# HiMoldeMaster

## Master of Science in Applied Informatics

## Single Vehicle Pickup and Delivery Problem with Multiple Commodities and Visit Windows

**Arne Gjengstø**

**Sigbjørn Vaksvik**

**Molde, 2008**

Molde University College

**Abstract**

Offshore platforms are supplied using a fleet of vessels originating from one depot which serves the installations in a region. The installations need to be supplied several different kinds of commodities, i.e. various fluids like water, drilling-liquids and fuel stored in tanks below deck in addition to other goods stored in containers etc. up on deck. Pickup and delivery demands for installations not beeing met may lead to delayed or halted operations on the installation. This is very costly and makes it important to plan routes for supply vessels so that they are used as efficiently as possible.

We consider a *Single Vehicle Pickup and Delivery Problem with Multiple Commodities* and extends this further to include *Visit Windows* and *Route Duration* constraints. The objective is to find a least cost route for a single vessel starting and ending at the supply base, visiting all platforms that have delivery and pickup demands for multiple commodities, while at the same time the vessel capacity must not be exceeded at any time for any of the commodities. In addition, for the second problem, some platforms can only be serviced during daytime and the tour duration must not exceed its maximum limit. This thesis describes mathematical models and heuristics capable of solving these problems. Computational results show that when the full capacity of the vessel is utilized, it is suboptimal to impose a Hamiltonian shape on the solution and is preferable to perform the delivery and pickup operations separately. Moreover, visiting customers twice may be dictated by feasibility considerations.

**Acknowledgements**

We would like to thank our supervisor professor Irina Gribkovskaia for valuable help and guidance during the work with this thesis. We would also like to thank Ahmed Zouari and Arne Borch Akselvoll for helpful tips.

Molde, May 26, 2008.
Arne Gjengstø and Sigbjørn Vaksvik

# Contents

# List of Figures



# List of Tables

# 1 Introduction

In this thesis we consider routing problems arising in the supply of offshore drilling-platforms off the Norwegian coast using supply vessels. Offshore platforms are supplied using a fleet of vessels originating from one offshore base which serves the installations in a region. The installations need to be supplied several different kinds of commodities, i.e. various fluids like water, drilling-liquids and fuel stored in tanks below deck. Commodities stored in tanks can also be dry, like cement and bentonite. Additional goods is stored in containers etc. up on deck. For an installation the consequences of not getting supplies and not getting rid of waste, empty containers and rented equipment may lead to delays and even full stop of operations. A stop of operations at a platform is not acceptable. Therefore it is important to plan routes for supply vessels so that they are used as efficiently as possible.

Planning routes where various commodities are involved can be a tedious process. Each of the platforms will have pickup and delivery demand. The vessel leaves the offshore base, the depot, with commodities to deliver at the platforms. Platforms do also have commodities that should be picked up by the vessel and transported back to the depot. This will have to be done without exceeding the capacity for each commodity. Some platforms are also closed at night so routes should be planned with this in mind. A vessel that arrives too early will have to wait until the platform is open for service. If a vessel arrives too late to complete the service before the platform closes for the night, the vessel will have to wait until the next morning. This is not a good option, though, so the risk of this happening should also be considered when planning a route. The tour must be completed within a specified time limit, which also increases the difficulty of finding a good route for the vessel. These extensions to the Vehicle Routing Problem makes the use of a second visit to a platform come in handy. This way, if there is not enough capacity to do both deliveries and pickups at the same time, the deliveries can be made at the first visit, increasing the available capacity. At a later time, this platform is visited again to pick up the commodities that could not be picked up at the first visit.

Two problems are considered in the thesis. The first is an expansion of the Capacitated Vehicle Routing Problem with the addition of multiple commodities, mixed pickup and delivery and limited to only one vehicle, more specifically, a supply vessel. In other words our core problem for this thesis is the One-to-many-to-one Multi Com-

modity Single Vehicle Pickup and Delivery Problem. One-to-many-to-one means that the vessel leaves an offshore base, visits all the platforms and performs services and then returns to the offshore base from where it started.

In the second problem Visit Windows and Route Duration are added. For both problems, two visits are allowed.

In this thesis we formulate mathematical models for these problems and a metaheuristic capable of generating solutions for large instances. The metaheuristic is a Tabu Search algorithm. Analyses are made to examine the robustness and accuracy of the Tabu Search algorithm. We also examine the effects on route shapes by increasing the utilization of the cargo capacity.

The rest of the thesis is built the following way: In Chapter 2 the theory background is discussed. Chapter 3 contains the problem description. In Chapter 4 the formulations for the mathematical models are presented. Tabu Search is described in Chapter 5. In Chapter 6 the computational experiments are analyzed. Chapter 7 gives conclusions reached and further suggestions for research.

# 2 Theory Background

This chapter begins with a description of the Vehicle Routing Problem (VRP) followed by extensions to VRP. Then solution methods are presented. Finally the relevant literature review is discussed.

## 2.1 Vehicle Routing Problem

Toth and Vigo (2002) described the Vehicle Routing Problem as follows. All customers correspond to deliveries and the demands are deterministic, known in advance, and may not be split. The vehicles are identical and based at a single central depot, and only the capacity restrictions for the vehicles are imposed. The objective is to minimize the total cost (i.e., a weighted function of the number of routes and their length or travel time) to serve all the customers.

## 2.2 Extensions to VRP

Below are short descriptions of some extensions to VRP:

- Various sized vehicles

  When dealing with a problem where some customers are close to the depot, a vehicle with less capacity than other vehicles can be used. The "normal-sized" vehicles can then visit the customers further away. This would give better utilization of vehicles, than if a large vehicle is used to deliver goods that would only require half the capacity of the vehicle.

- Multiple depots

  The customer can be served from several depots. Each depot has a set of available vehicles and each vehicle starts and finishes at the same depot. The location and demand of each customer is known in advance.

- Fleet size

  This problem consists of determining how many vehicles are needed to serve all customers at a lowest possible cost. The vehicles have predetermined capacity and the number of available vehicles may be set in advance.

- Time windows

  In general a time window denotes a time-interval in which the servicing of a customer must be performed. Variations in the implementation of time windows might allow starting before or after a time window, but then with an additional cost.

  - Single time window

    The customers have one specific time interval for when they can be serviced.

  - Multiple time windows

    Customers may have several time intervals for when they can be serviced. A vehicle arriving between two time windows must wait until the beginning of the next time window. This extension has primarily been examined for the multi-period VRP. Each customer must be visited a specified number of times within the planning horizon.

  - Soft time window

    When the problem includes soft time windows, the vehicle is allowed to start service a customer before or after the time window, but the vehicle incurs additional costs by doing so.

  - Hard time window

    When the time window is "hard", the vehicle must start and stop the servicing within the start and the end of the time window.

- Maximum tour duration

  A vehicle must complete its tour before a specified time limit.

- Multiple commodities

  The customer needs different types of commodities to be delivered and/or picked up. The commodities can share a common compartment or there are separate compartments for each commodity on the vehicle.

- Pickup

  A customer needs to dispose of some commodities in addition to receiving deliveries. At least two types of problems use this extension:

- VRP with Backhauls, where the deliveries for ALL customers must be completed, then the vehicle picks up commodities on its way back to the depot;

- the Pickup and Delivery problem, where the pickup can be performed in any order, as long as the delivery is done earlier or simultaneously as the pickup.

## 2.3 Solution Methods

Many methods have been used to solve vehicle routing problems. These include exact solution methods, approximation methods and heuristics. In the later years the focus has been on metaheuristics.

The exact methods will explore every possible solution for an instance of a problem. This may take a long time, but eventually the optimal solution will be found, if there is any. For large instances this method is not suitable, as it takes too long to explore all solutions.

Approximation methods find solutions within a a lower and upper bound that must be known in advance. Assuming a minimization problem, an upper bound can be found by applying a heuristic that gives a feasible solution. The lower bound can be found by relaxing the problem. Relaxation can be achieved by increasing the set of feasible solutions. Approximation methods need proof that the algorithm returns a value within the bound.

Heuristics are applied when solving time consuming problems, like large TSP or VRP instances. For small problem instances an exact solution method would be sufficient. Heuristics are categorized as either classical or modern. The classical heuristics stop when a local optimum is reached, whereas the modern heuristics can escape local optima by using various techniques. A local optimum is when you from one solution is not able to reach a solution with a better cost. Local Search behaves this way; It is an iterative search procedure that starts from an initial feasible solution and then improves the solution slightly every iteration by performing moves. When no more improvements can be done, we say that the search has reached a local optimum. One technique to overcome this might be to start the search procedure from different initial solutions when a local optimum is reached, hoping that the search will converge to different local optima. Then the best is chosen among the different solutions. Another approach might be to enter infeasible space in order to reach otherwise unreachable solutions. Penal-

ties are then incurred to the infeasible solutions so the feasible solutions will be more favourable, Cordeau, Laporte and Mercier (2001). Other examples of classical heuristics are the Savings Heuristic, Clarke and Wright (1964), and the Sweep Algorithm, Gillett and Miller (1974).

Metaheuristics are usually based on things that happen in real life, such as Simulated Annealing (thermodynamic processes), Kirkpatrick, Gelatt and Vecchi (1983), and Ant Colony Optimization (i.e. paths are marked as bad if the ants going a specific path do not return), Dorigo (1992). Tabu Search tries to simulate intelligent processes by implementing memory structures. Tabu Search is a common algorithm for solving VRP. Tabu Search was introduced by Fred Glover in 1986, as described in Gendreau (2003), and was simply a guided Local Search heuristic with short term memory in the form of tabu lists used to find neighboring solutions and decide what is the next move. Tabu Search is an improvement heuristic and requires an initial solution for which to apply the local search.

One thing that differs between heuristics and exact solution methods is that a heuristic has no guarantee of finding any solutions. In practice, however, a well implemented heuristic is capable of finding optimal and near-optimal solutions in a relatively short time.

Figure 1 shows a local optimum. The graph represents the solutions in the search space and the arrow indicates the local optimum. Assuming a minimization problem, the good solutions are close to the bottom of this graph. A classical heuristic will not find the global optimum, which can be found at the right end of the figure, close to the bottom.

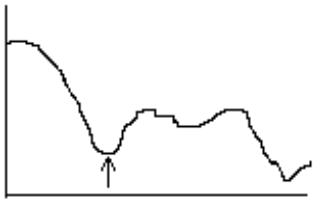

Figure 1: A local optimum in the solution space

## 2.4 Literature Review

Classical Vehicle Routing Problem, VRP, is a well known and commonly studied combinatorial problem that has many variations. VRP in itself is an NP-Hard problem, but with a small expansion to include i.e. Time Windows it becomes an NP-complete problem.

During the last few decades there have been written many articles for solving general VRP problems and variations thereof like VRPTW. One example is Desaulniers, Desrosiers and Solomon (2005). Their solution only allows one visit to every customer, only considers one commodity and uses a fleet of vehicles. They explore different strategies and variations including multiple time windows and soft time windows.

One article considering multiple-commodities is Choo, Poh and Wong (2005). The problem described involved multiple sources, discrete time periods, a fleet of vehicles and it aims to minimize the total number of discrete periods needed to complete an entire operation.

Cordeau et al. (2000) discuss a multi-commodity network flow formulation with time and capacity constraints for the VRPTW: How to derive upper and lower bounds using different approaches and describe how to adapt those methods to other variants of the problem like the Multiple TSP with Time Windows and VRP with Backhauls and Time Windows. For the latter problem it is assumed that all deliveries must be made before pickups, and they state that; 'more complex algorithms are however necessary when the pickup and delivery requests can be performed in any order'.

Lu, Dessouky (2006) suggest an insertion-based construction heuristic that evaluates the insertion cost based on the changes in the time window slack. This model is limited by a constraint stating that the customers pickup request must be served before its delivery request.

A metaheuristic for the time windowed vehicle routing problem is suggested in Chiang, Russel (2004), but the problem described contains multiple vehicles and is limited to one visit per customer.

Montané, Galvão (2006) presents an algorithm to solve the one-to-many-to-one VRP_SPD. Chen, Wu (2005) presents a similar algorithm.

A Vehicle Routing Problem with multiple interdependent time windows is explored in Doerner et al. (2008) where exact and heuristic algorithms for their problem are pre-

sented. Their problem concerns pickup of goods, in this case blood, that has a time-limit for when it needs to be delivered at the depot. Their mixed-integer objective function is minimizing the total travel distance between all successive stops on all tours, in addition they have suggested three heuristic solutions and a branch and bound approach to solve the problem.

For research a bit closer to our problem we have Landrieu, Mati and Binder (2001). The problem in this article has pickup locations that are visited first, then that load is delivered to a delivery location. Each location is only visited once. Each location has a time window for which the service must start. The service may end after the end of the time window if the service time is too long. They use two algorithms: Simple Tabu Search and Probabilistic Tabu Search and concluded that the Simple Tabu Search was fastest and most accurate. They state that the single vehicle pickup and delivery problem with time windows is one of the most difficult routing problems to solve, and for larger problems they had difficulties ensuring the existence of feasible solutions in less than one hour.

Mitrović-Minić (1998) did a survey about the Pickup and Delivery Problem with Time Windows and various methods to solve them. Multiple vehicles and multiple commodities were considered. Examples of problems solved by this problem are dial-a-ride problems (DARP), handicapped person transportation problems (HTP), courier company pickup and delivery problems (CCPDP). The First two problems deal with transporting people, while the third problem deals with transporting messages and parcels.

Tam, Tseng (2003) propose a micro-genetic algorithm for solving PDP-TW. They consider the number of vehicles used, the total traveling cost, the total schedule duration, and the drivers' total waiting time for the solutions.

Cordeau, Laporte and Mercier (2001) presented a unified tabu search heuristic for VRPTW and some of its generalizations, Periodic VRPTW, multi-depot VRPTW and later to the site-dependent VRPTW, Cordeau and Laporte (2001).

In a later article Cordeau, Laporte, Mercier (2004), modify and enhance their previously mentioned unified tabu search algorithm so that the beginning of service at a vertex can be postponed without violating time windows. This leads to an increase in computation time, but yields improved solutions.

Kammarti et al. (2004) presented a hybrid evolutionary approach to solve the single-vehicle Pickup and Delivery Problem with Time Windows. They allowed only one visit

to each customer and considered only one commodity.

Gribkovskaia et al. (2007) considered Single Vehicle Pickup and Delivery Routing Problem and Non-Simultaneous Services where 2 visits are allowed, formulated a mathematical model and implemented a tabu search algorithm based on Cordeau, Laporte and Mercier (2001). In Hoff et al. (2007) the multi-vehicle variant of a similar problem is considered and mathematical formulation with tabu search algorithm is provided. The tabu search algorithm was able to produce lasso-shaped solutions as well as general solutions. General solutions were reached by first duplicating each customer then generating a Hamiltonian solution on the extended set of customers. They found that general solutions outperform other solutions in term of cost, but the computation can be time consuming.

Zouari and Akselvoll (2007) is the only work we know that investigated on the same problem as we do, the One-to-Many-to-One Single Vehicle Pickup and Delivery Problem with Visit Windows. They formulated a mathematical model for solving the problem for small instances. In this model they used a subcycle elimination constraint that requires the solution to be manually controlled for subtours. If a subtour is found the data for the instance has to be manually altered to prevent that subtour. The process will have to be repeated until no more subtours are found.

# 3 Problem Description

In this chapter the problems considered in the thesis are described. Then there are discussions about splitting pickup between two visits and the benefits of two visits. Finally the purpose of the work is presented.

## 3.1 Problems Definitions

The first problem considered in the thesis is a variant of the one-to-many-to-one Single Vehicle Pickup and Delivery Problem. The offshore base supply commodities that the platforms require. At the same time the platforms have several different commodities that need to be transported back to the offshore base. The goal of this problem is to find the least cost route to pickup and deliver all the commodities without exceeding the vessel capacity limits for each commodity. It is also assumed that the vessel starts and ends the tour in the same offshore base. The vessel is allowed to visit each platform one or two times. If the platform is visited once, pickups and deliveries are performed simultaneously. If there are two visits, then in the first visit all the deliveries are made to free some space on the vessel and may do some of pickups, while on the second visit collections are performed.

In the second problem Visit Windows and Route Duration constraints are added. Visit Windows (VW) is a variant of Time Windows (TW). These visit windows are regular time intervals appearing at the same time every day of the week. This results in more opportunities to service the platform since the trip of a vessel is likely to be stretched over several days. VW are also stricter when it comes to the servicing. There are variations to this giving the opportunity to service outside the visit window at an additional cost, but in this thesis this is not allowed. The duration of the route must not exceed the maximum allowed duration.

## 3.2 Benefits of Two Visits

There are two benefits of allowing two visits at customers. The first is that a Hamiltonian solution may not be feasible. The second is that solutions with two visits may result in a lower cost.

**Increased Feasibility**

When a platform is night closed, it can happen that the visit window is not large enough to complete both delivery and pickup demands.

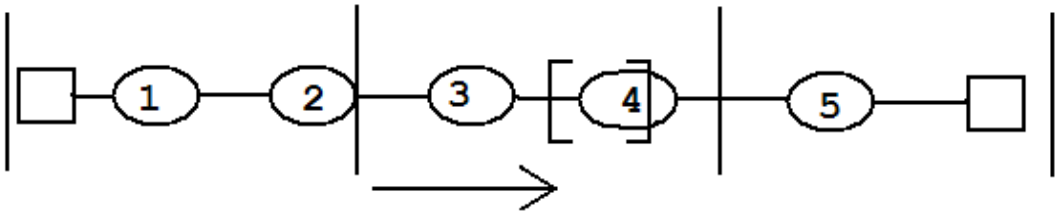


Figure 2: Vertex 4 violates visit window

Figure 2 shows a situation where the visit at platform 4 results in a violation of the visit window. The large vertical lines denotes the start and end of days. In total there are 3 days. The length of the circles represent the time spent for servicing the platform. The square brackets denote the start and end of the visit window for platform 4. The square boxes in the beginning and the end denote the offshore base and the circles denote the platforms. The length of the circles show the service time for the services performed. The length of the lines between the circles represents the travel time between platforms. When a platform has a visit window associated with it, the platform is open for service between 07:00 and 19:00. In Figure 2 the vessel arrives too late to complete the delivery and pickup services within the visit window.

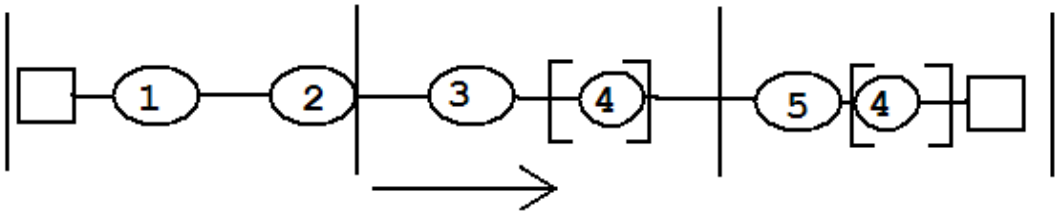


Figure 3: Vertex 4 is within visit window

By visiting the platform a second time, as shown in Figure 3, the solution becomes feasible, at the expense of a higher travel cost. The delivery service is performed at the first visit, while the pickup service is performed at the second visit.

**Cost Efficiency**

In this example four installations and two commodities are considered. Table 1 shows the capacity $q_h$ for each commodity $h$ together with the delivery and pickup demand for each installation $i$.

| | h=1 | | h=2 | |
|---|---|---|---|---|
| i | $p_{ih}$ | $d_{ih}$ | $p_{ih}$ | $d_{ih}$ |
| 1 | 2 | 9 | 5 | 3 |
| 2 | 9 | 4 | 6 | 4 |
| 3 | 7 | 2 | 1 | 10 |
| 4 | 1 | 1 | 2 | 5 |
| $q_h$ | 19 | | 22 | |

Table 1: Vessel capacity and delivery and pickup demand.

| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | | 626 | 686 | 589 | 591 |
| 1 | | | 61 | 118 | 69 |
| 2 | | | | 156 | 116 |
| 3 | | | | | 52 |
| 4 | | | | | |

Table 2: Travel time between installations and the depot.

Travel times in minutes between the different installations are shown in Table 2. The best Hamiltonian solution for this example is 0 - 4 - 1 - 3 - 2 - 0, which has the cost of 1620 minutes. If two visits are allowed, the route 0 - 3 - 2 - 1 - 4 - 3 - 0 will give a cost of 1516 minutes. This is a reduction in cost of 104 minutes. This new and better solution does not have a hamiltonian shape.

## 3.3 Split of Pickup

During the work on this thesis we have considered a few issues in how simplified the model and heuristic should be compared to real life. One of these issues are whether or not to split the pickup of different kinds of commodities over the two visits, that is to pickup commodity 1 on the first visit and commodity two on the second visit, or pickup both commodities on the second visit if you only have sufficient capacity to pick up one of the commodities on the first visit.

Using the travel cost matrix in Table 3 and demand as described below the table this question is explored:

| Travel cost | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | . | 626 | 686 | 589 | 591 |
| 1 | 626 | . | 61 | 118 | 69 |
| 2 | 686 | 61 | . | 156 | 116 |
| 3 | 589 | 118 | 156 | . | 52 |
| 4 | 591 | 69 | 116 | 52 | . |

Table 3: Travel cost between platforms

Vessel has maximum capacity of 82 units for commodity 1 and 2.

Platforms P1-P4 has the following pickup and delivery demand for commodity C1 and C2 respectively:

P1 has 15 and 27 delivery and 21 and 24 pickup demand.

P2 has 29 and 14 delivery and 13 and 27 pickup demand.

P3 has 25 and 18 delivery and 26 and 18 pickup demand.

P4 has 13 and 23 delivery and 22 and 13 pickup demand.

The vessel starts with 82 units of each commodity, having used all the vessel capacity.

| Cust. | Commo. | D | P | Sum |
|---|---|---|---|---|
| 0 | 1 | 82 | 0 | 82 |
| 0 | 2 | 82 | 0 | 82 |
| 4 | 1 | 69 | 0 | 69 |
| 4 | 2 | 59 | 0 | 59 |
| 2 | 1 | 40 | 13 | 53 |
| 2 | 2 | 45 | 27 | 72 |
| 1 | 1 | 25 | 34 | 59 |
| 1 | 2 | 18 | 51 | 69 |
| 4 | 1 | 25 | 56 | 81 |
| 4 | 2 | 18 | 64 | 82 |
| 3 | 1 | 0 | 82 | 82 |
| 3 | 2 | 0 | 82 | 82 |

Table 4: Optimal route and vessel load.

According to CPLEX, route P4 - P2 - P1 - P4 - P3 is the optimal route for this setup. In Table 4 the optimal route and load of each commodity is shown. The first column shows the number of the platform. The second column shows the type of commodity. Column 'D' shows the commodities that are left to deliver. Column 'P' shows the commodities that are picked up along the route. Column 'Sum' shows the total load of the vessel for each commodity during the tour.

Below is shown what might happen if you do not have space to pick up all types of commodities, but still decide to pick up those commodities that the vessel *can* hold:

If at P4 you deliver 13 of C1 and 23 of C2 resulting in 69 of C1 and 59 of C2 on the vessel with available space for 13 of C1 and 23 of C2.

There is not enough space to pickup everything of C1, but there is space for 13 of C2 resulting in 69 of C1 and 72 of C2 on the vessel with available space for 13 of C1 and 10 of C2.

At P2, 29 of C1 and 14 of C2 is delivered resulting in 40 of C1 and 58 of C2 on the vessel with available capacity for 42 of C1 and 24 of C2.

We now see that we do not have enough space to pick up C2 and have to make a second visit to P2 as well giving us two platforms needing two visits as opposed to only one in the optimal solution.

As you can see, this decision resulted in us not being able to find the optimal route for this setup. Because of this we have drawn the conclusion that it is better to pick up all commodities during the same visit. So if there is not enough room for at least one of the commodities we will not make a pickup of any commodity on that visit.

In regards to travel distance the decision to pick up all commodities during the same visit will have no negative effect as it will not lead to any extra detours apart from those already necessary. This also leads to potential fuel-savings and increased speed from having a lighter load, but Gribkovskaia, Halskau and Aas (2005) argue that the fuel consumption is almost fixed from the fact that the supply vessels usually cruise at "economical speed". That we have a fixed fuel consumption is shown by the problems considered in this thesis consisting of complete non-directed symmetrical graphs.

In regards to time spent loading/unloading we might lose a few minutes from not being able to fully utilize the crane in loading and unloading at the same time, but since it is possible to pickup different kinds of commodities simultaneously, like bulk and fuel in different tubes and containers using the crane, the time lost from not being able to

make simultaneous pickup and delivery will be negligible. This is due to the fact that you still have to spend the time to load the commodity you couldn't load on the first visit.

Considering visit windows those few minutes could potentially be "critical" in not exceeding the limits of the visit window.

As a result of the above arguments, and to simplify the model, we have decided not to consider those potentially lost minutes from not being able to load and unload at the same time.

When running problem instances with CPLEX it does happen that the commodities are picked up at different visits, but as long as it does not result in more second visits it does not really matter.

## 3.4 Work Purpose

The goal of this thesis is to describe and implement a Tabu Search heuristic capable of solving the problems described in Chapter 3.1. Using the work of Zouari and Akselvoll (2007) we will formulate mathematical models for these two problems. We will use the results from these models and compare them to the solutions found by our Tabu Search heuristic to find out how well our heuristic performs. The performance and robustness will also be examined. We will also check if increased capacity for vessels affects the solutions. Note that only small instances will be tested with the mathematical model since solving large instances will be too time consuming. We have used the algorithm from Gribkovskaia et al. (2007) and the work done by Cordeau, Laporte and Mercier (2001) as guidelines when implementing the algorithm to solve the two problems.

# 4 Mathematical Models

In this chapter the mathematical models used to solve the problems are presented.

## 4.1 Single-Vehicle Pickup and Delivery Problem with Multiple Commodities

The problem can be formalized as follows: Let $G = (V, A)$ be a complete graph with vertex set $V = \{0, ..., n\}$, where vertex 0 represents the depot and $n$ is the maximum number of customers. The arc is defined as $A = \{(i, j) : i, j \in V, i \neq j\}$. Each arc $(i, j) \in A$ has a non-negative length or cost $c_{ij}$ (usually equal to the travel time). Let $H = \{1, ..., p\}$ be the set of commodities representing types of entities to be transported. Each vertex, excluding the depot, can have a non-negative delivery and or pickup demand of at least one commodity $h$. $d_{ih}$ is the amount of commodity $h$ to deliver to vertex $i$ and $p_{ih}$ is the amount of commodity $h$ to pickup from vertex $i$. Only one vessel is used for the entire tour. $Y_{ih}$ shows if both delivery and pickup service is performed simultaneously for commodity $h$ at vertex $i$. $Y_{ih}$ is equal to 1 if both services are performed simultaneously, 0 otherwise. The second visit at a vertex $i$ is represented by a copy $i+n$. We set $p_{i+n,h} = p_{ih}$. If the pickup and delivery are performed simultaneously, vertex $i + n$ is not visited. The binary variable $Z_i$ indicates number of visits for customer $i$. $Z_i$ is equal to 1 if customer $i$ is visited only once, 0 otherwise. $X_{ij}$ describes the visiting sequence of customers. $X_{ij}$ is equal to 1 if customer $j$ is visited directly after customer $i$, 0 otherwise. $q_h$ denotes the capacity of commodity $h$. The continuous variable $W_{ih}$ denotes the pickup load of the vessel for commodity $h$ after vertex $i$ has been visited. The continuous variable $V_{ih}$ denotes the delivery load of the vessel for commodity $h$ after vertex $i$ has been visited.

In the model presented, variables must be interpreted as zero whenever they are not defined.

### Mathematical Formulation

*Notation*

**Sets:**

$H$ - Set of commodities

**Parameters**

$d_{ih}$ - Delivery demand of commodity $h$ for facility $i, \forall i \in 1, ..., n, \forall h \in H$

$p_{ih}$ - Pickup demand of commodity $h$ for facility $i, \forall i \in 1, ..., n, \forall h \in H$

$q_h$ - Capacity available for each commodity $h$ on the vessel, $\forall h \in H$

$c_{ij}$ - Extended cost matrix showing travelling cost between facility $i$ and facility $j$, $\forall i \in 1, ..., n, \forall j \in 1, ..., n$

**Variables**

$V_{ih}$ - Delivery load of commodity $h$ after visiting facility $i, \forall i \in 0, ..., 2n, \forall h \in H$

$W_{ih}$ - Pickup load of commodity $h$ after visiting facility $i, \forall i \in 0, ..., 2n, \forall h \in H$

$X_{ij}$ - Indicates if facility $j$ is visited immediately after facility $i, \forall i \in 0, ..., 2n, \forall j \in 0, ..., 2n$

$Y_{ih}$ - $\begin{cases} 1, \text{if pickup and delivery for commodity } h \text{ is performed simultaneously at facility } i \\ 0, \text{otherwise} \end{cases}$
, $\forall i \in 1, ..., n, \forall h \in H$

$Z_i$ - Indicates number of visits to customer $i$. $Z_i = 1$ if 1 visit is required, 0 otherwise, $\forall i \in 1, ..., n$

**Mathematical model**

$$(1) \min \sum_{i=0}^{2n} \sum_{j=0}^{2n} c_{ij} X_{ij}$$

$$(2) \sum_{j=0}^{2n} X_{ij} = 1, \forall i = 0, ..., n$$

$$(3) \sum_{i=0}^{2n} X_{ij} = 1, \forall j = 0, ..., n$$

$$(4)\ V_{0h} = \sum_{i=1}^{n} d_{ih}, h \in H$$

$$(5)\ 0 \leq V_{ih} + W_{ih} \leq q_h, \forall i = 0, ..., 2n, h \in H$$

$$(6)\ V_{jh} \geq V_{ih} - d_{jh} - (1 - X_{ij}) q_h, \forall i = 0, ..., 2n, \forall j = 1, ..., 2n, h \in H$$

$$(7)\ W_{jh} \geq W_{ih} + p_{jh} Y_{jh} - (1 - X_{ij}) q_h, \forall i = 0, ..., 2n, \forall j = 1, ..., n, h \in H$$

$$(8)\ W_{jh} \geq W_{ih} + (1 - Y_{j-n,h}) p_{j-n,h} - (1 - X_{ij}) q_h, \forall i = 0, ..., 2n, \forall j = n + 1, ..., 2n, h \in H$$

$$(9)\ Z_i \leq \frac{\sum_{h \in H} Y_{ih}}{|H|}, \forall i = 1, ..., n$$

$$(10) \sum_{i=0}^{2n} X_{ij} = 1 - Z_{j-n}, \forall j = n + 1, ..., 2n$$

(11) $\sum_{j=0}^{2n} X_{ij} = 1 - Z_{i-n}, \forall i = n+1, ..., 2n$

(12) $X_{ij} \in \{0,1\}, i \neq j, \forall i, j = 0, ..., 2n; j \neq i+n \; if \; 1 \leq i \leq n; j \neq i-n \; if \; i > n$

(13) $Z_i \in \{0,1\}, \forall i = 1, ..., n$

(14) $Y_{ih} \in \{0,1\}, \forall i = 1, ..., n, h \in H$

(15) $V_{ih} \geq 0, \forall i = 0, ..., 2n, h \in H$

(16) $W_{ih} \geq 0, \forall i = 0, ..., 2n, h \in H$

**Description of constraints**

- The objective function (1) expresses the minimization of the total travel time.
- Constraints (2) and (3) make sure that the first node associated with each customer is visited once, either for combined pickups and deliveries or for delivery only.
- Constraints (4) define the total load of all commodities on the vehicle when it is leaving the base.
- Constraints (5) express that when the vehicle has left a customer the combined pickup- and delivery-load of each commodity should not exceed the capacity for that commodity.
- Constraints (6) makes sure all delivery demands are satisfied, (7) and (8) make sure all pickups are collected. Constraints (6), (7) and (8) also prevent the formation of subtours.
- Constraints (9), (10) and (11) make sure that if all commodities for a given customer are delivered and picked up simultaneously at first visit then a second visit is not required. A second visit is required if at least one commodity is not picked up in the first visit.
- Constraints (9) is a linearization of $Z_i = \begin{cases} 1, if \sum_{h \in H} Y_{ik} = |H|, \forall i = 1, ..., n \\ 0, otherwise \end{cases}$

| $Y_1$ | $Y_2$ | $Z$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Figure 4: Relation between variables

Figure 4 shows the relation between variables $Y$ and $Z$. When the pickup and delivery for both commodity 1 and 2 are performed simultaneously, only one visit is required. In other words, $Z = 1$ when both $Y_1 = 1$ and $Y_2 = 1$.

- (12),(13) and (14) are binary requirements for variables.
- (15) and (16) represents lower bounds on the pickup- and delivery-load of each commodity.

## 4.2 Single-Vehicle Pickup and Delivery Problem with Multiple Commodities and Visit Windows

Let $[e_i, l_i]$ be the visit window associated with each installation $i$, where $e_i$ and $l_i$ denote respectively the earliest time when service may begin and the latest time when service may finish at installation $i$. If the supply vessel arrives to the installation before $e_i$, it will have to wait until it is possible to provide the service for the installation. If the vessel arrives to the installation later than $l_i$, then it has to wait until the next visit window starts. In both cases the waiting time will increase the duration of the route, which is undesirable. let $T$ denote the maximum route duration in hours and $t_h$ the service time in hours related to picking up or delivering one unit of commodity $h$. $D$ denotes a fixed time of the day at which the vessel leaves the depot, i.e. 4 p.m.. Let $B_i$ be a variable for the departure time in hours associated to each installation $i$. When the supply vessel leaves the offshore base, we set $B_0 = 0$. Let $U_i^a$ be an integer variable that denotes the day when the vessel arrives to installation $i$. $U_i^a = 0$ if the vessel arrives at installation $i$ on the same day it leaves the depot. $U_i^a = 1$ if the vessel arrives at installation $i$ one day

after it leaves the offshore base and so on until $\left\lceil \frac{T-l_i}{24} \right\rceil$. Let $U_i^d$ be an integer variable that denotes the day when the vessel leaves installation $i$. $U_i^d = 0$ if the vessel leaves installation $i$ on the same day it leaves the depot. $U_i^d = 1$ if vehicle leaves installation $i$ one day after it leaves the offshore base and so on until $\left\lceil \frac{T-l_i}{24} \right\rceil$. The model presented is linear, which the model in Zouari and Akselvoll (2007) was not.

In the model presented, variables must be interpreted as zero whenever they are not defined. To make sure that customers are served within the visit windows, additional elements are added to the model:

### Mathematical Formulation

*Notation*

**Added Parameters**

$e_i$ - earliest time, in hours, when service may begin at installation $i$, $\forall i = 1, ..., n$

$l_i$ - latest time, in hours, when service may finish at installation $i$, $\forall i = 1, ..., n$

$T$ - the maximum route duration, in hours

$t_h$ - service time, in hours, for picking up or delivering commodity $h$, $h \in H$

$D$ - fixed time, in hours, for when the vessel leaves the depot

**Added Variables**

$B_i$ - departure time in hours for installation $i$, $\forall i = 0, ..., 2n$

$U_i^a$ - denotes the day when the vessel arrives to installation $i$, $\forall i = 1, ..., 2n$

$U_i^d$ - denotes the day when the vessel departure from installation $i$, $\forall i = 1, ..., 2n$

**Added Constraints**

(17) $B_j \geq B_i + \frac{c_{ij}}{60} X_{ij} + \sum_{h \in H} d_{jh} \frac{t_h}{60} + \sum_{h \in H} p_{jh} \frac{t_h}{60} Y_{jh} - (1 - X_{ij})T, \forall i = 0, ..., 2n, \forall j = 1, ..., n$

(18) $B_j \geq B_i + \frac{c_{ij}}{60} X_{ij} + \sum_{h \in H} p_{j-n,h} \frac{t_h}{60} (1 - Y_{j-n,h}) - (1 - X_{ij})T, \forall i = 0, ..., 2n, \forall j = n + 1, ..., 2n$

(19) $e_i + 24U_i^a \leq D + (B_i - \sum_{h \in H} d_{ih} \frac{t_h}{60} - \sum_{h \in H} p_{ih} \frac{t_h}{60} Y_{ih}) \leq l_i + 24U_i^a, \forall i = 1, ..., n$

(20) $e_{i-n} + 24U_i^a \leq D + (B_i - \sum_{h \in H} p_{i-n,h} \frac{t_h}{60} (1 - Y_{i-n,h})) \leq l_{i-n} + 24U_i^a, \forall i = n + 1, ..., 2n$

(21) $e_i + 24U_i^d \leq D + B_i \leq l_i + 24U_i^d, \forall i = 1, ..., n$

(22) $e_{i-n} + 24U_i^d \leq D + B_i \leq l_{i-n} + 24U_i^d, \forall i = n + 1, ..., 2n$

$(23)\ B_i + \frac{c_{i0}}{60} X_{i0} \leq T, \forall i = 1, ..., 2n$

$(24)\ B_i \geq 0, \forall i = 0, ..., 2n$

$(25)\ U_i^a \geq 0 \text{ integer}, \forall i = 1, ..., 2n$

$(26)\ U_i^d \geq 0 \text{ integer}, \forall i = 1, ..., 2n$

**Description of constraints**

- Constraints (17) control the departure times from the current facility when only one visit is required.
- Constraints (18) control the departure time from the current facility when a second visit is required.
- Constraints (19) express that for the first visit to a facility, the arrival time must be within the visit window of the arrival day.
- Constraints (20) express that if a second visit to a facility is necessary, the arrival time must be within the visit window of the arrival day.
- Constraints (21) express that for the first visit to a facility, the departure time must be within the visit window of the departure day.
- Constraints (22) express that if a second visit to a facility is necessary, the departure time must be within the visit window of the departure day.
- Constraints (23) make sure the total route duration do not exceed the maximum route duration.
- Constraints (24) express non-negativity requirements for variables.
- Constraints (25) and (26) express non-negativity and integer requirements for variables.

# 5 Tabu Search

In this chapter a general description of tabu search is given, then the tabu search used to solve the problems is thoroughly described.

## 5.1 Tabu Search Algorithm

As mentioned earlier, Tabu Search was introduced by Fred Glover in 1986 and is a metaheuristic using a Local Search heuristic to find neighbouring solutions. Tabu Search is an improvement heuristic and requires an initial solution for which to apply the local search. The basic principle of Tabu Search is to let the search accept moves that lead to a worse solution to overcome local optima.

The search space of a TS heuristic is the set of all solutions that can be visited during a search, i.e. all feasible solutions. A neighbourhood is a collection of all possible new solutions that can be reached from the current solution. These solutions are usually represented by the moves required to reach the solutions.

### 5.1.1 Tabu Tenure and Aspiration Criteria

Tabu Search walks the neighbourhood by doing moves. Such a move can be to interchange the origin and destination of an arc in a solution. This kind of move is called 2-opt. For a Knapsack Problem a move can be to either take one item out of the backpack or to put one item in. A memory structure, called tabu list, records the recently performed moves and contains info about when the move is no longer tabu. The tabu mechanism is used to prevent cycling when moving away from local optima when non-improving moves are used. An example: the current solution $S$ has reached a local optimum. The best non-improving move $m^-$ leads to a worse solution $S^-$. Then the available moves are again evaluated and the best move $m^+$leads back to solution $S$. By declaring move $m^+$ tabu, another move would have to be performed, thus leading farther away from solution $S$. Moves remain tabu for a number of iterations, called tabu tenure. Tabu tenure can be of fixed or dynamic length during the search. When using dynamic length of a tabu tenure the entries in the list usually contain the iteration for when the move is no longer tabu.

Sometimes the tabu status can lead to not identifying good moves. There might be a

move that leads to a better solution, but the move is tabu. Aspiration criteria can then be applied to the search. The most common criterion for Tabu Search is to allow a move if it is tabu, when the move results in a new best solution so far.

### 5.1.2 A Simple Tabu Search Procedure

A simple procedure for tabu search is presented. Suppose that we are trying to minimize a function $f(S)$ and the best available move at each iteration is chosen from a list of possible moves.

***Notation***

- $S$, the current solution,
- $S^*$, the best known solution,
- $f^*$, value of $S^*$,
- $N(S)$, the neighbourhood of $S$,
- $\tilde{N}(S)$, the "admissible" subset of $N(S)$, the moves that meet the aspiration criteria or are non-tabu.

**Initialization**

Choose an initial solution $S_0$.

Set $S := S_0$, $f^* := f(S_0)$, $S^* := S_0$, $T := \varnothing$.

**Search**

While *termination criteria not satisfied* do

- Select $S$ in $argmin[f(S')]$; $S' \in \tilde{N}(S)$
- If $f(S) < f^*$, then set $f^* := f(S)$, $S^* := S$;
- Record tabu for the current move in $T$;

End while

### 5.1.3 Diversification and Intensification

After the search has been running a while we find attributes that are often used in the solutions. We can focus on these to make sure the best solutions in some areas of the search space are found. This phase is performed from time to time. The intensification phase is based on *intermediate-term memory*, such as a *recency memory*. This memory contains the number of consecutive iterations that various "solution components" have been present in the current solution without interruption. This mechanism is not always needed. Some search procedures do have a thorough search process, so adding the diversification process will be redundant and lead to more CPU time.

Diversification on the other hand, forces the search into previously unexplored areas of the search space. A search without diversification tends to explore most local attributes and may miss many interesting solutions that includes areas at the outer points of the search space. Diversification is usually based on some form of *long-term-memory* of the search, such as a *frequency memory*, where the total number of iterations various "solution components" have been present in the current solution, or have been involved in the selected moves. A proper search diversification is possibly the most critical issue in the design of TS heuristics. There are two major diversification techniques: *Restart diversification* and *continuous diversification*.

Restart diversification involves forcing a few rarely used components in the current solution and restarting the search from this point. Continuous diversification integrates diversification considerations directly into the regular searching process. This is achieved by *biasing* the evaluation of possible moves by adding a small term related to component frequencies to the objective.

A third diversification technique is to allow infeasible solutions. Many problem constraints in the definition of a search space often restricts the searching process too much and can lead to mediocre solutions. Constraint relaxation is an attractive strategy. This creates a larger search space that can be explored with "simpler" neighbourhood structures. Constraint relaxation can be implemented by dropping selected constraints from the search space definition and adding weighted penalties for constraint violations to the objective. Weights can be adjusted dynamically on the basis of the recent history of the search. If infeasible solutions were encountered the last iterations the weights are increased and if the solutions were feasible, the weights are decreased. A technique

called *strategic oscillation* consists of systematically modifying penalty weights to drive the search to cross the feasibility boundary of the search space to induce diversification.

### 5.1.4 Termination Criteria

In theory a search can run forever, but it has to be stopped at a point. The most common criteria for this purpose are:

- Run for a fixed number of iterations or a fixed amount of CPU-time
- After a given number of iterations without improvement in the objective function value.
- When the objective reaches a pre-specified threshold value.

### 5.1.5 Surrogate and Auxiliary Objectives

The true objective function can be quite costly to evaluate for many problems. An effective approach to handle this, is to evaluate neighbours using a *surrogate objective*, i.e. a function that is correlated to the true objective, but is less computationally demanding, in order to identify a (small) set of promising candidates. The true objective is then computed for this small set of candidate moves and the best one is selected.

The objective function may not provide enough information to effectively drive the search to more interesting areas of the search space. A situation might occur that all elements in the neighbourhood score equally with respect to the primary objective. In such a case, it is necessary to define an auxiliary objective function to orient the search. This function must in some way measure the desirable attributes of solutions. An effective auxiliary objective is not always easy to implement and may require a lengthy trial and error process.

## 5.2 Tabu Search Heuristic for the Single Vehicle Pickup and Delivery Problem with Multiple Commodities and Visit Windows

The algorithm is based on the works of Gribkovskaia et al. (2007), Cordeau, Laporte and Mercier (2001) and Cordeau, Laporte, Mercier (2004). The algorithm is allowed

to go into infeasible space to search for solutions and has the extension of multiple commodities, visit windows and duration limit for the tour.

### 5.2.1 Initial Solution

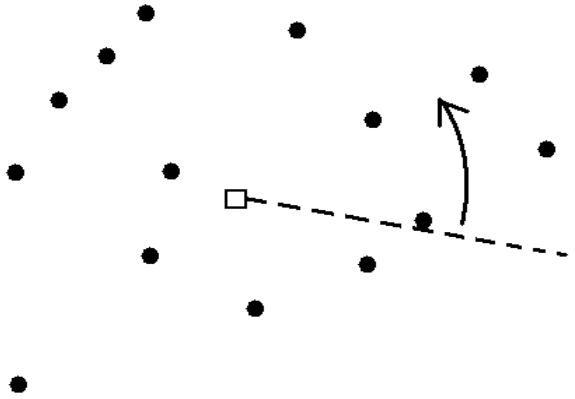

Figure 5: Sweep algorithm

Gillett and Miller (1974) proposed a construction heuristic called the Sweep algorithm that sorts vertices according to the angle with the depot, assuming euclidean distances for the instances, and then inserts the nodes into the route in that order. Figure 5 shows the depot in the middle as a square. An "arm" is sweeping the plane of vertices counter-clockwise (clockwise is also possible) and place them in that order in a list. We use such a heuristic to create the initial solution. The depot is inserted first, then the first vertex to visit is chosen randomly and from there on the other vertices are inserted in the same order as they appear in the sorted list of vertices. There are no requirements that the initial solution has to be feasible.

### 5.2.2 Main Features of the Tabu Search Heuristic

Below are descriptions of the main features implemented in our tabu search.

#### Load feasibility violations

Load feasibility is checked each time a vertex is visited. The total load infeasibility

$q(s)$ of a route is equal to the sum of load infeasibilities of all its vertices. This check simply consists of comparing the current load to the available capacity for the vessel, for each commodity.

**Duration feasibility violation**

Each instance has an associated maximum allowed time period $T$ associated with the traveling and servicing, given in hours. The total duration of the services and travel time $t$ is not allowed to exceed this limit. The violation of $T$ is given by $d(s) = \max(0, t-T)$.

**Visit window feasibility violation**

Platforms can be night closed. The visit window for vertex $i$ is given by $[e_i, l_i]$, where $e_i$ is the start of the visit window and $l_i$ is the end of the visit window. Vessels may arrive before the start of a visit window resulting in a waiting time $W_i = e_i - A_i$ for the vessel, where $A_i$ is the arrival time at vertex $i$. The total violation of visit windows $w(s)$ in a solution is equal to $\sum_{i=1}^{n}(\max(0, a_i - l_i))$. For vertices $i$ with visit windows, the allowed service period is given by the interval $[7, 19]$, all other vertices are open for service the entire day, shown by the interval $[0, 24]$. Cordeau, Laporte, Mercier (2004) uses a concept called forward time slack which we have included to postpone the beginning of service at a given node without causing any visit window violations. Given an ordered route $k = (v_0, ..., v_i, ..., v_q)$ where both $v_0$ and $v_q$ represents the depot, let $B_i$ denote the beginning of service at vertex $v_i$. Setting $B_0 = e_0$ and $B_i = \max\{A_i, e_i\}$ for $i = 1, ..., q$ is optimal in terms of minimizing visit window violations. This is because the vehicle leaves the depot as early as possible and the service at each vertex also begins as early as possible. Because of route duration constraints, an infeasible solution that has the properties just described, can be feasible if the departure from the depot is voluntarily delayed. The forward time slack is the largest increase in the beginning of service at vertex $v_i$ that will not cause any increase in visit window violation, thereby the $+$ symbol in the following formula. Forward time slack $F_i$ of vertex $v_i$ is defined by $F_i = \min_{i \leqslant j \leqslant q} \left\{ \sum_{i < p \leqslant j} W_p + (l_j - B_j)^+ \right\}$. Setting $B_0 = e_0 + F_0$ will yield a modified route of minimal total duration with equal violations of visit window constraints.

### Neighbourhood structure and attribute set

The neighbourhood $N(s)$ of solution $s$ is defined by all solutions that can be reached from $s$ by changing the number of visits for a vertex. When solution $s$ changes to the solution $s' \in N(s)$ a move is performed. The move can be expressed by the removal of attribute $(i, v)$ to $B(s')$, $(v \neq v')$. $\phi_i$ denotes the number of visits in a solution. The problems have two types of move. If a vertex has 1 visit only, a second visit is examined. If a vertex has 2 visits, removal of the second visit is examined. Some of these moves may not be valid. If a vertex that has the same predecessor as successor is removed, there will be two consecutive visits to the same vertex, which is not allowed. These are left out of the neighbourhood. Moves are performed as follows:

1. Insertion of the second visit at vertex $i$, $\phi_i = 1$: Suppose that in the solution $s$ vertex $i$ is visited once, that is $(i, 1) \in B(s)$. The second visit to vertex $i$ is inserted in the route minimizing the increase in the penalized function $f(s') = c(s') + \alpha q(s') + \beta d(s') + \gamma w(s')$. Hence, $(i, 2) \in B(s')$. The purpose of visiting vertex $i$ twice is to obtain a solution $s'$ with lower total load infeasibility, visit window infeasibility and route duration infeasibility than that of solution $s$. Visiting a vertex twice results in a higher total travel cost and may result in route duration violation. This is why the insertion of a second visit to vertex $i$ is based on the penalized function.

2. Deletion of the second visits at vertex $i$, $\phi_i = 2$: Suppose that in the solution $s$ vertex $i$ is visited twice, that is $(i, 2) \in B(s)$. A neighbour solution $s'$ is obtained by deleting the second visit to vertex $i$ from the route maximizing the decrease in $f(s')$, and reconnecting its predecessor and successor. The deletion of the second visit to $i$ implies that $(i, 1) \in B(s')$. In solution $s'$ the vertex $i$ is visited in the same order as its first visit in solution $s$: If costs satisfy the triangle inequality, the cost will not be increased if we delete a second visit. The deletion of a second visit may lead to increased load infeasibility and visit window infeasibility and a reduction in tour duration infeasibility.

### Tabu status of an attribute

Each move performed is defined by the attribute used. This attribute gets a tabu status

associated with it. If number of visits $v$ at customer $i$ is changed to $v'$, changing back to $v$ visits at vertex $i$ is not allowed for the next $\theta$ iterations. Parameter $\theta$ is a user controlled parameter and is changed every iteration.

### Aspiration criterion

During the search, an attribute may turn out to be very good and lead to a better solution than the best known solution so far, but the attribute is tabu. To overcome this, a rule is introduced. The rule is that the tabu status of an attribute $(i, v)$ can be revoked if this leads to a feasible solution with a smaller cost than the best known solution having that attribute. The move has to result in solution $s' \in N(s')$ with $q(s') = 0$, $d(s') = 0$, $w(s') = 0$ and $c(s') < \sigma_{iv}$, where $\sigma_{iv}$ is the aspiration level of attribute $(i, v)$. The initial set of $\sigma_{iv}$ is equal so $c(s)$ if $(i, v)$ belongs to the attribute set of the feasible solution and to $\infty$ otherwise. Every time a feasible solution $s$ is identified, the aspiration level of each of its attributes $(i, v)$ is updated to $\min\{\sigma_{iv}, c(s)\}$.

### Diversification

Without a diversification strategy, the evaluation of the best solution $s' \in M(s)$ would be based on the penalized function $f$. However, we wish to diversify the search, that is, give a higher chance of being selected to solutions $s' \in M(s)$ having an attribute $(i, v')$ that has not been frequently present in past solutions. The mechanism operates as follows: Any solution $s' \in M(s)$ such that $f(s') \geqslant f(s)$ is penalized with a term $p(s')$ proportional to the addition frequency $\rho_{iv}$ of the modified attribute, value of $c(s)$ and parameter $\zeta$. Let $\rho_{iv}$ denote the number of times attribute $(i, v)$ has been added to the solution and let $\lambda$ denote tabu iteration counter, where the parameter $\zeta$ is used to control the intensity of the diversification. Then the penalty term is $p(s') = \zeta c(s)\rho_{iv}/\lambda$. If $f(s') < f(s)$, we assume that $p(s') = 0$. Finally, the selection of the best solution $s' \in M(s)$ is based on a generalized function $g(s') = f(s') + p(s')$

### Termination criterion

A fixed number $\lambda$ of iterations as termination criteria. In this thesis the search terminates when $\lambda = 100.000$ iterations have been completed.

**Penalized objective function**

For a solution $s \in S$, let $c(s)$ denote the total routing cost, let $q(s)$ denote the total load feasibility violation of the route, let $d(s)$ denote the total duration violation of the route, and let $w(s)$ denote the total visit window violation of the route. Solutions $s \in S$ are evaluated with the help of the penalized cost function $f(s) = c(s) + \alpha q(s) + \beta d(s) + \gamma w(s)$, where $\alpha$, $\beta$ and $\gamma$ are positive parameters. The values of $\alpha$, $\beta$ and $\gamma$ are dynamically adjusted based on the recent history of the search. At each iteration the values of $\alpha$, $\beta$ and $\gamma$ are modified by a factor $(1 + \delta) > 1$, where $\delta$ is a positive parameter. If the current solution is load-feasible, the value of $\alpha$ is divided by $(1 + \delta)$; otherwise, it is multiplied by $(1 + \delta)$. The same rule applies for $\beta$ and $\gamma$.

### 5.2.3 Improvement Procedure

In the algorithm an intra-route procedure is applied each time a better solution is found, and regularly every $\varphi$ iterations. In this algorithm each vertex that has only 1 visit is evaluated. The steps are as follows:

Step1: Delete the current vertex if this does not change the number of visits to other vertices.

Step2: Reinsert this vertex in the route minimizing the penalized function f, that is without requiring feasibility of the resulting route. The vertex can be reinserted in the same position from which it was deleted.

Step3: Consider another vertex on which this procedure has not been applied yet, go to Step 1 and repeat until all vertices are checked.

### 5.2.4 Steps of the Tabu Search Algorithm

The initial solution $s_0$ is always based on the sweep algorithm. The initial solution is not guaranteed to be feasible. Below is a summary of the notations used for the tabu search algorithm:

**Notations used in the description of the tabu search algorithm**

| | |
|---|---|
| $(i, v)$ | Attribute: number of visits $v$ at customer $i$ |
| $B(s)$ | Attribute set of solution $s$ |
| $c(s)$ | Routing cost of solution $s$ |
| $q(s)$ | Total load violation of the vehicle during the route for given solution $s$ |
| $d(s)$ | Total duration violation of the vehicle during the route for given solution $s$ |
| $w(s)$ | Total visit window violation of the vehicle during the route for given solution $s$ |
| $N(s)$ | Neighbourhood of solution $s$ |
| $M(s)$ | A subset of $N(s)$ |
| $s, \widetilde{s}$ | Solutions |
| $s_0$ | Initial solution |
| $s^*$ | Best solution identified |
| $\alpha$ | Penalty factor for overload |
| $\beta$ | Penalty factor for duration violation |
| $\gamma$ | Penalty factor for visit window violation |
| $\delta$ | Parameter used to update $\alpha, \beta, \gamma$ |
| $\zeta$ | Factor used to adjust the intensity of the diversification |
| $\eta$ | Total number of iteration to be performed |
| $\theta$ | Tabu duration |
| $\lambda$ | Iteration counter |
| $\rho_{iv}$ | Number of times attribute $(i, v)$ has been added to the solution |
| $\sigma_{iv}$ | Aspiration level of attribute $(i, v)$ |
| $\tau_{iv}$ | Last iteration for which attribute $(i, v)$ is declared tabu |
| $\varphi$ | Parameter used for intra-route optimization |

The search process is defined by set of parameters $(\delta, \zeta, \theta, \eta, \varphi)$, and returns, after execution, the best feasible solution found $s^*$, if any. We assume that $c(s^*) = \infty$ if no feasible solution has been identified after execution of the algorithm.

1. Set $s := s_0, \alpha = 1$. If $s$ is feasible, set $s^* := s$.

2. For every $(i, v)$, do

   - Set $\rho_{iv} := 0$, $\tau_{iv} := 0$.

- If $(i, v) \in B(s)$ and $s$ is feasible, set $\sigma_{iv} := c(s)$; else, set $\sigma_{iv} := \infty$

3. For $\lambda$ = 1,...,$\eta$, do

   **a.** Update $\delta, \zeta, \theta$ :

   - $\delta :=$ random value from the open interval $(0, 1)$.
   - $\zeta :=$ random value from the open interval $(0, 1)$.
   - $\theta :=$ random value from the open interval $(0, round(10 \log 10n)$.

   **b.** Set $N(s) := \emptyset$.

   **c.** For each attribute $(i, v') \notin B(s)$ do

   - Create a solution $s'$ applying a move definition. Replace corresponding attribute $(i, v) \in B(s)$ by attribute $(i, v')$, i.e. $(i, v) \in B(s) \rightarrow (i, v') \in B(s')$.
   - Set $N(s) := N(s) \cup s'$.

   **b.** Set $M(s) := \emptyset$.

   **c.** For each $s' \in N(s)$ do

   - For $(i, v) \in B(s') \backslash B(s)$ such that $\tau_{iv} < \lambda$ or such that $c(s) + (f(s') - f(s)) < \sigma_{iv}$, set $M(s) := M(s) \cup \{s'\}$.

   **d.** For each $s' \in M(s)$, do

   - If $f(s') \geq f(s)$, set $g(s') := f(s') + \zeta c(s) \rho_{iv} / \lambda$; else, set $g(s') := f(s')$.

   **e.** Identify a solution $s' \in M(s)$ minimizing $g(s')$.

   **f.** For attribute $(i, v) \in B(s') \backslash B(s)$ do

   - Set $\rho_{ik} := \rho_{ik} + 1$ and $\tau_{iv} := \lambda + \theta$.

   **g.** If $s'$ is feasible ($q(s') = 0$ and $d(s') = 0$ and $w(s') = 0$), do

   - If $c(s') < c(s^*)$, set $s^* := s'$.
   - For each $(i, v) \in B(s')$, set $\sigma_{iv} := \min\{\sigma_{iv}, c(s')\}$.
   - Set $\alpha := \alpha / (1 + \delta)$; else, set $\alpha := \alpha(1 + \delta)$.
   - Set $\beta := \beta / (1 + \delta)$; else, set $\beta := \beta(1 + \delta)$.

- Set $\gamma := \gamma/(1+\delta)$; else, set $\gamma := \gamma(1+\delta)$.

**h.** Set $s := s'$.

**i.** If $q(s) = 0$ and $d(s) = 0$ and $w(s) = 0$ and $c(s) = c(s^*)$, or $\lambda = k\varphi, k = 1, 2, ...$, do Intra-route optimization:

- For each $(i, 1) \in B(s)$ do

  **(i)** Remove customer $i$ from the route in solution $s$ and reinsert customer $i$ in the route in solution $\widetilde{s}$ minimizing $f(\widetilde{s}) = c(\widetilde{s}) + \alpha q(\widetilde{s}) + \beta d(\widetilde{s}) + \gamma w(\widetilde{s})$ such that $B(s) = B(\widetilde{s})$.

  **(ii)** Set $s := \widetilde{s}$.

- If $q(s) = 0$ and $d(s) = 0$ and $w(s) = 0$ and $c(s) < c(s^*)$, set $s^* := s$.

**j.** Set $\lambda := \lambda + 1$.

*Note:* When running instances without visit windows, the violations for duration $d(s)$ and visit windows $w(s)$ are always treated as zero.

# 6 Computational Experiments

The mathematical models were modeled in AMPL. AMPL is a modelling language used to formulate mathematical models, which is then supplied to a solver, usually along with data for the instance. CPLEX is a solver which can solve problems formulated in AMPL. In this thesis CPLEX 9.0.0 was used.

The heuristic is written in C using Pelles C for Windows, version 5.00.6 beta 4. The program is available on the internet at the address

http://www.smorgasbordet.com/pellesc, last accessed 20.05.08. The tests for the heuristic were run on a computer with an Intel Core 2 Quad processor running at 3,65 GHz having 2 GB of memory, running Microsoft Windows XP. All instances solved by the heuristic were run for 100.000 iterations, except for the largest instances with visit windows; Some were allowed to run 100.000 iterations, but most of them were run for a maximum of 2 hours. The reason for this was that the thesis deadline was closing in rapidly. The tests performed with CPLEX were run on computers with Intel Pentium 4, 2.4 GHz CPU and 1 GB of RAM. These tests were run for a maximum of one hour.

## 6.1 Test Instances Generation

The capacity of the vehicles $q_h$for each commodity $h$ is equal to the maximum of the sum of delivery demand $d_{ih}$ for each vertex $i$ and the sum of pickup demand $p_{ih}$ for each vertex $i$ : $q_h = \max(\sum_{i=1}^{n} d_{ih}, \sum_{i=1}^{n} p_{ih})$. The service time for delivering and picking up each commodity is equal to 1 minute per unit. In other words the service time is equal to the pickup and delivery demand for each commodity.

For this thesis 40 small instances with number of nodes from 5 to 31 were generated. The first group of small instances are the same as those described in Gribkovskaia et al. (2007). Those instances were based on a CVRP problem and two parts of instances were made the following way. Let $q_i$ denote the demand of customer $i$ in the CVRP instance, let $d_i$ denote the delivery demand of customer $i$, and $p_i$ as the pickup demand of customer $i$ in our problem. In all instances with suffix 'b20', 7 in total with number of nodes from 16 to 31, the corresponding delivery and pickup demands are computed as follows:

$$d_i := q_i, \qquad p_i := \begin{cases} [(1-\beta)q_i] & \text{if } i \text{ is even} \\ [(1+\beta)q_i] & \text{if } i \text{ is odd} \end{cases}, i = 1, ..., n,$$

where $\beta$ is a real non-negative parameter smaller than 1. $\beta = 0.20$ is used to create these pickup demands. The second part of this group are 9 instances with number of nodes from 5 to 31. They were generated such that customers demands are geographically dependent. The instance names for this part have no suffix. Instances with the same initial name have the same $x_i$, $y_i$ coordinates and delivery demands $d_i$ as corresponding instances from the first part, but another pattern was used to calculate pickup demands $p_i$. Pickup demands of 30% of customers geographically closest to the depot were computed as 120% of their delivery demands. Pickup demands of 30% of customers furthest from the depot were computed as 80% of delivery demands. Pickup demands for the rest of the customers were computed as in the first group of instances.

The second group of small instances were downloaded from the internet at the URL http://www.fernuni-hagen.de/WINF/touren/probleme/. The instances found at that URL are large, so the 24 first instances were collected and reduced in size to a random size between 5 and 25. I.e. in an instance randomly selected to be of size 19 all but the 19 first vertices were removed to create the new smaller instances. The number of vertices is included first in the instance name, then follows the original instance name. The pickup demand for the first commodity in these instances were generated the same way as the instances with the suffix 'b20' that were used in Gribkovskaia et al. (2007).

The third group of instances are large instances derived from CVRP instances of VRPLIB containing between 7 and 484 vertices and between two and 25 vehicles. The instances chosen for this thesis have between 41 to 262 nodes, 13 instances in total. These instances can be found on the website http://www.or.deis.unibo.it/research_pages/ORinstances/VRPLIB/VRPLIB.html. Delivery and pickup demands were generated as follows. Let $q_i$ be the demand of customer $i$ in the original CVRP instance. Then we set $d_i := q_i$ and $p_i := \lfloor 0.8q_i \rfloor$ if $i$ is even or $p_i := \lfloor 1.2q_i \rfloor$ if $i$ is odd.

To generate the second type of commodity, $h = 2$, we let the delivery and pickup demand be the reverse of the delivery and pickup demand for commodity $h = 1$. For customers $i = 1, ..., n$ we let $d_{i2} = d_{n+1-i,1}$, and $p_{i2} = p_{n+1-i,1}$.

The delivery demand of the third commodity, $h = 3$, is generated by finding the average delivery demand of commodity 1 and 2 and multiply this average by a random number between $0.6$ and $1.6$: as represented by this formula: $d_{i3} = \left(\frac{d_{i1}+d_{i2}}{2}\right) \mathrm{rand}(0.6, 1.6)$. The pickup demand was generated in the similar way.

The heuristic and the mathematical model are general in the way that they accept instances without visit windows as well as instances with visit windows. For the instances with visit windows we identified and chose vertices that do not need more service time than the given visit window allows. To find a suitable allowed tour duration for the instances, we let the heuristic run on instances with no time windows, but modified the heuristic to calculate visit window violations and set the maximum allowed tour duration to a very large number of hours. Based on these calculations we chose the maximum allowed tour duration. The instances that have 1 visit window contain the suffix 'vw'. For instances with more than one visit window the suffix is 'vwX', where X is the number of platforms with visit windows. 'vw2' means there is 2 platforms with visit windows.

Instances with 2 commodities have the prefix '2c_' and the instances with 3 commodities have the prefix '3c_'. The number after this prefix is the number of vertices in the instance, including the depot. The rest of the filename is there to identify the instance.

There are 40 small instances run both on the mathematical model in CPLEX and using the Tabu-algorithm. All 40 small instances have been run with the cargo-space for the different commodities at full capacity, called 100% utilization. The instances have also been run with 5%, 10% and 20% extra capacity on the vessel for each commodity, further called 95%, 90% and 80% utilization. These instances are generated as follows. Let $q_h$ denote the capacity of commodity $h$ of the vessel, let $d_{ih}$ denote the delivery demand of commodity $h$ of platform $i$, let $p_{ih}$ denote the pickup demand of commodity $h$ of platform $i$. The capacity for commodity $h$ is then given by the formula $q_h = \alpha \max(\sum_{i=1}^{n} d_{ih}, \sum_{i=1}^{n} p_{ih})$ , where $\alpha$ denotes the utilization of capacity. For instances with 100% utilization, $\alpha = 1$. For instances with 95% utilization, $\alpha = 1.05$. For instances with 90% utilization, $\alpha = 1.1$. For instances with 80% utilization, $\alpha = 1.2$.

In addition, 14 instances of size 16 to 31 are selected to run with one commodity and 1 platform having a visit window. For the same 14 instances, 1 additional visit window is added and run with 1, 2 and 3 commodities with 100% utilization.

A modification of the mathematical model constrained to find only Hamiltonian solutions were run as well for all instances in group one and two.

The total number of small instances are then 1336 and the total number of large

instances is 208.

## 6.2 Computational Results

Below follow tables showing various data extracted from the output from our instances run with CPLEX and from the runs using the Tabu-search algorithm.

Due to a lack of available computers and the potentially very long runtime for each instance we had to limit the CPU time of the CPLEX tests to 1 hour per instance. For the runtime on CPLEX there seems to be other factors than problem-size playing a part as we during testing had one size 23 instance with a runtime of nearly 4.5 hours and a size 31 instance with a runtime of less than 4 minutes. In comparison the Tabu-algorithm ran 100000 iterations in 26 and 77 seconds for the same instances and found the optimal solution for the size 23 problem and came within 0.71% of the found solution by CPLEX for the size 31 problem. Those instances where CPLEX found the optimal solution is marked as bold in tables 5a, 5b, 6a and 6b.

3 test cases are examined: The first case is the VRP with Multiple Commodities (VRPMC). The second case is the VRPMC with Visit Windows and Route Duration limits. Case 1 and 2 are run on small instances and the results from the Tabu-algorithm are compared to results reported from the mathematical model run on CPLEX. The third case contains both of these problems, but are run for large instances with Tabu search only. Some of the instances had pickup demand generated with a geographical aspect in mind. These geographical aspects does not seem to have any effect due to the method used to generate instances with 2 and 3 commodities.

After the 3 test cases are examined, optimal solutions found by the mathematical model is compared to the solutions found by the Tabu algorithm. Then results from test case 3 is compared to results from the literature.

### 6.2.1 Test Case 1: Multiple commodities for small instances

The following tables show calculated gap, in percentage, between the results from our Tabu-algorithm and the results from CPLEX in terms of cost. Positive numbers indicate that AMPL found the better solution while negative numbers indicate that the Tabu-algorithm found the best result. Those having a gap of 0.00% indicates that the same solution was found both for CPLEX and the Tabu-algorithm. This could indicate that

the optimal solution is found, but we cannot say it with any certainty for those having a running-time of more than one hour in CPLEX as we did not find the confirmed optimal solution for those instances. Those instances where CPLEX found an optimal solution within the given time-restriction have been marked in bold.

To present all instances in one table would take too much space and not fit on one page, so the tables are split into 2 parts, shown by Table 5a and 5b. The 100% column indicates that the vessel capacity is fully utilized for every commodity when the vessel leaves the offshore base or when the vessel returns from its trip.

In the search for the largest problem-size the Tabu-algorithm would be able to solve to optimality, all instances at 80% utilization in Table 5a and Table 6a were run without time-limit. Instances 2c_030b20 and 3c_030b20 are not marked as optimal because CPLEX reported internal errors on these instances after 13 and 15 hours. However in the Hamiltonian-run, feasible solutions were found in the allotted 1 hour. In comparison both size 31 instances were solved by CPLEX in less than 10 and 4 minutes for two commodities and 31 and 17 minutes for three commodities.

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c_005 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_006 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_016 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_016b20 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_021 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_021b20 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_022 | **0,0** | **0,0** | **0,0** | -3,4 |
| 2c_022b20 | **0,0** | 0,0 | 0,0 | 0,2 |
| 2c_023 | **0,0** | 0,0 | 0,0 | 0,2 |
| 2c_023b20 | **0,0** | 0,0 | 0,0 | -1,1 |
| 2c_026 | **3,9** | **3,9** | 3,9 | 5,9 |
| 2c_026b20 | **3,9** | **3,9** | **3,9** | 0,5 |
| 2c_030 | 0,8 | 1,5 | 6,7 | -2,7 |
| 2c_030b20 | 0,7 | 0,3 | -0,6 | -9,2 |
| 2c_031 | **0,7** | **0,7** | **0,7** | 0,6 |
| 2c_031b20 | **0,7** | **0,7** | **0,7** | 1,5 |

Table 5a: TABU deviation from CPLEX in percent, 2 commodities, part1

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c_6_c110_3 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_6_c1_6_3 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_7_c1_2_2 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_8_c110_5 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_8_c1_4_3 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_9_c1_2_7 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_11_c110_7 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_11_c1_2_9 | **0,0** | 0,0 | 1,4 | -1,2 |
| 2c_12_c110_1 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_13_c1_4_4 | **2,1** | **2,1** | **2,1** | **0,0** |
| 2c_13_c1_6_1 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_14_c110_9 | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_15_c1_2_5 | 6,5 | 6,5 | 6,5 | -6,0 |
| 2c_15_c1_4_6 | 2,6 | 2,6 | 0,0 | -1,7 |
| 2c_16_c110_8 | **0,0** | **0,0** | **0,0** | 0,0 |
| 2c_19_c110_4 | 0,0 | 0,0 | 0,0 | -1,1 |
| 2c_19_c1_210 | 5,9 | 5,9 | 5,9 | -23,7 |
| 2c_19_c1_4_2 | 8,1 | 8,1 | 8,1 | -6,5 |
| 2c_20_c1_4_7 | 4,7 | 4,7 | 4,7 | 1,2 |
| 2c_20_c1_6_4 | 0,0 | 0,0 | 0,0 | 0,0 |
| 2c_21_c1_2_3 | 3,6 | 3,6 | 3,6 | -8,2 |
| 2c_22_c110_6 | 3,1 | 3,1 | 3,1 | 5,7 |
| 2c_23_c1_2_8 | 3,3 | 3,3 | 3,3 | -31,1 |
| 2c_23_c1_4_9 | 8,4 | 8,4 | 8,4 | -29,3 |
| Average | 1,5 | 1,5 | 1,6 | -2,7 |

Table 5b: TABU deviation from CPLEX in percent, 2 commodities, part2

As mentioned a gap of 0.00% could indicate an optimal solution giving a possible optimal Tabu-solution in the instance 2c023b20 with 95% utilization of the space on the vessel, but it cannot be said for certain looking only at this single solution, shown in tables 5b and 5a. The biggest instance where Tabu found the optimal solution as reported by CPLEX is the 2c023b20 instance with 80% utilization. Since both the solutions with 90% and 95% utilization have the same cost and route these are most likely optimal as well since the only change from 80% utilization is the restriction of less capacity of the vessel. As the name of the instance suggests this instance has 23 nodes including the depot.

The Tabu-solutions are compared against two runs of CPLEX, one "normal" search for the optimal solution and one with an additional constraint making it only search for Hamiltonian solutions. The goal is to show how the Tabu-algorithm compares to the best found solution, and then it can be seen that for 80%, 90% and 95% the average gap between the best found solution found using CPLEX, and the solution found using Tabu, is approximately 1.5%. But when the problem becomes more constrained with 100% utilization of the vessel capacity the situation is reversed and the Tabu-algorithm produces on average 2.7% better solutions than the best found solution generated by the mathematical model using CPLEX. This is due to CPLEX using more CPU time as the problem becomes more difficult. CPLEX will find optimal solutions given enough time, but in this thesis CPLEX is limited to 1 hour and is to that end unable to find optimal solutions for all instances. The solutions quality will in many cases get worse as the vessels cargo hold approaches full capacity for the various commodities.

Table 6a and 6b shows the same as Table 5a and 5b, but with three commodities instead of two.

3c031b20 100% did not produce a solution in any of the runs in CPLEX and is listed with a gap of -100,00% since the Tabu-algorithm was the only one to find a feasible solution.

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 3c_005 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_006 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_016 | **0,0** | **0,0** | **0,6** | **0,0** |
| 3c_016b20 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_021 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_021b20 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_022 | **0,0** | 0,0 | 0,0 | -5,0 |
| 3c_022b20 | **0,0** | 0,0 | 0,0 | 0,0 |
| 3c_023 | **0,0** | 0,0 | 0,0 | 0,2 |
| 3c_023b20 | **0,0** | 0,0 | 0,0 | -3,0 |
| 3c_026 | **3,9** | 3,9 | 3,9 | -7,3 |
| 3c_026b20 | **3,9** | **3,9** | 3,9 | 0,0 |
| 3c_030 | -0,4 | 5,8 | 5,0 | -4,6 |
| 3c_030b20 | -0,9 | -7,5 | -2,0 | -11,6 |
| 3c_031 | **0,7** | **0,7** | **0,7** | **0,6** |
| 3c_031b20 | **0,7** | **0,7** | **0,7** | -100,0 |

Table 6a: TABU deviation from CPLEX, 3 commodities, part1

In these two tables, 6a and 6b, for three commodities it can be seen that the biggest instance where both the mathematical model and Tabu found the optimal solution is instance 3c_023b20, same as for two commodities. The reason it is known that 3c_023b20 with 80% utilization is optimal is that this instance was run with no time limit using the mathematical model, it took 5.5 hours, and is included in the results to show how big solutions the Tabu-algorithm is able to solve to optimality. The Tabu algorithm used 34 seconds to go through the 100000 iterations and solve the instance.

Table 7 has a more compressed version of the previous tables and some additional info. The average cost gap and avg % gap shows the solution gap between the Tabu and the CPLEX generated solutions. From this view there is no clear difference between two and three commodities for 80%, 90% and 95% utilization, however it is worth noticing the big difference between 95% and 100% utilization of the vessel. These numbers indicate that CPLEX, when restricted to 1 hour, has more difficulties than the Tabu-algorithm in finding good solutions when the vessel is at full capacity.

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 3c_6_c110_3 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_6_c1_6_3 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_7_c1_2_2 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_8_c110_5 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_8_c1_4_3 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_9_c1_2_7 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_11_c110_7 | **0,0** | **0,0** | **0,0** | -0,3 |
| 3c_11_c1_2_9 | 0,0 | 0,0 | 2,4 | -0,7 |
| 3c_12_c110_1 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_13_c1_4_4 | **2,1** | **2,1** | **2,1** | **0,0** |
| 3c_13_c1_6_1 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_14_c110_9 | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_15_c1_2_5 | 6,5 | 6,5 | 6,5 | -14,2 |
| 3c_15_c1_4_6 | 2,6 | 2,6 | 0,0 | -3,7 |
| 3c_16_c110_8 | **0,0** | **5,2** | **5,2** | -1,1 |
| 3c_19_c110_4 | 0,0 | 0,0 | 0,0 | -1,5 |
| 3c_19_c1_210 | 5,9 | 5,9 | 3,2 | -15,2 |
| 3c_19_c1_4_2 | 8,1 | 8,1 | 8,1 | -3,8 |
| 3c_20_c1_4_7 | 4,7 | 4,7 | 4,7 | 3,9 |
| 3c_20_c1_6_4 | 0,0 | 0,6 | 0,6 | -5,9 |
| 3c_21_c1_2_3 | -1,0 | 3,6 | 1,6 | -24,1 |
| 3c_22_c110_6 | 3,1 | 3,1 | 0,0 | 1,8 |
| 3c_23_c1_2_8 | 2,3 | 2,7 | 1,4 | -30,8 |
| 3c_23_c1_4_9 | 8,4 | 5,9 | 8,4 | -20,8 |
| Average | 1,3 | 1,5 | 1,4 | -6,2 |

Table 6b: TABU deviation from CPLEX, 3 commodities, part2

| # Commo. | % Util. | avg. Cost-gap | avg. % gap | # TABU best | % TABU best |
|---|---|---|---|---|---|
| 2 | 80 | 9,5 | 1,5 | 24 | 60,0 |
| 2 | 90 | 9,6 | 1,5 | 24 | 60,0 |
| 2 | 95 | 9,6 | 1,6 | 25 | 62,5 |
| 2 | 100 | -19,3 | -2,7 | 32 | 80,0 |
| 2 | average | 2,4 | 0,4 | 26 | 65,6 |
| 3 | 80 | 8,6 | 1,3 | 27 | 67,5 |
| 3 | 90 | 11,0 | 1,5 | 23 | 57,5 |
| 3 | 95 | 9,5 | 1,4 | 23 | 57,5 |
| 3 | 100 | -32,5 | -6,2 | 36 | 90,0 |
| 3 | average | -0,9 | -0,5 | 27 | 68,1 |

Table 7: Average gap for TABU compared to CPLEX and number and percentage of instances where TABU found the best known solution.

The last column show that for the 80% utilization with two commodities, Tabu has for 60.0% of the instances found the optimal or best found solution compared to the solutions found using CPLEX. The column "# TABU best" tells how many of the instances has their best found solution found by the Tabu-algorithm. 65.6% of the instances with 2 commodities had solutions found by the Tabu-algorithm that were as good as or better than CPLEX limited to 1 hour. For instances with 3 commodities the share is 68.1%. Again it is seen that there is a big difference between 95% and 100% utilization of the cargo space and that the Tabu-algorithm with high utilization is better compared to the solutions found using AMPL. This goes for both two and three commodities. For three commodities there is a slight increase in best found solutions by Tabu for 80% and 100% utilization, other than that there is no difference worth mentioning between 80, 90 and 95 percent utilization and between two and three commodities for these levels of utilization.

In total Tabu found the best found solution in 214 of the 320 instances, that is in total 66.9% of the best found solutions.

| # Commo. | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2 | 55,0 | 45,0 | 45,0 | 42,5 |
| 3 | 52,5 | 40,0 | 37,5 | 40,0 |
| average | 53,8 | 42,5 | 41,3 | 41,3 |

Table 8a: Found optimal solutions by TABU in percent

| # Commo. | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2 | 67,5 | 57,5 | 55,0 | 42,5 |
| 3 | 65,0 | 52,5 | 50,0 | 42,5 |
| average | 66,3 | 55,0 | 52,5 | 42,5 |

Table 8b: Found optimal solutions by CPLEX in percent

Table 8a shows the percentage of instances solved to optimality by the Tabu-algorithm for the different utilizations. They are optimal since these instances were solved within the set time limit for CPLEX. These are mostly solutions with a problem size less than 22 vertices. Table 8b shows the total amount of instances solved to optimality using CPLEX. Looking at these two tables alone it might look as if CPLEX does a bit better than Tabu, at least for the instances with lower utilization of the cargo capacity. However, if the number of best found solutions, shown in Table 7, is considered instead of only the optimal solutions found, it can be seen that the number of tabu solutions better than, or equal to, the best found solution using the mathematical model are in favor of the Tabu-algorithm.

| # Commo. | % Util. | % TABU Ham | % best found Ham | % Ham found |
|---|---|---|---|---|
| 2 | 80 | 82,5 | 97,5 | 97,5 |
| 2 | 90 | 80,0 | 95,0 | 97,5 |
| 2 | 95 | 80,0 | 90,0 | 97,5 |
| 2 | 100 | 35,0 | 32,5 | 60,0 |
| 2 | average | 69,4 | 78,8 | 88,1 |
| 3 | 80 | 82,5 | 97,5 | 97,5 |
| 3 | 90 | 77,5 | 92,5 | 97,5 |
| 3 | 95 | 77,5 | 90,0 | 97,5 |
| 3 | 100 | 30,0 | 32,5 | 52,5 |
| 3 | average | 66,9 | 78,1 | 86,3 |

Table 9: Percentage of instances where solution produced by the TABU-algorithm had Hamiltonian-shape, where the best known solution had Hamiltonian-shape & percentage of instances where Ham-shape is known to exist

Observing the changes in the solution shapes is one of the goals in this thesis and is shown in Table 9. For 80%, 90% and 95% utilization with two commodities 82.5%, 80% and 80.0%, respectively, of the best Tabu produced solutions have Hamiltonian

solution shapes. For three commodities the numbers are 82.5% and 77.5%, while there are Hamiltonian solutions found for 97.5% of the 40 instances for each of the three lower utilization levels for both two and three commodities. For 100% utilization the numbers are 35.0% for two commodities and 30.0% for 3 commodities. This indicates a trend of a decreasing number of Hamiltonian shaped solutions for the Tabu solutions when increasing the utilization, and perhaps a tiny decrease in Hamiltonian solutions when going from two to three commodities.

For 100% utilization the number of Hamiltonian shaped solutions found are only 60.0% for two commodities and 52.5% for three commodities. In total Tabu found a Hamiltonian shaped solution for 68.1% of the instances. Hamiltonian solutions are found for 87.2% of the 320 possible instances. 78.5% of the best found solutions had Hamiltonian shape.

The final column shows the percentage of the Tabu-solutions having a Hamiltonian shape. This shows that for 80%, 90% and 95% utilization with both two and three commodities there were only one solution that did not find a Hamiltonian shaped solution. Observations show that, in general, the more constrained the problem becomes the less Hamiltonian solutions are found. For 100 percent utilization of the vessel the number of Hamiltonian shaped solutions plummet down to 32.5% for the best found solutions. A similar behaviour is shown in the last three columns for both two and three commodities, but for the last column the change is not that big.

| # Commo. | % Util. | % 0 double | % 1 double |
|---|---|---|---|
| 2 | 80 | 97,5 | 2,5 |
| 2 | 90 | 95,0 | 5,0 |
| 2 | 95 | 90,0 | 10,0 |
| 2 | 100 | 32,5 | 67,5 |
| 2 | Total | 78,8 | 21,2 |
| 3 | 80 | 97,5 | 2,5 |
| 3 | 90 | 90,0 | 10,0 |
| 3 | 95 | 90,0 | 10,0 |
| 3 | 100 | 32,5 | 67,5 |
| 3 | Total | 77,5 | 22,5 |

Table 10: Percentage of instances having zero or one installation requiring a second visit in the solutions made by the TABU-algorithm

Another goal of this thesis is to observe the changes in number of installations requiring two visits when subjected to various changes. Table 10 shows how many second visits were required in the best found Tabu-solutions. There seems to be no change worth mentioning from two to three commodities, but the number of vertices requiring two visits increase with higher utilization of the vessel. As in the previous tables there is a big change from 95% utilization to 100% utilization.

The "% 0 double" column indicates the percentage of instances with a best found solution where no installations require a second visit, these instances have a Hamiltonian shape.

### 6.2.2 Test Case 2: Added Visit Windows for small instances

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c_005vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_006vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_016b20vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_016vw | **0,6** | **0,6** | **0,6** | **0,0** |
| 2c_021b20vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_021vw | **2,3** | **2,3** | **2,3** | **6,5** |
| 2c_022b20vw | 5,9 | 5,9 | 5,9 | 12,7 |
| 2c_022vw | -15,2 | -1,8 | -32,9 | -100,0 |
| 2c_023b20vw | 3,4 | 3,4 | 3,4 | -100,0 |
| 2c_023vw | 1,5 | 1,5 | 1,5 | -0,9 |
| 2c_026b20vw | 6,4 | 6,4 | 6,4 | **6,4** |
| 2c_026vw | 4,7 | 4,7 | 4,7 | -100,0 |
| 2c_030b20vw | -100,0 | -100,0 | -100,0 | — |
| 2c_030vw | 9,7 | 12,9 | 6,4 | 14,8 |
| 2c_031b20vw | **1,1** | **1,1** | **1,1** | 2,0 |
| 2c_031vw | **2,0** | **2,0** | **2,0** | 1,0 |

Table 11a: Visit windows, TABU deviation from CPLEX, 2 commodities, part 1

Tables 11a and 11b show the cost gap in percentage for the Tabu solution compared to the best solution found by CPLEX. Positive numbers indicate that CPLEX found the best solution, while negative numbers indicate that Tabu found the best solution. 0,00%

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c_6_c110_3vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_6_c1_6_3vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_7_c1_2_2vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_8_c110_5vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_8_c1_4_3vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_9_c1_2_7vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_11_c110_7vw | **0,0** | **0,0** | **8,2** | -1,2 |
| 2c_11_c1_2_9vw | 0,0 | 0,0 | 2,4 | 6,3 |
| 2c_12_c110_1vw | **7,6** | **7,6** | **7,6** | **7,6** |
| 2c_13_c1_4_4vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_13_c1_6_1vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 2c_14_c110_9vw | **2,4** | **2,4** | **2,4** | **1,7** |
| 2c_15_c1_2_5vw | 6,5 | 6,5 | 6,5 | -6,3 |
| 2c_15_c1_4_6vw | 2,3 | 2,3 | 0,0 | 2,7 |
| 2c_16_c110_8vw | **0,0** | **0,0** | **0,0** | 1,1 |
| 2c_19_c110_4vw | 0,0 | 0,0 | 0,0 | -100,0 |
| 2c_19_c1_210vw | 15,4 | 15,4 | 15,4 | -100,0 |
| 2c_19_c1_4_2vw | 9,6 | 9,6 | 9,6 | -100,0 |
| 2c_20_c1_4_7vw | 4,7 | 4,7 | 4,7 | 16,2 |
| 2c_20_c1_6_4vw | 12,3 | 12,3 | 12,3 | -100,0 |
| 2c_21_c1_2_3vw | 100,0 | 100,0 | 100,0 | — |
| 2c_22_c110_6vw | 0,0 | 0,0 | 0,0 | 20,4 |
| 2c_23_c1_2_8vw | -0,5 | 2,0 | 5,0 | -100,0 |
| 2c_23_c1_4_9vw | 8,7 | 8,7 | 8,7 | -100,0 |
| Average | 2,3 | 2,8 | 2,1 | -20,2 |

Table 11b: Visit windows, TABU deviation from CPLEX, 2 commodities, part 2

indicates that Tabu and CPLEX both found the same solution, this indicates a likely optimal solution, but it cannot be said with 100% certainty unless CPLEX was able to finish within the given hour per instance. If the optimal solution to an instance is found the number is written in bold.

There are two instances where neither CPLEX nor Tabu were able to find a feasible solution, indicated by '—'. There are some instances where only the Tabu algorithm was able to find a solution. This is illustrated by the gap being -100.0%. There is also an instance, 2c_21_c1_2_3vw, where only the Hamiltonian run on CPLEX was able to find a solution, this is shown by the gap being 100.0%.

The biggest instance solved to optimality in this thesis with the Tabu algorithm was of size 23 without visit windows, and size 21 with visit windows for both two and three commodities.

For two commodities with visit windows it is observed that the average gap is just above 2% except for 100% utilization, but there are some instances with a gap of 5-10% and even one up to 15.4% which is not very good. Then there are also instances where the gap is just as big, or bigger, but in favour of the solution found by the Tabu-algorithm.

In tables 12a and 12b the cost gap for three commodities with visit windows is shown.

These tables show an average in favour of the Tabu-algorithm. Some of this is due to the fact that for two of the instances with three commodities only Tabu was able to find a feasible solution on all levels of utilization. Looking at individual instances there are some gaps in the 6-9% range, and for instance 3c_16_c110_8vw at 95% utilization the gap is 34,9%. This is not a good sign for the robustness of the Tabu-algorithm. However, the power of the Tabu-algorithm is shown in the number of instances where the Tabu-algorithm found feasible solutions in a few minutes while the mathematical model was unable to find any feasible solutions in its allotted one hour. And even if some instances have a bad gap, the average speaks in favor of the Tabu-algorithm. Again there are big gaps in favour of the Tabu-algorithm as well.

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 3c_005vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_006vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_016b20vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_016vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_021b20vw | **3,2** | **3,2** | **6,2** | **2,0** |
| 3c_021vw | **2,4** | **2,4** | **0,0** | -100,0 |
| 3c_022b20vw | 2,6 | -1,0 | 2,6 | -100,0 |
| 3c_022vw | -1,9 | 0,2 | 0,2 | -100,0 |
| 3c_023b20vw | 8,5 | 8,5 | -4,9 | -100,0 |
| 3c_023vw | 5,4 | -100,0 | -26,9 | -100,0 |
| 3c_026b20vw | 6,9 | 6,9 | 6,9 | 1,8 |
| 3c_026vw | 6,1 | 6,1 | 7,7 | -100,0 |
| 3c_030b20vw | -100,0 | -100,0 | -100,0 | -100,0 |
| 3c_030vw | -100,0 | -100,0 | -100,0 | -100,0 |
| 3c_031b20vw | **0,7** | **0,7** | 0,7 | -100,0 |
| 3c_031vw | 3,3 | 3,3 | 3,3 | 1,5 |

Table 12a: Visit windows, TABU deviation from CPLEX, 3 commodities, part 1

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 3c_6_c110_3vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_6_c1_6_3vw | **0,0** | **0,0** | **1,0** | **0,0** |
| 3c_7_c1_2_2vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_8_c110_5vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_8_c1_4_3vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_9_c1_2_7vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_11_c110_7vw | **0,0** | **0,0** | **0,0** | -3,0 |
| 3c_11_c1_2_9vw | 0,8 | 0,8 | 2,4 | -0,8 |
| 3c_12_c110_1vw | **14,4** | **14,4** | **14,0** | -100,0 |
| 3c_13_c1_4_4vw | **2,1** | **2,1** | **2,1** | **3,3** |
| 3c_13_c1_6_1vw | **0,0** | **0,0** | **0,0** | **0,0** |
| 3c_14_c110_9vw | **1,9** | **1,9** | **1,9** | **1,7** |
| 3c_15_c1_2_5vw | 6,5 | 6,5 | 6,5 | -100,0 |
| 3c_15_c1_4_6vw | 2,3 | 2,3 | 2,3 | -100,0 |
| 3c_16_c110_8vw | **7,5** | **0,0** | **34,9** | -2,1 |
| 3c_19_c110_4vw | 3,5 | 3,5 | 3,6 | -100,0 |
| 3c_19_c1_210vw | 3,2 | 3,2 | 1,0 | -100,0 |
| 3c_19_c1_4_2vw | 8,1 | 8,1 | 8,1 | -100,0 |
| 3c_20_c1_4_7vw | 12,4 | 12,4 | 12,4 | 5,3 |
| 3c_20_c1_6_4vw | 5,0 | 4,3 | -4,0 | -100,0 |
| 3c_21_c1_2_3vw | 5,5 | 7,0 | 2,5 | -100,0 |
| 3c_22_c110_6vw | 8,6 | 8,6 | 11,9 | 3,3 |
| 3c_23_c1_2_8vw | 0,7 | -6,5 | 1,9 | -100,0 |
| 3c_23_c1_4_9vw | 5,8 | 5,8 | 5,8 | -100,0 |
| Average | -1,9 | -4,9 | -2,4 | -47,2 |

Table 12b: Visit windows, TABU deviation from CPLEX, 3 commodities, part 2

| # Commo. | % Util. | avg. Cost-gap | avg. % gap | # TABU best | % TABU best |
|---|---|---|---|---|---|
| 2 | 80 | 5,8 | 2,3 | 20 | 50,0 |
| 2 | 90 | 7,5 | 2,2 | 19 | 47,5 |
| 2 | 95 | 6,1 | 2,1 | 18 | 45,0 |
| 2 | 100 | -121.1 | -23,0 | 25 | 62,5 |
| 2 | average | -25,4 | -4,1 | 21 | 51,3 |
| 3 | 80 | 5,1 | -1,9 | 15 | 37,5 |
| 3 | 90 | -12,3 | -4,9 | 18 | 45,0 |
| 3 | 95 | 9,7 | -2,4 | 17 | 42,5 |
| 3 | 100 | -228,7 | -47,2 | 33 | 82,5 |
| 3 | average | -56,5 | -14,1 | 21 | 51,9 |

Table 13: Visit windows, TABU average gap from CPLEX

Comparing the gap for two and three commodities with visit windows in Table 13 it is seen that the Tabu-algorithm produces overall better results for three commodities than for two commodities. The effects of increasing utilization has no clear effect for 80, 90 and 95 percent utilization. But when looking at 100% utilization it seems that increasing the utilization of the vessel leads to a gap in favour of the Tabu-algorithm.

The table also shows the percentage and number of all instances that has its best found solution from the Tabu-algorithm. Comparing the levels of utilization the percentage of best found solutions found by Tabu is steady at the lower levels of utilization, but at 100% it increases and Tabu finds 62.5% of the best found solutions for two commodities and 82.5% for three commodities. Some of this increase is from the time limit imposed on the CPLEX-runs.

Comparing two and three commodities it looks like three commodities lead to more best found solutions by Tabu when the utilization is 100%. For 80% utilization, CPLEX finds better solutions .

Table 14 shows the percentage of solutions where one or more installations require two visits, for solutions found by Tabu only. For the lower levels of utilization most solutions are Hamiltonian and have no installations requiring two solutions. For two commodities there is no change between 80%, 90% and 95% utilization. While for 100% utilization of the vessel capacity there are only 25% of the solutions that have Hamiltonian shape, as confirmed by Table 15. For 100% utilization and two commodi-

| # Commo. | % Util. | % 0 | % 1 | % 2 | % 3 | % 4 | % 5 | % 8 | % No solution |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 80 | 85,0 | 10,0 | 2,5 | 0,0 | 0,0 | 0,0 | 0,0 | 2,5 |
| 2 | 90 | 85,0 | 10,0 | 2,5 | 0,0 | 0,0 | 0,0 | 0,0 | 2,5 |
| 2 | 95 | 85,0 | 10,0 | 2,5 | 0,0 | 0,0 | 0,0 | 0,0 | 2,5 |
| 2 | 100 | 25,0 | 42,5 | 15,0 | 5,0 | 2,5 | 2,5 | 2,5 | 5,0 |
| 2 | Average | 70,0 | 18,1 | 5,6 | 1,3 | 0,6 | 0,6 | 0,6 | 3,1 |
| 3 | 80 | 82,5 | 12,5 | 5,0 | 0,0 | 0,0 | 0,0 | 0,0 | 0,0 |
| 3 | 90 | 82,5 | 10,0 | 5,0 | 2,5 | 0,0 | 0,0 | 0,0 | 0,0 |
| 3 | 95 | 67,5 | 20,0 | 7,5 | 0,0 | 2,5 | 2,5 | 0,0 | 0,0 |
| 3 | 100 | 20,0 | 55,0 | 12,5 | 7,5 | 5,0 | 0,0 | 0,0 | 0,0 |
| 3 | Average | 63,1 | 24,4 | 7,5 | 2,5 | 1.9 | 0,6 | 0,6 | 0,0 |

Table 14: Visit windows. Percentage of instances having X nodes requiring two visits in TABU generated solutions

ties there is one instance having as much as eight nodes requiring two visits. 27,5% of the solutions have more than one node requiring two visits.

For three commodities the number of instances having 1 or more nodes requiring two visits is bigger than for two commodities. Already at 90% utilization there is an instance with 3 nodes requiring two visits, and at 95% utilization there are instances with 4 and 5 such nodes. At 100% utilization the number of nodes in an instance requiring two visits decreases some, but there are still 55,0% of the instances having one such node, and 25% having more than one node requiring two visits.

| # Commo. | % Util. | % TABU Ham | % best found Ham | % Ham found |
|---|---|---|---|---|
| 2 | 80 | 85,0 | 97,5 | 97,5 |
| 2 | 90 | 85,0 | 97,5 | 97,5 |
| 2 | 95 | 85,0 | 97,5 | 97,5 |
| 2 | 100 | 25,0 | 40,0 | 50,0 |
| 2 | average | 70,0 | 83,1 | 85,6 |
| 3 | 80 | 82,5 | 92,5 | 92,5 |
| 3 | 90 | 82,5 | 90,0 | 90,0 |
| 3 | 95 | 67,5 | 85,0 | 92,5 |
| 3 | 100 | 20,0 | 27,5 | 45,0 |
| 3 | average | 63,1 | 73,8 | 80,0 |

Table 15: Visit Windows, TABU created solutions with Ham-shape & number of solutions where Ham-shaped solutions are known to exist

The number of problem instances where a feasible Hamiltonian solution is found is shown in Table 15. This table also shows how many of the best found solutions have Hamiltonian shape. As for most of the instances there are little differences between the three lower levels of utilization. However there is a clear decrease in the number of Hamiltonian shaped solutions when going from two to three commodities. There is also a large decrease in the number of Hamiltonian shaped solutions at 100% utilization for both two and three commodities compared to the lower levels of utilization.

For two commodities, on average 85.6% of the instances have a feasible Hamiltonian shaped solution found. For three commodities there are on average 80.0% Hamiltonian shaped solutions found. For the best found solutions there are 83.1% solutions with Hamiltonian shape for the instances with two commodities. For three commodities 73.8% of the instances have a Hamiltonian shaped solution. For all solutions with visit windows found by the Tabu-algorithm there are on average 70.0% Hamiltonian shaped solutions for 2 commodities and on average 63.1% Hamiltonian shaped solutions for 3 commodities.

In other words there is a trend across the board that when the number of commodities increase there is a decrease in the number of solutions with a Hamiltonian shape. There is also a decrease in Hamiltonian shaped solutions as the utilization of the vessel capacities increases.

| TABU | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c | 30,0 | 35,0 | 32,5 | 32,5 |
| 3c | 30,0 | 32,5 | 30,0 | 27,5 |
| average | 30,0 | 33,8 | 31,3 | 30,0 |

Table 16a: Percentage of instances with optimal solutions found by the TABU-algorithm for visit windows

| CPLEX | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c | 50,0 | 50,0 | 50,0 | 42,5 |
| 3c | 47,5 | 47,5 | 45,0 | 35,0 |
| average | 48,8 | 48,8 | 47,5 | 38,8 |

Table 16b: Percentage of instances with optimal solutions found by CPLEX for visit windows

Tables 16a and 16b show the percentage of instances with 1 visit window with optimal solutions found by Tabu and the percentage of optimal solutions found using the mathematical model.

The numbers show a trend that the amount of optimal solutions found decrease the more constrained the vessel capacity becomes, but it is not really visible before the load for the different commodities is at 100% utilization of the vessel capacity. For the number of optimal solutions found by the Tabu-algorithm the change is not big enough to be able to say anything. There is also a decrease in the number of optimal solutions found when going from two to three commodities.

Comparing the results from the Tabu-algorithm to the results found using the mathematical model, the Tabu algorithm found 68.2% of all the optimal solutions.

| Avg. Cost Gap | % MC | % MC VW | % Cost Change |
|---|---|---|---|
| 2c 80% | 1,5 | 2,3 | -0,2 |
| 2c 90% | 1,5 | 2,8 | -0,2 |
| 2c 95% | 1,6 | 2,1 | 0,2 |
| 2c 100% | -2,7 | -20,2 | 2,6 |
| 2c average | 0,4 | -3,3 | 0,6 |
| 3c 80% | 1,3 | -1,9 | 3,1 |
| 3c 90% | 1,5 | -4,9 | 2,2 |
| 3c 95% | 1,4 | -2,4 | 5,2 |
| 3c 100% | -6,2 | -47,2 | 3,9 |
| 3c average | -0,5 | -14,1 | 3,6 |

Table 17: Cost gap between solutions with and without visit windows

**Multiple Commodities vs Multiple Commodities with Visit Windows**

Looking at Table 17, '% MC' is the gap between solutions found by Tabu and CPLEX while the '% MC VW' column represents the gap between the solutions with visit windows found by Tabu and CPLEX. The '% Cost Change' column contains the change in cost for the best found solutions with visit windows: The cost for solutions with visit windows are subtracted from the best found cost in the corresponding instances without visit windows.

Looking at the columns showing average cost gap in percentage it seems that there is a difference in the cost gap between the instances with and without visit windows, especially for three commodities. For two commodities there is a lower average cost for the solutions with visit windows at 80% and 90% utilization. At 95% utilization there is a small increase in cost when adding visit windows, while for 100% utilization the increase is an average of 2.6%. When adding a commodity the trend of increasing cost for visit windows when increasing utilization is not so clear. However, when adding visit windows to three commodities the average cost increase by an average of 3.6%. This shows that the gap between the solutions with and without visit windows increase when adding visit windows, most likely because the problem becomes more complex.

| Two Visits TABU | % MC | % MC VW |
|---|---|---|
| 2c 80% | 2,5 | 12,5 |
| 2c 90% | 5,0 | 12,5 |
| 2c 95% | 10,0 | 12,5 |
| 2c 100% | 67,5 | 70,0 |
| 2c average | 21,3 | 26,9 |
| 3c 80% | 2,5 | 17,5 |
| 3c 90% | 10,0 | 17,5 |
| 3c 95% | 10,0 | 32,5 |
| 3c 100% | 67,5 | 80,0 |
| 3c average | 22,5 | 36,9 |

Table 18: Percentage of instances having one or more installation requiring two visits.

Table 18 shows the number of solutions with one or more installations requiring two visits. Without visit windows it seems that only the different level of utilizations of the vessel capacities had any effect. However, when adding visit windows there is an increase in instances where one or more platforms require two visits. This increase is best seen for instances with 3 commodities.

In Table 19 the amount of Hamiltonian shaped best found solutions are shown for the test instances with and without visit windows. For both cases a decrease in Hamiltonian shaped solutions is observed when going from two to three commodities. When adding the visit window an increase in solutions with Hamiltonian shape is observed,

| % Best Ham | % MC | % MC VW |
|---|---|---|
| 2c 80% | 82,5 | 97,5 |
| 2c 90% | 80,0 | 97,5 |
| 2c 95% | 80,0 | 97,5 |
| 2c 100% | 35,0 | 40,0 |
| 2c average | 69,4 | 83,1 |
| 3c 80% | 82,5 | 92,5 |
| 3c 90% | 75,5 | 90,0 |
| 3c 95% | 75,5 | 85,0 |
| 3c 100% | 30,0 | 27,5 |
| 3c average | 66,9 | 73,8 |

Table 19: Percentage of best known solutions with Hamiltonian shape for visit windows and without

but the increase is not so big for three commodities as it is for two commodities. The expected outcome was a decrease in Hamiltonian shaped solutions when visit windows was added.

| % TABU solved Ham | % MC | % MC VW |
|---|---|---|
| 2c 80% | 97,5 | 85,0 |
| 2c 90% | 95,0 | 85,0 |
| 2c 95% | 90,0 | 85,0 |
| 2c 100% | 32,5 | 25,0 |
| 2c average | 78,8 | 70,0 |
| 3c 80% | 97,5 | 82,5 |
| 3c 90% | 92,5 | 82,5 |
| 3c 95% | 90,0 | 67,5 |
| 3c 100% | 32,5 | 20,0 |
| 3c average | 78,1 | 63,1 |

Table 20: Percentage of TABU-solutions with Hamiltonian shape for visit windows and without

Table 20 shows the percentage of the Tabu–generated solutions that have Hamiltonian shapes. When adding visit windows the amount of Tabu generated solution with Hamiltonian shape decreases. This result of a decrease in Hamiltonian shaped solutions is the opposite of what is observed in Table 19.

### 6.2.3 Test Case 3: Large Instances

In Table 21 the solution shapes for the instances with multiple commodities are presented, sorted by the degree of utilization. An H illustrates a Hamiltonian solution shape, an L illustrates a Lasso solution shape and a G illustrates a General solution shape. All the solutions show the same trend: With increased capacity, the solution shapes tend to be of Hamiltonian shape. A lasso solution shape is a solution where the vessel visits one or more installations only dropping off the delivery demand, then the vessel visits all other installations performing both deliveries and pickups before it returns to the last installation where only delivery was made. Now the installations that only received its deliveries are visited in reverse order, and the pickup demand is met, so that the shape of the route resembles a lasso.

Only a few instances below 100% utilization have a non-Hamiltonian shape, while no instances with 100% utilization have Hamiltonian solutions. These observations are the same for the instances with 2 commodities as well as those with 3 commodities.

Table 22 shows the solution shapes for instances with visit windows. There is 1 Hamiltonian solution with 100% utilization, instance 2c_072b20 which have 2 commodities. The same instance, with 3 commodities, has a general solution. This shows that the effect of having one more commodity might affect the solution shapes. As for Table 21 the Hamiltonian solutions are heavily represented when the utilization is decreased.

The travel times for all the large instances are presented in Table 23a and Table 23b. These tables show that the travel time is almost the same for all instances with lower utilization than 100%. There are exceptions, though. I.e. instances 2c_200b20 and 2c_262b20 do in fact have a shorter travel time with 100% utilization than when the capacity is increased before visit windows are added. This might be the result of local optima in the Tabu Search, but since the costs are equal for the instances with 80%, 90% and 95% utilization this seems less likely. The total travel time when 100% of the cargo space is utilized is 1.4% higher than for the other degrees of utilization for 2 commodity instances. For the instances with 3 commodities the total travel time is 2.2% higher for the instances with 100% utilization than the instances with 95% utilization and 2.1% higher than the instances with 80% and 90% utilization. There is also a

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c_041b20 | H | H | H | L |
| 2c_045b20 | L | L | L | G |
| 2c_048b20 | H | H | H | G |
| 2c_051b20 | H | H | H | G |
| 2c_072b20 | H | H | H | G |
| 2c_076b20 | H | H | H | L |
| 2c_101b20 | H | H | H | G |
| 2c_111b20 | H | H | H | G |
| 2c_121b20 | G | G | G | G |
| 2c_135b20 | H | H | H | G |
| 2c_151b20 | H | H | H | L |
| 2c_200b20 | G | G | G | L |
| 2c_262b20 | H | H | H | G |
| 2c % Ham | 76,9 | 76,9 | 76,9 | 0,0 |
| 3c_041b20 | H | H | H | L |
| 3c_045b20 | L | L | L | G |
| 3c_048b20 | H | H | H | G |
| 3c_051b20 | H | H | H | L |
| 3c_072b20 | H | H | H | G |
| 3c_076b20 | H | H | H | L |
| 3c_101b20 | H | H | H | G |
| 3c_111b20 | H | H | H | G |
| 3c_121b20 | G | G | G | G |
| 3c_135b20 | H | H | H | G |
| 3c_151b20 | H | H | H | L |
| 3c_200b20 | G | G | G | L |
| 3c_262b20 | H | H | H | G |
| 3c % Ham | 76,9 | 76,9 | 76,9 | 0,0 |

Table 21: Solution shapes for large instances

| Instance | 80% VW | 90% VW | 95% VW | 100% VW |
|---|---|---|---|---|
| 2c_041b20 | H | H | H | H |
| 2c_045b20 | H | H | H | G |
| 2c_048b20 | H | H | H | G |
| 2c_051b20 | H | H | H | L |
| 2c_072b20 | H | H | H | G |
| 2c_076b20 | H | H | H | H |
| 2c_101b20 | H | H | H | G |
| 2c_111b20 | H | H | H | G |
| 2c_121b20 | H | H | H | G |
| 2c_135b20 | H | H | H | G |
| 2c_151b20 | G | G | G | G |
| 2c_200b20 | G | G | G | G |
| 2c_262b20 | H | H | H | G |
| 2c % Ham | 84,6 | 84,6 | 84,6 | 15,4 |
| 3c_041b20 | H | H | H | L |
| 3c_045b20 | G | G | G | G |
| 3c_048b20 | H | H | H | G |
| 3c_051b20 | H | H | H | L |
| 3c_072b20 | H | H | H | G |
| 3c_076b20 | H | H | H | L |
| 3c_101b20 | H | H | H | L |
| 3c_111b20 | H | H | H | L |
| 3c_121b20 | G | G | G | G |
| 3c_135b20 | G | G | G | G |
| 3c_151b20 | G | G | H | G |
| 3c_200b20 | G | G | G | G |
| 3c_262b20 | G | G | G | G |
| 3c % Ham | 53,8 | 53,8 | 61,5 | 0,0 |

Table 22: Solution shapes for large instances with Visit Windows

| Instance | 80 % | 90 % | 95 % | 100 % |
|---|---|---|---|---|
| 2c_041b20 | 370,24 | 370,24 | 370,24 | 376,48 |
| 2c_045b20 | 645,75 | 645,75 | 645,75 | 622,75 |
| 2c_048b20 | 33784,03 | 33784,03 | 33784,03 | 34302,34 |
| 2c_051b20 | 440,43 | 440,43 | 440,43 | 443,08 |
| 2c_072b20 | 197,26 | 197,26 | 197,26 | 212,42 |
| 2c_076b20 | 565,58 | 565,58 | 565,58 | 569,36 |
| 2c_101b20 | 663,88 | 663,88 | 663,88 | 668,51 |
| 2c_111b20 | 553,62 | 553,62 | 553,62 | 559,24 |
| 2c_121b20 | 598,61 | 598,61 | 598,61 | 598,86 |
| 2c_135b20 | 819,26 | 819,26 | 819,26 | 819,45 |
| 2c_151b20 | 778,61 | 778,61 | 778,61 | 761,32 |
| 2c_200b20 | 854,80 | 854,80 | 854,80 | 829,17 |
| 2c_262b20 | 2666,04 | 2666,04 | 2666,04 | 2672,17 |
| 2c Total | 42938,11 | 42938,11 | 42938,11 | 43435,15 |
| 3c_041b20 | 370,24 | 370,24 | 370,24 | 377,79 |
| 3c_045b20 | 645,75 | 645,75 | 645,75 | 621,33 |
| 3c_048b20 | 33784,03 | 33784,03 | 33784,03 | 34302,34 |
| 3c_051b20 | 440,43 | 440,43 | 440,43 | 445,19 |
| 3c_072b20 | 197,26 | 197,26 | 197,26 | 212,42 |
| 3c_076b20 | 565,58 | 565,58 | 565,58 | 566,67 |
| 3c_101b20 | 663,88 | 663,88 | 663,88 | 666,19 |
| 3c_111b20 | 553,62 | 553,62 | 553,62 | 554,91 |
| 3c_121b20 | 598,61 | 598,61 | 598,61 | 594,86 |
| 3c_135b20 | 819,26 | 819,26 | 819,26 | 823,2 |
| 3c_151b20 | 778,61 | 778,61 | 778,61 | 750,24 |
| 3c_200b20 | 854,80 | 854,80 | 854,80 | 847,73 |
| 3c_262b20 | 2666,04 | 2666,04 | 2666,04 | 2753,55 |
| 3c Total | 42938,11 | 42938,11 | 42938,11 | 43516,42 |

Table 23a: Cost/Travel time for large instances

| Instance | 80% VW | 90% VW | 95% VW | 100% VW |
|---|---|---|---|---|
| 2c_041b20 | 366,25 | 366,25 | 366,25 | 383,14 |
| 2c_045b20 | 645,75 | 645,75 | 645,75 | 659,66 |
| 2c_048b20 | 33831,73 | 33831,73 | 33831,73 | 34568,98 |
| 2c_051b20 | 440,43 | 440,43 | 440,43 | 445,21 |
| 2c_072b20 | 199,16 | 199,16 | 199,16 | 257,23 |
| 2c_076b20 | 562,27 | 562,27 | 562,27 | 565,12 |
| 2c_101b20 | 676,56 | 676,56 | 676,56 | 711,03 |
| 2c_111b20 | 571,36 | 571,36 | 571,36 | 580,47 |
| 2c_121b20 | 612,25 | 612,25 | 612,25 | 673,40 |
| 2c_135b20 | 884,72 | 884,72 | 884,72 | 914,44 |
| 2c_151b20 | 803,23 | 803,23 | 803,23 | 781,63 |
| 2c_200b20 | 868,53 | 868,53 | 868,53 | 878,92 |
| 2c_262b20 | 2730,71 | 2730,71 | 2730,71 | 2726,76 |
| 2c Total | 43192,95 | 43192,95 | 43192,95 | 44145,99 |
| 3c_041b20 | 371,51 | 371,51 | 371,51 | 383,35 |
| 3c_045b20 | 657,77 | 657,77 | 657,77 | 649,56 |
| 3c_048b20 | 33784,03 | 33784,03 | 33784,03 | 35947,64 |
| 3c_051b20 | 440,43 | 440,43 | 440,43 | 445,21 |
| 3c_072b20 | 197,73 | 197,73 | 197,73 | 219,56 |
| 3c_076b20 | 564,70 | 564,70 | 564,70 | 566,02 |
| 3c_101b20 | 666,61 | 666,61 | 666,61 | 690,22 |
| 3c_111b20 | 572,22 | 572,22 | 572,22 | 620,28 |
| 3c_121b20 | 607,17 | 607,17 | 607,17 | 658,17 |
| 3c_135b20 | 883,99 | 883,99 | 883,99 | 890,68 |
| 3c_151b20 | 791,26 | 791,26 | 791,26 | 773,47 |
| 3c_200b20 | 866,82 | 866,82 | 866,82 | 863,52 |
| 3c_262b20 | 2717,08 | 2717,08 | 2717,08 | 2853,25 |
| 3c Total | 43121,32 | 43121,32 | 43121,32 | 45560,93 |

Table 23b: Cost/Travel time for large instances with Visit Windows

consistent increase in cost when adding visit windows, but the cost increase is very small. The smallest increase is only 0,2% for 3 commodities at 95% utilization, while for 3 commodities at 100% utilization the cost increase when adding visit windows is 1.0%. The increase of 1% is the largest increase, for 2 commodities at 100% the increase is 0.5% and for the rest 0.3%. It should be noted that most of the increase comes from the two instances that has a very large and disproportionate cost compared to the other instances. Many of the instances with smaller costs actually experience a cost decrease when visit windows are added.

| % Util. | 2c | 3c | 2c VW | 3c VW |
|---|---|---|---|---|
| 80 | 42938,11 | 42938,11 | 43192,95 | 43121,32 |
| 90 | 42938,11 | 42938,11 | 43192,95 | 43121,32 |
| 95 | 42938,11 | 42938,11 | 43192,95 | 43121,32 |
| 100 | 43435,15 | 43516,42 | 44145,99 | 45560,93 |
| Average | 43062,37 | 43082,69 | 43431,21 | 43731,38 |

Table 24: Average Cost/Travel time for large instances

Averages of travel times for both the multiple commodity problems and visit windows problems are given in Table 24. When comparing the average travel times between instances with 2 and 3 commodities, we see that the average is the same for all levels of utilization until we reach 100% utilization. The average for instances with this level of utilization and 3 commodities is 0.2% higher than the ones with 2 commodities. When comparing the instances with visit windows, the result is a little bit different. Here the average for 2 commodities is slightly higher than for 3 commodities with 80%, 90% and 95% utilization. The instances with visit windows and 3 commodities at 100% utilization have the largest travel time. If we look at the total averages at the bottom row, we will see that the average travel time increases by the number of commodities and is also further increased with the addition of visit windows.

Table 25 shows the total number of installations that are visited twice for all instances. The columns '% 0' ,'% 1' and '% 2' show the number of solutions where 0, 1 or 2 installations, respectively, require two visits. Column '% 3 or more' shows how many solutions have 3 or more installations that require a second visit. The instances with 100% utilization have the highest number of second visits.

| Commo. & Util. | % 0 | % 1 | % 2 | % 3 or more |
|---|---|---|---|---|
| 2c 80% | 76,9 | 15,4 | 7,7 | 0,0 |
| 2c 90% | 76,9 | 15,4 | 7,7 | 0,0 |
| 2c 95% | 76,9 | 15,4 | 7,7 | 0,0 |
| 2c 100% | 0,00 | 53,8 | 30,8 | 15,4 |
| 2c Avg | 57,7 | 25,0 | 13,5 | 3,8 |
| 2c 80% vw | 84,6 | 0,0 | 15,4 | 0,0 |
| 2c 90% vw | 84,6 | 0,0 | 15,4 | 0,0 |
| 2c 95% vw | 84,6 | 0,0 | 15,4 | 0,0 |
| 2c 100% vw | 15,4 | 15,4 | 15,4 | 53,8 |
| 2c Avg vw | 67,3 | 3,8 | 15,4 | 13,5 |
| 2c Average | 62,5 | 14,4 | 14,4 | 8,7 |
| 3c 80% | 76,9 | 15,4 | 7,7 | 0,0 |
| 3c 90% | 76,9 | 15,4 | 7,7 | 0,0 |
| 3c 95% | 76,9 | 15,4 | 7,7 | 0,0 |
| 3c 100% | 0,0 | 69,2 | 15,4 | 15,4 |
| 3c Avg | 57,7 | 28,8 | 9,6 | 3,8 |
| 3c 80% vw | 53,8 | 23,1 | 15,4 | 7,7 |
| 3c 90% vw | 53,8 | 23,1 | 15,4 | 7,7 |
| 3c 95% vw | 61,5 | 15,4 | 15,4 | 7,7 |
| 3c 100% vw | 0,0 | 23,1 | 23,1 | 53,8 |
| 3c Avg vw | 42,3 | 21,2 | 17,3 | 19,2 |
| 3c Average | 50,0 | 25,0 | 13,5 | 11,5 |

Table 25: Percentage of installations requiring two visits for large instances

For 2 commodities with utilization 80%, 90% and 95% there is an increase in the number of Hamiltonian solutions when adding visit windows. For 2 and 3 commodities without visit windows, with 100% utilization, there are no Hamiltonian solutions. When visit windows are added for 2 commodities and 100% utilization, the number of Hamiltonian solutions increase. The number of installations per instance requiring 3 or more second visits are also increased. For 3 commodities and 100% vessel utilization the number of installations per instance requiring 3 or more second visits are increased, but there are no increase in Hamiltonian solutions. The average number of Hamiltonian solutions decrease when going from 2 to 3 commodities.

Table 26a shows the CPU time the tabu search for the large instances. These results are for instances without visit windows. Instances with 2 commodities are solved within 1 hour when the problem size is of 121 vertices or less. The CPU time for the instances are increased when introducing a third commodity. The largest instance with 2 commodities is solved within 11 hours. Another 2 and a half hour is needed for 3 commodities. The CPU time between various levels of utilization is negligent and shows that the heuristic is not affected by the level of utilization.

The CPU time needed for instances with visit windows is much higher than those without visit windows, as shown in Table 26b. Even the smallest instance now needs around 13 minutes to be solved, as compared to 2 minutes without visit windows. The largest instance without visit windows was limited to 10 hours. The CPU time for instances with visit windows was limited to 2 hours, except for instances with 2 commodities up to the size of 111. The CPU time for an instance with a visit window is on average increased by a factor of higher than 6 when compared to the corresponding instance without a visit window.

| Instance | 80% | 90% | 95% | 100% |
|---|---|---|---|---|
| 2c_041b20 | 140 | 141 | 141 | 141 |
| 2c_045b20 | 184 | 182 | 181 | 181 |
| 2c_048b20 | 227 | 223 | 224 | 229 |
| 2c_051b20 | 264 | 262 | 262 | 264 |
| 2c_072b20 | 713 | 710 | 710 | 718 |
| 2c_076b20 | 831 | 830 | 828 | 833 |
| 2c_101b20 | 1909 | 1912 | 1910 | 1930 |
| 2c_111b20 | 1897 | 1901 | 1901 | 1958 |
| 2c_121b20 | 3230 | 3243 | 3245 | 3283 |
| 2c_135b20 | 4522 | 4547 | 4547 | 4612 |
| 2c_151b20 | 6266 | 6282 | 6280 | 6303 |
| 2c_200b20 | 15524 | 15664 | 15650 | 15900 |
| 2c_262b20 | 37878 | 37884 | 37952 | 38833 |
| 2c Total | 73585 | 73781 | 73831 | 75185 |
| 2c Average | 5660,38 | 5675,46 | 5679,31 | 5783,46 |
| 3c_041b20 | 178 | 177 | 177 | 182 |
| 3c_045b20 | 235 | 233 | 233 | 229 |
| 3c_048b20 | 286 | 284 | 285 | 292 |
| 3c_051b20 | 336 | 334 | 334 | 339 |
| 3c_072b20 | 913 | 905 | 905 | 917 |
| 3c_076b20 | 1067 | 1053 | 1055 | 1051 |
| 3c_101b20 | 2450 | 2423 | 2420 | 2427 |
| 3c_111b20 | 2436 | 2414 | 2411 | 2486 |
| 3c_121b20 | 4169 | 4130 | 4123 | 4149 |
| 3c_135b20 | 5791 | 5756 | 5769 | 5946 |
| 3c_151b20 | 7973 | 8002 | 7994 | 7977 |
| 3c_200b20 | 19818 | 19885 | 19887 | 20017 |
| 3c_262b20 | 47466 | 47466 | 47509 | 47992 |
| 3c Total | 93118 | 93062 | 93102 | 94004 |
| 3c Average | 7162,92 | 7158,62 | 7161,69 | 7231,08 |

Table 26a: CPU time in seconds for large TABU-instances without Visit Windows

| Instance | 80% VW | 90% VW | 95% VW | 100% VW |
|---|---|---|---|---|
| 2c_041b20 | 1028 | 1028 | 1029 | 1039 |
| 2c_045b20 | 1927 | 1910 | 1908 | 1886 |
| 2c_048b20 | 2509 | 2493 | 2515 | 2508 |
| 2c_051b20 | 2036 | 1043 | 2048 | 1916 |
| 2c_072b20 | 10119 | 10237 | 10241 | 10563 |
| 2c_076b20 | 9589 | 7828 | 8063 | 10005 |
| 2c_101b20 | 19060 | 19871 | 19015 | 16583 |
| 2c_111b20 | 15963 | 16036 | 16033 | 16415 |
| 2c_121b20 | 7200 | 7200 | 7200 | 7200 |
| 2c_135b20 | 7200 | 7200 | 7200 | 7200 |
| 2c_151b20 | 7200 | 7200 | 7200 | 7200 |
| 2c_200b20 | 7200 | 7202 | 7200 | 7200 |
| 2c_262b20 | 7201 | 7210 | 7201 | 7211 |
| 2c Total | 98232 | 97458 | 96853 | 96926 |
| 2c Average | 7556,31 | 7497,77 | 7450,23 | 7455,85 |
| 3c_041b20 | 1365 | 1308 | 1290 | 1291 |
| 3c_045b20 | 2163 | 2160 | 2349 | 2358 |
| 3c_048b20 | 2924 | 2664 | 2938 | 2734 |
| 3c_051b20 | 2492 | 2562 | 2765 | 2389 |
| 3c_072b20 | 7200 | 7200 | 7200 | 7200 |
| 3c_076b20 | 7200 | 7200 | 7200 | 7200 |
| 3c_101b20 | 7200 | 7200 | 7200 | 7200 |
| 3c_111b20 | 7200 | 7200 | 7200 | 7200 |
| 3c_121b20 | 7200 | 7200 | 7200 | 7200 |
| 3c_135b20 | 7200 | 7200 | 7200 | 7200 |
| 3c_151b20 | 7200 | 7200 | 7200 | 7200 |
| 3c_200b20 | 7202 | 7203 | 7200 | 7200 |
| 3c_262b20 | 7202 | 7207 | 7202 | 7200 |
| 3c Total | 73748 | 73504 | 74144 | 73572 |
| 3c Average | 5672,92 | 5654,15 | 5703,38 | 5659,38 |

Table 26b: CPU time in seconds for large TABU-instances with Visit Windows

### 6.2.4 Tabu vs Optimal

| Commo | vw | CPLEX avg | Tabu avg | % Gap |
|---|---|---|---|---|
| 2 | 0 | 625,79 | 627,01 | 0,2 |
| 2 | 1 | 653,97 | 663,16 | 1,4 |
| 3 | 0 | 633,45 | 636,69 | 0,5 |
| 3 | 1 | 676,11 | 696,84 | 3,1 |
| | Avg | 647,33 | 655,93 | 1,3 |

Table 27: Cost gap between found Optimal using CPLEX and corresponding solutions from Tabu.

The gap between the instances where the mathematical model found the optimal solution and the solution found by the Tabu-algorithm on the same instances is shown in Table 27. For two commodities without visit windows the gap is only 0.2%. Both for two and three commodities there is an increase in the gap when adding a visit window to one of the installations in the instance. There is also an increase in gap when increasing the number of commodities from two to three. On average the gap between the solutions found by the Tabu-algorithm and the optimal solutions found by the mathematical model was 1.3%.

| Commo | vw | CPLEX avg | Tabu avg |
|---|---|---|---|
| 2 | 0 | 570,17 | 12,26 |
| 2 | 1 | 337,18 | 62,60 |
| 3 | 0 | 957,04 | 15,39 |
| 3 | 1 | 258,20 | 44,90 |
| | Avg | 530,65 | 33,79 |

Table 28: Difference in CPU-time between optimal solutions found by CPLEX and corresponding solutions found by Tabu

Looking at the average CPU time in Table 28 it is seen that the CPU time for 100000 iterations in the Tabu-algorithm is much shorter than the time to find the optimal solutions using CPLEX. Even with such a big difference in CPU time as is observed here the gap was only 1.3% as shown in Table 27.

The reason the CPU time for CPLEX is larger without visit windows is that for 80% utilization some of these instances had no timelimit and that made a big impact on the

average. This is carried over to the Tabu average as well since CPLEX were able to find the optimal solution for bigger instances that require more CPU time to be solved.

For both the CPLEX and the Tabu average there is a decrease in the average CPU time when going from two to three commodities with visit windows. This is caused by a lower number of instances solved to optimality for three commodities shown in Table 29. The instances that were solved to optimality were smaller and required less CPU time to be solved, thus the decrease in CPU time.

| Commo | vw | # CPLEX opt | # Tabu opt | % CPLEX opt | % Tabu opt |
|---|---|---|---|---|---|
| 2 | 0 | 89 | 75 | 55,6 | 46,9 |
| 2 | 1 | 77 | 54 | 48,1 | 33,8 |
| 3 | 0 | 84 | 68 | 52,5 | 42,5 |
| 3 | 1 | 70 | 48 | 43,8 | 30,0 |
| | Total & Avg | 320 | 245 | 50,0 | 38,3 |

Table 29: Number and percentage of solutions where the optimal solution is found for solutions generated by CPLEX and Tabu

In Table 29 the number and percentage of optimal solutions found by the mathematical model and the Tabu-algorithm is shown. The numbers tells us that a bigger percentage of the instances without visit windows than those with visit windows were solved to optimality. There is also a reduction when adding the third commodity. For all instances from test case 1 and 2 combined, the mathematical model found the optimal solution for 50.0% of all the instances while the Tabu algorithm found 38.3% of all the instances.

In the solutions solved to optimality an average of 82.1% had Hamiltonian shape. Only 78.7% of the solutions found using the Tabu-algorithm for the same instances had Hamiltonian shaped solutions. These numbers are shown in Table 30.

| Commo | vw | CPLEX avg % Ham | Tabu avg % Ham |
|---|---|---|---|
| 2 | 0 | 81,4 | 81,4 |
| 2 | 1 | 86,0 | 79,3 |
| 3 | 0 | 80,9 | 80,9 |
| 3 | 1 | 80,3 | 73,1 |
| | Avg | 82,1 | 78,7 |

Table 30: Percentage of solutions with Hamiltonian shapes found for solutions generated by CPLEX to optimality and the solutions from the corresponding instances solved by the Tabu-algorithm

### 6.2.5 Tabu vs Results from Literature

| Commo | vw | % Gap | Found % non-Ham |
|---|---|---|---|
| 1 | - | 0,0 | 14,3 |
| 1 | 1 | 3,9 | 64,3 |
| 1 | 2 | 2,8 | 71,4 |
| 2 | - | 1,7 | 71,4 |
| 2 | 1 | 5,9 | 78,6 |
| 2 | 2 | 7,8 | 100,0 |
| 3 | - | 1,4 | 64,3 |
| 3 | 1 | 5,5 | 85,7 |
| 3 | 2 | 6,6 | 92,9 |

Table 31: Gap between the best found solutions from the literature and the solutions to the corresponding instances from this thesis for instance size from 16 to 31. 100% utilization only. Also the percentage of non-Hamiltonian shaped solutions

In Table 31 and Table 32 the results from the run with the Tabu-algorithm in this thesis is compared to the best found results from Gribkovskaia et al. (2007). Table 31 compares the small instances of size 16-31. The numbers with one commodity without visit windows is the numbers from Gribkovskaia et al. (2007) and is the baseline for the gaps. To get a better picture of the effects from adding commodities a short run were made for these instances with only one commodity using the Tabu-algorithm presented in this thesis. A run was also made where the instances had two installations with visit windows instead of only one as is the case in the rest of the thesis.

Looking first at one commodity the gap is on average 3.9% when one of the installations in each instance has a visit window, and when two installations have a visit

window the gap decreases to 2.8%. For the baseline only 14.3% of the instances had a non-Hamiltonian shape, while with one and two instances with visit windows the percentage of non-Hamiltonian shaped solutions were 64.3% and 71.4%.

For two commodities the gap without visit windows i 1.7% and the amount percentage of non-Hamiltonian shaped solutions is 71.4%. When adding one visit window the gap increases to 5.9% and the non-Hamiltonian shaped solutions increase to 78.6%. This increase continues when adding the second visit window and the cost gap is 7.8%. For two commodities and two visit windows there is not a single Hamiltonian shaped solution.

When adding the third commodity there seems to be an overall decrease in gap compared to with two commodities, but still an increase compared to only one commodity. Without visit windows the gap is 1.4% and the percentage of non-Hamiltonian shaped solutions is 64.3%. For one visit window the gap is 5.5% and the percentage of non-Hamiltonian shaped solutions is 85.7%. For two visit windows these numbers are 6.6% and 92.9%.

| Instance | 1c Cost | 2c % Gap | 3c % Gap | 2c vw % Gap | 3c vw % Gap |
|---|---|---|---|---|---|
| 041b20 | 362,02 | 4,0 | 4,4 | 5,8 | 5,9 |
| 045b20 | 619,09 | 0,6 | 0,4 | 6,6 | 4,9 |
| 048b20 | 33523,71 | 2,3 | 2,3 | 3,1 | 7,2 |
| 051b20 | 433,6 | 2,2 | 2,7 | 2,7 | 2,7 |
| 072b20 | 202,82 | 4,7 | 4,7 | 26,8 | 8,3 |
| 076b20 | 555,07 | 2,6 | 2,1 | 1,8 | 2,0 |
| 101b20 | 653,96 | 2,2 | 1,9 | 8,7 | 5,5 |
| Avg | 5192,9 | 2,7 | 2,6 | 7,9 | 5,2 |

Table 32: Gap between the best found solutions from the literature and the solutions to the corresponding instances from this thesis for instance size from 41 to 101. 100% utilization only

Table 32 shows the gap between the solution created by the Tabu-algorithm in this thesis and the best found solution in Gribkovskaia et al. (2007) for the large instances ranging from 41 to 101 in size. Once again a decrease in the gap is seen when going from two to three commodities, just like in Table 31. For two commodities the gap is 7.9% with visit windows and 2.7% without, while for 3 commodities the numbers are 5.2% and 2.6%.

# 7 Conclusions and Further Research

Two problems were examined in this thesis. The first was the Single Vehicle Pickup and Delivery Problem with Multiple Commodities. The second problem was an extension of the first, adding Visit Windows and Route Duration limits. The objective is to find a least cost vessel route starting and ending at the offshore base, visiting all platforms that have demands, while at the same time the vessel capacities must not be exceeded at any time. In addition, for the second problem, the Visit Windows must not be violated and the route duration must not exceed its maximum limit. General solutions are allowed, that is installations may be visited once or twice. If an installation is visited once, all pickups and deliveries are performed simultaneously. If an installation is visited twice, all the deliveries are performed on the first visit and pickups may be performed. On the second visit pickups are performed.

The importance of generating the solution tools for these problems is derived by the practical applications the problems have in supply of oil and gas installations in the North Sea. The installations are supplied by supply vessels that are located at an offshore base. The routes for these supply vessels are currently created manually by a route planner. In this thesis a single vessel is considered, since the route planning is done for one vessel at the time. When adding multiple commodities and visit windows into the planning process it becomes a time consuming task. A heuristic can solve such problems in a relatively short time compared to what a manual planning process will. The heuristic will give the planner feasible solutions and the planner can then choose the ones that are least costly.

This thesis had several goals being:

- To formulate mathematical models for the One-to-Many-to-One Single Vehicle Pickup and Delivery problem with Multiple Commodities and with Visit Windows.
- To create a Tabu-search algorithm able to produce general solutions to instances of any size.
- To evaluate the performance of the Tabu-algorithm.

- To run tests using the Tabu-algorithm to see how the solution shape, cost and CPU time is affected when we change the problem characteristics in the following way:

  - Increase in number of commodities.

  - Addition of visit windows and constraint on the route duration.

  - Increase in capacity utilization for the vessel.

The results:

- The one-to-many-to-one Single Vehicle Pickup and Delivery Problem with Multiple Commodities and Visit Windows were formulated as MIP mathematical models. These models contain load controlling constraints that prevent generation of subtours.

- For computational testing 1544 instances were generated.

  - 2 groups of small instances, 40 instances in each group

  - 1 group of large instances, 13 instances.

  - Number of commodities : 2 and 3

  - Vessel capacity generated for 100%, 95%, 90% and 80% utilization.

  - In addition, all small instances were run with mathematical models with the constraint of allowing only Hamiltonian solutions.

  - Instances from all 3 groups were run without visit windows and with 1 platform having a visit window.

  - In addition to these 3 groups, 14 instances of size 16 to 31 were selected. These instances were run with 1 commodity with 1 visit window with 100% utilization for vessel capacity.

  - These 14 instances were also run with 2 visit windows for commodities 1, 2 and 3 with 100% utilization for vessel capacity.

- The mathematical models found the optimal solution for 49.5% of these instances within one hour. The maximal size of the instance solved to optimality was 31.

- Average CPU time for Tabu:

  - small instances without visit windows: 23.23 seconds

  - small instances with visit windows: 139.30 seconds

  - large instances without visit windows: 6439.12 seconds

  - large instances with visit windows: These instances had a two hour limit and the average is therefore omitted.

- The Tabu-algorithm ran for 100 000 iterations found the optimal solution for 38.5% of all small instances.

- The Tabu-algorithm has shown that within 26 seconds it is able to produce optimal solutions for instances up to the size of 23 nodes.

- 35.7% solutions created by Tabu algorithm are general where an instance has one or more installations requiring two visits.

- The Tabu-algorithm has an average decrease of 4.3% in cost compared to the CPLEX solutions found within one hour.

- For some instances while the mathematical model for one hour was unable to find any feasible solutions. The Tabu-algorithm was able to find a feasible solution in few minutes.

- There is almost no change in solutions cost when levels of vessel capacity utilization is up to 95%.

- Adding Visit Windows leads to increased cost of the solutions when compared to instances without Visit Windows. The solution cost in average increases by 3.1% with 100% utilization.

- There is a 20% increase in the number of solutions where one or more installations require two visits when visit windows are added for one installation per instance.

- When adding a second visit window, the number of general solutions increase further by 16%.

- Between 80%, 90% and 95% utilization there is almost no change in the number of solutions with installations requiring two visits.
- There is a 413% difference in the number of solutions with installations requiring two visits when changing the level of utilization of the vessel capacity from 95% to 100%.
- When the instances become larger, the amount of installations requiring two visits in the solution is increasing.
- There is a 16.9% increase in CPU-time shown when adding a commodity.
- When adding visit windows a 598,4% increase in CPU time to solve a problem occurs.
- There is a decrease in the number of Hamiltonian shaped solutions as the level of utilization increases.

| % Util | non-VW | VW |
|---|---|---|
| 80 | 92,5 | 80,2 |
| 90 | 89,6 | 80,2 |
| 95 | 86,8 | 75,5 |
| 100 | 24,5 | 18,9 |

- There is a 9% decrease in the number of Hamiltonian-shaped solutions when increasing the number of commodities.
- The solutions created by the Tabu-algorithm show the opposite, namely a decrease in Hamiltonian shapes when adding visit windows.

Possible additional expansions and tests for this heuristic:

- Capacitated platforms

  If the platforms have only limited space available, the problem becomes more complex. The pickups and deliveries might then have to be split in order to make feasible solutions. When there is low capacity on the platform, loading of the platform might take longer time since available space must be found. There might already be reserved space, but care must be taken to not crush anything when using the crane to load containers onto the platform deck.

- Transshipments

  - If a platform wants to send some commodities over to another platform, this could be considered if there is enough room on the supply vessel.

  - Also, if a platform has a lot of free space, the supply vessel could use this as an intermediary for storing containers, while the vessel is servicing other platforms. This could be considered if a route is scheduled to be near the capacity limits later in the route. This might lead to less time of handling and placing the containers in the middle of the route when the vessel was supposed to be almost full. However, it should only be considered if the platform in question is close to the path where the vessel will sail back to the offshore base or close to another platform which is scheduled for a visit.

- Include more than one visit window per day per vertex or different windows per day

  If platforms are scheduled for maintenance on a regular basis, there could be more than one visit window associated to those platforms.

- Multiple Visits

  When adding more commodities it could be interesting to see if multiple visits would give a better solution than with only one or two visits. In order to do this the demands for different commodities must be split. An example of three visits could be to deliver all commodities at the first visit, then return later for picking up another type of commodity and return at an even later point in the route to pickup the last type of commodity.

# References

Chen, J. F.; Wu, T.H. (2005). *Vehicle routing problem with simultaneous deliveries and pickups*. Journal of the Operational Research Society (2006) 57: 579-587.

Chiang; Russel (2004). *A metaheuristic for the vehicle-routing problem with soft time windows*. Journal of the Operational Research Society 55: 1298-1310.

Choo; Poh; Wong (2005). *A heuristic approach to the multi-period multi-commodity transportation problem*. Journal of the Operational Research Society (2005) 56: 708-718.

Clarke, G.; Wright, J. (1964). *Scheduling of Vehicles from a Central Depot to a Number of Delivery Points*. Operations. Research 12, 568-581.

Cordeau; Laporte (2001). *A tabu search algorithm for the site dependent vehicle routing problem with time windows*. INFOR 39: 292–298.

Cordeau; Desaulniers; Desrosiers; Solomon; Soumis (2000). *The VRP with Time Windows*. Les Cahiers du GERAD, G-99-13.

Cordeau, Jean-Francois; Laporte, Gilbert; Mercier, Anne (2001). *A Unified Tabu Search Heuristic for Vehicle Routing Problems with Time Windows*. Journal of the Operational Research Society 52: 928-936 .

Cordeau, Jean-Francois; Laporte, Gilbert; Mercier, Anne (2004). *Improved Tabu Search Algorithm for the Handling of Route Duration Constraints in Vehicle Routing Problems with Time Windows*. Journal of the Operational Research Society 55: 542-546.

Desaulniers, Guy; Desrosiers, Jacques; Solomon, Marius M. (2005). Chapter 3: Vehicle Routing Problem with Time Windows. Springer.

Doerner, Karl F.; Gronalt, Manfred; Hartl, Richard F.; Kiechle, Guenter; Reimann, Marc (2008). *Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows*. Science Direct, Computers & Operations Research 35: 3034-3048

Dorigo, M. (1992). *Optimization, learning and natural algorithms (in Italian)*. Ph.D. Thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy.

Gendrau, Michel (2003). *Chapter 2: An Introduction to Tabu Search.* In F. Glover and G.A. Kochenberger (eds.), "Handbook of Metaheuristics". Kluwer Academic Publishers.

Gillett, B.; Miller, L. (1974). *A heuristic algorithm for the vehicle dispatch problem*. Operations Research, 22, 340–349.

Glover, F. (1986). *Future Paths for Integer Programming and Links to Artificial Intelligence*. Computers and Operations Research 13: 533-549.

Gribkovskaia I.; Halskau Ø.; Laporte, G.; Vlček, M. (2007). *General Solutions to the Single Vehicle Routing Problem with Pickups and Deliveries*. European Journal of Operational Research 180: 568-584.

Gribkovskaia, I.; Halskau, Ø.; Aas, B. (2005). Routing of supply vessels serving oil and gas installations in the Norwegian Sea. Working paper, Molde University College.

Gribkovskaia, Irina; Laporte, Gilbert (2007). *One-to-Many-to-One Single Vehicle Pickup and Delivery Problems*. Les Cahiers du GERAD G–2007–04.

Gribkovskaia, I; Laporte, G.; Shlopak, A. (2007). *A Tabu Search Heuristic for a Routing Problem Arising in Servicing of Offshore Oil and Gas Platforms*. Journal of the Operational Research Society p 1-11 .

Hoff, A.; Gribkovskaia, I.; Laporte G.; Løkketangen, A. (2007). *Lasso Solution Strategies for the Vehicle Routing Problem with Pickups and Deliveries*. European Journal of Operational Research, doi:10.1016/j.ejor.2007.10.021.

Kammarti, R.; Hammadi, S. Borne, P.; Ksouri M. (2004). *A New Hybrid Evolutionary Approach For The Pickup And Delivery Problem With Time Windows*. 2004 IEEE International Conference on Systems, Man and Cybernetics.

Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. (1983). *Optimization by Simulated Annealing*. Science, Vol 220, Number 4598, pages 671-680.

Landrieu, Antoine; Mati, Yazid; Binder, Zdenek (2001). *A Tabu Search Heuristic for the Single Vehicle Pickup and Delivery Problem with Time Windows*. Journal of Intelligent Manufacturing 12: 497-508.

Lu, Quan; Dessouky, Maged M. (2006). *A New Insertion-Based Construction Heuristic for Solving the Pickup and Delivery Problem with Time Windows*. European Journal of Operational Research 175: 672-687.

Mitrović-Minić (1998). *Pickup and Delivery Problem with Time Windows: A Survey*. SFU CMPT TR 1998-12.

Montané, Fermín Alfredo Tang; Galvão, Roberto Diéguez (2006). *A Tabu Search Algorithm for the Vehicle Routing Problem with Simultaneous Pick-Up and Delivery Service*. Computers and Operations Research 33: 595-619.

Ropke,Stefan; Pisinger, David (2006). *A Unified Heuristic for a Large Class of Vehicle Routing Problems with Backhauls*. European Journal of Operational Research 171: 750-775.

Tam, Vincent; Tseng, Lois C.Y. (2003). *Effective Heuristics to Solve Pickup and Delivery Problems with Time Windows*. Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'03).

Toth, Paolo; Vigo, Daniele, eds. (2002). The Vehicle Routin Problem. Philadelphia: Society for Industrial and Applied Mathematics.

Zouari, Ahmed; Akselvoll, Arne Borch (2007). *Multi-Commodity Pickup and Delivery Supply Vessel Routing for Statoil: Analysis and Modelling*. Master's Thesis, Molde University College.