}$, the most constraining earliest and latest delivery dates. If $E_{S_{new}} > L_{S_{new}}$, or if $\sum_{i \in S_{new}} \delta_i > W$, the configuration is infeasible. Else, define $V(S_{new}) = \{(k, t) \in V, E_{S_{new}} \leq t \leq L_{S_{new}}\}$ the set of unused vehicles with an operating day compatible with all the visits in S_{new}. If $V(S_{new})$ is empty, the configuration is infeasible. Else *randomly* choose a vehicle in $V(S_{new})$ and build the new route as in the chosen configuration. The route addition moves are executed, like the k-split interchange, on a *best-accept* basis: all potential moves are examined, and the move producing the highest cost decrease is implemented.

Route Configurations			
No.	Route 1	Route 2	New Route
1	$0, b_{1p}, a_{1p}, 0$	$0, b_{2p}, a_{2p}, 0$	$0, p, 0$
2	$0, a_{1p}, 0$	$0, b_{2p}, a_{2p}, 0$	$0, b_{1p}, p, 0$
3	$0, b_{1p}, 0$	$0, b_{2p}, a_{2p}, 0$	$0, p, a_{1p}, 0$
4	$0, b_{1p}, a_{1p}, 0$	$0, a_{2p}, 0$	$0, b_{2p}, p, 0$
5	$0, b_{1p}, a_{1p}, 0$	$0, b_{2p}, 0$	$0, p, a_{2p}, 0$
6	$0, b_{1p}, 0$	$0, b_{2p}, 0$	$0, a_{1p}, p, a_{2p}, 0$
7	$0, a_{1p}, 0$	$0, a_{2p}, 0$	$0, b_{1p}, p, b_{2p}, 0$
8	$0, a_{1p}, 0$	$0, b_{2p}, 0$	$0, b_{1p}, p, a_{2p}, 0$
9	$0, b_{1p}, 0$	$0, a_{2p}, 0$	$0, a_{1p}, p, b_{2p}, 0$

Table 5.1: Route addition configurations, for 2-split deliveries

5.2.6 *Module 4*– VOLUME OPT

This last module concerns only the vehicle occupation and the customer's inventory level. It will leave the routing plan, and therefore the total cost, unchanged. Remember that, in Section 3.4, we restricted our search to solutions that leave all customers with empty inventory. If this is optimal when solely considering the routing cost, it can penalize alternative performance measures that will be used later. We will therefore apply the following procedure to our solution. It increases the volume delivered to customers when customer and truck capacities allow it. This is done as follows:

1. Initialize $t = T - 1$ (start from the end of the planning period)
2. If $t < 0$ STOP. Else let V be the set of non-idle vehicles operating on day t.
3. If V is empty set $t = t - 1$ and go to step 2. Else pick up a vehicle $k \in V$, and let S_k be the set of customers visited by k on day t
4. If S_k is empty, remove k from V and goto step 3. Else pick a customer $i \in S_k$.

5. Let $Q_{free}(k)$ be the available capacity on the vehicle. If $Q_{free}(k) = 0$, remove k from V and go to step 3. Let $Q_i^{max}(t) = min_{t' \leq t}(C_i - I_i^{t'})$, the maximum additional volume that can be delivered to i without affecting its capacity constraint during the planning horizon. Deliver an additional volume of $min(Q_{free}(k), Q_i^{max}(t))$ to customer i using vehicle k on day t. Update the inventory of customer i and truck occupation of vehicle k on day t, remove i from S_k and go to step 4.

5.2.7 Discussion on the empty inventory assumption

The previous module highlights the fact that, in Chapter 3, we restricted our search to solutions where the final inventory is empty at the end of the planning period. The *Module 4*– VOLUME OPT tries to compensate this, by delivering as much product as possible, without altering the routing costs. We should point out that this assumption still makes it difficult to buffer slight surges in demand. We believe that this difficulty can be overcome by implementing a rolling horizon framework when planning longer periods. More specifically, we recommend to use the technique implemented in Jaillet *et al.* (2002). Every week, solve the IRP on a 2-week horizon, and implement the first week only. In this fashion, the inventory won't run empty at the end of the first week, and the routing plan for the following period is *anticipated*. Moreover, this allows the planner to update the demand forecast on shorter term, which will lead to more accuracy.

5.3 Summary

This chapter focused on designing a heuristic that builds good feasible MVRPD, and therefore IRP solutions, given the equivalence of the two problems. The heuristic proposed is an adaptation of existing VRP techniques and Split-delivery VRP

improvement methods: an initial schedule is generated using a modified savings algorithm and then improved, using several neighborhood operators. Cost-saving moves splitting the demand of customers between vehicles are then sought and implemented. This heuristic was run on a wide range of instances, and compared with different benchmarks, as described in the next chapter.

Chapter 6

Computational results

In order to assess the quality of the heuristic described in Chapter 5, we carried out a series of computational experiments, which are reported in the following paragraphs. We start in Section 6.1 by introducing the hardware and software used to implement the heuristic, as well as the framework used to generate our instances. Two sets of experiments are then presented, which give two complementary pieces of information on our heuristic's performance. Section 6.2 starts by comparing our heuristic with the optimal solution, obtained with a commercial LP/IP solver. This comparison is carried out on a set of small instances only. We subsequently present, in Section 6.3 a myopic heuristic, that allows us to study instances of bigger size. In the various tables and analyses that follow, our heuristic will be referred as *CONST*, for "constructive" heuristic.

6.1 Generalities

6.1.1 Hardware and software

All the computations were made on the same desktop computer, with a Pentium 4 processor running at 2.6 GHz, and 512 MB of RAM. Our heuristic was implemented

using ILOG Optimization Suite tools: the constraint programming module ILOG Solver 6.0 helped us implement our local search strategy, as it integrates powerful constraint propagation methods. Moreover, the ILOG Dispatcher 4.0 simplified our description of the routing heuristics, given that it includes several constraints inherent to routing problems, such as route connectivity, capacity constraints, and flow conservation. Finally, our heuristic was coded in a C++ program that called these two modules using the C++ based ILOG Concert Technology 2.0.

6.1.2 Generation of the instances

The instances we studied were of various sizes, described by the length of their planning horizon T and the number of IRP customers considered n. For a given problem size, the instances were randomly generated as follows.

- *Distances*: All our instances are euclidian, that is the transportation costs are $c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$, where the (x_i, y_j) coordinates were randomly generated in the interval $[-500, 500]$.
- *Vehicle Capacity*: The vehicle capacity $W = 10,000$ was fixed, and all the other data was defined as fractions of that quantity.
- *Tank Sizes*: The set C_i of customer tank sizes was randomly generated in 2 distinct ranges, $[5, 100]$ and $[5, 200]$. These ranges are expressed as percentages of W. For example, the customers of an instance where the range $[5, 100]$ was used will have their capacities C_i in $[5\%W, 100\%W] = [500, 10,000]$;
- *Daily demand*: For each day of the planning period t and each customer i, the daily demand d_i^t was randomly generated in $[0, C_i]$.
- *Fleet size*: Finally, to avoid infeasibility, we allocated an important daily fleet to each instance. Indeed, on each day of the planning period, $K = n$ vehicles

are available, that is, there are as many vehicles as demand nodes. We will see later in our computational experiments that far less vehicles were needed.

6.1.3 Infeasibility

It is important to highlight that, by choosing an important fleet size, infeasibility problems were overlooked. Remember that, in the heuristic developed in Chapter 5, this infeasibility could be encountered at the end of "*Module 1*– INITIAL". Our focus is on the *cost* performance of our heuristic, and those feasibility issues were not tackled, and would therefore be interesting numerical extensions to this study.

Before describing our two sets of computational experiments, we wish to highlight the fact that, for *all* the instances studied, the solution provided by our MVRPD heuristic transposed into a feasible IRP solution, which empirically validates the procedure described in Chapter 4.

6.2 Comparison of the results with a commercial solver

In this section we describe the comparative experiments we made between *CONST* and the results obtained by a commercial IP solver, ILOG CPLEX 9.0. The original IP formulation presented in section 3.3 of the IRP was fed into the solver, and the comparison was drawn using several instances of reduced size: we considered time horizons $T \in [3, 5]$ and problem sizes $n \in [4, 10]$. As already explained, for each problem size, two runs were done: the instances in *SET 1* have customer tank sizes drawn in $[5\%, 200\%]W$ while the capacities were drawn in $[5\%, 100\%]W$ only for the instances of *SET 2*.

	SET 1		*SET2*	
	OPT	*CONST*	*OPT*	*CONST*
n	(CPU)	*% gap to OPT*	(CPU)	*% gap to OPT*
4	4216.83	4217.23	3914.02	**3914.02**
	(1 s.)	*0.01%*	(3 s.)	***0.00%***
5	3599.31	**3599.31**	3116.07	**3116.07**
	(5 s.)	***0.00%***	(4 s.)	***0.00%***
6	2525.45	**2525.45**	12787.1	13432.7
	(4 s.)	***0.00%***	(2 s.)	*5.05%*
7	11083.1	11436.2	5884.99	6003.6
	(15 s.)	*3.19%*	(1 s.)	*2.02%*
8	4293.56	4445.2	21045	22138.7
	(8 s.)	*3.53%*	(2500 s.)	*5.20%*
9	6728.64	7360.96	15420.1	15646.3
	(1131 s.)	*9.40%*	(40 s.)	*1.47%*
10	10836	11315.3	10922	11114
	(4500 s.)	*4.42%*	(1110 s.)	*1.76%*

Table 6.1: Comparison with a commercial solver, $T = 3$

Tables 6.1, 6.2 and 6.3 display the results obtained for a time horizon $T = 3, 4$ and 5 respectively. For each instance presented in those tables, several results are shown. The column *OPT* gives the value of the optimal cost found by *CPLEX*. Beneath this value, in brackets() is the CPU time, in seconds, needed by CPLEX to solve that particular instance. The second column, *CONST* indicates the cost obtained by our heuristic. Beneath this value, the relative difference with the optimum, computed as $\frac{CONST-OPT}{OPT}$ is displayed in italic as a *percentage(%)*. The instances where our heuristic *CONST* found the optimum are **bolded** in those tables. Note that the *CONST* running times were not reported in the tables. This is because, for all the instances, *CONST* required *less than one second (1 s.)* to build a routing plan.

The overall results are very satisfying. The average difference to the optimum obtained by CPLEX is 5.33%, and most of the instances were less than 10% above

	SET 1		*SET2*	
	OPT	*CONST*	*OPT*	*CONST*
n	(CPU)	*% gap to OPT*	(CPU)	*% gap to OPT*
4	4395.71	**4395.71**	10422.1	10467.5
	(8 s.)	***0.00%***	(3 s.)	*0.44%*
5	7112.62	7617.77	12265	12915.6
	(8 s.)	*7.10%*	(42 s.)	*5.30%*
6	4468.62	4661.21	7181.45	8052.12
	(3540 s.)	*4.31%*	(64 s.)	*12.12%*
7	6086.63	6686.48	13411.2	14837.3
	(210 s.)	*9.86%*	(251 s.)	*10.63%*
8	5192.32	5432.01	15161.2	17392.7
	(5123 s.)	*4.62%*	(4856 s.)	*14.72%*
9	11856	12304.9	13356	14122.7
	(7850 s.)	*3.79%*	(6524 s.)	*5.74%*
10	7522.3	8124.5	18865.23	19413.8
	(8522 s.)	*8.01%*	(10256 s.)	*2.91%*

Table 6.2: Comparison with a commercial solver, $T = 4$

that lower bound, as described in the histogram of Figure 6.1. Moreover, for five instances, *CONST* found an optimal routing plan.

The other interesting fact that comes out of this first study is the CPU time needed by CPLEX to find an optimum. Some small instances were solved in a few seconds. However, as soon as $n \geq 8$ or $t \geq 5$, the running times increase significantly and CPLEX cannot find solutions in less than an hour. The biggest instance took more than 12,000 seconds (more than 3 hours) to be solved, and our heuristic found a solution with a cost 1.62% above optimum, within a second.

We conducted experiments on bigger instances, which are not reported here ($n > 8$ and $T > 10$). For those instances, CPLEX was imposed a maximum running time of 5 hours, and could not find *any* feasible solution to the IRP within that time.

	SET 1		*SET2*	
	OPT	*CONST*	*OPT*	*CONST*
n	(CPU)	*% gap to OPT*	(CPU)	*% gap to OPT*
4	3783.8905	3997.09	12596.2	13758.5
	(12 s.)	*5.63%*	(2 s.)	*9.23%*
5	9203.2	9433.56	13600.25	14252.6
	(3800 s.)	*2.50%*	(4569 s.)	*4.80%*
6	7756.56	8681.6	20170.56	21680.9
	(5423 s.)	*11.93%*	(5462 s.)	*7.49%*
7	9965.37	10890.1	13200.32	14638.8
	(7852 s.)	*9.28%*	(6875 s.)	*10.90%*
8	13856.32	14375.1	31966.2	34595.6
	(7845 s.)	*3.74%*	(7542 s.)	*8.23%*
9	15954.21	17525.6	16805.85	18375.7
	(9521 s.)	*9.85%*	(8554 s.)	*9.34%*
10	11025.23	11204	27782.32	28821
	(12654 s.)	*1.62%*	(9532 s.)	*3.74%*

Table 6.3: Comparison with a commercial solver, $T = 5$

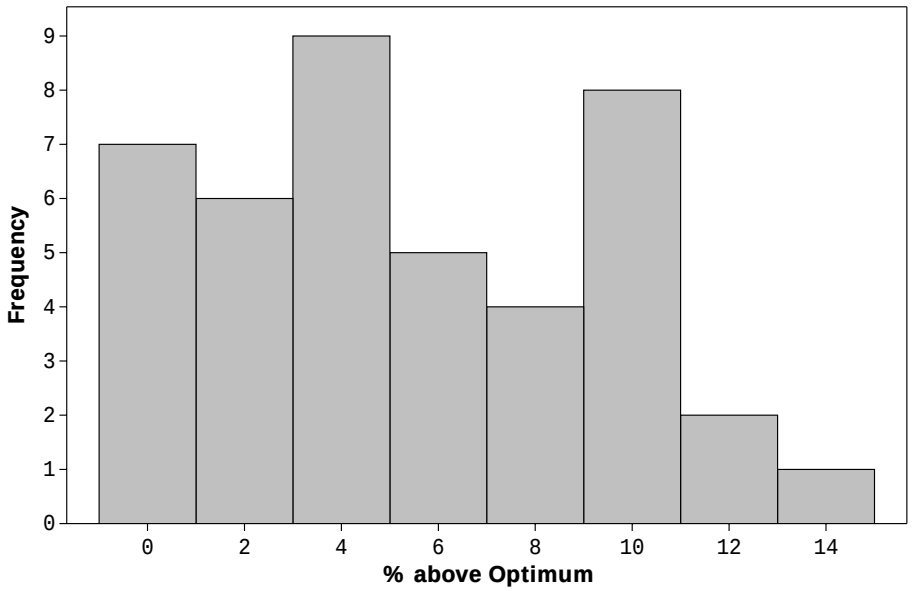

Figure 6.1: Gap to optimality

For real-world instances, it therefore becomes impractical to seek the optimum with a LP solver. However, we still need to assess the performance of our heuristic on bigger instances. This will be done by using another heuristic as a benchmark, as developed in the next section.

6.3 Comparison of the results with a myopic heuristic

To assess the quality of our heuristic on instances of a more consistent size, we compared our results with the output of a myopic IRP heuristic. After describing that heuristic, we will present the data sets used. We will then present several analyses of our results, in terms of total cost, fleet utilization, and inventory level.

6.3.1 Description of the alternative heuristic

The heuristic we used as benchmark will be referred to as *LATEST*. It consists of serving urgent customers only. Campbell *et al.* (2002) presented it as the "rule of thumb" in the industry, and it was used as a benchmark in Bertazzi *et al.* (2002). It can be described with the following rules:

1. Start from the beginning of the planning period, and consider one day at a time. On a given day, list the customers who will experience stockouts if they are not delivered on that day.

2. For each of those customers, create a route that serves the customer, that fills its tank to capacity. Create several full load direct deliveries if the volume needed exceeds the truck capacity.

3. If, on such a route, there is available truck capacity left, extend the route to nearby customers whose inventory is strictly less than capacity, on a least insertion cost basis.

4. All the customers served by a given route will be filled to capacity, except perhaps the last one, which will be served the remaining volume in the vehicle.

5. Do not initiate routes with customers who do not require a delivery on the day considered.

6.3.2 Data sets

Both heuristics were run on an extensive range of instances, obtained by varying the different parameters. 12 sets of problem sizes were examined by choosing $n = 15, 25, 35, 45$ and considering 3 sets of planning horizon $T = 5, 10, 15$, that is one, two and three working weeks. We did not feel the need to investigate longer planning horizons, as it would be unrealistic to forecast a deterministic demand over such a time period. For each problem size, and each capacity set ($[5, 100]$ and $[5, 200]$), four instances were generated and solved. We will consequently examine here a total of $4 \times 3 \times 2 \times 4 = 96$ instances.

The reader should know that several other data sets were examined. Indeed, in our experiments we considered a wide panel of customer capacity sets as narrow as $[5, 33]$ or $[132, 200]$, and we also built instances with correlated customer demand, therefore creating "peak" periods within the planning period. However, throughout these experiments, no significant patterns could be detected. We therefore chose to present here only the general, random instances. We believe this will be more synthetic, and therefore provide more insight on the quality of our solution.

We will now present the results obtained, by exploring three facets of the solution qualities. It is obvious that the *cost* obtained will be the main performance

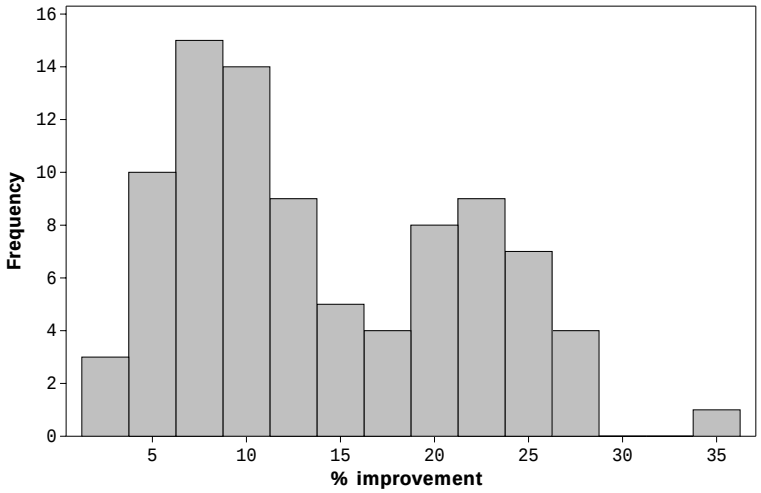

Figure 6.2: Cost improvement of *CONST* over *LATEST*

indicator. We will however study two other performance indicators, namely the fleet utilization, and the inventory behavior. Given that 96 instances have been studied, we will present only average results to highlight the influence of the different parameters. The interested reader can however refer to Appendix A to view the complete computational results.

6.3.3 Comparison of the cost obtained by the heuristics

As indicated, we will start by examining the performances of the two heuristics, *CONST* and *LATEST* in terms of total cost. In all the instances studied, the cost of the solution built by *CONST* was less than the one produced by *LATEST*. We will therefore write our results as a percentage relative improvement $\%imp = \frac{LATEST-CONST}{LATEST}$.

The overall distribution of our results, over all parameters can be seen in the graph in Figure 6.2.

On average, *CONST* finds solutions with a cost 13.6% less than the cost provided by *LATEST*. In some cases a relative improvement of up to 35% has been observed.

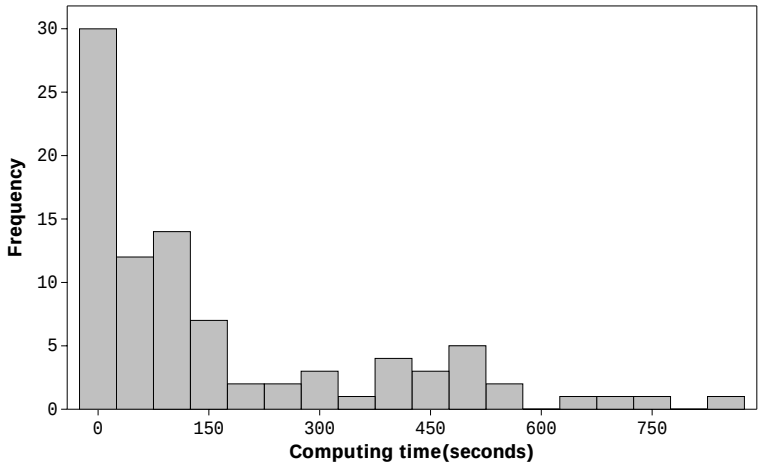

Figure 6.3: Computing time observed

The average computing time observed was 170 seconds, but this unfortunately hides disparity, as shown on the graph of Figure 6.3. Indeed, a minority of instances took up to 15 minutes to be solved. The graph shows however that the majority of the problems were solved within five minutes.

Table 6.4 describes the results obtained with the different parameters used to generate the instances: time horizon, number of customers and customer's capacity range.

Firstly, we can see that the number of nodes n does not really influence the relative performance of *CONST*, except perhaps in the set $n = 45$, which is above average with 17% improvement. This might be due to a deterioration of the quality of *LATEST* when dealing with a large number of customers. Additionally, no noticeable difference can be noticed between the two sets of capacity range.

Secondly, the main result of this table is that, for a very short planning horizon (5 days), our algorithm outperforms *LATEST* by 22%, but this difference drops to 10% for longer horizons. This can be interpreted as follows: *LATEST* proceeds with very costly deliveries, that are useless in short planning horizons, as the volume delivered will not be used. It however becomes more competitive when considering

Parameter	Value	%
T	5	*22.35*
	10	*10.42*
	15	*8.12*
n	15	*13.67*
	25	*13.13*
	35	*13.37*
	45	*17.29*
Range of C_i	[5,100]	*14.71*
	[5,200]	*13.24*
Average		*13.63*

Table 6.4: % improvement of *CONST* over *LATEST*, for different parameters

longer planning horizons.

The other explanation would be that the quality of *CONST* deteriorates with longer planning horizons. This could be possible, and would reflect the difficulty in efficiently distributing the delivery volumes throughout the planning horizons. That is, our algorithm, *CONST*, would lack the ***fluidity*** necessary to produce good solutions to the SDVRP. The heuristic is indeed basically built around a non-split VRP algorithm. One way to verify this would be to implement a heuristic that searches wider solution spaces and gives a higher degree of freedom to the sizes of the delivery volumes and the "split" deliveries. One approach that could be adapted in future research is the SDVRP tabu search heuristic proposed by Archetti *et al.* (2003).

We believe however that the quality of our heuristic does NOT deteriorate with the length of the planning horizon, as other performance indicators, developed hereafter, remain very stable with T. The first of these alternative performance measures is relative to the fleet of vehicles.

6.3.4 Study of the fleet utilization

In what follows, we will focus on two indicators concerning the fleet of vehicles used. The first one is K_{max}, the ***number of trucks needed*** to implement the schedule. It is easily computed as the maximum of the fleet sizes, over all the days of the planning period. The second one, K_{avg}, is the ***average number of trucks used***. From these two metrics we can easily deduce the ratio of idle trucks in the fleet, $K_{idle} = \frac{K_{max} - K_{avg}}{K_{max}}$. The values of these indicators are presented in Table 6.5.

We can see that, overall, *CONST* uses a fleet of 13.3 vehicles, whereas *LATEST* needs an average of 17.6 trucks. This represents a 25% relative difference.

Moreover, the utilization of the fleet within the planning period is far from being optimal with the solutions provided by *LATEST*: on average, 38% of the fleet is idle. *CONST* provides better results, as it has a smaller idle fleet, in absolute and relative terms(only 27%). Note that, while the relative size of the idle fleet stays stable for all parameters in the solutions provided by *CONST*, this is not the case for *LATEST*. The situation worsens with longer planning horizons, and we can see that for $T = 15$, almost half (43%) the fleet needed stays idle. These results are very satisfying, as they deal with a more tactical aspect of the routing plan, that is, the fleet sizing. Moreover, our algorithm would be able to adapt to problems incurring a fixed cost on each vehicle of the fleet.

We will now provide similar analysis on the inventory behavior of the customers.

6.3.5 A note on the inventory behavior

The last facet to be examined when providing solutions to the inventory routing problem is the inventory behavior of the different customers. Given the variety of

		LATEST			*CONST*		
Variable	***Value***	K_{avg}	K_{max}	$\%K_{Idle}$	K_{avg}	K_{max}	$\%K_{Idle}$
T	5	*13.23*	*19.63*	*33%*	*10.32*	*14.22*	*27%*
	10	*11.41*	*18.20*	*37%*	*10.45*	*14.33*	*27%*
	15	*8.67*	*15.11*	*43%*	*8.34*	*11.48*	*27%*
n	15	*5.51*	*9.56*	*42%*	*4.97*	*7.41*	*33%*
	25	*10.12*	*15.67*	*35%*	*8.94*	*12.33*	*28%*
	35	*14.74*	*23.08*	*36%*	*12.91*	*17.63*	*27%*
	45	*18.16*	*28.14*	*35%*	*15.02*	*19.71*	*24%*
C_i	5-100	*7.51*	*12.30*	*39%*	*6.50*	*9.51*	*32%*
	5-200	*15.39*	*23.90*	*36%*	*13.41*	*17.81*	*25%*
Average		*11.10*	*17.65*	*38%*	*9.70*	*13.34*	*27%*

Table 6.5: Fleet size and utilization, for different parameters

customers, with each having his own capacity C_i and his own demand pattern, we decided to measure this inventory level as a **percentage** of C_i. This value was then averaged over T and n. The values of this measure of the inventory behavior in the solutions obtained by *CONST* and *LATEST* were compared in Table 6.6, that distinguishes the different data sets. The figures in this table clearly show that *LATEST* is making an uncontrolled use of inventory. Indeed, on average, the customer's inventory levels are at 71% of their capacity C_i, whereas, for *CONST*, this level drops to 45%. This is however understandable, because of the very nature of *LATEST* and *CONST*. In *LATEST* each time a customer is visited, it must be "filled" to capacity whenever possible, whereas, in *CONST*, a customer's inventory must be left empty at the end of the planning period.

We see this difference in inventory behavior as positive performance for *CONST*. Indeed, this means it uses more efficiently the resource buffer available, that is the inventory, to obtain a good routing plan and reduce routing costs.

Variable	***Value***	*LATEST*	*CONST*
T	5.00	71.62%	40.56%
	10.00	70.96%	47.23%
	15.00	72.20%	48.45%
n	15.00	71.92%	46.27%
	25.00	71.34%	46.19%
	35.00	71.19%	44.58%
	45.00	71.95%	42.49%
C_i	5-100	72.96%	44.06%
	5-200	70.02%	46.47%
Average		71.49%	45.27%

Table 6.6: Average inventory level, as a % of C_i

Moreover, this means that our algorithm would have the potential to be adapted to problems where an inventory holding cost h would be charged for each unit of product present in the customer's inventory on each day of the planning period. We believe however that in such a case the whole heuristic would need to be redesigned, to account for these additional costs during the construction-improvement phases. The heuristic proposed by Bertazzi *et al.* (2002), for example, explicitly takes into account the holding costs.

This analysis of the inventory behavior of the customers was the last facet of the computational results we wished to present. We can therefore conclude this chapter.

6.4 Summary

This chapter described the computational experiments conducted: the cost of the solutions found by our heuristic were compared with the optimum on small instances, and with the output of a myopic heuristic on a wide panel of problems. In the first

set of experiments, our heuristic was on average within 5.33% above the optimum. In the second set of experiments, we noticed a 13.6% cost saving over the myopic policy, a far more rational utilization of the daily fleet, and a tighter use of the inventory buffer.

Chapter 7

Conclusion and future research

This thesis has presented an innovative analysis of the Inventory Routing Problem(IRP). Indeed, the IRP is, till now, regarded as a marginal routing problem, given that the routing decisions could not be related to fixed-size orders. This study showed that, in fact, the IRP was nothing more than a Split-Delivery Vehicle Routing Problem (SDVRP), with multiple delivery periods and delivery dates constraints, referred to as the MVRPD. We explained how an IRP can be transposed to this demand-driven VRP, and formally showed the equivalence between the two problems. Once this equivalence was shown, we looked into the classic VRP literature, as well as in the SDVRP studies, to find constructive methods adapted to the problem we defined. This allowed us to design a heuristic that we tested on numerical instances. The results were satisfying, as our heuristic gave results 5.33% above optimum in small instances, and, on average, presented a 13% cost reduction over a myopic heuristic.

We believe however that our contribution was more about highlighting the intrinsic nature of the IRP than about designing a powerful heuristic. There is therefore ample room for refinement and improvement of our solution method, and we will give some indications of what could be done hereafter:

First of all, we believe it would be interesting to design a more customized heuristic for the MVRPD to take into account specific features of the problem created. Indeed, what we did, when designing our solution method, was to simply adapt existing robust methods to our problem. An effort should be made, for example, to take into account the fact that, in the problem created, *a lot* of demand points share the same geographical location. This information could be used, when locally improving solutions, to reduce the number of potential moves examined, and therefore to reduce the computational effort needed.

Second of all, we wish to highlight, once again, the *Split-Delivery* nature of the IRP. This implies that future research could try to adapt the different solution techniques experimented in the SDVRP literature, and listed in Section 2. We believe that the biggest drawback of the solution method we have chosen (inspired by Dror & Trudeau (1990)) is the lack of *fluidity* of the demand. Other methods should therefore be applied to our problem. The tabu search developed by Archetti *et al.* (2003) appears to be the best candidate. Note as well that the local improvement operators developed in Sierksma & Tijssen (1998) gave promising results, and allowed more fluidity in the customer demand than the traditional 2-opt and r-opt operators we implemented in our study. On a related issue, we believe that the cutting-plane developed by Belenguer *et al.* (2000) for the SDVRP could be adapted to produce optimal solutions for middle-size instances, and lower bounds of a good quality for bigger instances, which could help assess the quality of the different heuristics. Above all, the new development in the SDVRP literature should be watched closely, as the topic has been far from extensively studied.

If our transposition allows us to use such a wide panel of solution methods, it can also inspire us to include additional features to the IRP, by considering the different VRP variants. The first feature that comes to our mind is the *time windows* constraint. Indeed, extensive work has been done on the VRP with time windows

(VRPTW) and could be adapted here. The results of such a study could be directly compared with the work on the IRP conducted by Campbell *et al.* (2002), that explicitly takes into account time windows and operating modes. The authors of the latter suggested another extension to the IRP, that could easily be implemented here: the multi-depot IRP, where several suppliers serve the customers. Again, our transformation would help implement the solution methods used in the multi-depot VRP method.

One last feature that should be explored using our approach is the inventory holding costs considerations. Indeed, we completely overlooked this aspect of the IRP in our study, in order to focus on the routing plan and the inventory *constraints*. It is straightforward however to include these holding costs in our formulation, by assigning a day-dependent cost of associating each delivery volume to a given vehicle. Integrating these additional costs into the objective when building and improving feasible solutions will highlight the tradeoff between inventory holding costs and routing costs. Such a study would allow an interesting comparison with the figures obtained by Bertazzi *et al.* (2002).

Bibliography

Agarwal, Y., Mathur, K. & Salkin, H.M. (1989). A set-partitioning-based exact algorithm for the vehicle routing problem. *Networks*, **19**, 731–749. 2.2.1.1

Altinkemer, K. & Gavish, B. (1989). Parallel savings based heuristic for the delivery problem. *Operations Research*, **39**, 456–469. 2.2.1.2

Anily, S. & Federgruen, A. (1990). One warehouse multiple retailer systems with vehicle routing costs. *Management Science*, **36**, 92–114. 2.1.1, 2.1.2

Anily, S. & Federgruen, A. (1991). Rejoinder to "comments on one warehouse multiple retailer systems with vehicle routing costs". *Management Science*, **37**, 1497–1499. 2.1.2

Anily, S. & Federgruen, A. (1993). Two-echelon distribution systems with vehicle routing costs and central inventories. *Operations Research*, **41**, 33–47. 2.1.2

Archetti, C., Hertz, A. & Speranza, M. (2003). A tabu search algorithm for the split delivery vehicle routing problem. Technical report Cahiers du GERARD G-2003-18, École des Hautes Études Commerciales de Montréal, Canada. 2.2.2, 5.2.2, 6.3.3, 7

Archetti, C., Savelsbergh, M.W.P. & Speranza, M. (2004). Worst-case analysis for split delivery routing problem. *Transportation Science*, to appear. 2.2.2, 5.2.2

Baita, F., Ukovich, M., Pesenti, R. & Favaretto, D. (1998). Dynamic routing-and-inventory problems: a review. *Management Science*, **A 32**, 585–598. 2.1.1

Bard, J.F., Huang, L., Jaillet, P. & Dror, M. (1998). A decomposition approach to the inventory routing problem with satellite facilities. *Transportation Science*, **32**, 189–203. 2.1.3

Baskell, T. (1967). Bases for vehicle fleet scheduling. *Operational Research Quarterly*, **18**, 281–295. 2.2.1.2

Belenguer, J.M., Martinez, M.C. & Mota, E. (2000). A lower bound for the split delivery routing problem. *Operations Research*, **48**, 801–810. 2.2.2, 5.2.1, 7

Berman, O. & Larson, R.C. (2001). Deliveries in an inventory/routing problem using stochastic dynamic programming. *Transportation Science*, **3**, 192–213. 2.1.5

Bertazzi, L., Speranza, M.G. & Ukovich, W. (1997). Minimization of logistics costs with given frequencies. *Transportation Research*, **31**, 327–340. 2.1.2

Bertazzi, L., Paletta, G. & G.Speranza, M. (2002). Deterministic order-up-to level policies in an inventory routing problem. *Transportation Science*, **36**, 119–132. 1.3, 2.1.4, 3.1, 4.1, 6.3.1, 6.3.5, 7

Bodin, L.D., Golden, B.L., Assad, A.A. & Ball, M. (1983). Routing and scheduling of vehicles and crews, the state of the art. *Computers & Operations Research*, **10**, 63–212. 2.2.1

Bramel, J. & Simchi-Levi, D. (1995). A location based heuristic for general routing problems. *Operations Research*, **43**, 649–660. 2.1.2

Breedam, A.V. (1994). *An analysis of the behavior of heuristics for the vehicle routing problem for a selection of problems with vehicle-related, customer-related, and time-related constraints*. Ph.D. thesis, University of Antwerp. 2.2.1.3

Campbell, A., Clarke, L. & Savelsbergh, M. (2002). Inventory routing in practice. In *The vehicle routing problem*, 309–330, Society for Industrial and Applied Mathematics. 2.1.4, 3.1, 6.3.1, 7

Chan, L.M.A., Federgruen, A. & Simchi-Levi, D. (1998). Probabilistic analysis and practical algorithms for inventory-routing problems. *Operations Research*, **46**, 96–106. 2.1.2

Chien, T.W., Balakrishnan, A. & Wong, R.T. (1989). An integrated inventory allocation and vehicle routing problem. *Transportation Science*, **23**, 67–76. 2.1.4

Christofides, N., Mingozzi, A. & Toth, P. (1979). The vehicle routing problem. In N. Christofides, A. Mingozzi & P. Toth, eds., *Combinatorial Optimization*, 315–338, Wiley, Chichester, UK. 2.2.1.2

Clarke, G. & Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, **12**, 568–581. 2.2.1.2

Desrochers, M. & Verhoog, T.W. (1989). A matching based savings algorithm for the vehicle routing problem. Technical report Cahiers du GERARD C-39-04, École des Hautes Études Commerciales de Montréal, Canada. 2.2.1.2

Desrochers, M., Desrosiers, J. & Solomon, M.M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, **40**, 342–354. 2.2.1.1

Dror, M. & Ball, M. (1987). Inventory/routing: Reduction from an annual to a short-period problem. *Naval Research Logistics*, **34**, 891–905. 2.1.3

Dror, M. & Trudeau, P. (1989). Split delivery routing. *Transportation Science*, **23**, 141–145. 2.2.2, 5.2.5, 5.2.5.2, 5.2.5.2

Dror, M. & Trudeau, P. (1990). Split delivery routing. *Naval Research Logistics*, **37**, 383–402. 2.2.2, 7

Dror, M., Ball, M. & Golden, B. (1985). A computational comparison of algorithms for the inventory routing problem. *Annals of Operations Research*, **4**, 3–23. 2.1.3

Dror, M., Laporte, G. & Trudeau, P. (1994). Vehicle routing with split deliveries. *Discrete Applied Mathematics*, **50**, 239–254. 2.2.2

E. Hadjiconstantinou, A.M., N. Christofides (1995). A new exact algorithm for the vehicle routing problem based on q-paths and k-shortest paths relaxations. *Annals of Operations Research*, **61**, 21–43. 2.2.1.1

Federgruen, A. & Simchi-Levi, D. (1995). Analysis of vehicle routing and inventory-routing problems. In *Network Routing. Vol. 8 of Handbooks in Operations Research and Management Science*, 297–373, North-Holland, Amsterdam, Netherlands. 2.1.1

Federgruen, A. & Zipkin, P. (1984). A combined vehicle routing and inventory allocation problem. *Operations Research*, **32**, 1019–1037. 2.1.1, 2.1.4

Fisher, M.L. (1994). Optimal solution of the vehicle routing problems using minimum k-trees. *Operations Research*, **42**, 626–642. 2.2.1.1

Fisher, M.L. & Jaikumar, R. (1978). A decomposition algorithm for large-scale vehicle routing, working paper, Dept. of Decision Sciences, University of Pennsylvania, Philadelphia. 2.1.4

Frizzel, P.W. & Giffing, J.W. (1992). The bounded split delivery vehicle routing problem with grid network distance. *Asia-Pacific Journal of Operational Research*, **9**, 101–116. 2.2.2

Frizzel, P.W. & Giffing, J.W. (1995). The split delivery vehicle scheduling problem with time windows and grid network distance. *Computers & Operations Research*, **22**, 655–667. 2.2.2

Gallego, G. & Simchi-Levi, D. (1990). On the effectiveness of direct shipping strategy for the one-warehouse multiple retailer r-systems. *Management Science*, **36**, 240–243. 2.1.2

Gendreau, M. & Laporte, A.H.G. (1994). A tabu search heuristic for the vehicle routing problem. *Management Science*, **40**, 1276–1290. 2.2.1.4

Golden, B., Assad, A. & Dahl, R. (1984). Analysis of a large scale vehicle routing problem with an inventory component. *Large Scale Systems*, **7**, 181–190. 2.1.3

Golden, B.L., Wasil, E.A., Kelly, J.P. & Chao, I.M. (1998). Metaheuristics in vehicle routing. In *Fleet Management and Logistics*, 33–56, Kluwer, Boston, MA. 2.2.1.4

Hall, R.W. (1991). Comments on one warehouse multiple retailer systems with vehicle routing costs. *Management Science*, **37**, 1496–1497. 2.1.2

Jaillet, P., Bard, J.F., Huang, L. & Dror, M. (2002). Delivery cost approximations for inventory routing problems in a rolling horizon framework. *Transportation Science*, **36**, 292–300. 2.1.3, 5.2.7

Karp, R. (1972). Reducibility among combinatorial problems. In R. Miller & J. Tatcher, eds., *Complexity of Computer Computations*, 85–104. 1.1

Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, **59**, 345–358. 2.2.1

Laporte, G. & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. *Annals of Discrete Mathematics*, **31**, 147–184. 2.2.1.1

Lin, S. (1965). Computer solution of the travelling salesman problem. *Bell System Technical Journal*, **44**, 2245–2269. 2.2.1.3

Mole, R.H. & Jameson, S.R. (1976). A sequential route-building algorithm employing a generalized savings criterion. *Operational Research Quarterly*, **27**, 503–511. 2.2.1.2

Mullaseril, P.A., Dror, M. & Leung, J. (1997). Split-delivery routeing heuristics in livestock feed distribution. *Journal of the Operations Research Society*, **48**, 107–116. 2.2.2

Or, I. (1976). *Travelling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking.*. Ph.D. thesis, Department of Industrial Engineering and Management Science, Northwestern University, Evanston, IL. 2.2.1.3, 5.2.4

Osman, I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, **41**, 421–451. 2.2.1.4

Ralphs, T.K., Kopman, L., Pulleyblank, W.R. & Trotter, L.E. (2003). On the capacitated vehicle routing problem. *Mathematical Programming*, **94**, 343–359. 2.2.1.1

Sierksma, G. & Tijssen, G.A. (1998). Routing helicopters for crew exchanges on off-shore locations. *Annals of Operations Research*, **76**, 261–286. 2.2.2, 7

Taillard, E.D. (1993). Parallel iterative search methods for vehicle routing problems. *Networks*, **23**, 661–673. 2.2.1.4

Toth, P. & Vigo, D. (2002). *The Vehicle Routing Problem*. SIAM monographs on discrete mathematics and applications. 2.2.1

Trudeau, P. & Dror, M. (1992). Stochastic inventory routing: Route design with stockouts and route failures. *Transportation Science*, **26**, 171–184. 2.1.3

Webb, I.R. & Larson, R.C. (1995). Period and phase of customer replenishment: A new approach to the strategic inventory/routing problem. *European Journal of Operational Research*, **85**, 132–148. 2.1.5

Yellow, P. (1970). A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, **21**, 281–283. 2.2.1.2

Appendix A

Comparison of CONST with LATEST

In Section 6.3 of Chapter 6 we presented our numerical results in terms of average results, in order to give a synthetic insight on the performance of *CONST*. The reader can find in the following table the results obtained in each of our individual instances. For each instance, the table presents the problem characteristics (number of days T, the number of customers n, the maximum allowed ratio of customer capacity to truck capacity C_{max}) and the results obtained by both $LATEST$ and $CONST$: total cost, average inventory $\bar{I}$, average number of trucks used K_{avg}, maximum number of trucks needed K_{max} and computing time CPU.

Instance			LATEST					CONST				
T	n	C_{max}	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU
5	15	100	19069	0.76	3.60	8	0	14970	0.50	3.00	8	1
5	15	100	12678	0.73	3.60	6	1	10880	0.40	3.20	5	2
5	15	100	12989	0.69	3.00	4	1	11404	0.34	2.40	5	1
5	15	100	26663	0.71	5.00	6	1	20866	0.36	4.00	5	1
5	15	200	59283	0.72	12.00	18	1	48486	0.40	9.80	13	7
5	15	200	33466	0.78	6.60	13	0	21561	0.44	4.60	6	3

Instance			LATEST					CONST				
T	n	C_{max}	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU
5	15	200	51253	0.70	9.00	12	0	38944	0.41	7.00	8	4
5	15	200	21084	0.68	6.00	11	0	16909	0.37	5.00	6	3
5	25	100	29320	0.69	6.20	10	1	22700	0.43	5.00	7	5
5	25	100	39015	0.72	7.20	10	1	29527	0.42	5.40	9	6
5	25	100	31568	0.73	7.60	12	1	24648	0.39	5.80	9	7
5	25	100	52884	0.72	7.80	12	1	41023	0.36	6.00	10	4
5	25	200	67406	0.70	11.00	14	2	54234	0.47	8.80	12	8
5	25	200	52562	0.71	13.20	19	1	43165	0.46	11.00	13	21
5	25	200	57938	0.69	15.60	23	1	41988	0.39	11.60	16	14
5	25	200	76443	0.71	14.80	20	1	56572	0.42	11.20	16	12
5	35	100	78318	0.71	12.80	15	2	59571	0.38	9.40	13	19
5	35	100	32340	0.75	9.80	18	2	23185	0.41	7.80	11	12
5	35	100	63189	0.75	13.60	19	2	46537	0.38	10.20	14	18
5	35	100	40995	0.74	8.80	12	2	30910	0.40	6.60	11	11
5	35	200	116855	0.67	22.00	30	3	93871	0.41	17.40	23	115
5	35	200	90874	0.72	23.00	29	4	72591	0.40	18.60	26	164
5	35	200	139724	0.67	21.80	27	3	112122	0.41	17.60	24	116
5	35	200	85943	0.73	20.20	29	3	61532	0.37	14.40	19	88
5	45	100	60106	0.71	10.60	15	2	47150	0.43	8.20	11	23
5	45	100	58683	0.73	13.20	22	2	44269	0.44	10.40	15	32
5	45	100	70544	0.73	13.80	20	2	55188	0.40	10.80	15	29
5	45	100	83505	0.75	14.40	25	2	65771	0.45	11.40	16	36
5	45	200	95618	0.72	24.80	41	2	76188	0.40	19.20	24	120
5	45	200	160296	0.68	29.00	43	3	126105	0.42	23.00	29	201
5	45	200	111021	0.74	25.40	38	2	89189	0.36	19.80	29	157
5	45	200	113887	0.72	27.80	47	3	86105	0.36	21.60	27	187
10	15	100	44855	0.75	4.00	7	1	36488	0.47	3.60	6	9
10	15	100	54022	0.70	3.80	7	1	49073	0.43	3.50	7	11
10	15	100	42057	0.68	3.80	7	1	37845	0.47	3.50	6	6
10	15	100	39687	0.73	4.70	8	1	35205	0.48	4.30	7	9
10	15	200	66216	0.72	8.60	16	2	59607	0.53	8.30	11	47
10	15	200	111858	0.66	11.20	17	3	105712	0.55	10.90	14	71
10	15	200	70482	0.72	7.60	12	1	60804	0.49	6.80	10	23
10	15	200	72622	0.64	8.10	14	2	66239	0.53	7.50	11	32
10	25	100	64608	0.75	7.10	11	2	56008	0.45	6.50	10	75
10	25	100	53094	0.73	5.80	10	2	48116	0.45	5.20	7	48

Instance			LATEST					CONST				
T	n	C_{max}	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU
10	25	100	67172	0.71	6.10	11	2	58851	0.46	5.60	9	28
10	25	100	104707	0.72	8.30	13	3	95514	0.47	7.80	11	55
10	25	200	98825	0.71	12.00	20	3	88664	0.46	10.80	14	137
10	25	200	156451	0.72	12.00	17	2	141738	0.49	10.80	16	158
10	25	200	101558	0.70	12.10	16	3	91197	0.54	11.00	16	114
10	25	200	127026	0.68	16.60	23	3	120034	0.51	15.80	19	298
10	35	100	87562	0.75	9.00	16	3	75450	0.46	7.80	10	88
10	35	100	86597	0.72	9.00	14	3	78239	0.45	8.20	11	83
10	35	100	113151	0.72	10.70	20	3	100587	0.47	9.60	14	89
10	35	100	75741	0.72	9.10	12	2	64529	0.44	8.20	13	146
10	35	200	146516	0.69	18.40	28	3	134562	0.49	17.10	22	464
10	35	200	240728	0.69	22.30	35	5	222462	0.48	20.60	29	831
10	35	200	240566	0.69	19.20	37	4	224666	0.47	18.00	25	519
10	35	200	187712	0.70	17.50	32	3	173910	0.46	16.30	22	514
10	45	100	136335	0.71	13.70	21	4	119662	0.44	12.10	16	479
10	45	100	115435	0.73	10.10	14	4	107317	0.45	9.40	12	411
10	45	100	136976	0.73	12.80	21	5	120277	0.43	11.30	15	361
10	45	100	110330	0.72	12.40	23	5	93136	0.43	11.00	14	418
10	45	200	330157	0.65	25.6	39	7	292887	0.48	24.9	30	562
10	45	200	220843	0.68	18.4	27	6	199700	0.46	17.8	23	456
10	45	200	200916	0.70	19.60	27	9	176323	0.46	17.60	21	654
10	45	200	189059	0.69	26.60	37	7	173914	0.48	24.50	32	552
15	15	100	36576	0.80	1.60	3	2	31078	0.50	1.73	4	22
15	15	100	39409	0.76	1.53	4	2	30215	0.47	1.67	3	20
15	15	100	38634	0.74	3.47	6	2	35368	0.48	3.40	6	36
15	15	100	56614	0.78	2.53	5	2	52420	0.46	2.53	4	19
15	15	200	96508	0.70	7.33	13	3	90640	0.51	7.07	9	112
15	15	200	76663	0.74	6.47	12	2	69181	0.48	5.93	9	146
15	15	200	86978	0.69	8.20	15	2	80115	0.51	8.07	11	116
15	15	200	119192	0.69	6.93	13	3	112397	0.51	6.87	11	92
15	25	100	97110	0.76	6.20	11	3	88931	0.46	6.07	8	237
15	25	100	147688	0.71	7.20	13	4	141671	0.45	6.80	10	324
15	25	100	142275	0.74	7.53	11	5	127968	0.47	7.00	10	168
15	25	100	76142	0.72	6.27	12	5	69533	0.47	6.07	9	238
15	25	200	159841	0.69	12.87	23	6	147789	0.51	12.20	16	744
15	25	200	200422	0.70	12.73	20	4	189077	0.53	12.13	15	550

Instance			LATEST					CONST				
T	n	C_{max}	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU	Cost	$\bar{I}$	K_{avg}	K_{max}	CPU
15	25	200	187784	0.70	12.60	22	4	175697	0.51	12.00	16	709
15	25	200	222658	0.71	14.13	23	4	216260	0.53	13.93	18	642
15	35	100	112525	0.74	8.93	15	4	100577	0.44	8.47	12	469
15	35	100	148300	0.73	8.73	18	5	139735	0.47	8.27	11	303
15	35	100	140276	0.73	10.40	15	5	128910	0.45	9.73	13	439
15	35	100	129383	0.72	7.40	15	4	119631	0.45	7.27	10	537
15	35	200	302239	0.70	17.00	30	7	291068	0.50	16.27	21	554
15	35	200	286059	0.71	17.73	30	5	275275	0.51	17.07	23	523
15	35	200	226855	0.68	18.73	33	14	212912	0.51	18.00	24	621
15	35	200	216857	0.67	17.67	25	12	205914	0.49	17.00	22	254
15	45	100	139127	0.73	11.13	22	6	129874	0.50	10.47	14	865
15	45	100	151317	0.72	12.87	22	9	140714	0.50	12.13	16	660
15	45	100	135012	0.71	11.20	20	5	122781	0.43	10.47	15	745
15	45	100	141897	0.71	12.47	20	10	131163	0.47	11.73	16	530
15	45	200	275519	0.70	21.40	34	9	257188	0.48	19.93	27	403
15	45	200	356177	0.70	21.93	34	7	334511	0.51	20.47	30	732
15	45	200	219026	0.67	24.87	34	7	207885	0.50	23.87	32	768
15	45	200	298449	0.70	21.40	27	7	282534	0.52	20.27	31	678

Table A.1: Comparison of *CONST* with *LATEST*