The Vehicle Routing Problem With Drop Yards: A Dynamic Programming Approach

Master Thesis
Industrial Engineering & Management
University of Twente

Quirijn Steeman

March 22, 2012

This thesis is written in fulfillment of the degree Master of Science in Industrial Engineering and Management.

Institute:
University of Twente
School of Management and Governance
Department of Operational Methods for Production and Logistics

University Supervisors:
J.M.J. Schutten
M. Mes

ORTEC Supervisors:
A.L. Kok
J. Gromicho

Author:
Quirijn Steeman
March 2012

Acknowledgements

I wrote this thesis as a graduation project for the study Industrial Engineering & Management at the University of Twente. Many people helped me to accomplish this.

First, I thank ORTEC for giving me the opportunity to write this thesis in an environment where many of the theories are used in practice. It gave me a better feeling of the subject of research.

I thank Joaquim Gromicho, for the support and ideas he gave me in doing this research. I also thank him for the efforts he has made to make the implementation of the algorithm possible.

I thank Marco Schutten, Leendert Kok, and Martijn Mes for their input and critical reading. This has helped me a lot to improve the thesis.

Finally, I thank Jelke van Hoorn for his input and help with the C++ implementation.

Thank you all,
Quirijn Steeman

Contents

1 Introduction **9**
1.1 ORTEC and OTD . . . 9
1.2 Vehicle Routing Problem . . . 9
1.3 Drop Yards . . . 10
1.4 Research Questions . . . 12
1.5 Outline . . . 13

2 Literature Review **14**
2.1 Traveling Salesman Problem and the Vehicle Routing Problem . . . 14
2.1.1 Variants of the Vehicle Routing Problem . . . 15
2.2 Methods to solve the VRP . . . 16
2.2.1 Exact Algorithms . . . 16
2.2.2 Heuristics . . . 17
2.3 Giant Tour Representation . . . 19
2.4 Drop Yards . . . 20
2.5 Conclusions . . . 22

3 Dynamic Programming **23**
3.1 DP Algorithm for the Traveling Salesman Problem . . . 23
3.2 Order Relations . . . 25
3.3 The DP Algorithm Adjusted for the Vehicle Routing Problem . . . 26
3.4 Conclusions . . . 27

4 Basic Model **28**
4.1 The Model . . . 28
4.2 Solving the Basic Model with Dynamic Programming . . . 29
4.2.1 Drop Yard Nodes . . . 29
4.2.2 Loads . . . 30
4.2.3 Order of the Nodes in the Solution . . . 30
4.2.4 Costs and State Description . . . 31
4.2.5 Multiple Vehicles . . . 31
4.2.6 Example of the DP Algorithm Applied to the Basic Model . . . 31
4.3 Conclusions . . . 32

5 Extensions of the Basic Model **34**
5.1 One of Multiple Drop Yards is used in a Tour 34
5.1.1 Inclusion in the Model 35
5.1.2 Use of the Depot as the Drop Yard 35
5.2 More than two Trailers on a Truck 36
5.2.1 Inclusion in the Model 36
5.3 Customer Visits with Multiple Trailers, Before and After Drop Yard Visits 37
5.3.1 Inclusion in the Model 37
5.3.2 State Description 38
5.3.3 Order Relations 38
5.4 Multiple Drop Yards may be used in one Tour 38
5.4.1 Inclusion in the Model 39
5.5 Conclusions 40

6 Computational Experiments **42**
6.1 Implementation 42
6.2 Test Cases 43
6.3 Test Instances 43
6.4 Results 44
6.4.1 Results for the VRP 44
6.4.2 Results for the VRPDY 45
6.4.3 Results for the VRPDY with Vehicle Customers 46
6.5 Conclusions 46

7 Conclusions and Further Research **48**
7.1 Conclusions 48
7.2 Further Research 49

Bibliography **51**

A Basic Model Example **53**

Summary

Introduction

Transportation companies deliver goods to customers. Their goal is to do that as efficiently as possible. To do this, they can, for example, minimize the distance traveled by their vehicles. They do this by assigning the loads to the vehicles and creating routes for the vehicles in an efficient way. The problem of assigning loads to vehicles and creating optimal routes is called the Vehicle Routing Problem (VRP). The standard VRP consists of several customers, which all have a demand and a location. Further, there is a fleet of vehicles that has to fulfill the demand of the customers and all vehicles start their routes from the depot. In general, the vehicles are specified as units that hold limited capacity to carry load. However, in some countries, companies can use large vehicles consisting of a pulling vehicle and multiple trailers. These vehicle combinations are sometimes too large to visit the customer locations. For this reason, trailers have to be decoupled to enable the vehicle to serve these customers. This decoupling happens on so called drop yards.

In the literature, the VRP is extensively described and many solution approaches are suggested. In most of the problems, the vehicle is modeled as a single unit and the use of drop yards is not supported. There are some authors that describe problems, such as the Truck and Trailer Problem, that includes the use of vehicles that consist of two parts, a truck and a trailer. In these problems, the trailer can be parked at customer locations, to enable the truck to deliver goods to other customers, but they make no use of dedicated drop yards. The objective of this research is to incorporate the use of drop yards in the VRP. Additionally, we propose a method to solve this new Vehicle Routing Problem with Drop Yards (VRPDY).

Dynamic Programming

This research is performed at ORTEC, one of the largest providers of advanced planning software. Part of the software of ORTEC is dedicated to solve VRPs. For the calculation of the solutions for the VRP, one main algorithm ORTEC uses is Dynamic Programming (DP). Therefore, we propose a solution methodology for the VRPDY based on DP. DP is a technique that divides problems in a sequence of smaller and simpler problems. These smaller problems are solved sequentially in multiple stages, and in the last stage, the optimal solution is found.

The DP algorithm ORTEC uses is based on the DP algorithm for the Traveling Salesman Problem (TSP) of Held and Karp (1961). The TSP is a special case of the VRP. In the TSP, there is only one vehicle with unlimited capacity that has to visit the customers in one tour. In the VRP, the customers are visited by a number of vehicles and each vehicle has a limited capacity. Therefore, the TSP is a VRP with one vehicle with unlimited capacity. To make use of the DP algorithm for the TSP, we represent the solution of the VRP as a TSP solution by means of the Giant Tour Representation (GTR). In its GTR, a VRP solution is represented as one large tour by connecting the finish of one vehicle tour with the start of the next vehicle tour. By connecting the last vehicle tour with the first tour, one complete tour is created.

Vehicle Routing Problem with Drop Yards

We start the VRPDY with a basic model and after that we extend it with additional features. In the basic model, there is one vehicle, consisting of a pulling vehicle and two trailers. Each customer can only be visited with one trailer at the time, which means that a drop yard must be used and two tours from the drop yard are performed by the vehicle. To solve this problem with DP, we need to create extra nodes for the drop yards and we define the relations between the nodes in the problem.

Extensions

The basic model is just a simple representation of the use of drop yards in the VRP. We extend this model to resemble reality more.

The first extension of the basic model, which contains one vehicle with two trailers, is to enable it for vehicles with more than two trailers. Thus, the vehicle can make multiple single trailer tours from the drop yard. To accomplish this, we created extra nodes for the extra trailers.

Next to the case that a vehicle has more than two trailers, it could be that more than one drop yard is available. In our second extension, the vehicle can use one of these drop yards in its tour. With each trailer, a tour from the drop yard can be made. In this model, we modeled the depot as a drop yard as well, to enable the vehicle to do single trailer tours from the depot. In some cases this could be cheaper than using another drop yard. In the case only the depot is used as the drop yard, the solution would be similar to the solution of a VRP.

The next extension is to enable the vehicle to visit customers that can handle the whole vehicle combination. Some customers can be visited with all trailers coupled to the vehicle and can therefore be visited before the drop yard is visited. Therefore, we divide the set of customers in two parts, vehicle customers and trailer customers. Vehicle customers can handle the complete vehicle with all trailers and the trailer customers can only have one trailer at their site.

With the last extension, it is possible for vehicles to use more than one drop yard in their tours. It is possible to use a trailer multiple times, in tours from different drop yards. In all previous models, the trailers are all used once. In this way, the vehicle can go to a drop yard and decouple trailers to do a single trailer tour. After that, it couples the trailers, and it has the choice to do a single trailer tour with one of the other trailers from the same drop yard, or to go to the next drop yard and do a tour there. During the travel from one drop yard to another, it is possible to serve vehicle customers.

Computational Results

We test our DP algorithm for the VRPDY with a C++ implementation and compare the results with the results of the VRP. To indicate the quality of the algorithm we also test the VRP with our algorithm and compare the results with the best known results for the test instances. We test the VRPDY model with multiple drop yards. Next to that, we test the VRPDY where also vehicle customers are present. For the tests we make use of well known instances for the VRP. The computations are done with a restricted version of the DP algorithm. We do this to limit the use of memory and computational times. By doing this, the guarantee of obtaining an optimal solution is lost.

The solution values of the VRPDY are on average 3% higher than the costs of the VRP solutions calculated with our algorithm. This means that in many of the cases the use of drop yards does not pay off. The differences per instance are large and the results of 4 of the 15 instances show values that are better than the best known solutions. The best results are obtained in instances that have clusters of customer locations relatively far from the depot. In these cases, it pays off to have a drop yard relatively close to the customer locations. The solution values of the cases with vehicle customers are 3% better than those of the VRPDY. This is what we expect, since the VRPDY with vehicle customers is more flexible than the VRPDY without vehicle customers.

Conclusions

The Dynamic Programming approach to solve the VRPDY seems to be a flexible approach. The models can be adjusted to include new features. The results show that the use of drop yards can improve the solutions. The results are the best in the cases that the drop yard is near a group of customers that are at a distance from the depot. We have also seen that the solutions can be even better when there is a mix of vehicle customers and trailer customers.

Chapter 1

Introduction

In the transportation business, companies deliver goods to customers with a fleet of vehicles. The goal of these companies is to deliver these goods as efficiently as possible to minimize the costs. Therefore, they want to minimize the number of vehicles that are used and the distances in the routes traveled by these vehicles. This problem of creating optimal routes for the vehicles and assigning customers to the vehicles is the Vehicle Routing Problem (VRP). Companies often use large vehicles to deliver goods to the customers. In some cases, the vehicles consist of a tractor carrying two or more trailers. Not all customers can accommodate such large vehicle combinations. To enable truckers to visit these customers, and to continue their tours, they have to decouple and park some trailers temporarily. This decoupling takes place at so-called drop yards. In this research, we look into the VRP in combination with the use of drop yards.

We perform this research at ORTEC, which provides decision support systems that solve VRPs. In this research we incorporate drop yards in the VRP, and solve this new VRP model using Dynamic Programming. In this chapter, we introduce ORTEC, the Vehicle Routing Problem, and drop yards. We conclude this chapter with the research questions and an outline how we answer these research questions.

1.1 ORTEC and OTD

ORTEC was founded in 1981 and is currently one of the largest providers of advanced planning and optimization software and also provides consulting services. ORTEC provides these services in two business areas: logistics and finance. The software and services that ORTEC delivers cover many topics from workforce scheduling to vehicle routing & dispatch on the logistics area and from asset management to real-estate management on the financial area. On the logistics side, ORTEC Transport and Distribution (OTD) is one of ORTEC's main products. OTD supports the planning and execution of transportation and distribution processes. It is a multi-user application that helps planners to create and improve their day-to-day transportation schedules, but it can also be used for tactical and strategic purposes.

1.2 Vehicle Routing Problem

One of the main features in OTD is vehicle routing. In a Vehicle Routing Problem, a fleet of vehicles has to visit a set of customers, delivering goods (deliveries) at minimum costs, taking into account all existing restrictions. These restrictions could be, for example, the capacity of the vehicles or time windows for the deliveries. An important property of the VRP is that the starting and end location of the vehicles is the same. The VRP is well known to be a difficult problem to solve, in fact it is NP-hard (Cordeau et al., 2007) even in its simplest forms. Therefore, the calculation time for finding an optimal solution will increase enormously as the number of locations to visit grows. For small instances, an optimal solution can be found with exact algorithms, but if the number of customers grows, heuristics are needed to obtain good solutions in reasonable time.

In OTD, the method that is used to find solutions for the VRP is based on Dynamic Programming (DP). Dynamic Programming is an exact method that implicitly enumerates all possible solutions of a problem. An important characteristic of Dynamic Programming is that the main problem is split up in sub problems and that there is some recurrence relation. An optimal solution is obtained by solving a system of recurrence relations (Bellman, 1957).

1.3 Drop Yards

In some countries, it is common to use large combinations of vehicles. A truck can carry up to four trailers (or "pups"). These so called road trains are only allowed on some roads, mostly in remote areas. In urbanized areas, with more traffic, the large combinations are prohibited. In these areas trucks are only allowed to carry one trailer at a time. To visit customer locations where multiple trailer vehicles cannot come, the surplus of trailers must be unhooked and parked until they are used for later deliveries. This parking happens in so called drop yards. Drop yards are used in various ways:

1. A driver leaves the depot with a truck carrying for example 2 trailers. Only 1 trailer at a time can visit the customer locations. The driver first drives to the drop yard where 1 trailer is unhooked, and he continues to the first customer with the other trailer. He completes his route with that trailer until it is empty and returns to the drop yard to swap trailers. With the other trailer he performs a route until that trailer is also empty, after which he collects the first trailer at the drop yard and heads back to the depot (See Figure 1.1).

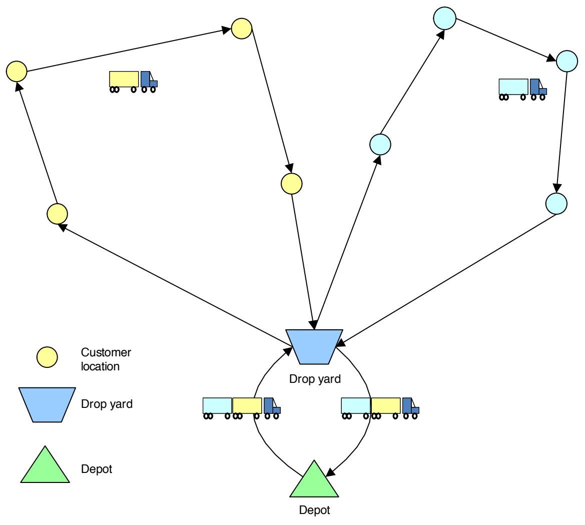


Figure 1.1: Basic drop yard use

2. If a customer location is capable to handle large vehicle combinations, it is possible to plan this visit before or after a local tour from the drop yard. Figure 1.2 shows a customer visit before a local tour from the drop yard is executed.

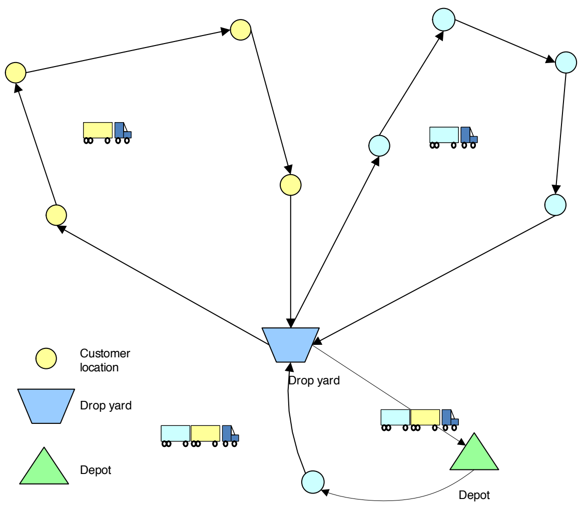


Figure 1.2: Customer visit before drop yard

3. A drop yard can also be used in a large tour where a vehicle visits multiple drop yards. For example, a multi-trailer truck drives from the depot to the drop yard. There, trailers are uncoupled and the vehicle performs a local route with one trailer, ending at the drop yard again. Next, all trailers are coupled and the combination drives to the second drop yard where the vehicle performs a local tour with one trailer again. This continues until all loads are delivered. Then the total combination returns to the depot (Figure 1.3).

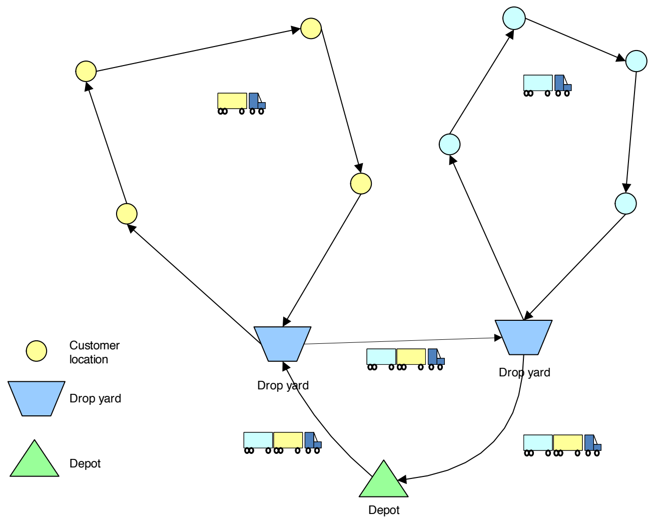


Figure 1.3: Multiple drop yards in 1 tour

4. A driver starts his route, serving customers. As some loads become available at a later moment at the depot, a second driver leaves the depot with a two-trailer truck combination and heads for the drop yard. There he parks one of the trailers and starts deliveries with the other trailer. After the first driver finishes his tour, he swaps trailers at the drop yard.

He leaves the empty trailer and continues deliveries with the full trailer. At the end of the tours when the drivers return to the depot, one of them picks up the trailer at the drop yard (Figure 1.4). This process is called wave planning. The wave planning can be a static planning as well as a dynamic planning. It is static in the case the planners know before planning which loads will be available for the second driver. In dynamic planning, the loads for the second driver are uncertain at the moment of planning. After a plan is made, new loads for the second driver can become available.

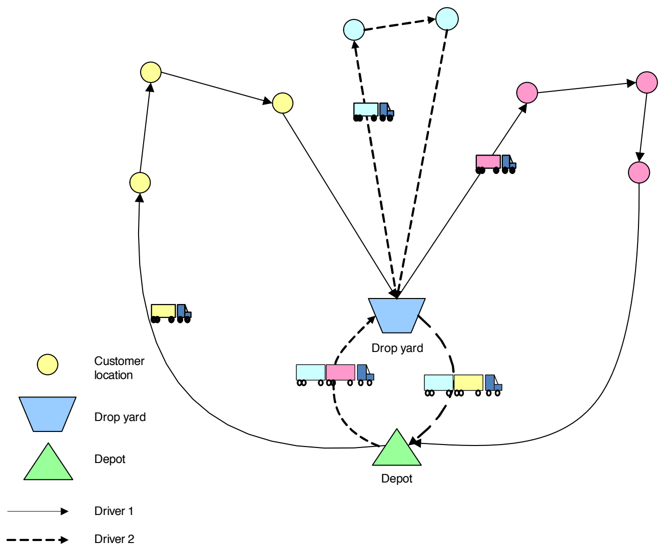


Figure 1.4: Drop yard used by more drivers

The situations as described above can be very important for companies using multiple trailers for their vehicles. Thus for these situations it is useful to make the use of drop yards part of the VRP.

1.4 Research Questions

The goal of this research is to incorporate drop yards in the VRP and to solve this new problem, which we call the Vehicle Routing Problem with drop yards (VRPDY). We start with a simple model that includes the most basic use of the drop yard. In this model, only 1 vehicle is available with 2 trailers. The vehicle has to visit a number of customers who all can handle only 1 trailer at the time. There is 1 drop yard available that must be used to be able to deliver to the customers. From that model we continue and add some extensions to solve models for more complex situations. Our research question is:

How can we include drop yards in a VRP model, and how can we solve the resulting model?

To find an answer to this research question we split the research in smaller pieces. To do this we have formulated sub questions. First we want to see what the literature says about the Vehicle Routing Problem and its variants, including drop yards in combination with vehicle routing and what elements of these findings we can use for our model. We also have to know what methods are used to solve the VRP. Then, we want to know how the method works that ORTEC is currently using to solve VRPs. Next, we want to know how we can include the use of drop yards in the solution methodology of ORTEC. Further, we want to extend the use of drop yards, to be able to handle more complex situations and we want to know what the solution quality is, compared to the VRP.

1. What elements from the literature about VRP in combination with drop yards can we use in our model of drop yards in the VRP?
2. What DP approach to solve the VRP is used by ORTEC?
3. How can drop yards be included in the VRP and how can we solve it with a DP algorithm?
4. With which extensions can we improve the basic model and how can we include these in the model?
5. How does the VRP with drop yards perform, compared to the VRP?

1.5 Outline

The remainder of this report is as follows. In Chapter 2 we describe the Vehicle Routing Problem and its variants. Additionally we describe several methods to solve the VRP. We also describe the Traveling Salesman Problem (TSP), which is a special case of the VRP and we describe how we can represent the solution of the VRP as a TSP by means of the Giant Tour representation, which we need to solve the VRP with the Dynamic Programming algorithm as described in Chapter 3. We also give an overview of how other authors describe the use of drop yards in combination with the VRP.

As stated before, in Chapter 3 we describe the Dynamic Programming algorithm for the TSP and how it can be used to solve the VRP. In Chapter 4 we include the drop yards in the VRP model and propose a method to solve this new problem with DP. This model only covers the basic use of the drop yards. In Chapter 5 we introduce extensions to the model from Chapter 4 to make it more realistic. In Chapter 6 we describe how we have implemented our model with the extensions in C++ and we show test results of computations with this implementation on problem instances to see how the VRPDY performs with respect to the standard VRP. Finally in Chapter 7 we draw conclusions and give suggestions for further research.

Chapter 2

Literature Review

In this chapter, we elaborate on the Traveling Salesman Problem (TSP) and the Vehicle Routing Problem. In Section 2.1, we describe the similarities and differences of the TSP and VRP and several variants of the VRP. In Section 2.2, we describe methods to solve the VRP. In Section 2.3, we describe how we can represent the solution of the VRP in terms of a TSP solution, which is used in our approach to solve the VRPs. Finally in Section 2.4, we give an overview of what is available in the literature about drop yards in combination with the VRP.

2.1 Traveling Salesman Problem and the Vehicle Routing Problem

In the TSP, there is a set of customers. Each customer must be visited exactly once. There is one vehicle available to visit these customers. This vehicle must make a complete tour by finishing its route in the same location as it started. The goal is to find a route that minimizes the traveling cost of the vehicle.

In the VRP, multiple vehicles are available and each vehicle has to do a complete tour. Each vehicle has limited capacity. Each customer has a demand, which will reduce the available capacity of a vehicle when it is loaded. A restriction in the VRP is that the amounts loaded to the vehicle stays within capacity of the vehicle. In the VRP routes for each vehicle must be found, that minimize the total traveling costs. The TSP and the VRP show many similarities. In fact, the TSP is a special case of the VRP, since the VRP with 1 vehicle that has infinite capacity, reduces to the TSP.

The VRP can be represented with a directed graph $G = (V, A)$, with $V = \{0, .., n\}$ a set of vertices (or nodes) representing customers and the depot and A a set of arcs (i, j) connecting these vertices. In the VRP, the vertex 0 represents the depot from where m vehicles with capacity Q are dispatched to visit a set of customers, which are represented by the remaining vertices. In this problem, every vehicle that is available is used in a route. The cost to travel over an arc from node i to j is c_{ij}. These costs are the distance, travel time, or any kind of other costs that occur between a node pair. Let q_i be the demand in node i. Let $r(S)$ be the minimum number of vehicles required to serve a customer subset $S \subseteq V \backslash 0$. The value of $r(S)$ can be found by solving an associated Bin Packing Problem (BPP) with bin set S with capacities Q. The BPP is NP-hard and thus difficult to solve. To avoid solving the BPP, this number can be approximated by $\lceil \sum_{i \in S} q_i / Q \rceil$, which is a lower bound (Cordeau et al., 2007). A route starts and finishes at the depot, and each customer is visited once. The objective is to visit all customers at minimal cost.

Toth and Vigo (2001) give the following ILP formulation:

$$Minimize \sum_{i,j \epsilon V} c_{ij} x_{ij} \tag{2.1}$$

subject to :

$$\begin{aligned}
\sum_{i \in V} x_{ij} &= 1 \qquad \forall j \in V \backslash 0, && (2.2)\\
\sum_{j \in V} x_{ij} &= 1 \qquad \forall i \in V \backslash 0, && (2.3)\\
\sum_{j \in V} x_{0j} &= m, && (2.4)\\
\sum_{i \in V} x_{i0} &= m, && (2.5)\\
\sum_{i \notin S} \sum_{j \in S} x_{ij} &\geq \lceil \sum_{i \in S} q_i / Q \rceil \qquad \forall S \subseteq V \backslash 0, S \neq \emptyset, && (2.6)\\
x_{ij} &\in \{0,1\} \qquad \forall \, i,j \in V && (2.7)
\end{aligned}$$

In this formulation, x_{ij} is a binary variable indicating whether arc (i,j) is traversed in the solution. If (i,j) is traversed, x_{ij} is 1 and otherwise it is 0. Constraints (2.2) and (2.3) make sure that every customer is visited exactly once. Constraints (2.4) and (2.5) ensure that m routes are created, one for each vehicle. Constraints (2.6) play a double role. First, given a set $S \subseteq V \backslash 0$ of customers with a total demand $\sum_{i \in S} q_i$, there must be at least $\lceil \sum_{i \in S} q_i / Q \rceil$ vehicles to serve this set of customers. This ensures that the amount loaded on a vehicle never exceeds its capacity. Second, it ensures that no sub tours are created by connecting customer j in subset S to a customer i that is not in that subset. The last constraints (2.7) indicate that x_{ij} are binary variables. The individual vehicle routes can be recovered from the solution by following the paths from the depot. For example, if a particular x_{0j} has value 1 and thus customer j is in that route, then the next customer in that route is customer i where x_{ji} is 1. The route finishes with the travel from the last customer to the depot, x_{i0}. All vehicle routes can be recovered this way.

2.1.1 Variants of the Vehicle Routing Problem

The VRP as described above is also called the classical VRP or the capacitated VRP (CVRP), because it takes the capacities of the vehicles into account. There exist many variants of this Vehicle Routing Problem. Each variant is an extension or generalization of the classical VRP and incorporates elements that occur in real life to make the VRP model more realistic. Of course, all kinds of combinations of the variants are possible. We name a few of these variants and describe the characteristics.

Cordeau et al. (2007) mention the VRP with time windows (VRPTW). In a VRPTW the service of a customer i must start in a given time window $[a_i, b_i]$. This means that serving customer i may not start before time a_i and the service may not start later than b_i, which is the end of the time window. If the vehicle arrives too early at the location of customer i, it must wait until the beginning of the time window of the service. Thus the VRPTW introduces the duration of travels and service times at customers.

In the Multiple Depot VRP (Tansini et al., 2003), more than one depot exists. Routes from different depots are possible. At the end of each route, each vehicle must return to the depot from which it departed. Generally, the Multiple Depot VRP (MDVRP) is solved in two parts. The first part assigns customers to a depot and the second part computes the routes from the depot.

In a VRP with pickups and deliveries (VRPPD), a vehicle must first drive to a customer to pickup a load before it drives to the location of delivery. The sequence of visiting the locations is thus important. In the Pickup and Delivery Problem (PDP), which is a subclass of the VRPPD, there are paired pickups and deliveries. This means that the load has to be picked up at a specific location and be delivered at one other location. In the Pickup and Delivery Vehicle Routing Problem, the pickups and deliveries are unpaired. This means that the loads that are picked up can fulfill the demand of any delivery customer. The VRPPD is similar to the VRP with backhauls, which also has pickups and deliveries. The difference between these two problems is that in VRP with backhauls first all deliveries are done. After all deliveries are done, the vehicle collects other loads to bring these back to the depot (Toth and Vigo, 2001). In the VRPPD, the load that is picked up is the load that has to be delivered at a later point in the route, which is not the case in the VRP with Backhauls where the deliveries and pickups are independent of each other.

In dynamic VRPs (DVRP), opposing to the classical VRP, not all information is available for the planners during construction of the routes. Additionally, the information, such as the number of orders or the time windows of deliveries, may change after construction of the routes. An example of new information can be an incoming priority order from a customer when the vehicles already started with their routes (Larsen, 2000).

In the stochastic VRP, there are components with an unknown value. Basically, there are four variables that can be stochastic: customers, demand, travel times, and service times. In the first case a customer is present with a probability p and not present with probability $(1-p)$. In case 2, the size of the demand can be stochastic and in the last two cases, the times of travel or the times of service can vary. The stochastic VRP is much more difficult to solve than the deterministic VRP (Cordeau et al., 2007). The difference between the stochastic VRP and the dynamic VRP is that in the dynamic VRP, the information available could change during execution of the routes, such as arrivals of new orders. In the stochastic VRP, the routes are created with stochastic information and the execution phase is static.

2.2 Methods to solve the VRP

There are generally two approaches for solving the VRPs. The first is to try to find an optimal solution by using an exact algorithm, which often only can be applied to small instances. The other option is to use an approximate method, a heuristic, which does not guarantee to find the optimal solution, but it can solve large instances in practical computation time. We first describe exact algorithms that are developed to solve the VRP after which we describe several heuristics that are used.

2.2.1 Exact Algorithms

Exact algorithms are algorithms that find an optimal solution for the problem. The simplest algorithm is the complete enumeration of all possible solutions and the best solution is the one with the lowest costs. Examples of other exact algorithms are Dynamic Programming and Branch and Bound algorithms. In this section, we discuss the Branch and Bound algorithm briefly. We discuss the Dynamic Programming algorithm in Chapter 3. In the Branch-and-Bound algorithm, the focus is on splitting the set of all feasible solutions in subsets of solutions. This is the branching procedure of the algorithm. During this process as many subsets of solutions as possible are eliminated for further investigation. This elimination is based on values of the upper and lower bound of the objective value of the subset. In a minimization problem, the global upper bound of the solution value is the value for best solution found so far in the process. For each subset the lower bound is calculated. If in a minimization problem the lower bound of the objective value in one subset is higher than the global upper bound, this subset can be eliminated because it will not lead to a better solution than the best solution found until then. The quality of the bounds and the speed to calculate the bounds in branch-and-bound are crucial for the algorithm to be effective. The algorithm stops when the subset of possible solutions is reduced to one, or when the upper bound of a subset is the same as the best lower bound of all solutions.

2.2.2 Heuristics

An alternative for using exact algorithms is using a heuristic. A heuristic is a method that finds a satisfactory solution in short time. Heuristics are used, for example, when it is not practical to use an exact algorithm. For the VRP, we distinguish 2 types of heuristics, namely route construction heuristics and route improvement heuristics. We briefly describe these heuristic types. Cordeau et al. (2002) describe four attributes that indicate good VRP heuristics. These attributes are accuracy, speed, simplicity, and flexibility. Most of the heuristics have the assumption that there is no fixed number of vehicles available. However, each vehicle has a limited capacity.

Route Construction Heuristics

A well-known route construction heuristic is the savings algorithm of Clark and Wright. The start situation of this algorithm is to visit each customer in a separate route. The basis of this algorithm is to combine two routes by connecting a customer from one route to a customer in the other route, if this reduces the costs and the sum of the amounts of the loads of the customers is not exceeding the vehicle capacity. If i is the last customer of one route and j is the first customer of the second route then the savings s of combining these routes are $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. If s_{ij} is positive, then it is cheaper to join the two routes. The algorithm stops when there are no positive savings left when combining routes.

Another construction heuristic is the insertion heuristic. There are a few variants of this heuristic. Two of these are the nearest insertion heuristic and the farthest insertion heuristic. The nearest insertion heuristic starts with two locations that are close together. Then the next location is chosen by removing an edge from the route in cluding new location in the route at that place. The location that increases the total distance the least is chosen. This process continues until all locations are inserted in the route. In the farthest insertion heuristic, instead of choosing the locations that have the smallest distance in between, the locations with the largest distance are chosen as initial solution. After that, the location that increases the total distance the most is inserted in the route. The reasoning behind this is that there could be locations that are difficult to fit into the route, and by handling these as first, a better route can be found.

The last type of heuristics we mention here are the so-called two-phase heuristics. Two-phase heuristics divide the VRP in two sub problems and solve these sub problems sequentially. In the cluster-first-route-second heuristic, the sub problems are the clustering of the customers and the routing of the vehicles. For each vehicle a cluster of customers is made. The clusters of customers can be achieved in different ways. One method is the sweep algorithm. The sweep algorithm begins with an arbitrary customer. Then, a line is drawn between the depot and that customer. That line is moved either clock wise or counter clock wise, with the depot as center. Only one direction is used during the heuristic. While making this circle, each customer that comes on the line is added to the cluster, as long as the total demand of that cluster stays within the capacity of the vehicle. In case no capacity of the vehicle is left, a new cluster is made for the next vehicle. This procedure continues until each customer is assigned to a cluster. After these clusters are made, optimal routes are made by solving a TSP within each cluster (Cordeau et al., 2007).

The route-first-cluster-second heuristic is also a two-phase heuristic. In this heuristic, first a route is constructed through all customers. Then this route is split in multiple routes, such that the demand of the customers in a route never exceeds the capacity constraints of the vehicle of that route.

Route Improvement Heuristics

When a solution of a VRP is found with a heuristic, other heuristics can be applied to improve this solution. Examples of such improvement heuristics are local search heuristics. Local search heuristics move to neighborhood solutions, until a stopping criterion is reached. As we illustrate later on, neighborhood solutions are solutions that are created by slightly modifying the original solution. The quality of local search heuristics is highly dependent on the criteria of selecting the neighborhood solutions.

The 2-opt and 3-opt heuristics are examples of local search heuristics. With these heuristics, respectively 2 or 3 connections between customers in a solution are exchanged. These exchanges can be done within one route or the connections can be exchanged between different routes. The new solution found is called a neighborhood solution of the original solution. All 2-opt and 3-opt heuristics choose the new solution if it is better than the current solution. Then a new iteration of exchanges of connections is done, etc. The heuristics stop after a given number of iterations or when no neighbor solution is better than the current solution.

In simulated annealing, a randomly chosen neighbor solution is accepted to be the new starting point for the neighborhood search, when it is better than the current solution. When the neighbor solution is worse than the current solution, it is accepted with a probability p (the acceptance probability) and rejected with probability $1 - p$. The acceptance probability is based on a function that is depending on the difference in the objective values of the current solution and that of the neighborhood solution, and a temperature factor. At high temperatures, the neighborhood solution has a higher chance to be selected than at lower temperatures. During the calculations, the temperature decreases by a predetermined cooling factor. With a higher cooling factor, the calculations take a shorter time, but with lower cooling factors it is more likely to find a better solution. From the accepted solution the neighborhood search continues. By also accepting neighborhood solutions that are worse than the current solution, it is possible to escape from local optima.

Tabu search is another local search heuristic. Tabu search starts with an initial solution. Then the neighborhood solutions are determined and the best of these solutions is chosen as the current solution and the previous is put on a list, the tabu list. From the new current solution the new best neighbor is selected. The tabulist is of a predetermined length and the solutions that are on that list may not be chosen as the current solution. This prevents the algorithm to return to earlier found solutions. If the maximum number of solutions on the list is reached, the oldest is dropped from that list. Each solution is compared with the best solution so far, and when it is better it is remembered as the new best solution.

Population search algorithms, such as genetic algorithms, search through the solution space by creating new solutions from a pool of good solutions (parent solutions). Child solutions are created by combining two parent solutions selected from the set of available good solutions. Within these new child solutions several mutations are done. These new (mutated) solutions replace the worst of the set of parent solutions. From this new set of parent solutions again two are combined, et cetera.

The last category of heuristics we discuss are the learning mechanisms. These heuristics use the best properties of solutions found in previous iterations to create a new solution. In ant colony algorithms for example, behavioral elements from ant colonies are used. Ants lay pheromone on the trails they use. Thus, when a trail is used more often, then the amount of pheromone on that trail is higher. Ants have the tendency to use the trails that have high levels of pheromone more often than the other trails, because it is likely this will lead to a food source. The ant colony heuristic uses this principle in the creation of the VRP solution. If a particular edge is used in a good solution, it gets a higher probability to be selected in the solutions of the following iterations. Thus with each iteration a little more is learned about the best solution.

2.3 Giant Tour Representation

The method we use for finding solutions for the VRPDY is based on a method to solve the TSP. The TSP is a special case of the VRP. Unlike the VRP, in the TSP a tour must be found for only one vehicle or person. Since we want to find solutions for the VRP, we have to represent it as a TSP solution. We can represent the solution of the VRP as a TSP solution by means of the Giant Tour Representation as introduced by Funke et al. (2005). The solution of the TSP is one tour starting in the depot, connecting the customer nodes, and finishing in the depot again. The VRP solution has a multiple of such routes $(r_1, .., r_n)$, where each route r_i represent a route performed by vehicle i. To represent the VRP solution as one giant route, we have to introduce new nodes. For each vehicle route r_i, the depot node is divided into a start node s_i and a finish node f_i. Thus, the set of nodes V is adjusted and consist of three subgroups of nodes, the start nodes S, request nodes R, and finish nodes F. The request nodes represent required visits to customers. All nodes have to be visited exactly once. The Giant Tour Representation of a VRP solution is the tour that is created by connecting the depot finish node f_i of route r_i with the start node s_{i+1} of the next route r_{i+1}. By connecting the finish node of the last route with the start node of the first route a complete tour is created. Figure 2.1 shows a solution of a VRP with 5 sub tours and Figure 2.2 shows the Giant Tour Representation of that solution.

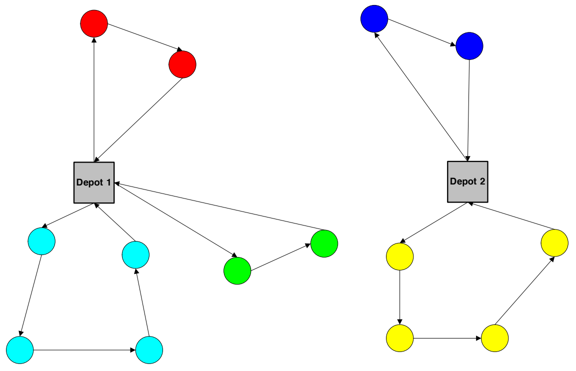


Figure 2.1: Solution of a VRP with 2 depots and 5 vehicles

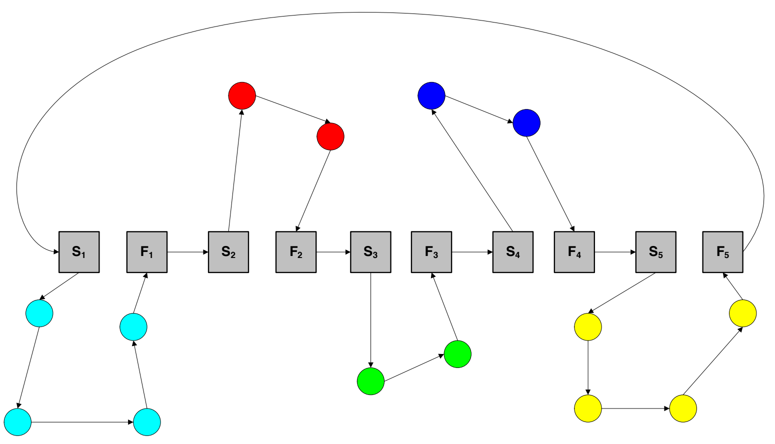


Figure 2.2: GTR of the VRP solution

2.4 Drop Yards

In the literature, little can be found about the use of drop yards in combination with the vehicle routing problem as we have described in Section 1.3. Some authors describe similar problems where vehicles with trailers are used. In their problems, the vehicle consists of two parts, a motorized vehicle, often called the truck, and a trailer, which cannot move on its own. Both parts can carry loads from customers. In our model we have a tractor pulling multiple, independent trailers. The tractor in our problem is not able to load goods and is only used for pulling the trailers. The tractor can perform single trailer tours with each of the trailers. A second difference is that in most of the literature the trailers are decoupled at customer locations instead of a dedicated drop yard as it happens in our problem. Oth, however, describes dedicated parking places in their VRP with Trailers and Transshipments. We discuss some of the cases as described in the literature.

In the Vehicle Routing Problem with trailers as described by Gerdessen (1996), the trailer is parked at a customer site, so that the truck can do a tour on its own. Gerdessen (1996) addresses a case in the Dutch dairy distribution. In the problem, there is a homogeneous fleet and each customer site can be a parking place for trailers. The complete combination of truck and trailer is called the vehicle. Thus, a customer can be visited by a truck or the vehicle, consisting of the truck and trailer. Each customer site has a level of inconvenience for the vehicle, which is denoted by maneuvering time. Difficult customer sites have higher maneuvering times. To save time at customer sites and to save fuel by using the truck on its own, without the trailer, for a part of the routes, the trailer is parked at a customer site in the route. Each trailer is parked exactly once and each customer is visited by a vehicle or truck exactly once.

Gerdessen (1996) proposes several heuristics to solve the vehicle routing problem with trailers. Gerdessen describes 3 constructive heuristics and 7 improvement heuristics. Construction heuristic I is very straightforward, it first solves a standard VRP without considering the parking of trailers. After that it solves a Traveling Salesman Problem with Trailers (TSPT) for every route resulting from the VRP. In this problem the route is divided in a truck route and a vehicle route, and a parking place for the trailer is chosen. This approach is simple, but can lead to many 'difficult' customers in one vehicle route. Heuristic II solves this problem by first dividing the set of customers into a truck set and a vehicle set. Next, truck routes are created for the set of truck customers. A parking place, which is a vehicle customer site, is inserted in each truck route. These parking places are used as seed for the vehicle routes that are created next. Heuristic III starts with the creation of truck routes and vehicle routes. In the next step these routes are connected at the best places possible.

The improvement heuristics of Gerdessen are divided in 2 kinds. The first kind of improvement heuristics consist of exchanging customers between different routes. Gerdessen uses 4 of these heuristics. The first exchanges chains of customers between 2 similar routes, which are two vehicle routes or two truck routes. The second heuristic exchanges chains of vertices between dissimilar routes, thus between a vehicle route and a truck route. The last 2 heuristics are a 2-opt and a 3-opt between vehicle routes. The second kind of improvement heuristics is focusing on improving the sequence of vertices within one vehicle route. Gerdessen (1996) describes 3 of these heuristics. The first contains improvement of a single route, a vehicle route or a truck route. The second heuristic consists of exchanges of vertices between the truck part and the vehicle part of one route. The last improvement heuristic is the application of a TSPT heuristic to the customers of a route.

Semet and Taillard (1993) allow vehicles to decouple and couple trailers at multiple locations. These locations are customer sites just as in the case of Gerdessen. In the problem of Semet and Taillard (1993), stores are visited to deliver loads. It contains a heterogeneous fleet with 21 trucks and 7 trailers. The set of customers is divided in a set of trailer stores, which can receive deliveries by the truck-trailer combination, and truck stores, which can only receive deliveries by truck only. Each trailer store can be used as a parking place for the trailer. While the truck does a subtour visiting truck stores, the trailer is unloaded. Semet and Taillard (1993) solve the problem with Tabu Search techniques. To investigate the neighborhood of the solution they move orders from one route to another. They applied their method for a company in Switzerland.

Semet (1995) applies a two-phase heuristic method to solve the Partially Accessibility Vehicle Routing Problem (PACVRP). The PACVRP contains, in addition to capacity constraints, accessibility constraints for the vehicles at customer locations. This means that not all vehicles can visit every customer. In the first phase of the heuristic, trailers are assigned to trucks and customers are assigned to trucks or road-trains (the combination of a truck and trailer). In the second phase, routes are constructed by solving TSPs or the TSP under Accessibility Constraints (ACTSP), which is similar to the PACVRP, with the difference that a route for only 1 truck and 1 trailer is constructed. For the trailer assignment problem of the first phase, Semet (1995) develops a branch-and-bound algorithm. In addition, Semet provide an ILP formulation for the PACVRP by extending the formulation for the ACTSP.

Schreuerer (2004, 2006) calls the problem of the VRP with trailers the Truck and Trailer Routing Problem (TTRP). In the TTRP, there is a heterogeneous fleet of trucks and a homogeneous fleet of trailers. Additionally there are two types of customers, truck customers and vehicle customers. Each truck customer can only be visited by a truck and each vehicle customer site can be used as parking place for the trailers. Scheuerer (2004) propose two new construction heuristics for the TTRP, a cluster heuristic and a sweep heuristic. Scheuerer (2004) also develop a new Tabu Search algorithm for the TTRP. Finally, approaches for a multi depot version and a periodic version of the TTRP are proposed.

Drexl (2007a) discusses the VRP with Trailers and Transshipments (VRPTT). In this problem, there are autonomous vehicles (lorries) and non-autonomous vehicles (trailers). Each vehicle can be a collection vehicle or a support vehicle, where collection vehicles are used to collect supplies from customers and the support vehicles operate as mobile depots and cannot visit any customers because of accessibility constraints at the customer locations. There are intermediate locations that can be used as parking places for the different vehicles or as a place to transfer loads between vehicles. Trailers can be pulled by any vehicle during its tour and the vehicles are not assigned to any particular depot. Drexl (2007a) proposes a MIP formulation for this problem. Drexl (2007a) and Drexl (2007b) also propose a MIP formulation for the TTRP and a Branch-and-Price algorithm to solve the problem.

Lin et al. (2009) propose a simulated annealing heuristic for the TTRP. For their neighborhood search, they use swaps and insertions of customers, and changes in vehicle types.

2.5 Conclusions

In this chapter, we discussed the Traveling Salesman Problem and various variants of the Vehicle Routing Problem. We also discussed some of the methods to solve the VRP and methods to improve the solutions of the VRP. We showed that the TSP is a special case of the VRP and that we can represent the VRP solution as a TSP solution by means of the Giant Tour Representation. Finally, we discussed the research that is done by other authors that incorporate the use of trailers in the VRP. We have seen that the main difference with our problem of the VRP with drop yards is that all problems of the other authors contain motorized vehicles with loading capabilities and one trailer, instead of a tractor, which cannot handle any loads, pulling multiple trailers. In our problem, the vehicle can do tours with each trailer separately. Further, in most problems, the trailers are parked at customer sites instead of dedicated drop yards in the VRPDY.

Chapter 3

Dynamic Programming

According to Bradley et al. (1977), Dynamic Programming is an optimization approach that can solve complex problems by dividing it in a sequence of simpler sub problems. These sub problems are solved in multiple stages, and each stage consists of multiple partial solutions, that are called states. In each stage a part is added to each partial solutions. In the last stage the optimal solution is found. The Dynamic Programming approach can be used for many types of problems. For the TSP, in each stage one node is added to each partial solution. Each stage consists of multiple states, which each contain a different sequence of nodes that are already planned.

In this chapter, we explain the Dynamic Programming algorithm for the TSP proposed by Held and Karp (1961) and demonstrate it on a small example. We explain the effect of order relations between nodes on the number of states that have to be investigated during the algorithm. Finally we explain how the Giant Tour Representation of Chapter 2 can be used in combination with the order relations to solve the VRP with the Dynamic Programming algorithm.

3.1 DP Algorithm for the Traveling Salesman Problem

Held and Karp (1961) formulate a Dynamic Programming algorithm that solves the Traveling Salesman Problem. The algorithm works as follows. Let V be the set of all nodes representing all customer locations with 0 the departing node. Given $S \subseteq V \backslash \{0\}$ and $i \in S$, let $C\{S,i\}$ be the cost of the route with minimum cost that starts in node 0, visits all nodes in set S and ends in i. Then the recurrence relation is

$$\begin{aligned} C(\{i\},i) &= c_{0i} && (3.1) \\ C(S,i) &= \min_{j \in S \setminus \{i\}} \left[C(S \setminus \{i\},j) + c_{ji}\right] && (3.2) \end{aligned}$$

The second part of the recurrence relation 3.1 can be explained as follows. Consider the shortest path from node 0 to node i, visiting all nodes from $S \setminus \{i\}$, that has node $j \in S$ immediately preceding node i. Since the nodes in $S \setminus \{i\}$ are visited in optimal order, the length of this path is $C(S \setminus \{i\},j) + c_{ji}$. By taking the minimum over all choices of j we obtain the minimum of $C(S,i)$ (Held and Karp, 1961). The ultimate goal is to find the minimum cost of a complete tour, terminating in node 0, which is given by

$$C(V,0) = \min_{i \in V \setminus \{0\}} \left[C(V \setminus \{0\},i) + c_{i0}\right] \qquad (3.3)$$

Algorithm (3.1) shows the algorithm of Held and Karp.

Algorithm 3.1 The Dynamic Programming algorithm of Held and Karp

INPUT :	A set of nodes V, and costs c_{ij} that gives the cost between every i and j in V
OUTPUT :	Cost of the optimal solution

```
for all $i \in V \setminus \{0\}$ do
  $C(\{i\}, i) = c_{0i}$
end for
for all $k$ =2 to $|V \setminus \{0\}|$ do
  for all $S \subset V \setminus \{0\}$ with $|S| = k$ do
    for all $j \in S$ do
      $C(\{S\}, j) = \infty$
      for all $m \in S \setminus \{j\}$ do
        if $C(S \setminus \{j\}, m) + c_{mj} < C(S, j)$ then
          $C(S, j) = C(S \setminus \{j\}, m) + c_{mj}$
        end if
      end for
    end for
  end for
end for
$C(V, 0) = \min_{i \in V \setminus \{0\}} C(0, V, j) + c_{j0}$
```

The algorithm starts with the paths from the depot to the first customers. In each following stage a node is added to each path. This implies that the number of stages is equal to the number of nodes that have to be planned. Each stage consists of multiple states with costs $C\{S, i\}$. The end node i in a state can be reached via multiple paths through the set of nodes S, but only the path with the least cost is memorized. At the last stage the optimal tour from 0, through all customer nodes, and back to 0 is found. We apply this algorithm to a small example.

Example: DP Algorithm for the TSP

We have 3 customers that have to be visited by 1 vehicle, see Figure 3.1. The vehicle starts at the depot. Matrix C contains the costs between every 2 nodes.

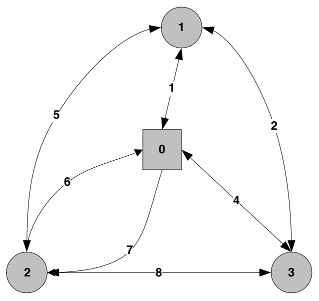


Figure 3.1: Example

$$C = \begin{pmatrix} - & 1 & 7 & 4 \\ 1 & - & 5 & 2 \\ 6 & 5 & - & 8 \\ 4 & 2 & 8 & - \end{pmatrix}$$

The first step is to set the costs from the start node 0 to each node i to be the cost c_{0i}. This is always the first travel in a route. Thus:

$$\begin{aligned} C(\{1\},1) &= 1 \\ C(\{2\},2) &= 7 \\ C(\{3\},3) &= 4 \end{aligned}$$

The following step is to add a node to each set S from the previous stage, thus the size of each set S becomes 2. We continue with the next stage which contains the following costs:

$$\begin{aligned} C(\{1,2\},1) &= C(\{2\},2) + c_{21} = 7+5 = 12 \\ C(\{1,2\},2) &= C(\{1\},1) + c_{12} = 1+5 = 6 \\ C(\{1,3\},1) &= C(\{3\},3) + c_{31} = 4+2 = 6 \\ C(\{1,3\},3) &= C(\{1\},1) + c_{13} = 1+2 = 3 \\ C(\{2,3\},2) &= C(\{3\},3) + c_{32} = 4+8 = 12 \\ C(\{2,3\},3) &= C(\{2\},2) + c_{23} = 7+8 = 15 \end{aligned}$$

In the next stage there are 2 possibilities to reach each state and we choose the minimum of these options.

$$\begin{aligned} C(\{1,2,3\},1) &= \min\{C(\{2,3\},2) + c_{21}, C(\{2,3\},3) + c_{31}\} \\ & \min\{12+5, 15+2\} = 17 \\ C(\{1,2,3\},2) &= \min\{C(\{1,3\},1) + c_{12}, C(\{1,3\},3) + c_{32}\} \\ &= \min\{6+5, 3+8\} = 11 \\ C(\{1,2,3\},3) &= \min\{C(\{1,2\},1) + c_{13}, C(\{1,2\},2) + c_{23}\} \\ &= \min\{12+2, 6+8\} = 14 \end{aligned}$$

Finally, we have to complete the tour by connecting the last node to the start location.

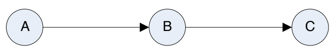
$$\begin{aligned} C(\{1,2,3\},0) &= \min\{C(\{1,2,3\},1) + c_{10}, C(\{1,2,3\},2) + c_{20}, C(\{1,2,3\},3) + c_{30}\} \\ &= \min\{17+1, 11+6, 14+4\} = 17 \end{aligned}$$

There are 2 optimal tours, namely 0-3-1-2-0 and 0-1-3-2-0. Both have costs of 17.

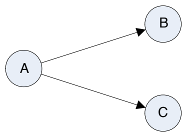
In the algorithm, each state (S, i) is the optimal path from the start node to node i visiting all nodes in set S. All other paths to i are not used in further stages. This means that to find an optimal path through all nodes, we do not explicitly evaluate every possible path.

3.2 Order Relations

Order relations indicate how two nodes relate to each other. We need to introduce order relations to be able to create feasible solutions. The order relations indicate which nodes could be planned



(a) Dependent order relations, version 1



(b) Dependent order relations, version 2

Figure 3.2

at a certain stage in the algorithm. There are restrictions on where the nodes in DP can be planned in the solution. Whether a node can be planned depends on other nodes that already are planned or not. For example, in the TSP the last depot node can only be planned after all other nodes are planned in the solution. These kinds of relations between nodes are listed in order relations. The solution is only feasible if all order relations that are defined hold. If one or more of the order relations are violated the solution is not feasible. To get a feasible solution the order relations should be checked before a node is selected. We denote the order relations as follows:

$$p\left(a,b\right) = \begin{cases} 1 & \text{if node } a \text{ has to precede node } b \\ 0 & \text{otherwise} \end{cases}$$

$$p_d\left(a,b\right) = \begin{cases} 1 & \text{if node } a \text{ has to precede node } b \text{ directly} \\ 0 & \text{otherwise} \end{cases}$$

The second variant of the order relations indicate that two nodes must be connected in a solution and that no other nodes may be planned between them. Greve (2006) discusses the effect of order relations on the number of states that is investigated per stage. If node b can only be planned after node a for example, then all options with node b planned before node a are infeasible. The number of states is reduced because all states with node b in it while a is not planned do not have to be investigated. If multiple order relations are present, these can be either dependent or independent. Two order relations are dependent if one node is occurring in both relations. If we have 3 nodes (a, b, and c), then we have 2 forms of dependent order relations. In the first form, the relations are $p(a,b) = 1$ and $p(b,c) = 1$. In the second form of dependent order relations, the order relations are $p(a,b) = 1$ and $p(a,c) = 1$. Figure 3.2 shows both situations.

In case the dependency of the restrictions is of the first form then the number of states in DP is reduced with factor $\frac{1}{2}$. In case the second form of dependency is present the number of states is reduced with factor $\frac{5}{8}$. For independent restrictions the number of states is reduced with $\frac{1}{4}$ per restriction. For a more extensive explanation of the reduction of states in the DP algorithm we refer to Greve (2006).

3.3 The DP Algorithm Adjusted for the Vehicle Routing Problem

By using the GTR we can rewrite the VRP as a TSP. Then, we can use the DP algorithm of Held and Karp for the TSP to solve the VRP. Since we use the GTR we introduce new nodes. The routes for each vehicle are planned subsequently and we introduce a start node and finish node, representing the start and finish of each vehicle tour. Before we can use the DP algorithm we have to define the order relations that involve these nodes. The order relations in this new problem are:

$$\begin{aligned} p(s_i, f_i) &= 1 & (3.4)\\ p_d(f_i, s_{i+1}) &= 1 & (3.5)\\ p(s_1, r) &= 1 & (3.6)\\ p(r, f_n) &= 1 & (3.7) \end{aligned}$$

Order relation 3.4 ensures that the finish node of a tour is always planned after the start node of that tour. With order relation 3.5 each vehicle route is finished before a new vehicle route starts. Since this is a direct relation, no other nodes can be planned in between. Request nodes can only be planned after the first route start and before the finish node of the last route, which is secured in order relations 3.6 and 3.7.

An advantage of the Dynamic Programming approach is that it is very flexible. It is very easy to include new restrictions of all kinds next to the order restrictions. One of these restrictions could be a loading restriction. For example a load of a customer can only be loaded in a vehicle that has a cooled compartment. Other restrictions could be time window restrictions for the deliveries. At every state these restrictions should be evaluated.

3.4 Conclusions

In this chapter, we described the Dynamic Programming algorithm of Held and Karp (1961) to solve the TSP and gave an illustration of how it works. Further, we described the influence of order relations on the number of states in the DP algorithm. We showed that there are two kinds of dependent order relations, which both reduce the number of states, with different factors. We showed how we can adjust the algorithm to enable it so solve the VRP. We used the GTR, which we discussed in Chapter 2 for this purpose. Finally, we mentioned that the DP algorithm is flexible, since it is not difficult to include restrictions to the problem.

Chapter 4

Basic Model

In this chapter we describe the basic model of the VRP with drop yards. We also describe a DP approach, based on the DP described in Chapter 3, to solve this model. At the end of this chapter we will show an example of this basic model. In the following chapter we extend this model to apply it to more realistic situations.

4.1 The Model

In a VRP model, all customers must be served with a number of vehicles. Each vehicle performs a tour from the depot and returns to that depot when it is finished. For simplicity, we assume that only deliveries take place and that all loads are available at the depot before the tour starts. In our basic model we start with a problem with 1 vehicle. At the end of this chapter we explain how to adjust the model to cope with multiple vehicles. Our vehicle has 2 trailers, which both have limited capacity. Next, we assume that the total load of 1 customer never exceeds the capacity of one trailer. Thus if a load for a customer is too large, it should be represented to the model as multiple loads, such that each load is smaller than the trailer capacity. The tour of the vehicle is similar to the tour in a regular VRP since it also starts and finishes at the depot. The difference is that its first move from the depot is to drive to a drop yard to decouple one of the trailers. We assume that every customer can only be visited by vehicles with only 1 trailer and because we have 2 trailers, the drop yard must always be visited first before deliveries can take place. From the drop yard the vehicle makes a tour visiting customers until the trailer is empty. After that it returns to the drop yard to switch trailers. With the other trailer now coupled, the vehicle does a tour again to deliver the loads from that trailer. Thus during solving the problem, the loads of the customers must be assigned to one of the trailers. When finished, the vehicle returns to the drop yard to collect the trailer that was left behind. After coupling, the combination travels back to the depot and the total tour is finished. Figure 4.1 shows an example of the VRP with a drop yard. This model is very similar to the regular VRP, the only differences are the first and the last connection of the route. These connections are the travels between the depot and the drop yard.

To summarize:

1. We start with the situation where only deliveries take place. We assume that the loads are available at the depot. Consequently, the tour in this situation will start at that depot.

2. We assume that only 1 drop yard is available.

3. Only 1 truck is available.

4. The truck carries 2 trailers.

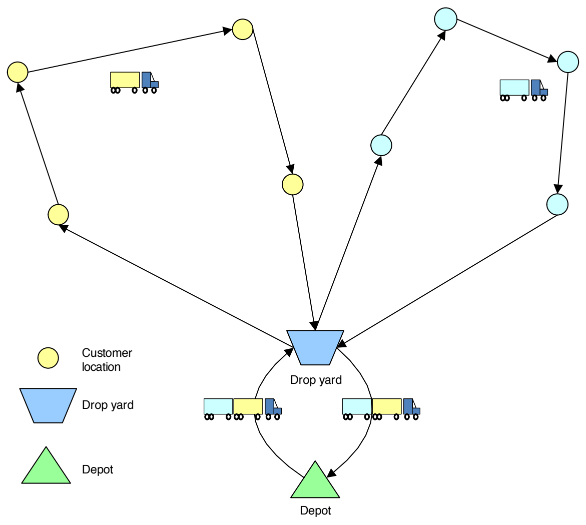


Figure 4.1: VRP with drop yard

5. All customers can only be visited with 1 trailer.
6. A tour starts with a travel from the depot to the drop yard. There, one of the trailers is decoupled and then customers are visited. When the carried trailer is empty it is switched with the other trailer at the drop yard. After visiting the remaining customers, the vehicle drives back to the drop yard where both trailers are coupled. From there the combination travels back to the depot.
7. The capacity of the trailers is limited.
8. The load of the customer may never exceed the capacity of a trailer. If a load is larger than the trailer capacity it should be split into multiple smaller loads. The input of the algorithm contains only loads that do not exceed the trailer capacity.

4.2 Solving the Basic Model with Dynamic Programming

In this section we describe the elements that are used to enable DP to solve the basic model. We develop an algorithm and we will demonstrate it with our example used in Chapter 2, in which we include a drop yard.

4.2.1 Drop Yard Nodes

In the DP for the VRP we described in Chapter 2, we split the depot node in two nodes, a start node and a finish node. We did this because in the DP algorithm we can use each node only once and we needed to keep the Giant Tour intact. Since the depot is used twice in each vehicle route, as a start and finish point for each route, we needed the extra depot nodes. In the model with the drop yards, the vehicle visits the drop yards multiple times, thus we create extra nodes for each drop yard visit. The drop yard is visited 3 times by the truck, each time with a different configuration of trailers. The first time it carries 2 trailers and one is decoupled. The second time the trailers are switched. The trailer on the truck is decoupled and the trailer that waited at the drop yard is coupled to the truck. The third and last time the trailer is at the drop yard the waiting trailer is coupled to the combination so that both trailers are coupled. To keep the Giant

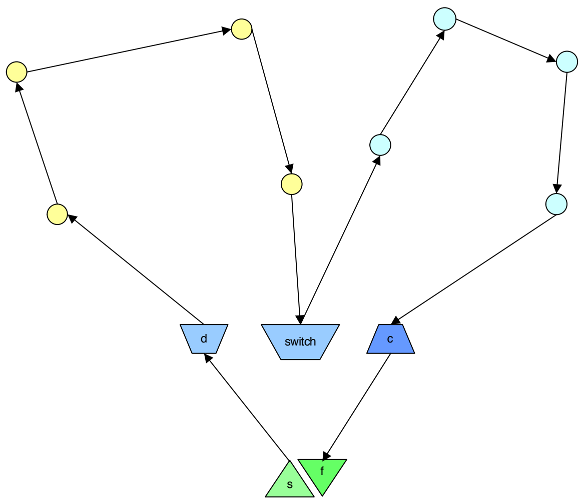


Figure 4.2: VRP with 3 drop yard nodes

Tour intact, we need to split the drop yard in 3 nodes, a decouple node for the first decouple action d, a switch node sw for the switch action, and a couple node c for the last couple action. Creating these extra nodes enables us to use the DP algorithm of Held and Karp. Figure 4.2 shows how the drop yard nodes are used in the tour.

4.2.2 Loads

The loads have to be delivered at the request nodes. Since we assume that all loads are available at the depot at the start of the route, we do not have to consider the pickups in our planning. Consequently, the sequence of the request nodes is not implied by any order relations. For every load a request node is made. Thus if a customer has a demand that contains multiple loads, multiple nodes are created for that customer. At every request node, a load is delivered and is removed from the trailer. After the drop yard visit, the trailer visits several request nodes and returns to the drop yard afterwards. Which request nodes and in what sequence is determined in the DP algorithm. In a tour, the sum of the loads that are in the trailer may not exceed the capacity of that trailer. Thus, before a request node is added to the solution, the remaining capacity of the trailer must be checked whether the load can be added to the trailer. At the moment the request node is added to the solution, the customer and the load are automatically assigned to a trailer. This is the trailer that is currently coupled to the truck, which is trailer 1 if the switch node is not planned yet and it is trailer 2 after the switch node is planned. The capacity of that trailer is then reduced by the amount of the load of the customer.

4.2.3 Order of the Nodes in the Solution

In this model a few order relations exist. The first node in the solution must always be the start node. The last node in the solution must always be the finish node. Since we do not allow customer visits with more than 1 trailer, we must connect the decouple node of the drop yard to the start node. We can only plan the request nodes after the decouple node is planned. We can only plan the couple node, for coupling the trailers again, when all request nodes are planned. In the basic model, there are no dependencies between request nodes, thus the request nodes can be planned in any order in the solution. The switch node must be planned after the decouple node and before the couple node. As no nodes but the finish nodes are left, this finish node is connected to the couple node. Thus, the order relations in this model are:

$$\begin{aligned}
p_d\,(s,d) &= 1 && \text{The decouple node must come after the start node}\\
p\,(d,r) &= 1 && \forall r \in R \text{ all request nodes must be planned after the decouple}\\
p\,(d, switch) &= 1 && \text{The switch node must be planned after the decouple node}\\
p\,(switch, c) &= 1 && \text{The couple must be planned after the switch node}\\
p\,(r,c) &= 1 && \forall r \in R \text{ all request nodes must be planned before the couple}\\
p_d\,(c,f) &= 1 && \text{The finish node can only be planned after the couple node}
\end{aligned}$$

We omit the order relation $p\,(s,f)$ because the other order relations together guarantee that the finish node is planned after the start node.

4.2.4 Costs and State Description

In the cost matrix the drop yard nodes are added as well as the corresponding costs of the travel to and from the drop yard nodes. Because the capacity of the trailer is an important issue we should also check whether it is possible to add loads to the trailer during the DP. To keep track of the current available capacity we incorporate this in the calculations. We add the factor capacity to the state description. The state description Z then reflects a cost component, the set of planned nodes, the last node, and a component that represents the remaining capacity of the current trailer in that state. Thus $Z = \{C\,\{S, i\},\ S,\ i,\ \text{remaining capacity}\}$. The capacity is changed every time a request node is added. If in two states, the same customer set is visited, one state is only dominating the other if both the costs and the remaining capacity are better. Thus, the costs must be lower than the other state and the remaining capacity must be higher as well. In any other case, we keep two copies of the state with the same customer set. We apply DP to our new model in the example in Section 4.2.6.

4.2.5 Multiple Vehicles

In the basic model, we assume that there is only one vehicle. We can extend this model for the use of multiple vehicles. If we have an instance with more than 1 vehicle with multiple trailers, we want to enable that each vehicle can visit every customer if that is necessary. To enable this, all these multiple trailer vehicles must have the possibility to visit the drop yard to decouple and couple trailers. For this, there must be drop yard nodes for each vehicle. Thus, each vehicle has its own drop yard couple, switch and decouple node. These drop yard nodes should all be visited. In addition, for each vehicle a start and finish node must be made to mark the start and finish of that vehicle tour. Because we use the GTR to make a complete solution, we plan each vehicle subsequently. First, a vehicle tour must be completed before a next vehicle tour can be started. Thus, the start node of vehicle i can only be planned after the finish node of vehicle $i-1$ is planned. No customer nodes or other nodes can be planned in between these nodes.

4.2.6 Example of the DP Algorithm Applied to the Basic Model

We include a drop yard in the example of Section 3.1 (see Figure 4.3). The cost of travel between the drop yard and the other nodes is also indicated. We assume that the capacity of the trailers is the same for both and is 10 units. We still have 3 customers and the demand of each customer is 4 units.

	s	f	d	sw	c	1	2	3
s	-	-	4	-	-	-	-	-
f	0	-	-	-	-	-	-	-
d	-	-	-	0	-	2	3	8
sw	-	-	-	-	0	2	3	8
c	-	4	-	-	-	-	-	-
1	-	-	-	2	2	-	5	2
2	-	-	-	3	3	5	-	8
3	-	-	-	8	8	2	8	-

Table 4.1: Costs between nodes

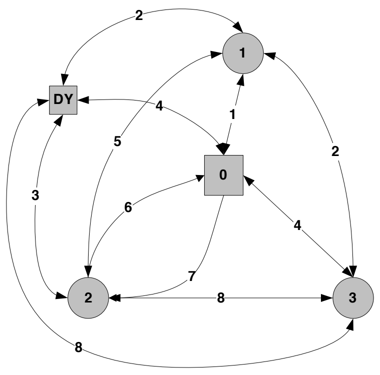


Figure 4.3: VRP with drop yard example

Table 4.1 shows the costs between nodes including the drop yard nodes. We present the outcomes of the DP in Table 4.2. In Table 4.2, each small table shows the states created with a set of nodes. Each row in such a small table shows the costs of a state for a given end node. For example, the second small table in Table 4.2 shows the states with nodes s and d. The end node of the state is d, the costs are 4, and the remaining capacity of the current vehicle is 10. We give a full overview of the calculations of this example in Appendix A.

Solution

There are 4 optimal solutions for this problem which all have costs of 26. These solutions are:

1. $s - d - 2 - sw - 3 - 1 - c - f$
2. $s - d - 2 - sw - 1 - 3 - c - f$
3. $s - d - 1 - 3 - sw - 2 - c - f$
4. $s - d - 3 - 1 - sw - 2 - c - f$

In the optimal solutions, the loads of 1 and 3 are combined and delivered with one of the trailers. Load 2 is delivered to the customer with the other trailer.

4.3 Conclusions

In this chapter we discussed the most simple use of drop yards in a vehicle route. In our problem we assumed that there is 1 truck carrying 2 trailers and that there is only 1 drop yard. In the DP

$\{s\}$	
s	$(0, 10)$

$\{s, d\}$	
d	$(4, 10)$

$\{s, d, 1\}$	
1	$(6, 6)$

$\{s, d, 2\}$	
2	$(7, 6)$

$\{s, d, 3\}$	
3	$(12, 6)$

$\{s, d, 1, 2\}$	
1	$(12, 2)$
2	$(11, 2)$

$\{s, d, 1, 3\}$	
1	$(14, 2)$
3	$(8, 2)$

$\{s, d, 2, 3\}$	
2	$(20, 2)$
3	$(15, 2)$

$\{s, d, 1, sw\}$	
sw	$(8, 10)$

$\{s, d, 2, sw\}$	
sw	$(10, 10)$

$\{s, d, 3, sw\}$	
sw	$(20, 10)$

$\{s, d, 1, 2, sw\}$	
1	$(12, 10)$
2	$(11, 10)$
sw	$(14, 10)$

$\{s, d, 1, 3, sw\}$	
1	$(22, 10)$
3	$(16, 10)$
sw	$(16, 10)$

$\{s, d, 2, 3, sw\}$	
2	$(23, 10)$
3	$(18, 10)$
sw	$(23, 10)$

$\{s, d, 1, 2, 3, sw\}$	
1	$(28, 2)$
1	$(20, 2)$
1	(25,6)

$\{s, d, 1, 2, 3, sw\}$	
2	$(27, 2)$
2	$(24, 2)$
2	$(19, 6)$

$\{s, d, 1, 2, 3, sw\}$	
3	$(14, 2)$
3	$(19, 2)$

$\{s, d, 1, 2, 3, sw, c\}$	
c	$(22, 0)$

$\{s, d, 1, 2, 3, sw, c, f\}$	
f	$(26, 0)$

Table 4.2: Example Basic Model

algorithm we introduced three new drop yard nodes, a decouple node, switch node, and a couple node. Together with these new nodes, we introduced new order relations for the problem to enable DP to use the drop yard nodes correctly. Finally we adjusted the state description by adding the attribute capacity, which indicated the capacity of the trailer that is coupled to the truck in that state.

Chapter 5

Extensions of the Basic Model

The basic model from Chapter 4 contains a simple and straightforward use of drop yards. The real life use of drop yards, however, can be very different from that of the model. We select some situations that are likely to occur in reality and include these in the model. In this chapter we describe extensions of the basic model. Each new extension also includes the properties of all extensions described before. However, each exstension can also be used as an extension of the basic model, independently of the other extensions.

First of all, companies may have more than one drop yard available. A choice should be made which drop yard to use in the vehicle tours. We describe a solution for this case in Section 5.1. We also describe how we can use the depot as a possible drop yard, which means that the vehicles do single trailer tours. We show that this is also a solution to the case that only one trailer is needed to serve all customers.

In Section 5.2, we describe the use of more than two trailers. In some countries, the number of trailers pulled by a vehicle is not limited to 2. This means that the pulling vehicle has to visit the drop yard more often to decouple and couple trailers.

In Section 5.3, we describe the situation that customers allow multi trailer vehicles on their site. When it is more efficient to drive directly to such a customer instead of driving to the drop yard first, this should be made possible. These customers are visited with the whole vehicle combination, which can take place before the drop yard is visited or after all trailer tours from the drop yard are finished.

Finally, in Section 5.4, we describe the situation that more than one drop yard is used in one vehicle tour. To prevent the vehicle to travel very long distances with one trailer, it could be easier to travel from one drop yard, where a trailer tour is done, to another drop yard to do a second trailer tour. Between these two drop yards, the trailer should be able to visit customers with the whole vehicle combination, if this is possible at the customer location. In addition to this, we show how a trailer can be used multiple times for a trailer tour.

5.1 One of Multiple Drop Yards is used in a Tour

It could be possible that a company has more than one drop yard location available. In that case, a choice has to be made which drop yard to use in the tour of a vehicle. The vehicle serves all customers from that drop yard. In this section, we describe how this can be modeled. We also describe the situation that the depot is considered as one of the available drop yards. When the depot is chosen as the drop yard, the vehicle performs trailer tours directly from the depot. This could be favorable when many customer locations are between the depot and the other drop yards. If, in that case, one of the other drop yards is used, the vehicle has to make a detour to visit that drop yard first and consequently the tour will be longer. Another advantage of using the depot as a drop yard is when the total amount of the load in a tour is less than or equal to the capacity of one trailer. Then it is not necessary to visit a drop yard to decouple trailers and it is much more efficient to do a single trailer tour directly from the depot.

5.1.1 Inclusion in the Model

In this model, we assume that when a vehicle uses a drop yard to decouple a trailer, it uses this drop yard also for the other trailer. In other words, a vehicle may not use a second drop yard in its tour. Before any decoupling of a trailer is done, there is a choice of where the first decouple takes place, thus a drop yard should be chosen for the tour. We describe two approaches to deal with this situation.

The first approach is to create a decouple node, switch node, and a couple node for every drop yard available. For each vehicle we create such a node set. We restrict ourselves to use not more than one drop yard in the vehicle tour, thus after the first decouple node of a drop yard is planned, the nodes of the other drop yards are not allowed in the solution anymore. The number of available nodes to plan can be reduced by not allowing the obsolete drop yard nodes to be planned after the moment that the decouple node is planned. The remainder of the DP algorithm is now the same as in the basic model, with only the switch node and a couple node for 1 drop yard that can be used. The DP algorithm is finished when all request nodes are planned and the tour is completed. The remaining drop yard nodes of the other drop yards stay unplanned. A consequence of this approach is that all options with all drop yards are investigated. This can lead to a high number of states that is investigated and increases the computation times. One way to deal with this problem is to limit the number of available drop yards to choose from. For example, one can preselect 2 or 3 drop yards, from a larger set, that can be used in the algorithm. This, however, causes that the algorithm does not necessarily leads to an optimal solution.

The second approach is to postpone the decision for which drop yard to use until the next request node is planned. As before, the decouple node is planned after the start node but this time no costs between these nodes are calculated and the drop yard node does not represent a specific drop yard yet. Instead, the costs for the route are calculated at the request node that is planned next. While adding the request nodes to the solution, the costs for the route are calculated including the decouple node. This is calculated for each possible location for the drop yard, which is given in the set of drop yards that can be used. The drop yard location that results in the lowest cost is stored in the current drop yard location for the decouple node. For each request node, switch node and couple node that is added, the best location for the drop yard is recalculated. Obviously, the location must be the same for each drop yard node, since we only allow the use of one drop yard in the vehicle tour. This approach results in fewer nodes for the drop yard: only 1 decouple node, 1 couple node, and 1 switch node are needed per drop yard. This also implies much fewer nodes in the DP algorithm that have to be planned and much fewer options that have to be investigated. Since the recalculation of the tours is quick the total calculation time will be shorter than in the first approach.

5.1.2 Use of the Depot as the Drop Yard

We can imagine that using the depot as a drop yard can be useful in some situations. For example, when many or all request nodes are located between the depot and the available drop yards. In that case, the vehicle must first pass all customers to decouple trailers at the drop yard, while it is shorter to go directly to the first customer to deliver the first load. To prevent the vehicle to use this long route and to allow the vehicle to deliver loads to customers directly from the depot with a single trailer, we enable that the vehicle uses the depot as a drop yard. To accomplish that, we add the depot to the set of available drop yards to choose from. Thus we make a decouple node, a couple node, and switch nodes for this drop yard. In the cost matrix, the cost from other depot nodes to the drop yard nodes, representing the depot, are 0 and the costs from these drop yard nodes to the request nodes are the same as the other distances from the depot to the customers.

If the amount of the loads from all customers together in a tour of a vehicle is at most the capacity of one trailer, it is not necessary to use two trailers and, consequently, it is unnecessary to visit a drop yard. In this case, all customers can be served with one trailer. This trailer can drive directly from the depot to the first customer, because no trailers have to be decoupled at the drop yard. In the basic model, we assume that the truck carries two trailers. This means that the second trailer can stay at the depot in this case, which saves travels from and to the drop yard with an empty trailer. In the solution of the DP algorithm calculations for this case, the switch node is directly followed by the couple node of the vehicle and the costs of travel between these nodes is 0.

5.2 More than two Trailers on a Truck

In the basic model of Chapter 4, two trailers were coupled to a truck. However, it should be possible that a truck carries more than two trailers. In this situation, we still assume that only one trailer at a time can visit customer locations. Consequently, more than one trailer has to be decoupled at the drop yard to get a truck with only one trailer. From the drop yard, more local tours are performed, namely one for every trailer. The route of the vehicle combination is not very different from the route in the basic model. The truck drives from the depot to the drop yard, where trailers are decoupled so that only one trailer is left on the truck. With that trailer a tour is done delivering goods to customers. Then the trailer is switched with another trailer at the drop yard and a tour is done with that trailer, then it is switched again, et cetera, until each trailer is used in a tour. After each trailer is used, all trailers are coupled to the truck. Then, the complete combination drives back to the depot.

5.2.1 Inclusion in the Model

The main difference with the original situation is that at the decouple node, more than one trailer is decoupled. In fact, all trailers but one are decoupled. At the first arrival at the drop yard the first trailer stays on the truck and the others are decoupled. After the tour of trailer 1 it is switched with the next trailer. After all trailers have performed a tour, all of them are coupled. Thus we have a decouple node and a couple node, just as in our basic model. Next to the decouple and couple node we have several switch nodes. If we have k trailers, then we have $k-1$ switches between trailers and thus we need $k-1$ switch nodes. The order relations in this model are:

$$\begin{array}{rcll}
p_d\left(s,d\right) & = & 1 & \text{The decouple node must come after the start node} \\
p\left(d, switch_1\right) & = & 1 & \text{The first switch node must be planned after the decouple node} \\
p\left(switch_n, switch_{n+1}\right) & = & 1 & \text{Switch node } n \text{ precedes switch node } n{+}1 \\
p\left(switch_{k-1}, c\right) & = & 1 & \text{The couple must be planned after the last switch node} \\
p_d\left(c,f\right) & = & 1 & \text{The finish node can only be planned after the couple node}
\end{array}$$

Similar to the basic model the couple node can only be planned after all request nodes are planned. Also the other order relations fix the order relation between the start an end node.

5.3 Customer Visits with Multiple Trailers, Before and After Drop Yard Visits

Some customers have enough space to handle vehicles with more than one trailer. It should be possible to visit these customers with larger vehicle combinations. If the location of such a customer is on the route or close to the route from the depot to the drop yard it may cost less to deliver the load at that moment instead of incorporating this delivery in a tour from the drop yard. The customer can be visited when the vehicle is driving to the drop yard, or after the couple of the vehicles when the tours from the drop yard are finished. Then the loads can be delivered on the way back to the depot. It is not necessary to include the customer visit in a tour from the drop yard with one trailer. Even if the customer location is not on the path between the depot and the drop yard, it could be cheaper to make a detour to deliver the load to that customer with the whole vehicle combination. For example, if the locations from the other customers are clustered together, and the customer that can handle multiple trailers is located apart.

From the situation described above, we distinguish two kinds of customers: customers that allow multiple trailers on their site and customers that can only be visited with 1 trailer at a time. From now we will refer to the first kind of customer as a *vehicle customer* and to the latter as a *trailer customer*. In our model we require that trailers are decoupled or switched at a drop yard, resulting in just one trailer coupled to the truck, which is needed to serve the trailer customers. Vehicle customers can be served with 1 trailer in a drop yard tour, or by a vehicle with more than 1 trailer. The load of a customer cannot be divided to be loaded in multiple trailers. For serving the vehicle customers with multiple trailers, the truck carries all available trailers. For example, in this model we do not allow that a 3-trailer truck decouples 1 trailer at the depot to visit a customer with 2 trailers. Further, during the route construction, the loads are assigned to the trailers. For delivering loads at the vehicle customer locations with the complete vehicle, it does not matter in which trailer the load is loaded, similar to the fact that we do not determine where any load is placed in the trailer. However, the assignment of the loads has an influence on the remainder of the vehicle tour, since the load of vehicle customers take part of the capacity of the trailers.

5.3.1 Inclusion in the Model

Since each customer could handle a different number of trailers, the request nodes should have an attribute that indicates whether the corresponding customer is a vehicle customer or a trailer customer. Request nodes that represent vehicle customers can now directly succeed the start node. If it is a request node for a trailer customer, then a drop yard visit must be planned first, before this node could be planned, and after the request node, a couple node must be planned to couple the trailers back to the vehicle. Thus, for every request node, it should be registered what kind of customer it represents, but also, at any moment, the number of trailers that are currently coupled to the truck should be known. In the model, we assume that at the drop yard all trailers but one are decoupled. This means that the vehicle has two states, in one state it carries all trailers and in the other state it carries only one. Before the first drop yard node is planned, the maximum number of trailers is coupled. At that moment, only vehicle customers could be planned or the decouple node of the drop yard could be planned. The other request nodes are unavailable at that moment, otherwise we would create an infeasible solution. After the decouple node is planned, only 1 trailer is left on the truck and the other trailers are parked at the drop yard. At that moment, the switch nodes are available as well as all request nodes.

For the request nodes representing vehicle customers, we have to assign the loads to one of the trailers that are carried by the truck. We assign the loads to the first trailer that is on the truck if it has enough capacity left to add the load. If this trailer is full, the load is assigned to the next trailer. The capacity of the trailer where the load is loaded is reduced by the amount of the load.

By doing this, the guarantee of optimality for the solutions is lost, since we assign the load to one trailer in a state and we do not have separate states for load assignments of the same load to other trailers. For the trailer customers that are planned after the decouple node, the load assignment is the same as in the basic model. The load of a customer must always be loaded in one trailer, thus we cannot divide the load over multiple trailers. Thus, if the trailer is not full, but the complete load cannot be assigned to the trailer, this load cannot be added to the trailer.

5.3.2 State Description

It is important to know which trailers are on the truck at all times, at every state in the DP algorithm. In our model, this is k before the drop yard decouple node is planned or 1 after the drop yard decouple node is planned. After the couple nodes are planned the number of trailers on the truck is k again. Further, it is important to know what the remaining capacities of these trailers are, because in a tour from a drop yard with a trailer, we must know which loads still fit in the trailer, if any. Therefore, we give each trailer a number and we add these to the state description, so that in each state it is clear which trailers are coupled. Together with the trailer number for each trailer on the truck, we give the remaining capacity for each of these trailers. Then, the state description becomes as follows: $Z = \{C(\{S\}, i),\ S,\ i,\ \{trailers\},\ \{capacities\}\}$, where $\{trailers\}$ is the set of trailers coupled to the truck in that state and $\{capacities\}$ the remaining capacities corresponding to those trailers. For example, a state description could look like this: $Z = (15, \{0,1,2,3,4,5\}, 5, \{1,2\}, \{4,10\})$. In this state, trailer 1 with remaining capacity 4 and trailer 2 with remaining capacity 10 are coupled to the truck.

5.3.3 Order Relations

The order relations of this model are similar to the order relations of the previous model. We split the set of request nodes R in two parts: set R_v, representing the vehicle customers and set R_t representing the trailer customers. We also split the order relations for the request nodes in two parts. The first part is quite obvious, namely all request nodes must be planned between the start node and the finish node. The second part is that the request nodes representing customers that cannot handle the total vehicle combination, must be planned between the decouple and the couple node of the drop yard. Below, we show the order relations for this extension.

$$\begin{array}{rcll}
p(s,d) & = & 1 & \text{The decouple node must come after the start node} \\
p(s,r) & = & 1 & \forall r \in R_v \text{ All vehicle customers must be planned after the start node} \\
p(d,r) & = & 1 & \forall r \in R_t \text{ All trailer customers must be planned after the decouple node} \\
p(d, switch_1) & = & 1 & \text{The first switch node must be planned after the decouple node} \\
p(switch_n, switch_{n+1}) & = & 1 & \text{Switch node } n \text{ precedes switch node } n\text{+}1 \\
p(switch_{k-1}, c) & = & 1 & \text{The couple must be planned after the last switch node} \\
p(r,c) & = & 1 & \forall r \in R_t \text{ All trailer customers must be planned before the couple node} \\
p(r,f) & = & 1 & \forall r \in R_v \text{ All vehicle customers must be planned before the finish node} \\
p(c,f) & = & 1 & \text{The finish node can only be planned after the couple node}
\end{array}$$

5.4 Multiple Drop Yards may be used in one Tour

The use of multiple drop yards within one vehicle tour can be cheaper than delivering to customers from one drop yard. In this section we describe how we can adjust the model to make this possible and also to visit vehicle customers during the travel between two drop yards.

The following example clarifies the advantage of the use of multiple drop yards in a vehicle tour. Consider, a vehicle with two trailers that has to deliver goods in two cities that both have a drop yard nearby. There are also deliveries available for customers between the two cities. These customers are vehicle customers. The customers in the cities are trailer customers. Assume that for the deliveries at trailer customers, only one trailer at a time may be used. Thus, the vehicle must first stop at a drop yard to decouple a trailer. The vehicle decouples a trailer at the drop yard near the first city to make a tour. After finishing the deliveries in the first city, all trailers are coupled at the drop yard to drive to the second city. During the travel between the two cities, the vehicle could deliver goods to vehicle customers with both trailers coupled. At the second city, the second trailer is decoupled to deliver goods in that city and when it is finished the vehicle drives back to the depot.

5.4.1 Inclusion in the Model

In our previous cases, we had 3 kinds of nodes for a drop yard per vehicle: a decouple node, several switch nodes, and a couple node. This means that each trailer on the vehicle is used at a drop yard. Even if only one trailer has actually a tour from that drop yard, all the switch nodes must be planned before the vehicle is finished with that drop yard. In the new situation, we want to make the use of the trailers more flexible. We make the decoupling and coupling of a trailer independent of the other trailers. Thus, we make a decouple node and a couple node per drop yard for each trailer. We do that for each drop yard, to enable that each trailer can be used at each drop yard. At a drop yard, a decouple node is planned for a trailer. After a number of request nodes, the tour of the trailer is ended by planning the couple node of that trailer. This is different from the previous cases where a switch node indicated the end of the trailer tour. To make a tour with the next trailer, the decouple node of that trailer must be planned. This decouple node can be for the same drop yard as the first trailer, but also for another drop yard. Between the couple node of the first trailer and the decouple node of the second trailer, all trailers are coupled. This means that in that situation vehicle customers can be planned. Figure 5.1 shows a vehicle tour with two

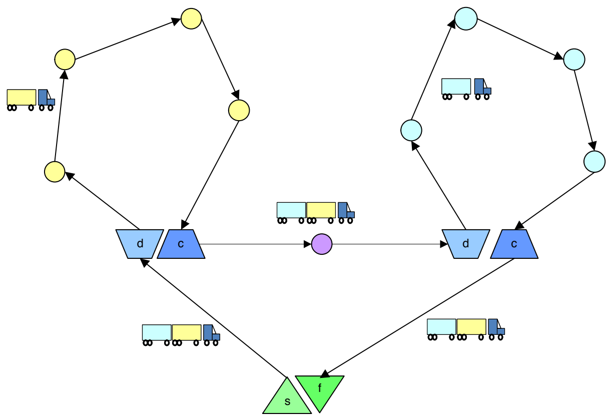


Figure 5.1: Tour with decouple and couple nodes for each trailer

trailers and two drop yards. In the picture, there is one trailer tour from each drop yard. Note that at each drop yard, also a tour with the other trailer is possible. For example, if all of the yellow trailer's capacity is used for the tour from the first drop yard, a second tour with the blue trailer is possible from the same drop yard.

We create drop yard nodes for each trailer at each drop yard. This makes it possible to use a trailer multiple times. The order relations of the new situation become:

$$
\begin{array}{rcll}
p\,(s, d_1) & = & 1 & \text{The first decouple node must be planned after the start node} \\
p\,(s, r_v) & = & 1 & \text{All vehicle customers must be planned after the start node} \\
p\,(d_n, c_n) & = & 1 & \text{The couple node representing a drop yard of a trailer} \\
 & & & \text{must be planned after the decouple node of the same drop yard}
\end{array}
$$

Each trailer customer node must be planned
between two drop yard nodes of the same drop yard:

$$
\begin{array}{rcll}
\forall r \in R_t & \exists k & & \\
s.t. & & & \\
p\,(r, c_n) & = & 1 & \forall n \geq k \\
p\,(r, c_n) & = & 0 & \forall n < k \\
p\,(d_n, r) & = & 1 & \forall n \leq k \\
p\,(d_n, r) & = & 0 & \forall n > k
\end{array}
$$

$$
\begin{array}{rcll}
p\,(c_n, d_{n+1}) & = & 1 & \text{Couple node } n \text{ precedes decouple node } n{+}1 \\
p_d\,(r, d_n) & = & 0 & \forall r \in R_t \text{ no trailer customers may be planned before a decouple node} \\
p_d\,(c_n, r) & = & 0 & \forall r \in R_t \text{ no trailer customers may be planned after a couple node} \\
p\,(c_k, f) & = & 1 & \text{All couple nodes must be planned before the finish node} \\
p\,(r, f) & = & 1 & \forall r \in R. \text{ All customers must be planned before the finish node}
\end{array}
$$

Similar to the case of Section 5.1, we can determine for each drop yard node pair which drop yard they represent. We also can postpone that decision by determining during the algorithm, which drop yard location belongs to the drop yard nodes.

5.5 Conclusions

In this chapter, we proposed 4 extensions of the original VRP with drop yards from Chapter 4. The first extension is the selection of one drop yard per vehicle tour. This drop yard is selected from a set of multiple drop yards. We proposed two variants to solve this problem. In the first variant, we created drop yard nodes for each of the drop yards and let the DP algorithm sort out which of these is selected. The other variant created nodes for one drop yard and the location of this drop yard is determined while planning the request nodes. With this extension, we added the depot to the set of possible drop yards in a tour. By doing this we allow trailer tours from the depot when this is cheaper. Adding the depot to the drop yards also solved the case that only one of the two trailers is used in a vehicle tour. It is not necessary anymore to drive with an empty trailer.

The second extension to the basic problem is the use of more than 2 trailers on a truck. We enabled this by adding one extra switch node per trailer to the set of drop yard nodes. Solving the problem is identical to the problem with two trailers.

The third problem we solved is visiting customers with the complete vehicle. These customers we call vehicle customers. We allow these customers to be visited before and after the drop yard visits, but they can also still be visited with only a trailer. For this model we adjusted the state description once again so that the capacities of all trailers are known at the moment a vehicle customer is added. The load of the vehicle customer is added to the first trailer on the truck that has enough capacity.

The last extension is the use of multiple drop yards in one vehicle tour. This means that a vehicle can do trailer tours from different drop yards. For this model, we replaced the switch node for a couple and a decouple node. Thus we have a couple-decouple node pair for each trailer. By doing this, we also allow vehicle customers to be served when a vehicle travels from one drop yard to another.

Chapter 6

Computational Experiments

In the previous chapters, we developed a DP approach to solve a new problem, the VRPDY. The VRPDY is related to the VRP. In fact the VRP is a capacitated VRP (CVRP). Therefore, we select 15 instances for the VRP and add drop yards to convert them to instances for the VRPDY. To see how the VRPDY performs, we implemented the algorithm for the VRPDY in C++ to compare the results with the results of the VRP. For the comparison, we select some benchmark instances for the VRP and run the algorithm with these instances and compare the results with the best known results of these instances.

This chapter is organized as follows: in Section 6.1, we briefly describe how we implemented the DP algorithm. In Section 6.2, we describe the test cases for the VRPDY. In Section 6.3, we describe the data we use for the test cases. Finally, in Section 6.4, we show and discuss the results of the tests.

6.1 Implementation

We implemented the algorithm for the VRPDY as described in Section 5.4 since it is the most comprehensive model of the VRPDY we described. In this section, we discuss the implemented DP algorithm for the VRPDY in C++ and the adjustments we made to ensure reasonable computation times and memory use.

As the number of customers and the number of vehicles increase, the total number of nodes in the VRPDY increases. Consequently, the number of states to evaluate in the DP algorithm increases as well. The number of states in the DP algorithm will be so large that it takes a very long time to evaluate them all. Additionally, a large state space will require a vast amount of process memory of the computer. Therefore, we restrict the state space of each stage. We do that by limiting the number of states in each stage, which we call the stage width (H). We do that by only expanding H states in each stage. Each state reflects a partial solution and we take the states with the lowest cost to continue with in the next stage. We take the states that have the lowest cost, because we expect that these have a higher probability to result in a good solution than a state that represents a partial solution that has high costs. If two partial solutions have the same cost, the one that has the most request nodes planned is preferred above the other. If after the comparison of cost and number of request nodes two partial solutions are equal, we choose the partial solution with the highest demand planned. Note that with the limitations on the state space, not all possible states are evaluated. This an implication on the optimality of the solutions found. Optimality cannot be guaranteed anymore, since it is possible that the state that reflects the partial solution that eventually leads to the optimal solution is discarded during the calculations.

The cost between two nodes in our models are reflected by the distance between the two locations that are reflected by these nodes. In our models, we do not consider the time dimension. Thus, a travel between two locations, as well as loading and unloading actions, will not take any time. Since the costs between a decouple node and a couple node of a drop yard are 0, it is likely that solutions with a decouple and a couple node sequentially are selected to expand. This means that no request nodes are planned between them and this possibly leads to infeasible solutions when these request nodes cannot be planned at other places in the solution. To prevent this, we restrict the planning of the couple node of a drop yard in our implemented algorithm. While not all trailer customer request nodes are planned, the couple node can only be planned if the previous node is a request node.

When all request nodes are planned, the drop yard couple node is planned and the finish node for the vehicle is planned. If the last remaining request node represents a vehicle customer, the finish node of the vehicle is planned directly after the request node. It is unnecessary to plan any other drop yard nodes or start new vehicle tours and therefore the algorithm is terminated. Thus it is possible that a feasible solution can be found without all drop yard nodes planned. For the same reason, solutions with not all vehicle start and finish nodes planned are also possible. The vehicles that are represented by these nodes are not used in the solution.

6.2 Test Cases

We test our algorithm on benchmark instances to see how it performs. First, we test the algorithm for the VRP and compare the results with the best-known results of these instances. This will give an idea of the quality of the implemented algorithm.

Second, we test the VRPDY according to the problem we described in Section 5.4. Thus, there are multiple drop yards, and each trailer may be used in multiple drop yard tours. Additionally, the depot is modeled as a drop yard, to make single trailer tours are possible from the depot. In the tests, each vehicle carries two trailers. We compare the results of the VRPDY with the results of the VRP.

Finally, we include vehicle customers in the benchmark instances. Thus some customers can be visited without first dropping a trailer at a drop yard. We compare the results to the results of the VRPDY since it is an extension of the VRPDY.

6.3 Test Instances

To test the VRPDY, we select 15 benchmark instances for the VRP from www.branchandcut.org (2011). The instances are from Augerat et al. (set P and set B), Christophides and Eilon (set E), and Fisher (set F). We select a mix of instances of different sizes and with different numbers of vehicles. The test instances are named according to the format P-n16-k8, where P indicates the instance set and n the number of customer locations plus the depot location in the original instance. The letter k in the instance names indicates the number of vehicles in that VRP instance.

For the VRPDY, we add 2 drop yards to the set of locations to each instance. The drop yards are located at a place where they are likely to be used, thus, near clusters of customers if possible and at a certain distance from the depot, because it is not likely that a drop yard is used when it is located close to the depot. We use 2 drop yards per instance to keep the cases simple. In fact, there are 3 drop yards available to the vehicles since they can also use the depot as a drop yard.

Next, we set the number of trailers for per vehicle to 2 for each VRPDY instance. The capacity of each trailer in an VRPDY instance is the same as the capacity of a vehicle in the corresponding VRP instance. For a fair comparison, the number of trailers in the VRPDY instances is the same as the number of vehicles in the corresponding VRP instance. In case there are an odd number of vehicles in the VRP instance, the number of trailers is one higher in the VRPDY instance, since we keep the number of trailers per vehicle to 2. Note that this extra trailer is not necessarily used in the solution.

For the instances with vehicle customers, we mark half of the customers per instance as a vehicle customer. We select these customers randomly per test instance. We compare the solutions of the tests with the vehicle customers with solutions of the VRPDY without vehicle customers.

6.4 Results

In this section we discuss the results for the test cases as described in Section 6.2. The stage width used per instance varies, due to the variation in memory use per instance. For each instance, we choose H as large as possible with respect to the allowed memory use. In our implementation, the maximum amount of memory that can be used by the algorithm is 2 GB. This has some implications for the results, since we loose the guarantee of optimality of the solutions. Table 6.1 shows the results of the calculations of the calculations of the VRP and a comparison with the best known solution values of the VRP. Column 'Stage Width' states the maximum stage size of the calculations of the instance. Table 6.2 shows the results of the calculations of the VRPDY and the VRPDY with vehicle customer instances and gives a comparison with the VRP results.

Instance	Best known VRP solutions	VRP with our implementation	Gap	Stage Width(H)
P-n16-k8	450	450	0%	100000
P-n19-k2	212	212	0%	100000
P-n20-k2	216	216	0%	100000
P-n22-k8	603	603	0%	100000
P-n40-k5	458	472	3%	50000
P-n76-k4	593	655	10%	10000
P-n101-k4	681	822	21%	10000
B-n31-k5	672	677	1%	100000
B-n34-k5	788	795	1%	100000
B-n44-k7	909	1002	10%	50000
E-n22-k4	375	375	0%	100000
E-n30-k3	534	545	2%	100000
F-n45-k4	724	882	22%	50000
F-n72-k4	237	301	27%	10000
F-n135-k7	1162	1374	18%	10000
Average			8%	

Table 6.1: Results of calculations with VRP instances.

6.4.1 Results for the VRP

Table 6.1 shows the results of the calculations of the VRP instances and the comparison with the best known solution values. Column 'VRP with our implementation' shows the results for the VRP calculated with our algorithm. Column 'Best known VRP solutions' shows the best known results for the VRP instances. From column 'Gap', we see that the solutions calculated with our algorithm have on average 8% higher costs than the best known values. If we look closer to the

Instance	VRP with our implementation	VRPDY	VRPDY with vehicle customers	Gap VRP-VRPDY	Gap VRPDY-VRPDY with vehicle customers
P-n16-k8	450	397	366	-12%	-8%
P-n19-k2	212	224	224	6%	0%
P-n20-k2	216	234	225	8%	-4%
P-n22-k8	603	631	575	5%	-9%
P-n40-k5	472	567	530	20%	-7%
P-n76-k4	655	777	739	19%	-5%
P-n101-k4	822	913	947	11%	4%
B-n31-k5	677	559	566	-17%	1%
B-n34-k5	795	688	688	-13%	0%
B-n44-k7	1002	820	817	-18%	0%
E-n22-k4	375	425	383	13%	-10%
E-n30-k3	545	543	543	0%	0%
F-n45-k4	882	965	897	9%	-7%
F-n72-k4	301	350	346	16%	-1%
F-n135-k7	1374	1337	1337	-3%	0%
Averages				3%	-3%

Table 6.2: Results of the calculations with VRPDY instances

results, we see that for smaller instances the gap between the VRP solutions that are generated with our implementation and the best known VRP result is 0%. For the larger instances, the gap increases. This can be explained by the fact that in larger instances the maximum stage width is reached in an earlier stage and the partial solutions that lead to the optimal solution are not further investigated. Further, we see that the instance with 72 customers has a worse result than the instance with 135 customers, while the maximum allowed stage width is the same. This is caused by the fact that in the instance with 135 customers, the density of the customer locations is higher than in the instance with 72 customers. This make that a wrong decision in an earlier stage has less impact on the decisions in the following stages, since it can be corrected with relatively little extra costs. The difference between costs of planning 2 customers A and B in the sequences A-B or B-A is relatively small.

6.4.2 Results for the VRPDY

Column 'VRPDY' in Table 6.2 shows the results for the VRPDY. Column 'Gap VRP- VRPDY' shows the gap between the VRPDY and the VRP calculated with our algorithm (shown in column 'VRP with our implementation'). We see that on average the solution value of the VRPDY is 3% higher than the solution values of the VRP, while for the VRPDY, one would expect solutions that are at least as good as the solutions for the VRP. We also see that the differences between the values of the two solutions have a high variation. We also see that for the instances P-n16-k8, B-n31-k5, B-n34-k5, and B-n44-k7, the VRPDY has better solution values than the best known solution values for the VRP. This shows a high potential of this model for the VRPDY.

We explain the results as follows. In the VRPDY there are more nodes to plan than in the corresponding VRP. The extra nodes are nodes for the drop yards that are not in the VRP. In other words, the problem to solve is more complex. This would not have been a problem if we did not limit the state space and all states could be evaluated and expanded. In our case, the state space is limited and we only expand the solutions with the lowest cost. This could cause that partial solutions, that are not among the best, are discarded while they would have led to good solutions when all nodes are planned. In the calculations for the VRP and the VRPDY, we used the same stage width (H) for each solution. Thus, since the VRPDY has more nodes, the solutions of the two problems can be different. From the instances with VRP results that have a gap of 0% with the best known results for the VRP, 4 of the 5 have higher costs in the VRPDY result. This shows that the stage width has a great influence on the results and that this limit is reached quicker in the VRPDY instances.

We also have seen a great variation in the difference in quality between the solutions of the VRP and those of the VRPDY. Some of the solutions are much better than those of the VRP, while others are much worse. This can be explained by the nature of the instances. The solutions of the VRPDY are good for instances that have the following properties: the customers are located in multiple clusters and the drop yards are located close to these clusters or between the clusters. Another aspect that occurs in some of the instances with good solutions for the VRPDY, is that the clusters of customer locations are located relatively far from the depot and that the drop yards are located closely to the customer clusters. The savings are obtained by driving once from the depot to the drop yard for a vehicle with two trailers instead of traveling twice for two vehicles with one trailer.

For the instances where the VRPDY solution is worse than the solution of the VRP, the customer locations are more scattered over the area. The drop yards are located in between the customer nodes and it is not obvious to visit the drop yard first instead of performing trailer routes from the depot. In these cases, especially when the instance contains more vehicles, the problem becomes much more complex since there are many extra nodes.

6.4.3 Results for the VRPDY with Vehicle Customers

Columns 'VRPDY with vehicle customers' and 'Gap VRPDY- VRPDY with vehicle customers' of Table 6.2 respectively show the results for the instances where we included vehicle customers and the difference of these results with the results of the VRPDY. On average, the solutions for the VRPDY with vehicle customers have 3% less costs than the values for the VRPDY solutions. On Average, the results of the VRPDY with vehicle customers have the same costs as the VRP instances. For 13 of the 15 instances, the results for the VRPDY with vehicle customers are equal or better than the VRPDY results. This is what we expect, since the VRPDY with vehicle customers has more flexibility in the sequence that the customers can be planned. The vehicle customers can be planned in a tour from the drop yard, but they can be planned directly from the depot as well, when no trailers are decoupled.

6.5 Conclusions

We have seen that the use of drop yards can improve the vehicle routes that we calculated for the VRP with our DP algorithm. Some of the solutions are better than the best known solutions for the VRP. However, we have also seen that in some cases the solution is much worse. This difference is mainly caused by the nature of the instances. Instances that have customer locations in a cluster at a distance from the depot, tend to have a good solution value when a drop yard is used. Instances that have customer locations scattered all over the area, tend to benefit less from the use of drop yards. In these cases the depot is the best drop yard and it is more beneficial to solve the more simple model of the VRP.

Another reason we discussed, for the higher values of the VRPDY solutions, is that the VRPDY is more complex. The reason for this is that the number of nodes that is in the VRPDY is higher. For each trailer and for each drop yard, there are drop yard couple and decouple node pairs available. In larger instances this can add many extra nodes to the problem. For practicality we limited the state space, and in combination with the higher complexity of the VRPDY, we have seen that in some cases the solutions are not as good as for the VRP.

For the case the vehicle customers are included, we have seen that for most of the instances, the algorithm produced better results for the VRPDY than when no vehicle customers are included.

Chapter 7

Conclusions and Further Research

In this research, we developed a model for the VRP that incorporates the use of drop yards (VRPDY). We extended a Dynamic Programming algorithm for the TSP and the VRP to be able to solve the VRPDY. In this chapter, we draw conclusions in the first section, and in Section 7.2 we give some options for further research.

7.1 Conclusions

Literature

In the literature, the VRP is studied extensively. Many variants of the VRP have been described. There are only a few variants that have similarities to the VRPDY. Examples of the variants are the Vehicle Routing Problem with Trailers, the Truck and Trailer Problem, and the Partial Accessibility Vehicle Routing Problem. There are similarities, but these problems differ on a few points from our model. In our model, we have a vehicle consisting of a motorized truck that pulls several independent trailers. Each of the trailers can carry load, while the pulling vehicle is not able to do that. In the problems from the other authors, there is a truck, which can carry load, pulling one trailer. In the Truck and Trailer Problem, the VRP with Trailers problems, and the Partial Accessibility Vehicle Routing Problem, the trailer can be parked at customers' sites, while in the VRPDY, a dedicated drop yard is available for this purpose.

Dynamic Programming

This research is performed at ORTEC. For solving VRPs, ORTEC uses Dynamic Programming in their software. Therefore, we also use this approach to solve VRPs that incorporate the use of drop yards. We showed that we can use the Giant Tour Representation (GTR) to represent a VRP solution as a TSP solution. By using the GTR, we apply Dynamic Programming to the VRP. In the GTR, the depot is divided into start nodes and finish nodes of each vehicle tour, and the vehicle tours are connected with each other by connecting the start and finish nodes of two successive routes. The correct sequence of all customers and depot nodes is controlled with order relations. In the basic model we created for the VRPDY, there is one vehicle containing two trailers. There is one drop yard and every customer can only be visited with one trailer at a time. We introduced three new node types to the model for the decoupling, switching, and coupling of trailers. A feasible solution for this problem is guaranteed by the order relations between the nodes.

Extensions

We added several extensions to the basic model. In the first extension, multiple drop yards are available. A vehicle can use one of these drop yards in its tour. We suggested two approaches to include this in the model. The first and most straightforward approach is to create drop yard nodes for each available drop yard, such that each trailer can be used at each drop yard. The second approach is to create one set of drop yard nodes for each trailer. These nodes are not representing particular drop yards. The decision which drop yard the nodes represent is postponed until the construction of the routes.

The second extension we suggested is the use of more than two trailers on the vehicle. For each extra trailer, we introduced an extra switch node. With new order relations the correct use of the extra trailer is ensured.

The third extension is the incorporation of customer visits with multiple trailers. Some customers can be visited with the whole vehicle combination. For this, it is unnecessary for the vehicle to visit a drop yard. To model this, we made a distinction in customer types, namely vehicle customers and trailer customers.

The last extension is the use of multiple drop yards in one vehicle tour. In this variant, a vehicle can use a drop yard to do a local tour with one trailer. At the end of that local tour, it couples all trailers again and it continues to another drop yard. There it can do a local tour with the same trailer as before or one of the other trailers. In between the drop yard visits it can serve vehicle customers.

Test Results

We obtained results of 15 test instances for the VRP and added drop yards and trailers to convert them to instances for the VRPDY. We compared the results of the calculations for the VRPDY with the results for the VRP. For this comparison we used the model from the last extension. The extension with vehicle customers is tested separately. We tested the instances with a restricted version of the DP algorithm. With this restriction, the number of states investigated in the algorithm is limited. If we compare the results of the VRPDY with the results of the VRP, then we see that the values are on average 3% higher, but that there is a large difference per instance. For instances that have customers clustered at a distance from the depot, the VRPDY scores better than the instances where customers are spread all over the area. For 4 of the test instances the values are lower than the best-known solution for the VRP. Finally, we added the vehicle customers to the test instances. The solutions for these instances were on average 3% better than the VRPDY and for 5 of the instances better solutions were found than the best known solutions for the VRP. These results are explained by the fact that with the introduction of the vehicle customers, there is more flexibility in planning the vehicles and the drop yards.

7.2 Further Research

In this research, we discussed the modeling of drop yards in the VRP. We started with a basic model and added some extensions to that model. However, there are possible extensions to this model. In this section we name a few and we also discuss how to extend the model further.

Loads Larger than Trailer Capacity

In our models, we assume that a load of a customer always fits in one trailer. We make sure that the input of the model was according to this assumption by setting the capacity of each trailer in our instances for the VRPDY the same as the capacity of the original vehicles in the VRP instances. In all the instances we used, all trailers of an instance have the same capacity.

However, in real life, the size of loads varies and not every trailer has to be equal. If a fleet consists of large trailers and small trailers as well, not all customer loads may fit in the smallest trailer. Another issue is when a vehicle customer has a load larger than the capacity of a single trailer, it should be possible to divide the load over the trailers of that vehicle.

Time Windows

In our models we do not consider time windows. Time windows can be easily included in the model by keeping score of the time dimension at every state. When adding a node to the partial solution, the time restrictions of that node should be tested against the time dimension at that state for feasibility.

Trailers are used by Multiple Vehicles

In some cases it can be useful to allow that trailers are used by different vehicles. For example, empty trailers can be parked and collected by vehicles that return to the depot. Another case is the wave planning we described in Chapter 1. To make this possible, each trailer must be an individual resource, with no connections to the trucks or other trailers. The routes must be planned per trailer. Important is that a time element must be introduced to keep track of the times the trailers are at the customers or the drop yard.

Decoupling of a Variable Number of Trailers at a Drop Yard

In our models, we only allow single trailer tours from the drop yard. This means that all trailers except for one are decoupled at the drop yard. The model becomes more flexible when the number of trailers that are used for a tour from the drop yard is variable. We can imagine that it is sometimes better to park one trailer and continue with two trailers when a three-trailer truck is used.

Location of the Drop Yards

In our models and calculations, we assumed that the locations of the drop yards were determined before we started calculating routes. Before drop yards are taken in use, the best possible locations should be determined. Further research is needed to include the problem of determining the best drop yard locations, which leads to better routes in the VRPDY solutions.

Bibliography

Bellman, R. E., 1957. Dynamic Programming, courier dover publications (2003) Edition. Princeton University Press.

Bradley, S., Hax, A., Magnanti, T., 1977. Applied mathematical programming. Addison-Wesley Reading, MA.

Cordeau, J., Gendreau, M., Laporte, G., Potvin, J., Semet, F., 2002. A guide to vehicle routing heuristics. Journal of the Operational Research Society 53 (5), 512–522.

Cordeau, J., Laporte, G., Savelsbergh, M., Vigo, D., 2007. Vehicle Routing. Handbooks in Operations Research and Management Science 14, 367–428.

Drexl, M., 2007a. On Some Generalized Routing Problems. Ph.D. thesis, RWTH Aachen University, Germany.

Drexl, M., 2007b. A branch-and-price algorithm for the truck-and-trailer routing problem. Tech. rep., RWTH Aachen University, Germany.

Funke, B., Grünert, T., Irnich, S., 2005. Local search for vehicle routing and scheduling problems: Review and conceptual integration. J. Heuristics 11 (4), 267–306.

Gerdessen, J., 1996. Vehicle routing problem with trailers. European Journal of Operational Research 93 (1), 135–147.

Greve, H., 2006. Dynamic Programming as a General Framework for Routing Problems.

Held, M., Karp, R., 1961. A dynamic programming approach to sequencing problems. In: Proceedings of the 1961 16th ACM national meeting. ACM New York, NY, USA, pp. 71–201.

Larsen, A., 2000. The dynamic vehicle routing problem.

Lin, S., Yu, V., Chou, S., 2009. Solving the truck and trailer routing problem based on a simulated annealing heuristic. Computers and Operations Research 36 (5), 1683–1692.

Scheuerer, S., 2004. Neue Tabusuche-Heuristiken für die logistische Tourenplanung bei restringierendem Anhängereinsatz, mehreren Depots und Planungsperioden. Ph.D. thesis, Wirtschaftswissenschaftlichen Fakultät, Universität Regensburg.

Scheuerer, S., 2006. A tabu search heuristic for the truck and trailer routing problem. Computers and Operations Research 33 (4), 894–909.

Semet, F., 1995. A two-phase algorithm for the partial accessibility constrained vehicle routing problem. Annals of Operations Research 61 (1), 45–65.

Semet, F., Taillard, E., 1993. Solving real-life vehicle routing problems efficiently using tabu search. Annals of Operations Research 41 (4), 469–488.

Tansini, L., Urquhart, M., Viera, O., 2003. Comparing assignment algorithms for the Multi-Depot VRP. Internet: http://www. ng. edu. uy/inco/pedeciba/bibliote/reptec/TR0108. pdf, Abruf am 3.

Toth, P., Vigo, D., 2001. The Vehicle Routing Problem. Society for Industrial and Applied Mathematics.

www.branchandcut.org, 2011. www.branchandcut.org.
URL `http://branchandcut.org`

Appendix A

Basic Model Example

Stage 1

$$C(\{d\}, d) = (c_{sd}, cap) = (4, 10)$$

Stage 2

$$\begin{aligned}
C(\{d,1\},1) &= (C(\{d\},d) + c_{d1}, cap) \\
&= (4+2, cap) = (6,6) \\
C(\{d,2\},2) &= (C(\{d\},d) + c_{d2}, cap) \\
&= (4+3, cap) = (7,6) \\
C(\{d,3\},3) &= (C(\{d\},d) + c_{d3}, cap) \\
&= (4+8, cap) = (12,6)
\end{aligned}$$

Stage 3

$$\begin{aligned}
C(\{d,1,2\},1) &= (C(\{d,2\},2) + c_{21}, cap) \\
&= (7+5, cap) = (12,2) \\
C(\{d,1,2\},2) &= (C(\{d,1\},1) + c_{12}, cap) \\
&= (6+5, cap) = (11,2) \\
C(\{d,1,3\},1) &= (C(\{d,3\},3) + c_{31}, cap) \\
&= (12+2, cap) = (14,2) \\
C(\{d,1,3\},3) &= (C(\{d,1\},1) + c_{13}, cap) \\
&= (6+2, cap) = (8,2) \\
C(\{d,2,3\},2) &= (C(\{d,3\},3) + c_{32}, cap) \\
&= (12+8, cap) = (20,2) \\
C(\{d,2,3\},3) &= (C(\{d,2\},2) + c_{23}, cap) \\
&= (7+8, cap) = (15,2) \\
C(\{d,1,sw\},sw) &= (C(\{d,1\},1) + c_{1sw}, cap) \\
&= (6+2, cap) = (8,10) \\
C(\{d,2,sw\},sw) &= (C(\{d_1,2\},2) + c_{2sw}, cap) \\
&= (7+3, cap) = (10,10) \\
C(\{d,3,sw\},sw) &= (C(\{d,3\},3) + c_{3sw}, cap) \\
&= (12+8, cap) = (20,10)
\end{aligned}$$

Stage 4

$$\begin{aligned}
C\left(\{d,1,2,sw\},1\right) &= \left(C\left(\{d,2,sw\},sw\right)+c_{sw1},cap\right)\\
&= (10+2,cap)=(12,6)\\
C\left(\{d,1,2,sw\},2\right) &= \left(C\left(\{d,1,sw\},sw\right)+c_{sw2},cap\right)\\
&= (8+3,cap)=(11,6)\\
C\left(\{d,1,2,sw\},sw\right) &= \left(\min\left\{C\left(\{d,1,2\},1\right)+c_{1sw},C\left(\{d,1,2\},2\right)+c_{2sw}\right\},cap\right)\\
&= \left(\min\left\{12+2,11+3\right\},cap\right)\\
&= (14,10)\\
C\left(\{d,1,3,sw\},1\right) &= \left(C\left(\{d,3,sw\},sw\right)+c_{sw1},cap\right)\\
&= (20+2,cap)=(22,6)\\
C\left(\{d,1,3,sw\},3\right) &= \left(C\left(\{d,1,sw\},sw\right)+c_{sw3},cap\right)\\
&= (8+8,cap)=(16,6)\\
C\left(\{d,1,3,sw\},sw\right) &= \left(\min\left\{C\left(\{d,1,3\},1\right)+c_{1sw},C\left(\{d,1,3\},3\right)+c_{3sw}\right\},cap\right)\\
&= \left(\min\left\{14+2,8+8\right\},cap\right)\\
&= (16,10)\\
C\left(\{d,2,3,sw\},2\right) &= \left(C\left(\{d,3,sw\},sw\right)+c_{sw2},cap\right)\\
&= (20+3,cap)=(23,6)\\
C\left(\{d,2,3,sw\},3\right) &= \left(C\left(\{d,2,sw\},sw\right)+c_{sw3},cap\right)\\
&= (10+8,cap)=(18,6)\\
C\left(\{d,2,3,sw\},sw\right) &= \left(\min\left\{C\left(\{d,2,3\},2\right)+c_{2sw},C\left(\{d,2,3\},3\right)+c_{3sw}\right\},cap\right)\\
&= \left(\min\left\{20+3,15+8\right\},cap\right)\\
&= (23,10)
\end{aligned}$$

Stage 5

$$\begin{aligned}
C\left(\{d,1,2,3,sw\},1\right) &= (\min\{C\left(\{d,2,3,sw\},2\right)+c_{21},\\
&\qquad C\left(\{d,2,3,sw\},3\right)+c_{31},\\
&\qquad C\left(\{d,2,3,sw\},sw\right)+c_{sw1}\},cap)\\
&= \left(\min\left\{23+5,18+2,23+2\right\},cap\right)\\
&= (20,2),(20,2),(25,6)\\
C\left(\{d,1,2,3,sw\},2\right) &= (\min\{C\left(\{d,1,3,sw\},1\right)+c_{12},\\
&\qquad C\left(\{d,1,3,sw\},3\right)+c_{32},\\
&\qquad C\left(\{d,1,3,sw\},sw\right)+c_{sw2}\},cap)\\
&= \left(\min\left\{22+5,16+8,16+3\right\},cap\right)\\
&= (27,2),(24,2),(19,6)\\
C\left(\{d,1,2,3,sw\},3\right) &= (\min\{C\left(\{d,1,2,sw\},1\right)+c_{13},\\
&\qquad C\left(\{d,1,2,sw\},2\right)+c_{23},\\
&\qquad C\left(\{d,1,2,sw\},sw\right)+c_{sw3}\},cap)\\
&= \left(\min\left\{12+2,11+8,14+8\right\},cap\right)\\
&= (14,2),(19,2),(22,6)
\end{aligned}$$

Stage 6

$$\begin{aligned}
C\left(\{d,1,2,3,sw,c\},c\right) &= (\min\{C\left(\{d,1,2,3,sw\},1\right)+c_{1c},\\
&\qquad C\left(\{d,1,2,3,sw\},2\right)+c_{2c},\\
&\qquad C\left(\{d,1,2,3,sw\},3\right)+c_{3c}\},cap)\\
&= \left(\min\left\{28+2,20+2,25+2,27+3,24+3,19+3,14+8,19+8,22+8\right\},cap\right)\\
&= (22,0)
\end{aligned}$$

Stage 7

$$\begin{aligned}
C\left(\{d,1,2,3,sw,c,f\},f\right) &= \left(C\left(\{d,1,2,3,sw,c\},c\right)+c_{cf},cap\right)\\
&= (22+4,cap)=(26,0)
\end{aligned}$$

$\{s\}$	
s	$(0,10)$

$\{s,d\}$	
d	$(4,10)$

$\{s,d,1\}$	
1	$(6,6)$

$\{s,d,2\}$	
2	$(7,6)$

$\{s,d,3\}$	
3	$(12,6)$

$\{s,d,1,2\}$	
1	$(12,2)$
2	$(11,2)$

$\{s,d,1,3\}$	
1	$(14,2)$
3	$(8,2)$

$\{s,d,2,3\}$	
2	$(20,2)$
3	$(15,2)$

$\{s,d,1,sw\}$	
sw	$(8,10)$

$\{s,d,2,sw\}$	
sw	$(10,10)$

$\{s,d,3,sw\}$	
sw	$(20,10)$

$\{s,d,1,2,sw\}$	
1	$(12,10)$
2	$(11,10)$
sw	$(14,10)$

$\{s,d,1,3,sw\}$	
1	$(22,10)$
3	$(16,10)$
sw	$(16,10)$

$\{s,d,2,3,sw\}$	
2	$(23,10)$
3	$(18,10)$
sw	$(23,10)$

$\{s,d,1,2,3,sw\}$	
1	$(28,2)$
1	$(20,2)$
1	(25,6)

$\{s,d,1,2,3,sw\}$	
2	$(27,2)$
2	$(24,2)$
2	$(19,6)$

$\{s,d,1,2,3,sw\}$	
3	$(14,2)$
3	$(19,2)$

$\{s,d,1,2,3,sw,c\}$	
c	$(22,0)$

$\{s,d,1,2,3,sw,c,f\}$	
f	$(26,0)$

Table A.1: Example Basic Model