MAGISTERARBEIT

Titel der Magisterarbeit

„Heuristics for the Free Newspaper Delivery Problem“

Verfasserin

Sabine Krenbek Bakk.

angestrebter akademischer Grad

Magistra rerum socialium oeconomicarumque (Mag. rer. soc. oec.)

Wien, 2009

Studienkennzahl lt. Studienblatt:	A 066 915
Studienrichtung lt. Studienblatt:	Magisterstudium Betriebswirtschaft
Betreuer:	Univ.-Doz. Dr. Karl Dörner
Mitwirkender Assistent:	Dr. Fabien Tricoire

1 The Free Newspaper Delivery Problem

The spreading of free newspapers increased rapidly during the last 10 years. Free newspapers were introduced in many countries world wide. In 2008 the free newspapers are distributed in 58 countries.[1] The requirements of one of the companies are analysed and taken as example for the master thesis. The aim of this document is to design and analyse a suitable optimisation method for the given problem. We propose a heuristic solution approach consisting of a construction heuristic and an improvement heuristic.

1.1 Problem description

In order to distribute free newspapers to underground and tramway stations the delivery company needs to build vehicle routes. The vehicle visits several stations (nodes) on its routes which start and end at the depot. A vehicle has limited capacity, but it is allowed that more than one trip is performed by it. The capacity and the consumption rate for each station is given, whereas it should be taken into account that the inventory level at a station should never exceed capacity of the available boxes or become zero. Secondary to the consumption at each station, there is production at the depot while the newspaper delivery takes place. The production rate is given. It is a daily changing problem as the thickness of newspapers is different every day and therefore the vehicle capacity and the capacity at the stations need to be calculated every day.

The goal of the company is

1. to consume all produced newspapers by distributing them to the stations, where they are actually consumed, and
2. to fulfil the distribution by using as few vehicles as possible, and as few trips as possible.

A more detailed overview is given as followed:

- ***Fleet***

  The fleet is homogenous. Each vehicle performs only one route at a time, but a route can consist of several trips.

---

[1] [WIK01]

- ***Horizon***

  The horizon starts at 4:00 a.m. and ends at 9:00 a.m. This time frame is splitted into periods. Consumption takes place only after 5:00 a.m. and ends at 9:00 a.m. Each station needs to be visited before that point of time to avoid a loss of consumption. Latest time for delivery is 8:30 as it is required to deliver before consumption takes place.

- ***Information on the depot***

  Production at the depot takes place between 1:00 and 7:00. The production rate of 40.000 newspapers per hour is fixed. The produced newspapers become available only at the end of each hour. This means that the first newspapers are available only at 2:00. At 9:00 the inventory level at the depot needs to be zero. In the current environment delivery starts at 4:00 with the quantity of 120.000 available newspapers which are produced from 1:00 to 4:00.

- ***Information on the station***

  The newspapers are stored in boxes at the stations. There are four kinds of boxes, with capacities of 2, 4, 8 or 10 batches. The total capacity at the station depends on the sizes and number of boxes that are placed there. There are some constraints that hold for all stations:

  - All stations should be visited before 5:00
  - The beginning time window is different for every station
  - The consumption rate for a period and a station is fixed. It can differ from station to station.
  - A stock-out should be avoided before 9:00 and it should occur at 9:00.

1.2 Literature Review

The described Free Newspaper Delivery Problem is similar to an inventory routing problem (IRP) with some differences, namely

- a stock-out at the supplier and the customer at end of the planning horizon is required
- not all products are available when delivery starts
- routes span over several periods

The route building for various periods is considered in the Periodic Vehicle Routing Problem (PVRP). This was the reason for basing literature review for the solution approach on the known methods for the IRP and PVRP.

1.2.1 Vendor managed Inventory Routing Problem[2]

The vendor managed inventory-routing problem (IRP) combines the components inventory control and vehicle routing. It is a variation of the vehicle routing problem (VRP) which is described as daily problem where customers place orders and the delivery company assigns the requested amounts to routes. The objective is to minimise the total distance travelled.

The inventory routing problem (IRP) involves repeated distribution of a single product from a single facility, where the supplier manages the inventory levels at the customer. Two contradictory objectives need to be balanced. Stock-outs at customer sites should be avoided and distribution cost are minimised. The difference to VRP is that IRPs are based on usage rates instead of customer orders. It is obvious that these problems are more complicated than VRP as storage quantities needs to be taken into account on both sides, the supplier and customer sites, when determining the delivery quantities. Inventory cost can occur at the supplier and at the customer location. Its amount can vary from situation to situation and be zero for one or more locations. Beneath the inventory consideration there are some other characteristics that can be described for an IRP:

- Finite or infinite planning horizon
- Deterministic or stochastic production and consumption rates
- Constant or varying production and consumption rates over time
- Production and consumption take place at discrete time instants continuously
- Choice of the optimal delivery policy
- Are all possible policies allowed when choosing the optimal delivery policy or only specific classes, e. g. fill to the maximum

---

[2] [GOL01] p. 49 ff

Inventory routing problems are defined on a graph $G = \{V, E\}$, where $V=\{S, 1, \ldots, n\}$ is the set of vertices and E is the set of edges. Vertex S represents the supplier (depot) and vertices 1, …, n represent the customers. For each edge (i, j) in E the travel time t_{ij} and the cost c_{ij} are given. The distribution of products will be performed by a fleet of m vehicles with the capacity of Q. In case of discrete time, the usage or consumption rate is symbolized by q_i as the quantity of product consumed per unit of time by customer i. In case of continuous time, the consumption or usage rate is represented by u_i. The initial inventory level is denoted by I^0_S for the supplier and I^0_i for each customer i. According to this notation the inventory levels at time t are I^t_S and I^t_i. The inventory holding capacity are represented by C_S and C_i for both supplier and customers, with the inventory cost h_S and h_i. The parameter H shows the length of the planning horizon.

These decisions need to be made:

- When to visit each customer?
- What is the delivery quantity for each customer for each visit?
- Which delivery routes are used?

In Golden (2008)[3] a literature study of inventory routing problem is given. IRP are classified in inventory routing problems with deterministic product usage, inventory routing problems with stochastic product usage and problems that can not be assigned into one of these categories. One of the approaches that is mentioned in this overview is the Decomposition Approach by Campbell and Savelsbergh (2004).

1.2.2 Period Vehicle Routing Problem (PVRP)[4]

The Period Vehicle Routing Problem is a generalization of the classical vehicle routing problem, where routes are established for more periods of time. The PVRP is defined as a graph G = (N, A), where N represents the set of nodes including all customers and the depot. Each arc belonging to the set of arcs A has assigned cost $c_{i,j}$ for all arcs (i,j). For the planning horizon D a schedule is established to determine which customer is visited

[3] [GOL01] p. 66ff
[4] [GOL01] p. 73 ff

on each day. The objective of the PVRP is finding routes to minimize the total travelling cost while satisfying the other constraints like vehicle capacity and visit requirements.

For solving a PVRP three decisions needs to be taken:

- Select a schedule for each node out of the known candidates
- Assign the nodes that should be visited on each day to vehicles
- Built routes for each vehicle for each day.

The evolution of models and solution methods for the PVRP is shown in Figure 1. The problem was presented by Beltram and Bodin (1974)[5] for describing the assignment for hoist compactor trucks in municipal waste collection. The articles of Russels and Igo (1979)[6] and Christofides and Beasley (1984)[7] provide formal definitions of the PVRP as mixed integer problems.

Solution methods for the PRVP are ranging from Classical heuristics, Metaheuristics and Mathematical programming based methods which are shown in the picture below.

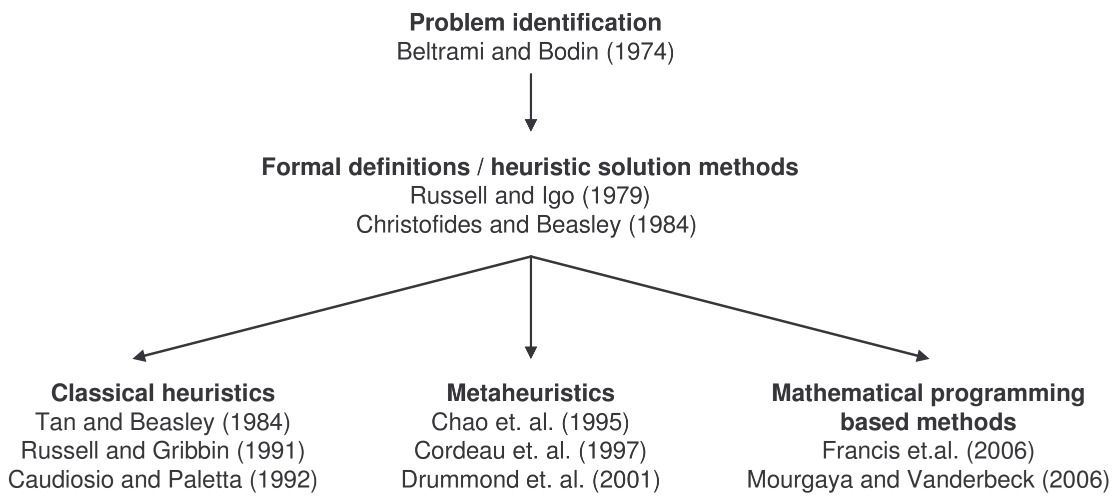

Figure 1: Evolution of models and solution methods for the PVRP[8]

[5] [BEL01]
[6] [RUS01]
[7] [CHR01]
[8] cf. [GOL01] p. 76

An example where methods out of these three categories were applied to a variant of the PVRP is the article "Delivery strategies for blood products supplies" by Hemmelmayr (2009) et. al.[9]. The methods used are the Clark and Wrights Savings algorithm, as well as the 3-opt-algorithm and the VNS.

1.2.3 A Decomposition Approach for the Inventory-Routing Problem[10]

The Decomposition Approach of Campbell and Savelsbergh is based on the idea that you split the solution process into two phases. In the first phase a longer time horizon of k-days is examined and the second is used for planning a shorter j-days horizon. It is assumed that only the short-term plan is finally executed.

In the first phase "Planning" customers and corresponding amounts for delivery are assigned to days. For this phase an IP-Model was solved in Campbell and Savelsbergh (2004) [CAM01]. The objective function of this model was the minimisation of the total cost. Upper bounds and lower bounds for customer demands are given. In this planning step it is taken into account that amounts identified for customer delivery on a specific day does not exceed the vehicle capacity and time required for the route is not higher than the time available.

In addition possible extensions to this basic model were presented:

- ***Fixed and Variable Stop Times:*** Fixed time stops are time amounts that need to be considered when deliveries take place at a customer. It can be the same for all customers or differ for each customer. The time represented by variable time stops depend on the amount of units delivered.

- ***Operating Modes***: are blocks of different usage rates that are assigned to different days.

- ***Time Windows***: for delivery are assigned to each of the customers

[9] [HEM02]
[10] [CAM01]

- ***Driver Availability***: The availability of drivers can vary for the various days, this impacts the route planning.

- ***Driver Restrictions***: Legal restrictions concerning the working time of the drivers might be needed to be considered.

- ***Order-Only Customers***: customers where the inventory is not managed by the supplier. These customer place single orders, as known from the classical VRP.

- ***Multiple Deliveries per Day***: In some cases customers will need more than one delivery a day.

1.2.4 Creation of delivery plans[11]

In Campbell and Savelsbergh (2004) [CAM02] some heuristics for the creation of delivery plans were explained and compared:

- ***Early Method***: Delivery to customer is planned as early as possible, the scheduling decision regarding the route for preceding customers are taken into account and delivery amount is the maximum at the time of delivery.

- ***Late Method***: Delivery to customer is planned as late as possible, the scheduling decision regarding the route for preceding customers are taken into account and delivery amount is the maximum at the time of delivery.

- ***Greedy Method***: The maximum possible is delivered to every customer while considering scheduling decisions at the preceding customers on the route.

- ***Maximum Usage Method***: Here the maximum amount possible is delivered to the customer with the highest usage rate among the unrouted customers.

---

[11] [CAM02]

1.2.5 Vehicle Routing Problem with Time Window[12]

The second phase of the Decomposition Approach which is used for solving the Free Newspaper Problem is the vehicle routing problem with time windows (VRPTW). The VRPTW is described below.

Let G = (V, E) be a connected digraph. V is a set of nodes consisting of all customer nodes and the depot. Each node can be serviced within a specified time window, if the vehicle arrives before the time window starts, it needs to wait at no cost until the appropriate time. The time window is defined through the earliest time e_i and the latest time l_i. b_i is the time when delivery begins. E is a set of arcs, where c_{ij} are the cost for that arc and t_{ij} its associated travel time. The service time s_i at each node is included in the travel time. Each customer has a demand of q_i, which needs to be satisfied. A set of identical vehicles with capacity Q is used to build routes and visit the customers. Each customer needs to be visited once and its service needs to start within its time window. One route starts and ends at the depot. The objectives are minimizing the number of routes, the distance travelled or the duration of the routes.

Construction Heuristics for VRPTW[13]

Route construction heuristics are used to build a feasible solution for the VRPTW. Some of them select one unrouted node after the other, based on some cost minimisation criterion, until all nodes are routed and a feasible solution has been created. These algorithms often need to consider restrictions of vehicle capacity and time window constraints which must not be violated. Sequential methods construct one route at a time and parallel methods allow building more routes at the same time.

In Bräysy and Gendreau (2005) [BRA03] some construction heuristics for VRPTW are explained. Some of them are summarised in the following overview.

---

[12] [BRA03]
[13] [BRA03]

- ***Solomon (1987)***[14] describes some heuristics for the VRPTW.

<u>*Savings Heuristic:*</u>

Solomon describes an extension to the savings heuristic of Clarke and Wright (1964). At the beginning the supply for each customer takes place in a separate route. If two of those routes are combined so that two customers i and j are served in one route, a saving of $S_{ij} = d_{i0} + d_{j0} - d_{ij}$ is gained. In the Savings-algorithm of Clark and Wright the arc (i ,j) linking customers i and j with the maximum S_{ij} is chosen. Two routes are connected on that arc, if this solution is feasible. This step is repeated until no more saving can be performed. In addition to that basic algorithm a waiting time limit is set for a route to ensure spatial and temporal closeness of customers. An example for the savings methods is given in Figure 2

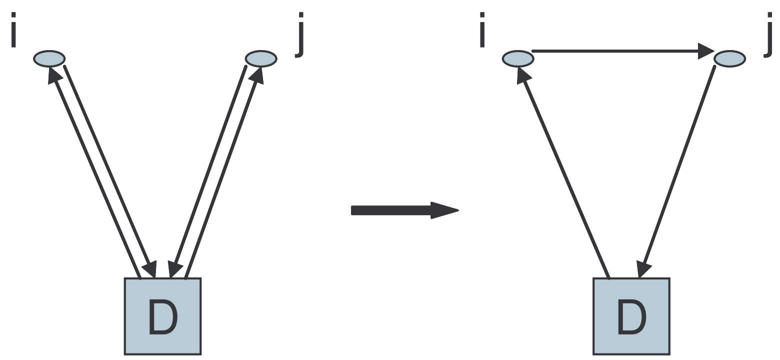

Figure 2: The savings heuristic

<u>*Time-oriented nearest neighbour*</u>*:*

A route is started by adding the unrouted customer closest to the depot. The following iterations append the customer that is closest to the last inserted point into the route. This step is repeated as long a customer can be found that can be added without violating the constraints. A new route is inserted when the search for a feasible customer fails or no more unrouted customers are left. The closest customer is defined by the following formulas:

[14] [SOL01]

$T_{ij} = b_j - (b_i + s_i)$

$v_{ij} = l_j - (b_i + s_i + t_{ij})$

$c_{ij} = \delta_1 d_{ij} + \delta_2 T_{ij} + \delta_3 v_{ij}$,

the weights needed in that formula are defined as

$$\delta_1 + \delta_2 + \delta_3 = 1, \delta_1 > 0, \delta_2 > 0, \delta_3 > 0.$$

Insertion Heuristic:

Solomon explained three insertion heuristics, but in this document only the I1 is presented as it is the best out of the three.

First each route is initialised by building this route depot - the route that is unrouted and farthest away from the depot – depot. Then the customers are assigned to the route step-by-step. In each iteration one unrouted customer is assigned by performing these two steps.

1. Compute best feasible insertion place for each unrouted customer on the current route that is described by $(i_o, i_l, i_2, . .., i_m)$. This position is described by

   $$c_l(i(u), u, j(u)) = \min[c_1(i_{p-1}, u, i_p)], p= 1, ..., m$$

   Where c_l is calculated by those formulas:

   $c_{11}(i,u,j) = d_{iu} + d_{uj} - \mu d_{ij}$,

   $$\mu > 0$$

   $c_{12}(i, u, j) = b_{j_u}, - b_j$

   $c_1(i, u, j) = \alpha_1 c_l(i, u, j) + \alpha_2 c_{12}(i, u, j)$

   $$\alpha_1 + \alpha_2 = 1, \alpha_1 \geq 0, \alpha_2 \geq 0$$

2. Find best customer u* to insert into the tour between i(u*) and j(u*), the selection criterion is described by

   $c_2(i(u^*), u^*,j(u^*)) = \text{optimum}[c_2(i(u), u.j(u))]$

   where u is unrouted and feasible.

   c_2 defined as

   $c_2 (i, u, j) = \lambda d_{Ou} - c_1(i, u, j)$

   $$\lambda \geq 0.$$

- ***Potvin and Rousseau (1993)***[15] describe a parallel version of Solomon's insertion heuristic I1. Instead of building only one route at a time, m routes were initialised. The characteristics for choosing the seed customers for the routes are the same as used by Solomon. A regret measure over all routes is considered during the selection of the next customer to be inserted.

- ***Ioannou et al. (2001)***[16] uses also ideas of the generic sequential insertion framework by Solomon. This method adds new criteria for customer selection and insertion, which are inspired by the greedy look-ahead solution approach of Atkinson(1994)[17].

Improvement Heuristics for the VRPTW[18]

In Bräysy and Gendreau (2005) [BRA03] and [BRA04] improvement heuristics and meta-heuristics for the VRPTW are described. Two of these methods are OR-Opt and VNS which are described here in more details.

1.2.5.1 Route Improvement Heuristic: OR-opt[19]

The following generic algorithm is usually used to change a solution with the aim of getting a new improved one:

Step 1. Generate an initial feasible solution.
Step 2. Modify the current solution to get a new improved feasible solution.
Step 3. Repeat Step 2 until no more improvement is possible (i.e. a local optimum has been reached).[20]

A well known approach for the second step of this algorithm is the k-opt of Lin[21], where k edges of one tour are replaced with k new edges. A tour is *k-optimal* (k-opt) if it is

[15] [POT02]
[16] [IAO01]
[17] [ATK01]
[18] [BRA02]
[19] [POT01], [SAV01]
[20] Cf [POT01]
[21] [LIN01]

impossible to find a shorter tour by performing a k-interchange. Examples for k=2 and k=3 will be provided to demonstrate the way k-opt works.

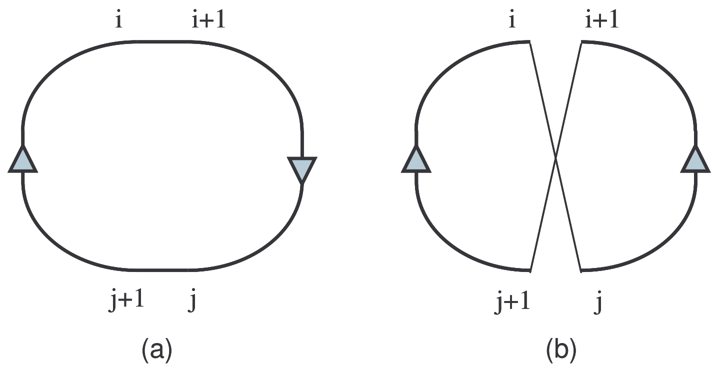

Figure 3: A 2-interchange[22]

A 2-interchange is performed by deleting the arcs between (i, i+1) and (j, j+1) shown in "Figure 3 – (a)" and replacing them by adding the arcs (i, j) and (i+1, j+1). The route between j and i+1 is reversed ("Figure 3 – (b)"). An improvement can be achieved, if route costs of tour (b) are lower than those of tour (a). When N is the number of nodes in the route, $\binom{N}{2}$ steps are needed for verifying 2-optimallity of it and a time complexity of $O(N^2)$ for finding this solution.

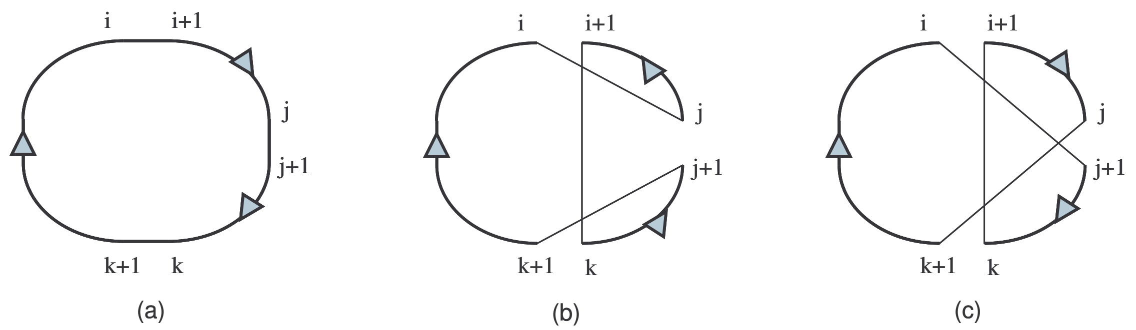

Figure 4: Two ways to perform a 3-interchange[23]

As opposed to the case k=2, there are eight ways of replacing the arcs if k=3. "Figure 4" gives an example of two ways performing such a move. In route (b) the edges (i, i+1), (j, j+1) and (k, k+1) are replaced by (i, j), (k, i+1) and (j+1, k+1), whereas the route between j and i+1 is inverted as well as the route between k and j+1. The route

[22] [SAV01]
[23] [SAV01]

represented by (c) consists of the new arcs (i, j+1), (j, k+1) and (k, i+1). $\binom{N}{3}$ steps are needed to determine the 3-optimallity of the route and a time complexity of $O(N^3)$ for verification.

For VRPTW the feasibility of the route needs to be checked after moving to an improved solution. This feasibility check has the complexity of O(N). In addition to this check often infeasible solutions are found due to the inverting of the route or due to the time windows that need to be considered at each node. A method that simplifies 3-opt and gives also good results is OR-opt.

The OR-opt procedure considers only those 3-interchanges that insert one, two or three consecutive vertices between two other vertices. A further advantage for VRPTW is that it is easier to verify feasibility. As only a subset of the neighbourhoods of 3-opt is examined also the complexity is lower. The time complexity to verify OR-opt is $O(N^2)$.

The example in the figure below shows one or-opt-move, where the sequence of customers (1, 2) is inserted between the customers 3 and 4.

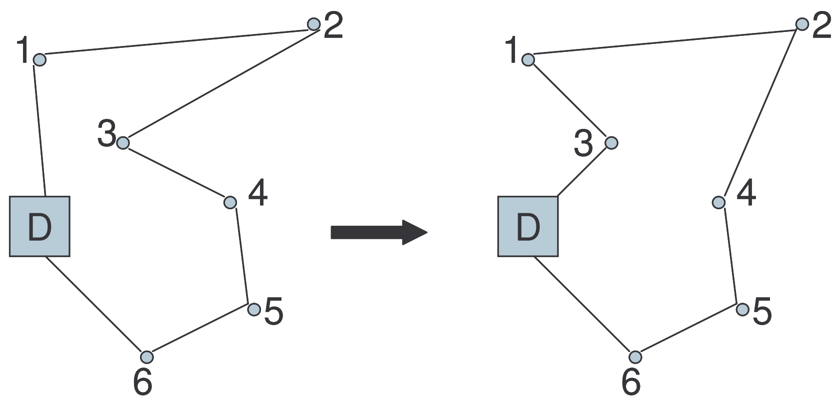

Figure 5: Moving the sequence of customers (1, 2) between customers 3 and 4[24]

[24] [POT01]

1.2.5.2 Variable Neighbourhood Search (VNS)

VNS is a recent metaheuristic for solving combinatorial and global optimization problems. This method was presented by Mladenovic and Hansen (1997). They suggest a method where a systematic change of neighbourhoods within a local search is performed. "Contrary to most other local search methods VNS does not follow a trajectory, but explores increasingly distant neighbourhoods of the current incumbent solution, and jumps from there to a new one if and only if an improvement was made. In this way often favorable characteristics of the incumbent solution, e.g. that most variables are already at their optimal value, will be kept and used to obtain promising neighboring solutions. Moreover, a local search routine is applied repeatedly to get from these neighboring solutions to local optima."[25]

[25] [MLA01]

2 Solution methods for the free newspaper delivery problem

The free newspaper delivery problem is defined on a graph $G = \{V, E\}$, where $V=\{S, C\}$ is the set of vertices and E is the set of edges. Vertex S represents the supplier (depot) and vertices C representing the customers 1, …, n. For each edge (i, j) element of the set of edges E the travel time t_{ij} is known.

The set of delivery periods τ consists of H periods where H shows the length of the planning horizon. The distribution of products will be performed by a fleet of vehicles V. The vehicle capacity is limited by Q. The consumption rate per customer and period is denoted by d_{it}.

The initial inventory level at the production plant is denoted by $\overline{B}$. Its final inventory level B_T is zero since all newspapers should be delivered to a station. According to this notation the inventory levels at time t are B_t. During the planning horizon production takes place at the plant with a production rate p.

The initial inventory level I_{0i} and the final inventory level I_{Hi} for each node i are zero. According to this notation the inventory levels at time t are I_{ti}. The inventory level at each customer is limited by K_i.

2.1 A Decomposition Approach for the free newspaper delivery problem

The implemented algorithm is based on the idea of the decomposition approach of Campell and Savelsbergh (2004)[26].

In the first phase "Planning" it is determined which customer will be visited with which quantity in a period. These periods are not days but rather smaller periods of 30 to 60 minutes each. The second phase "Scheduling" where the creation of the tour plan takes

[26] [CAM01]

place, is implemented as insertion heuristic described by Solomon (2004)[27], in a slightly modified version since at the beginning of the planning horizon not all newspapers are available. This problem is more difficult to solve than a VRPTW, because not all newspapers are available when the first vehicles leave the depot. An additional restriction considering the production capacity needs to be considered as vehicles can only leave the depot when sufficient newspapers are available to load it. The concept of Stepwise Releasing is introduced to take this into account, which is placed between Phase 1 and Phase 2.

The delivery plan created in Phase 1 is converted in a Vehicle Routing Problem with time windows. For each delivery planned at a station a virtual customer is created in a list. These virtual customers are described by

- the identification number of the station,
- the demand for that delivery,
- the period, where delivery takes place
- start and end of the time window for the planned delivery – the following table show the values that are used for each periods.

Period	Starttime	Endtime
1	0	60
2	60	90
3	90	120
4	120	150

Period	Starttime	Endtime
5	150	180
6	180	210
7	210	240
8	240	270

- the planned delivery time

If a station needs to be visited in the periods zero, two and five - three virtual customers are inserted into the set of customers (Delivery Customer List).

Release Cycles are established to ensure that nodes are assigned to a route only if enough newspapers are available at a depot when the vehicle leaves. A Release Cycle is a subset of all existing virtual customers at the beginning of a specific period, where nodes are considered only if they can be delivered with the capacity that has been produced so far at the depot.

[27] [SOL01]

A tourplan created without this consideration could lead to an infeasible solution. Routes that start in period zero and deliver customers until the last period (seven) are allowed. This means that eventually the number of newspapers needed to load the trucks in period zero may exceed the amount available. The following table shows the total sum of all newspapers considered for the planned visits during a period in the line "demand of newspapers (total)". As opposed to the amount planned for customer delivery, the newspapers that are available at the beginning of a period are shown.

Station Number	Period 0 4:00-5:00	Period 1 5:00-05:30	Period 2 05:30-6:00	Period 3 6:00-6:30	Period 4 6:30-7:00	Period 5 7:00-7:30	Period 6 7:30-8:00	Period 7 8:00-8:30
0	90	0	0	0	0	0	0	0
1	810	0	0	0	0	0	0	0
2	225	0	0	150	0	125	0	0
3	720	0	0	480	0	400	0	0
4	45	0	0	0	0	0	0	0
5	270	0	0	180	0	150	0	0
6	135	0	0	0	0	0	0	0
7	315	0	0	210	0	175	0	0
8	1305	0	0	870	0	725	0	0
...	...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	...	...
208	150	0	0	0	0	0	0	0
209	650	0	0	520	0	0	130	0
210	666	0	666	0	0	668	0	0
211	150	0	0	0	0	0	0	0
demand of newspapers (total)	*120000*	*0*	*28483*	*51517*	*0*	*37085*	*2915*	*0*
release of new newspapers	*120000*	*40000*	*0*	*40000*	*0*	*40000*	*0*	*0*

Figure 6: Comparison demand and supply of newspapers for each period

In a period where new newspapers are available, a list of virtual customers is created. Stations are considered, if visits are planned for that period and the successor periods where no newspapers are disposed. For the test instance the following scenario is implemented:

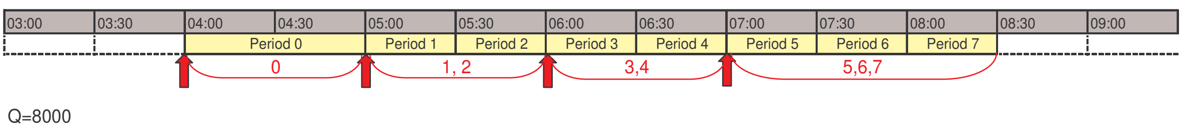

Figure 7: Concept of stepwise releasing of virtual customers

Periods when newspapers are available are marked with the arrows, namely at the beginning of the periods zero, one, three and five. The numbers above the brackets show periods from which virtual customers are considered during the Delivery Customer List creation of that step – the so-called Release Cycles.

2.2 Phase 1 - Creation of delivery plan for the free newspaper delivery problem

In Campbell and Savelsbergh (2004)[28] it is shown that for instantaneous delivery all methods except the early method achieve the same results. Based on that knowledge the following scenario is implemented for the creation of the delivery plan. The idea is to deliver in the first period as many newspapers as possible. The maximum that can be delivered to a station is either the box-capacity or the total estimated consumption. As shown in the description of test instance it is not possible to fill the boxes either with the whole daily demand or the total box-capacity, therefore the below described algorithm is used. This method considers both the number of available newspapers and the estimated consumption per period. It is assumed that delivery takes place as late as possible – in other words the customer is visited in the period before a stock-out would occur.

In each period the current available newspapers (current inventory) for all the stations are estimated. It is calculated by subtracting the cumulated delivery amounts (first period to current period) from the total number of delivered newspapers for the station. The inventory level for first period under observation is zero.

The open demand at a station for a given period is the cumulated unsatisfied demand from the current period to the last period.

In each period it is calculated if the current inventory can satisfy the estimated consumption for that period. If there are not enough newspapers available to satisfy the estimated consumption for the period, a visit to that customer is planned. If the open

[28] [CAM02]

demand for the customer is lower than the box-size, the delivery amount is the open demand. Otherwise the delivery amount is calculated by dividing the total open demand of this customer by the minimal number of deliveries needed to satisfy this demand. The table below shows an example for that calculation.

Node ID	Open Demand	Box-capacity		Min. Number Deliveries	Amount Newspapers
0	100	32000		1	100
1	900	1200		1	900
2	500	400		2	200
3	1600	800		2	400
4	50	32000		1	50
5	600	400		2	200
6	150	32000		1	150
7	700	400		2	200
8	2900	1600		2	800
9	2900	1200		3	400
10	1200	600		2	300
11	600	600		1	600
12	300	200		2	100
13	600	400		2	200
14	300	32000		1	300
15	100	200		1	100

Figure 8: Example for the calculation of the delivery amount

After the calculation of delivery amounts for all customers in one period is finished the determination of the feasibility of this plan is performed. This is done by calculating if the total amount of newspapers to be delivered in this period exceeds the number of newspapers available. In case that amount of newspapers to be delivered is greater than the available newspapers a recalculation of delivery amounts is performed in the following way. For each node the last period where inventory level will be higher than zero is identified. The delivery amount for the station is reduced by the number of newspapers that are considered for satisfying the estimated consumption of that period.

The recalculation is demonstrated in the example in Figure 9 – only a few nodes are used for that purpose. The column "Open Demand-begin of period" represents the part of the total estimated consumption that is not covered by the delivery amounts planned in the prior periods at a station. In our example the delivery period zero is considered. The column "Planned Delivery amount Newspapers" shows the results of delivery amount calculation before the feasibility check. In this example the total delivery amount of newspapers is 7.384 while it is assumed that there are only 7.000 available. The last period where the inventory level is greater than zero after subtracting the estimated consumption of as many periods as possible from the cumulated delivery amounts of the node are presented in the column “Last Period, where newspapers available at station”. It is the third period for Node ID three. The column “Reduction” contains the amount that is planned for that period. It is 240 newspapers for Node ID three. The reduction step is performed as long as the total delivery amount exceeds the number of newspapers available. For the last node in our example, on Node ID seven only the amount needed for reaching that value is considered, not the number of newspapers that are considered at the period for that customer. Only 54 newspapers are calculated for reducing delivery for node seven instead of 105. The new delivery amount decreased by the reduction can be found in “Planned Delivery Amount – Supply capacity considered”.

				Demand						
Node ID	Open Demand - begin of period	Box-capacity	Planned Delivery Amount Newspapers	Period 0 05:00-05:30	Period 1 05:30-06:00	Period 2 06:00-06:30	Period 3 06:30-07:00	Last Period, where newspapers available at station	Reduction	Planned Delivery Amount - Supply capacity considered
3	1600	800	800	80	160	320	320	3	240	560
5	600	400	300	30	60	120	120	3	90	210
7	700	400	350	35	70	140	140	3	54	296
8	2900	1600	1450	145	290	580	580	3	0	1450
9	2900	1200	967	145	290	580	580	2	0	967
10	1200	600	600	60	120	240	240	3	0	600
12	300	200	150	15	30	60	60	3	0	150
13	600	400	300	30	60	120	120	3	0	300
18	1100	400	367	55	110	220	220	2	0	367
21	500	400	250	25	50	100	100	3	0	250
22	900	600	450	45	90	180	180	3	0	450
23	600	200	200	30	60	120	120	2	0	200
24	1200	400	400	60	120	240	240	2	0	400
25	2400	800	800	120	240	480	480	2	0	800
	Total Demand		7384					Total Demand adjusted		7000

Figure 9: Example for calculation of delivery amounts including feasibility check

2.3 Phase 2 - Solving the Vehicle Routing Problem for the free newspaper delivery problem

The I1-Heuristic described by Solomon[29] is used as construction heuristic for the problem. It was slightly modified to meet the production capacity constraint with the concept of Release cycles explained above. For each Release Cycle the I1-Heuristic is performed, the results are added to the set of routes after each step.

[29] [SOL01]

The improvement heuristics that were applied to the starting solution created by the construction heuristic are the OR-opt and VNS.

The OR-opt-procedure was implemented as described in Savelsbergh, 1995[30] following this general algorithm.

Or-Opt

1. for i Є all routes
2. i = find_1-opt_solution (i)
3. i = find_2-opt_solution(i)
4. i = find_3-opt_solution(i)
5. end for

The second improvement heuristic that is used is the VNS. It is based on the basic structure introduced by Mladenović and Hansen (1997).[31] (cf. Figure 10)

<u>Initialisation:</u>

Select the set of neighbourhood structures $N_\kappa(\kappa = 1;\dots;\kappa_{max})$, that will be used in the search

find an initial solution x

choose a stopping condition

<u>Main step</u>:

Repeat the following until the stopping condition is met:

Set $\kappa \leftarrow 1$;

Repeat the following steps until $\kappa = \kappa_{max}$:

<u>Shaking</u>. Generate a point x' at random from κ^{th} neighbourhood of x ($x' \in N_\kappa(x)$);

<u>Iterative improvement</u>. Apply some local search method with x' as initial solution; denote with x'' the so-obtained local optimum;

<u>Acceptance decision</u>. If this local optimum x'' is better than the incumbent,

[30] [SAV01]
[31] [MLA01]
[32] [POL02]

VNS is based on a set of pre-selected neighbourhood structures Nκ (κ =1, . . . , $κ_{max}$). The stopping condition can be maximum CPU time allowed, maximum number of iterations or maximum number of iterations between two improvements.

An initial solution is modified in a so-called shaking step to get by randomly selecting a solution from the first neighbourhood. An iterative improvement is applied to the solution found in the prior step. If no new incumbent solution is found, a switch in neighbourhood is performed before the next shaking step followed by an iterative improvement takes place. If a new incumbent solution is found the shaking step starts with the first neighbourhood.

The set of neighbourhoods used for shaking is one of the most important decision for VNS. Each neighbourhood should provide a balance between perturbing the incumbent solution and retain the good parts of the incumbent solution. Neighbourhood structures that are often used are the so-called “move” and “cross”-operators.

The move operator inserts a part of one route into a different route. An example can be found in “Figure 11”, where customers x'_2 and y_2 are moved from second route to the first one. In my algorithm up to three customers are relocated.

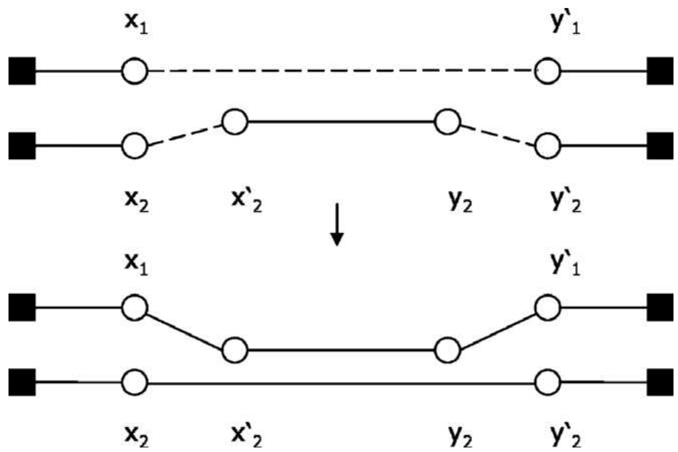

Figure 11: The move operator[33]

The cross exchange operator exchanges two segments of two different routes as illustrated in Figure 12. In this example the segment from customer x'_1 to y_1 of the first

[33] [HEM01]

route is exchanged with the segment x'$_2$ to y_2 of the second route. A segment length of up to six customers is considered in the implemented algorithm.

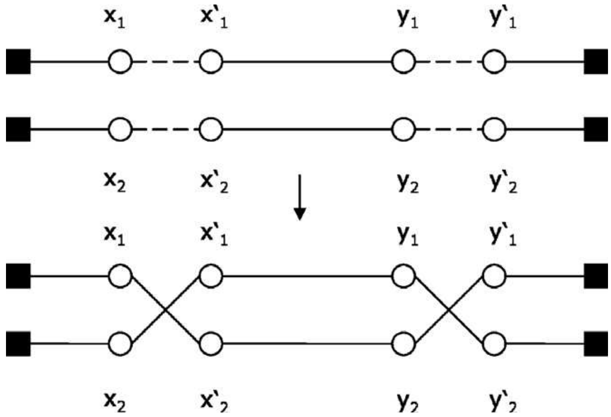

Figure 12: The cross exchange operator[34]

One important difference to the VNS for the VRPTW is that the neighbourhood structures are applied only to routes of the same release cycle. Which means the first route that is candidate for the neighbourhood is chosen randomly from the set of all routes. The second one is selected only out of the routes that are in the same release cycle as the first one.

The neighbourhood structures which are used in the shaking step are move and cross-exchange. The table below gives an overview of the neighbourhood structure that is used.

K	Operator	Min. segment length	Max. segment length
1	Move	1	min(1, n)
2	Move	1	min(2, n)
3	Move	1	min(3, n)
4	Cross	1	min(1, n)
5	Cross	1	min(2, n)
6	Cross	1	min(3, n)
7	Cross	1	min(4, n)
8	Cross	1	min(5, n)
9	Cross	1	min(6, n)

Figure 13: Neighbourhood Structure for VNS[35]

---

[34] [HEM01]
[35] [HEM01]

The iteration improvement procedure that was used is OR-Opt, but only for the tours that are changed in the shaking-step. The stopping criterion that was chosen is “number of iterations without improvement”. The procedure was tested with several values, namely 10, 100, 1000, 10.000 and 100.000 iterations without improvement

3 Experiment

The aim of this experiment is the implementation and analysis of the described methods for both the construction heuristic and the two improvement heuristics. The analysis of each phase was performed and described separately. The solutions are analysed with respect to quality of solution, robustness, flexibility and running time.

The algorithms are implemented in C++. The experiment was performed on a computer with this configuration:

- Intel Core 2 Duo CPU with 2,66 GHz
- 4 GB RAM

The experiment is based on the following assumption

- As starting point one vehicle is assigned to one route. It is not considered that consecutive routes can be seen as a trip and aggregated to larger route.
- There is a fixed stop time of one minute considered for each visit.
- It is assumed that you can delivery each number of newspapers to a station. They are not considered to be bundled into bunches.
- A different starting time window at the various stations is not considered. The solution is based on the assumption that all stations are accessible at 4:00.
- The consumption is different for all stations but the distribution of consumption rate is assumed to be equal for all stations.
- Different thicknesses of newspapers are not considered – for this experiment it is assumed that this is fixed. Therefore capacity of vehicles and boxes are considered to be fixed for the whole experiment.

3.1 Description of the test-instance

The test-instance consists of a set of customers with 213 nodes which are numbered starting with zero. The depot is included in that set. Its node-ID is 212.
The following information is given for each node:

- The ***total consumption*** of each node for the whole planning horizon (eight periods of one day). These consumption values vary between 50 and 11.000 newspapers. The distribution of the consumption values is shown in the graph.

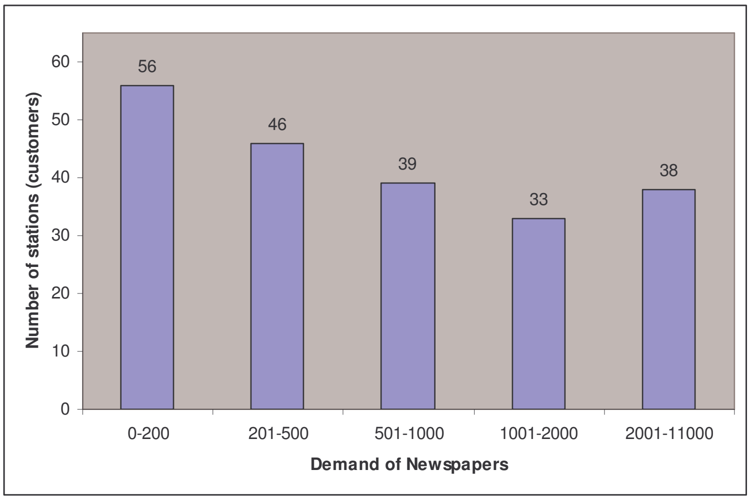

Figure 14: Distribution of daily demand for all stations

- The ***total box-capacity of the station***, where the newspapers are stored. There are different categories of boxes available with the capacity of 200, 400, 700, 800, 1400 or "Abwurfstellen" with unlimited capacity available. At each station there are one or more boxes of these categories placed. The total box-capacity of the station is the sum of capacities of the single boxes available at the station. "Abwurfstellen" are considered with the maximum capacity of 32.000 in the test instance.

- x-coordinate and y-coordinate of the node

- ***Distance and Time*** to all other nodes including the depot
    - only the time matrix is considered for all heuristics that are described later
    - the service time is assumed to be fixed and one minute for all stations

The length of the consumption periods is assumed to be 30 minutes. They are starting at 5:00. Consumption for each period is given as percentage of the total consumption. The values are assumed to be the same for all customers. An overview about the distribution of the consumption rates over the various periods is given in Figure 15. In the first line the start time of the period is given. The estimated consumption for the period 5:00 – 5:30 is 5 % of the total consumption for a customer.

Starttime of period	05:00	05:30	06:00	06:30	07:00	07:30	08:00	08:30
Consumption rates	5%	10%	20%	20%	10%	15%	10%	10%

Figure 15: Consumption rates

It is required that delivery is finished when consumption starts. Therefore the estimated consumption for a period at a customer must be satisfied one period earlier. There are eight periods considered for delivering newspapers to a station. The first period is 60 minutes long considering that delivery could start at 4:00 and no consumption takes place until 5:00. The next delivery periods are 30 minutes and end at 8:30. Figure 16 shows the relationship between the consumption of newspapers in each period and the delivery period, when those newspapers need to be delivered latest.

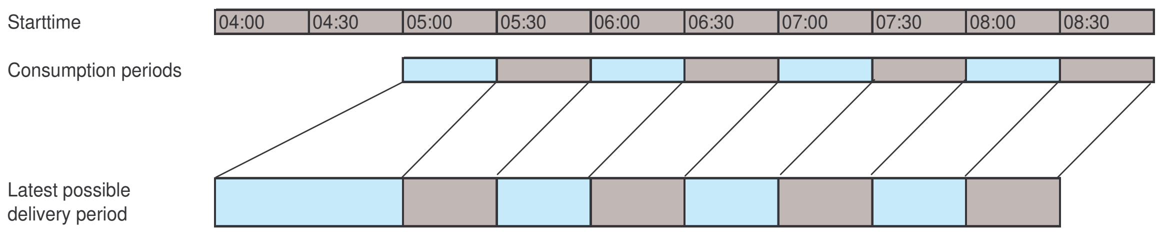

Figure 16: Consumption vs. delivery period

The analysis of the relation between the total capacity per station and the consumption shows that the consumption for one period at all stations is lower than its capacity at this station.

The minimum number of visits at a station is defined by the total consumption divided by the total box-capacity of the station. The consumption of many stations can be satisfied by delivering only once. The distribution of the number of visits per station is shown in the chart. As presented in Figure 17 there is no station that needs not be visited more than four times per day when looking only at the minimum number of visits.

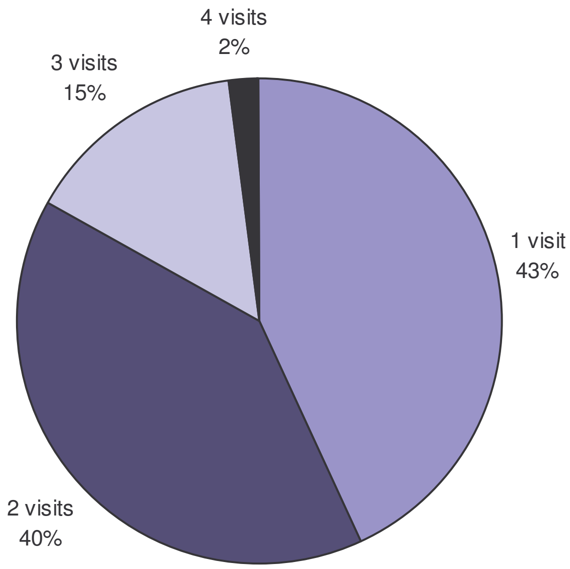

Figure 17: Overview of minimum frequency of visits per station

The estimated consumption of all stations is 250.250 newspapers but the number of newspapers available for all periods is only 240.000 – which means the values for consumption exceeds the supply. Further analysis show that not all boxes can be filled with either the estimated total consumption for the customer if this is lower than the boxsize, or the full capacity of its boxes in the first period. If the delivery amount per customer in the first period is considered as min{total box-capacity, total consumption} the sum for all customers is 145.850. This is more than the amount that is available at 4:00 o'clock, the assumed start there for delivering the customers.

3.2 Phase 1: Create Delivery Plan

With the delivery plan that is created by this algorithm - the delivery amount of 88 stations is equal to consumption and for 124 stations the delivery amount is lower. Figure 18 shows how many stations are visited in the various periods and Figure 19 presents the corresponding total amount of newspapers that will be delivered.

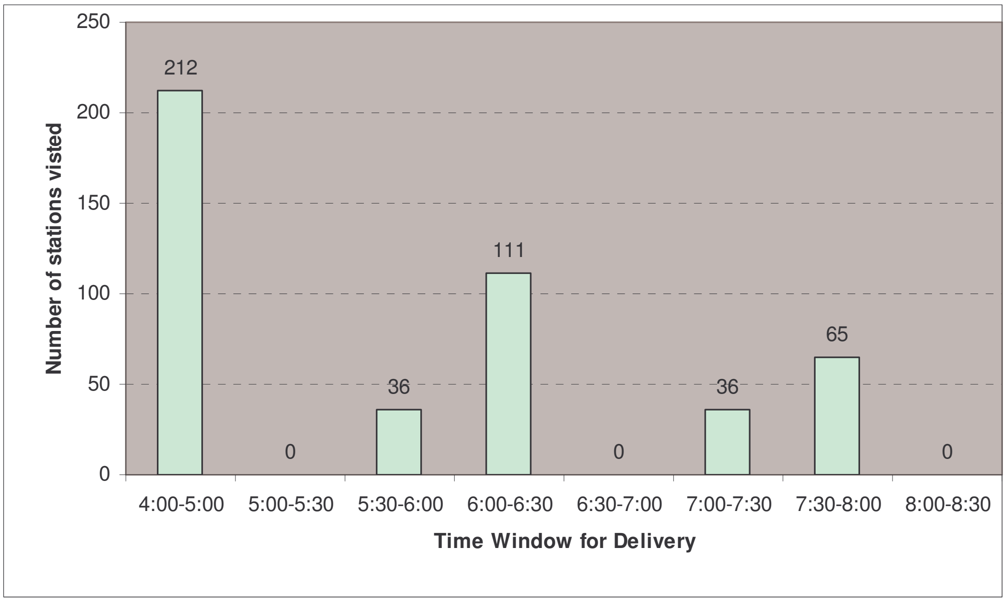

Figure 18: Number of customer visited in each period

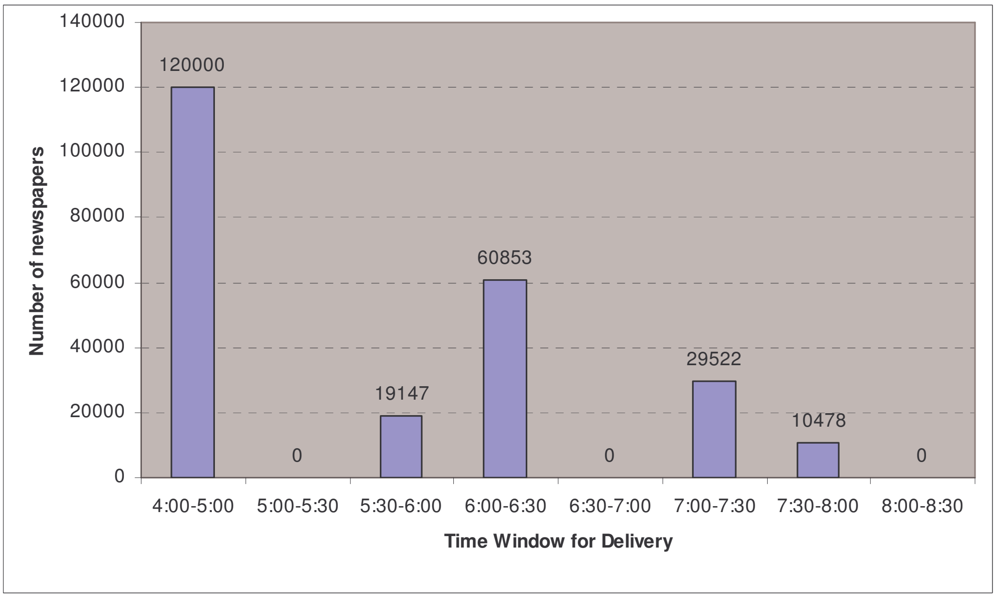

Figure 19: Total Amount of newspapers delivered in each period

A stock-out of newspapers, where cumulated consumption of the current period and its predecessors exceeds cumulated supply for the corresponding time frame at station, occurs only in the last period. The total shortage is less than 10 %. Figure 20 shows if the whole estimated consumption is covered by the amount of delivered newspapers or if there are gaps. The results are grouped in those categories

- "0 % = x " represent the number of stations where consumption is equal to delivery amount
- "0 % < x < = 5 %" is used for stations where the gap is between 0% and 5%.
- "5 % < x < 10 %" where the gap is greater than 5%.

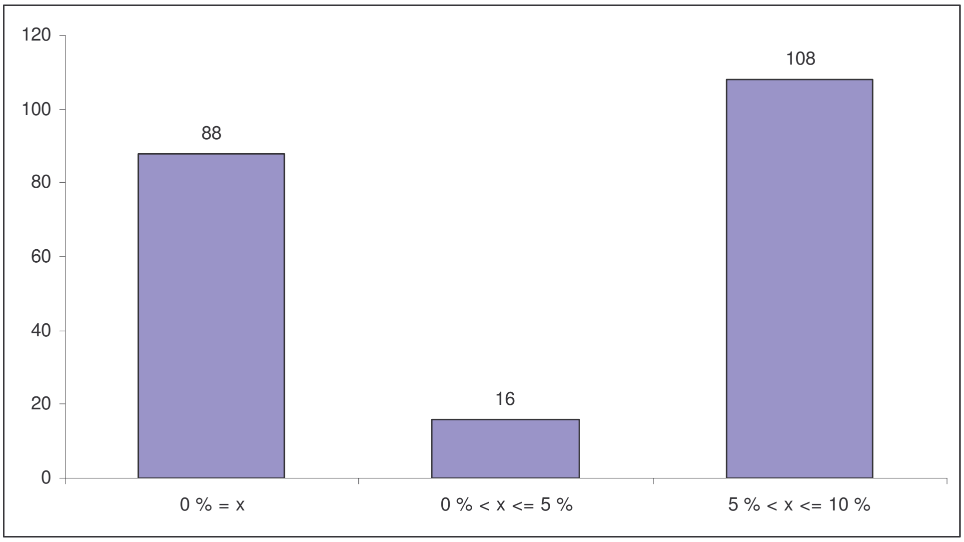

Figure 20: Customer grouped by gaps between delivery amount and consumption

In Figure 21 the theoretical lower bound of vehicles is shown. It is calculated as the total delivery amount per period divided by the capacity of one vehicle which is defined as 8.000 newspapers. The time window constraints are not considered in that calculation.

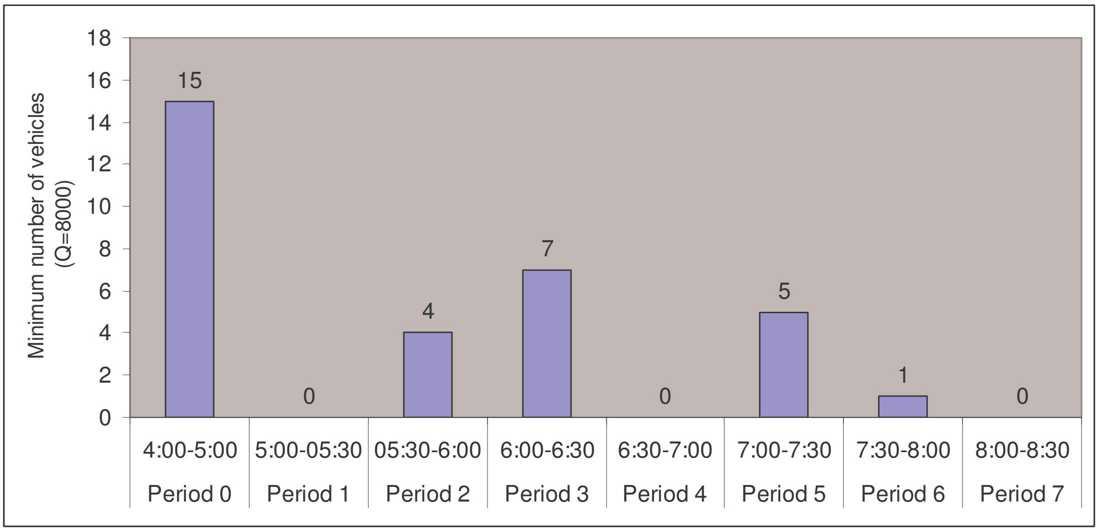

Figure 21: Theoretical lower bound of vehicles per period

3.3 Phase 2: Solving the VRPTW

The algorithms created for solving the Vehicle Routing Problem with Time Window (VRPTW) are analysed due to those criteria

- ⇨ results of the solutions with respect to time and quantities of the tours
- ⇨ number of vehicles needed for tours started at the beginning of a release cycle
- ⇨ number of vehicles needed in total
- ⇨ compare several parameter sets discussed by Solomon (2004)[36]
- ⇨ create scenarios – where some constraints are relaxed

3.3.1 Construction Heuristic

First of all the scenario that was prior described and leads to a feasible solution was analysed. The vehicle capacity is assumed to be 8.000 newspapers. The scenario is mentioned as "Scenario 1: With stepwise release" or standard scenario in the following text.

As shown in table Figure 22 the best result for the construction heuristic run, when considering the travel time as optimization criteria, was found for the parameter set (α_1=0, α_2=1, μ=1, λ=1) with cost of 4.178,42 and the worst one for (α_1=1, α_2=0, μ=1, λ=2) with cost of 4362,63. "Net travel time" is the travel time without waiting time, whereas the "Total Tour time" is the sum of the waiting time and net travel time.

As depicted in column "Average quantity" this value is relatively low with 3.380 newspapers of the best solution in this set. This number is decreasing the worse the solution gets. A similar effect can be observed with the number of tours. The less tours are in the solution the better is the result with respect to the net travel time.

[36] [SOL01]

	μ	λ	α 1	α 2	Number of tours	Total waiting time	Net travel time	Total tour time	Average waiting time	Average travel time	Average quantity	Minimal quantity	Maximal quantity	Average Customers in tour
Scenario 1: with stepwise release														
	1	1	1	0	72,00	23,49	4179,00	4202,49	0,33	58,04	3333,33	485,00	7955,00	8,39
	1	1	0	1	71,00	3,43	4178,42	4181,85	0,05	58,85	3380,28	80,00	7537,00	8,48
	1	2	1	0	74,00	50,12	4315,02	4365,14	0,68	58,31	3243,24	340,00	7872,00	8,22
	1	2	0	1	75,00	41,75	4362,63	4404,38	0,56	58,17	3200,00	340,00	7627,00	8,13

Figure 22: Scenario 1 - Construction Heuristic

Figure 23 shows the number of vehicles needed in one release cycle compared to the number of vehicles that would be needed in the theoretical lower bound solution. A huge difference between those two numbers can be observed – especially in the third release cycle where the minimum number is 7 vehicles and the number in the solutions is 26 or 27, which is approximately three times higher.

	μ	λ	α 1	α 2		1st release cycle	2nd release cycle	3rd release cycle	4th release cycle	Number of Tours
Minimum Vehicles needed - with stepwise release										
Min Vehicles						15	4	7	5	31
Scenario 1: with stepwise release										
	1	1	1	0		25	7	27	13	72
	1	1	0	1		25	7	26	13	71
	1	2	1	0		27	7	27	13	74
	1	2	0	1		27	8	27	13	75

Figure 23: Number of vehicles needed for the various solutions of scenario 1

When assuming that only 20 vehicles are available, it is obvious that there is a bottleneck in all scenarios in the first and third release cycle. The reason for the big difference is that the average load of each vehicle is less than 50% of the maximum load, which holds for all solution tested with the various parameter sets. The strongest constraint in all cases is the very restrictive time window.

	μ	λ	α_1	α_2	1st release cycle	2nd release cycle	3rd release cycle	4th release cycle
Average Number of Elements								
	1	1	1	0	10,48	7,14	6,23	9,29
	1	1	0	1	10,48	7,14	6,27	9,77
	1	2	1	0	9,85	6,29	6,11	9,77
	1	2	0	1	9,85	6,50	6,11	9,77
Average Quantity								
	1	1	1	0	4800,00	2735,29	2318,96	2897,14
	1	1	0	1	4800,00	2735,29	2340,50	3076,92
	1	2	1	0	4444,44	3109,57	2253,81	3076,92
	1	2	0	1	4444,44	2393,38	2253,81	3076,92
Average Travel Time								
	1	1	1	0	70,43	53,12	43,88	64,67
	1	1	0	1	71,92	54,00	43,79	66,46
	1	2	1	0	69,64	52,88	43,52	68,42
	1	2	0	1	70,11	51,18	43,52	68,10

Figure 24: Construction heuristic – Detailed Analysis of all release cycles

A more detailed analysis of the construction heuristic's results in Figure 24 shows that if there are nodes with different delivery periods or longer time windows in one release cycle, higher tour average travel times and a higher number of elements per tour can be observed. In our example this is true for first release cycle where the nodes for the first period are scheduled. This period lasts 60 minutes compared to 30 minutes which is the length of the other periods. A second example can be found in the fourth release cycle where nodes are visited in two periods, compare to the other release cycles where delivery takes place in only one period.

3.3.2 OR-Opt

In Figure 25 and Figure 26 the results after the Or-Opt is shown. The total travel time which is used as criterion for the optimization is reduced by about 15 % for all parameter sets.

	μ	λ	α_1	α_2	Number of tours	Total waiting time	Total travel time	Total tour ime	Average waiting time	Average travel time	Average quantity	Minimal quantity	Maximal quantity	Average Customers in tour
	1	1	1	0	72,00	55,11	3562,99	3618,10	0,77	49,49	3333,33	485,00	7955,00	8,39
	1	1	0	1	71,00	28,76	3549,34	3578,10	0,41	49,99	3380,28	80,00	7537,00	8,48
	1	2	1	0	74,00	72,21	3628,73	3700,94	0,98	49,04	3243,24	340,00	7872,00	8,22
	1	2	0	1	75,00	59,36	3666,35	3725,71	0,79	48,88	3200,00	340,00	7627,00	8,13

Figure 25: Results for OR-Opt

μ	λ	α_1	α_2	Total travel time			Total tour time			Average travel time		
				before Or Opt	after Or Opt	Change in percent	before Or Opt	after Or Opt	Change in percent	before Or Opt	after Or Opt	Change in percent
Scenario 1: with stepwise release												
1	1	1	0	4179,00	3562,99	-15%	4202,49	3618,10	-14%	58,04	49,49	-15%
1	1	0	1	4178,42	3549,34	-15%	4181,85	3578,10	-14%	58,85	49,99	-15%
1	2	1	0	4315,02	3628,73	-16%	4365,14	3700,94	-15%	58,31	49,04	-16%
1	2	0	1	4362,63	3666,35	-16%	4404,38	3725,71	-15%	58,17	48,88	-16%

Figure 26: Comparision Results Construction Heuristic - Or-Opt

3.3.3 VNS

VNS was executed with different completion criteria, which were defined as x iterations without improvement. Meaning when an improvement was found after x-1 iterations, the iteration count was set to zero. The values that were used for x were 10, 100, 1000, 10000 and 100.000. The acceptance criterion allows only improvements. A move is counted as iteration when it leads to a feasible solution, otherwise it was discarded. The VNS was performed 10 times for each scenario with each parameter set.

The following chart (Figure 27) gives an overview of the results of these test runs for the parameter set (α_1=0, α_2=1, μ=1, λ=1). This is used as example because it returns the highest value after the or-opt-optimization. The results of the other parameter sets can be found in Appendix B.

Number Iterations without improvement	Cost			Improve-ment	Time (s)			Total Iterations		
	Avg	Min	Max		Avg	Min	Max	Avg	Min	Max
10	3542,96	3536,64	3549,34	0,36%	0	0	1	18	10	95
100	3457,18	3446,11	3478,79	2,91%	4	1	32	936	176	2.030
1.000	3428,21	3418,47	3447,03	3,69%	27	15	247	6.133	3.592	18.068
10.000	3414,44	3402,53	3427,66	4,14%	112	61	1.373	29.225	16.824	117.335
100.000	3419,55	3395,38	3440,50	4,34%	500	380	8.249	116.642	105.795	494.152
Cost after Construction Heuristic: 4178,42 Cost after Or-Opt: 3549,34										

Figure 27: Results VNS with different stopping criteria - Parameter Set (α_1=0, α_2=1, μ=1, λ=1)

3.3.4 Further analysis by the means of other scenarios

To analyse the impact of the restriction regarding quantities and time windows the following scenarios are tested. These tests should also help analysing if an improvement of the solution is possible when relaxing the constraints.

The scenarios used for analysing the algorithms are described and pictures of the delivery periods are presented. These scenarios are based on the standard scenario (Scenario 1 – With stepwise release). Only the differences of the new scenarios are described below. The consumption periods remain the same.

Scenario 2 – Without stepwise release

This scenario describes a solution where the production constraint is relaxed and only one Release Cycle is considered. In this scenario it is assumed that all newspapers are available at the beginning of the planning horizon at 4:00.

- The supply side is relaxed in that scenario – it is not considered if enough newspapers are available.
- The impact on the number of tours is analysed – if any conclusions could be found about the minimum number of tours.

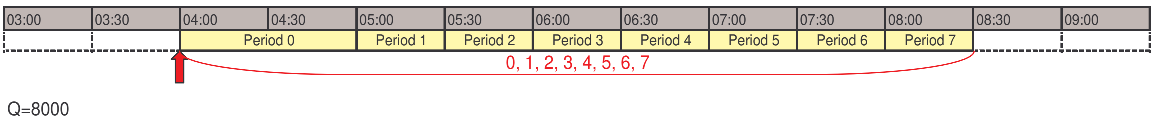

Figure 28: Without stepwise release

Scenario 3: Stepwise release – with extensions of time windows by 30 minutes

In this scenario the end-time of each period is postponed by 30 minutes.

- The restriction that newspapers should be available when the demand occurs is relaxed.
- The impact on the demand side is neglected – it is not considered that newspapers remain at the station when they are delivered too late.
- The extension of the time window by 30 minutes means that delivery is allowed in the period where consumption takes place.

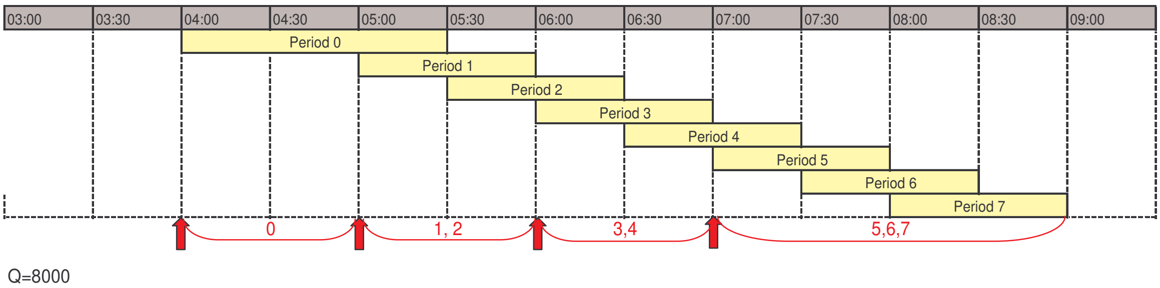

Figure 29: Stepwise release - extend time window by 30 minutes

Scenario 4: Stepwise release – increase vehicle capacity by 2.000

In this scenario the vehicle capacity is increased to 10.000 newspapers. No further adjustments are made.

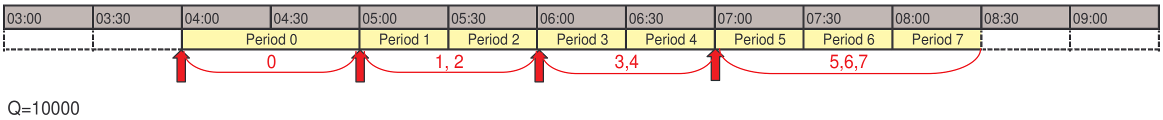

Figure 30: Stepwise release - increase vehicle capacity by 2000

Scenario 5: Stepwise release – increase first period to 120

In this scenario the first period begins 60 minutes earlier so that the duration is increased to 120 minutes. The availability constraint is not met here. The goal is to analyse the impact if tours are started prior to 4:00 am – which would be possible as the first newspapers are available at 2:00 am. In this scenario it is assumed that 120.000 newspapers are available at 3:00.

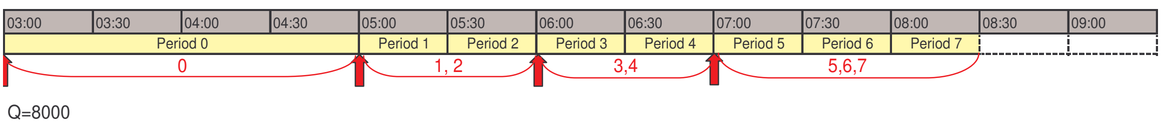

Figure 31: Stepwise release - increase first period to 120

These five scenarios were executed with all parameter sets with the four different parameter sets for α_1, α_2, μ and λ for the construction heuristic. The various parameter sets described by Solomon are (α_1=1, α_2=0, μ=1, λ=1), (α_1=0, α_2=1, μ=1, λ=1),

(α_1=1, α_2=0, μ=1, λ=2) and (α_1=0, α_2=1, μ=1, λ=2). The results show that no parameter returns the best values for all scenarios. The results of the best parameter set each scenario are used in the following figures. The results for all parameter sets for each scenario can be found in Appendix B.

In the Figure 32 the results for various scenarios of the construction heuristic are shown. Generally it can be said that the less restrictive the scenario with respect to the time window the better the solution. The number of tours and the total travel cost are considered as indicators for the quality of the solution.

The time constraints are relaxed in Scenario two, three and five. In Scenario 2 the release cycles are considered and tours can last longer. In Scenario 3 and Scenario 5 time windows are extended.

	μ	λ	α1	α2	Number of tours	Total waiting time	Total travel time	Total tour ime	Average waiting time	Average travel time	Average quantity	Minimal quantity	Maximal quantity	Average Customers in tour
Scenario 1: with stepwise release	1	1	0	1	71	3,43	4178,42	4181,85	0,05	58,85	3380,28	80	7537	8,48
Scenario 2: without stepwise release	1	1	1	0	31	2845,01	3377,22	6222,23	91,77	108,94	7741,94	3233	8000	16,84
Scenario 3: extend time window by 30 minutes	1	1	1	0	47	12,55	3991,34	4003,89	0,27	84,92	5106,38	1120	7992	11,79
Scenario 4: increase vehicle capacity by 2000	1	1	1	0	71	23,49	4167,44	4190,93	0,33	58,70	3380,28	485	9605	8,48
Scenario 5: extend first period to 120 minutes	1	1	0	1	64	3,43	4296,60	4300,03	0,05	67,13	3750,00	80	8000	9,19

Figure 32: Results Construction heuristic - best parameter set for each scenario

Not much improvement can be gained when relaxing the vehicle capacity constraint as can be seen in scenario 4. The solution improved only few when increasing the capacity by 2.000 newspapers per vehicle. The maximum improvement that was gained was for parameter set (α_1=1, α_2=0, μ=1, λ=2) with only 0.79 % cost improvement and a tour reduction of two tours. No changes can be gained by increasing the vehicle capacity by 8.000 newspapers. Nearly no cost reduction can be gained and for the parameter sets (α_1=1, α_2=0, μ=1, λ=1) and (α_1=0, α_2=1, μ=1, λ=1) there was no change. This was surprising as the vehicle capacity was double compared to the standard scenario.

A relationship that was observed for all scenarios and parameter sets is that if the number of tours is low the total travel time is also low. The opposite effect is observed for the average quantity and the average number of customers in a tour. These numbers increase when the number of tours go down.

	μ	λ	α1	α2	1st release cycle	2nd release cycle	3rd release cycle	4th release cycle	Number of Tours
Minimum Vehicles needed - with stepwise release	-	-	-	-	15	4	7	5	**31**
Minimum Vehicles needed - without stepwise release	-	-	-	-	30	0	0	0	**30**
Scenario 1: with stepwise release	1	1	0	1	25	7	26	13	**71**
Scenario 2: without stepwise release	1	1	1	0	31	0	0	0	**31**
Scenario 3: extend time window by 30 minutes	1	1	1	0	19	5	14	9	**47**
Scenario 4: increase vehicle capacity by 2000	1	1	1	0	24	7	27	13	**71**
Scenario 5: extend first period to 120 minutes	1	1	0	1	18	7	26	13	**64**

Figure 33: Number of tours needed -best parameter sets for each scenario

The number of tours needed for each of these scenarios can be found in Figure 33. The number of tours decreases when the restrictions regarding time are relaxed. This is true for increasing the time windows or taking the concept of stepwise releasing out of operation. An increase of capacity has no significant impact with respect to the number of tours.

The lines "Minimum Vehicles Needed" show the minimum of vehicles that would be needed if all constraints were ignored. It is calculated by dividing the number of newspapers that will be delivered in a release cycle by the vehicle capacity.

	μ	λ	α1	α2	Construction Heuristic	OR-Opt	VNS
Scenario 1: with stepwise release	1	1	0	1	4178,42	3549,34	3395,38
Scenario 2: without stepwise release	1	1	1	0	3377,22	2656,10	-
Scenario 3: extend time window by 30 minutes	1	1	1	0	3991,34	2917,59	2662,47
Scenario 4: increase vehicle capacity by 2000	1	1	1	0	4167,44	3543,08	-
Scenario 5: extend first period to 120 minutes	1	1	0	1	4296,60	3386,12	-

Figure 34: Results Improvement Heuristics for best parameter sets per scenario

In Figure 34 the results of the improvement heuristic Or-Opt and VNS for the best parameter sets of the mentioned scenarios are shown – whereas VNS was only performed for scenario 2 and scenario 3. In column VNS the best result of all runs for that scenario are shown.

4 Conclusions

In this master thesis the free newspaper delivery problem was introduced and solved. The free newspaper delivery problem deals with delivering all newspapers produced at the depot to underground and tramway stations. The newspapers are delivered and produced within a defined time horizon. The goal is to minimize travel cost while avoiding a stock-out at the stations during the time horizon of delivery and getting a stock of zero at each station at its end. The free newspaper delivery problem is NP-hard.

The heuristic used for construction a solution to the Free Newspaper Delivery Problem uses a 2-phase decomposition approach. In the first phase a delivery schedule is created which determines the quantities that should be delivered at each node at each period. Each planed visit is used as virtual nodes identified by the node ID of the station and the period where delivery should take place. In the second phase the routes are created by using the I1 heuristic by Solomon[37] in a modified way. Not all newspapers are available when the first vehicle leaves the depot to start its first route. Therefore virtual nodes are released only if there are enough available products at the depot so that planned capacities for these nodes can be satisfied. This concept was introduced as release cycles.

This Construction Heuristic finds feasible solutions in less than one second. A relaxation of time constraints leads to better solutions but increasing the vehicle capacity has nearly no impact on the solution. The created delivery plan shows periods with a high number of customers to deliver and some with zero customers. The number of available newspapers is less than the estimated consumption, but in the delivery plan this effect was observed only in the last period. It was not tested if a better diversification of deliveries and delivery amounts would help decreasing the number of vehicles needed. The improvement heuristics VNS and Or-Opt that were applied to the solution found by the construction heuristic return a better solution than its starting point. The heuristics were only implemented in neighbourhoods within the VRP. Changes in the delivery plan where not considered.

[37] [SOL01]

5 Table of figures

6 References

[ATK01] Atkinson, J. B (1994): A greedy look-ahead heuristic for combinatorial optimisation: An application to vehicle scheduling with time windows. J. Oper. Res. Soc. 45 673-684.

[BEL01] Beltrami, E. J.; Bodin, L.D. (1974): Networkks and vehicle routing for municipal waste collection. Networks 4 (1), 65-94

[BRA01] Bräysy, O. (2003): A Reactive Variable Neighborhood Search for the Vehicle-Routing Problem, INFORMS Journal on Computing, 2003 /15/4, p.347

[BRA02] Bräysy, O. (2002): Fast local searches for vehicle routing problem with time windows, INFOR, Nov.2002, 40,2, p.319

[BRA03] Bräysy, O., Gendreau, M. (2005): Vehicle Routing Problem with Time Window, Part I: Route Construction and Local Search Algorithms, Transportation Science, Feb. 2005, Vol. 39, Iss1, pg.104

[BRA04] Bräysy, O., Gendreau, M. (2005): Vehicle Routing Problem with Time Window, Part II: Metaheuristics, Transportation Science, Feb. 2005, Vol. 39, Iss1, pg.119

[CAM01] Campbell, A., Savelsbergh, M. (2004): A Decomposition Approach for the Inventory Routing Problem, *Transportation Science*, Volume 38, Number 4, Pages 488-502.

[CAM02] Campbell, A.; Savelsbergh, M. (2004): Delivery Volume Optimization, *Transportation Science*, Volume 38, Number 2, 210-223

[CAM03] Campbell, A.; Savelsbergh, M. (2004): Efficient Insertion Heuristics for Vehicle Routing and Scheduling Problems, *Transportation Science*, Volume 38, Number 3, 369-378.

[CAM04] Campbell, A.; Savelsbergh, M. (2002): Inventory Routing in Practice, *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications

[CAM05] Campbell, A.; Savelsbergh, M.; Clarke, L.; Kleywegt, A. (1998): The Inventory Routing Problem', *Fleet Management and Logistics*, Kluwer Academic Publishers, 95-112.

[CHA01] Chao, I.-M.; Golden, B.L.; Wasil, E.A. (1995): An improved heuristic for the period vehicle routing problem, Networks 26(1), p. 25-44

[COR01] Cordeau, J.-F.; Gendreau, M ; Laporte, G. (1997) : A tabu search heuristic for periodic an multi-depot vehicle routing problems, Networks 30(2), p. 105-119

[CHR01] Christofides, N.; Beasley, J. E. (1984): The period routing problem, Networks 14(2), 237-256

[DRU01] Drummond, L.M.A.; Ochi, L.S.; Vianna, D.S. (2001): An asynchronous parallel metaheuristic for the period vehicle routing problem, Future Generation Computer Systems 17(4), p. 379-386

[FRA01] Francis, P.M.; Smilowitz, K.R; Tzur, M. (2006): The period vehicle routing problem with service choice, Transportation Science 40(4); p. 439-454

[GAU01] Gaudioso, M.; Paletta, G. (1992): A heuristic for the period vehicle routing problem, Transportation Science 26(2), p. 86-92

[GOL01] Golden, B. et. al. (2008): The Vehicle Routing Problem – Latest Advances and New Challenges, Springer Science and Business Media, New York

[HEM01] Hemmelmayr, V.; Doerner, K.; Hartl, R. (2009): A variable neighborhood search heuristic for periodic routing problems, European Journal of Operational Research. Amsterdam: Jun 16. Vol. 195, Iss. 3; p. 791

[HEM02] Hemmelmayr, V. et.al. (2009): Delivery strategies for blood products supplies, OR Spectrum (2009) 31:707–725

[IOA01] Ioannou, G.; Kritikos, M.; Prastacos, G. (2001): A greedy look-ahead heuristic for the vehicle routing problem with time windows. J. Oper. Res. Soc. 52 523-537.

[KIN01] Kindervater, G.A.P.; Savelsbergh, M.W.P. (1997): Vehicle routing: Handling edges exchanges windows, Local Search in Combinatorial Optimization, Wiley, Chichester, pp. 337–360.

[LIN01] Lin, S. (1965): Computer solutions of the traveling salesman problem, Bell Syst. Tech. J. 44, 2245-2269.

[MLA01] Mladenovic, M., Hansen, P.: Variable Neighborhood Search, Computer Ops Res. Vol. 24, No.11, pp. 1097-1100, 1997

[MOI01] Moin, NH., Salhi, S. (2007): Inventory routing problems : a logistical overview, Journal of the Operational Research Society 58, pp.1185 – 1194

[POL01] Polacek, M., Hartl, R., Dörner, K. (2004): A Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows, Journal of Heuristics, 10, 2004, p. 613-627

[POL02] Polacek, M. et.al. (2008): A Cooperative and Adaptive Variable Neighborhood Search for the Multi Depot Vehicle Routing Problem with Time Windows, BuR – Business Research Vol.1, Iss.2, p.207-218

[POT01] Potvin, J., Rousseau J. (1995): An Exchange Heuristic for Routeing Problems with Time Windows, Journal of the Operational Research Society, 46, 1433-1446

[POT02] Potvin, J.-Y., Rousseau, J.-M. (1993): A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. Eur. J. Oper. Res. 66 331-340.

[RUS01] Russell, R. A.; Gribbin, D. (1991): A multiphase approach to the period routing problem, Networks 21 (7), p. 747 - 765

[RUS01] Russell, R. A.; Igo, W. (1979): An assignment routing problem, Networks 9(1), p. 1-17

[SOL01] Solomon , M. (1987): Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints, Operations Research, Vol. 35, No.2, pp. 254-265

[SAV01] Savelsbergh, M. (1985): Local search in routing problems with time windows, Annals of Operations Research 4(1985/6) p. 285-305

[TAM01], Tam, V., Ma, K.T. (2004): Combining Meta-Heuristics to Effectively Solve the Vehicle Routing Problems with Time Windows, Artificial Intelligence Review 21, 87-112

[TAN01] Tan, C. C. R.; Beasley, J. E. (1984): A heuristic algorithm for the period Vehicle routing problem, Omega 12 (5), p. 497-504

[WIK01] Free Daily Newspaper, http://en.wikipedia.org/wiki/Free_daily_newspaper, call from 2009-08-13

Appendix A Detailed Algorithms

A-1 Create Delivery Plan

The generalized algorithm for creating the delivery plan – the result of this algorithm are stored in the variable DeliveryMatrix[***nPeriod***][***nCustomer***].

CreateDeliveryplan ()

```
1.  int OpenDemand[nCustomer] initalized by the total demand at
                                              each station (node)
2.  int Boxsize[nCustomer] initalized by the total box-capacity at
                                              each station (node)
3.  int DeliveryAmount[nCustomer] initalized by total demand at each station
                                   divided by its minimum number of visits
4.  int Box_Curr_Fillgrade[nCustomer] initalized by zero at each station

5.  int newspapers, AmountDelSoFar
6.  int TotalDeliveryAmount initalized with zero
7.  int period, customer
8.  int tempPeriod

9.  for (period = 0 to (nPeriods-1))
10.     newspapers = getProducedNewspapers (period)
11.     for (customer = 0 to (nCustomers-1))
12.         if ( OpenDemand[customer] > 0)
13.             if (Box_Curr_Fillgrade[customer] -
                        getDemand( period, customer)<=0)
14.                 if  (OpenDemand[customer] > Boxsize[customer])
15.                   DeliveryMatrix[period][customer]
                              =DeliveryAmount[customer]
16.                 else
17.                     DeliveryMatrix[period][customer]=
                              OpenDemand[customer]
```

```
                    end if
                    Box_Curr_Fillgrade[customer] =
                              Box_Curr_Fillgrade[customer] +
                              DeliveryMatrix[period][customer]
                    if (Box_Curr_Fillgrade[customer] > Boxsize[customer])
                        DeliveryMatrix[period][customer]=
                                    DeliveryMatrix[period][customer] +
                                    Boxsize[customer] -
                                    Box_Curr_Fillgrade[customer]
                      Box_Curr_Fillgrade[customer] = Boxsize[customer]
                    end if
                    OpenDemand[customer] = OpenDemand[customer]  -
                                       DeliveryMatrix[period][customer]
               end if
               Box_Curr_Fillgrad[customer] = Box_Curr_Fillgrad[customer] -
                                          getDemand( period, customer)
                TotalDeliveryAmount= TotalDeliveryAmount +
                                          DeliveryMatrix[period][customer]
          else
               DeliveryMatrix[period][customer]=0
          end if
     end for

    customer = 0
    while (TotalDeliveryAmount > newspapers)
        AmountDelSoFar=
                TotalAmountDeliveredUntilPeriod (customer,period);
        tempPeriod =
                getLatestPeriodSatisfyingDemand(customer, AmountDelSoFar);
        if (TotalAmountDeliveredUntilPeriod(customer,tempPeriod) >
                                                      AmountDelSoFar)
            Reduction=AmountDelSoFar-
                TotalAmountDeliveredUntilPeriod(customer,tempPeriod-1)
```

```
        else
            Reduction=getDemand(customer,period)
        endif
        DeliveryMatrix[period][customer]= DeliveryMatrix[period][customer] -
                                                                  Reduction
        if  (DeliveryMatrix[period][customer] < 0)
             Reduction = Reduction + DeliveryMatrix[period][customer]
             DeliveryMatrix[period][customer]=0
        endif
        Box_Curr_Fillgrad[customer]= Box_Curr_Fillgrad[customer]-Reduction
        OpenDemand[customer]=OpenDemand[Customer]+Reduction
        TotalDeliveryAmount = TotalDeliveryAmount – Reduction
        if ( customer < nCustomer)
            customer = customer + 1
        else
            customer = 0
        end if
    end while

end for
```

The constant ***nCustomer*** represents the number of Customers, while ***nPeriod*** is used for the number of periods.

Now a short overview of the functions used in ***CreateDeliveryplan***:

- ***getProducedNewspapers*** returns the number of newspapers that are available until the period handed over as parameter.

- The function ***getDemand*** gives the demand for the period and the customer that are represented by the parameters back.

- The amount of newspapers that is delivered to a customer for a period is returned by ***TotalAmountDeliveredUntilPeriod.***

- ***getLatestPeriodSatisfyingDemand*** hand back the last period where the inventory minus cumulated demand of newspapers is greater than zero, the cumulated amount of newspapers that is handed over as second parameter is compared to the cumulated demands . The customer is used as the first parameter.

A-2 Convert Delivery Plan to VRPTW

The algorithm ***ConvertDeliveryPlanToVRPTW*** returns a list of virtual customers, where each element consists of the above described information. The following schema shows how it is implemented:

ConvertDeliveryPlanToVRPTW (StartPeriod, EndPeriod, DeliveryMatrix)

```
1. DCL={} //Delivery Customer List
2. for period= StartPeriod to EndPeriod
3.         for customer=0 to nCustomer-1
4.                 if DeliveryMatrix[period][customer] > 0
5.                         DCL = addElement(DCL, customer, period);
6.         endfor
7. end for
8. return DCL
```

The Parameters StartPeriod and EndPeriod for the function ***ConvertDeliveryPlanToVRPTW*** are introduced to ensure that a stepwise release can be performed.

A-3 VRPTW – Construction Heuristic

The Tourplan is represented by a list of routes of type DCL-List. The following algorithm is used, for its creation.

CreateTourPlan ()

1. list<virtual customers> DCL ={}   //Delivery Customer List
2. list<DCL> Tourplan ={}
3. startperiod=0
4. while (startperiod < (nPeriods-1))
5.   if (***NewNewspapersAreAvailable*** (startperiod))
6.   endperiod=***GetLatestPeriodOfReleaseCycle***(startperiod)
7.   DCL= ***CreateCustomerList*** (Deliverplan,startperiod,endperiod)
8.   Tourplan=***CreateRoutesAndExtendTourPlan*** (Tourplan, DCL, TimeMatrix)
9.   endif
10. end while
11. return Tourplan

The function ***NewNewspapersAreAvailable*** check whether newspapers are released in the period passed as parameter, it returns TRUE when newspapers are available and FALSE otherwise. ***GetLatestPeriodOfReleaseCycle*** returns the last period before the next release of newspapers, the period for the start of the evaluation is handed over as parameter.

CreateRoutesAndExtendTourPlan create tours for the Delivery Customer List (DCL) which is the second parameter of the function and the third parameter is the TimeMatrix, which includes the time needed to get from one node to all others including the service time at the node. The tours are built based on the procedure I1 described by Solomon for solving the VRPTW[38]. Return value is the tourlist extended by the new routes, which were passed to the function as second parameter. The following algorithm provides more details about the implementation.

CreateRoutesAndExtendTourPlan (Tourlist, DCL, TimeMatrix)

1. int DepotID initizalised by the ID of the depot
2. boolean newTourNeeded, capacityCheck
3. list<virtual customers> currentTour ={}

[38] [SOL01]

```
<virtual customer> currentCustomer   // pointer to an <virtual customer> element
int currentTourNumber

currentTour=CreateNewTour (DepotID, DCL, startperiod);
currentTourNumber, Tourlist=AddTourToTourList(Tourlist, currentTour)
newTourNeeded=FALSE

while (not isempty(DCL))
    capacityCheck=FALSE
    best_c2=MIN_C2

    for (currentCustomer = all listelements in DCL)
        if ( ( getTourCapacity(currentTour)+getDemand(currentCustomer) ) <= VEHICLE_CAPACITY)
            c1,PositionToInsert = CalculateC1 (currentTour, currentCustomer)
            if (c1 != MIN_C1)
                c2 = CalculateC2 (currentCustomer, c1)
                if (c2 >= best_c2)
                    best_c2=c2
                    currentBestCustomer=CurrentCustomer
                    BestInsertPositionInTour=PositionToInsert
                    capacityCheck=TRUE
                end if
            end if
        end if
    end for

    if (capacityCheck=FALSE)
        newTourNeeded=TRUE
    end if
```

```
    if ((best_c2 != MIN_C2) AND (newTourNeeded=FALSE))
        currentTour=insertCustomerInTour(currentTour, currentBestCustomer,
                                                   BestInsertPositionInTour)
        DCL=deleteCustomerFromDCL(DCL, currentBestCustomer)
        Tourlist = ExchangeTourList (Tourlist, currentTour, currentTourNumber)
    else
        newTourNeeded = TRUE
    end if

    if (newTourNeeded =TRUE
        currentTour=CreateNewTour (DepotID, DCL, startperiod);
        currentTourNumber, Tourlist=AddTourToTourList(Tourlist, currentTour)
    end if
end while

return Tourlist
```

The constants ***MIN_C1*** and ***MIN_C2*** represent very small values, they are used for identifying if C1 or C2 is found. The constant ***VEHICLE_CAPACITY*** represents the maximum capacity of a vehicle which is in our example 8000 newspapers.

The following functions are used

- The function **AddTourToTourList** adds the currentTour on the last position in the array Tourlist, which is returned as well as the position where it is inserted in the currentTourNumber.

- ***ExchangeTourList*** returns the Tourlist where the tour handed over as parameter replaces the tour on the position number, which is the third parameter.

- ***deleteCustomerFromDCL*** returns the Delivery Customer List (DCL) which is the first parameter of this function. The Customer handed over as second parameter is deleted from that list.

- ***InsertCustomerInTour*** inserts the customer handed over as second parameter currentBestCustomer before the position represented by the third parameter BestInsertPositionInTour in the tour which is the first parameter and returns the adjusted Tour again. The delivery times of all elements of the tour are adjusted according to change needed for this insertion.

- The function ***getTourCapacity*** calculates the total capacity of all virtual customers that belong to the tour handed over as parameter and returns this value.

- The Function ***CreateNewTour*** creates a new tour consisting of the depot, the customer that is farthest away from the depot and the depot again. The customer that is farthest away from the station is selected from the list handed over as second parameter DCL. The third parameter startperiod is used to calculate the beginning of the tour. It is used to ensure that the tour doesn't leave the depot earlier than the newspapers are available. If the starttime would be defined to be zero, tours that start before the availability of the newspapers would be allowed. Starttime of the depot, which is the last element in the list, is the same as from the first node. The endtime is 999999, to avoid problems due to restrictions of that parameter.

- The function ***CalculateC1*** returns the C1-value, described by Solomon [39] in his article, and the position for the best position to insert the customer in the tour. Parameters handed over to the function are the tour as first parameter and the customer as second parameter. The function also checks if it is feasible to insert the customer in Tour. If no feasible solution can be found the return value for C1 is the constant **MIN_C1**. The detailed algorithm can be found below.

- ***CalculateC2*** returns the C2-value, described by Solomon in his article, with the information provide by the parameters currentCustomer and c1. The algorithm can be found below.

[39] [SOL01]

```
CalculateC1 (Tour, uCustomer)
1.  boolean feasible
2.  double tempDeliveryTime
3.  double c11, c12, c1
4.  double currentBestC1=MIN_C1
5.  pointer <virtual customer>  iCustomer, jCustomer, fc_Customer,
    PositionToInsert

6.  for ( iCustomer = firstcustomer(currentTour) to
    penultimateCustomer(currentTour) ),
                    ( jCustomer = secondCustomer (currentTour) to
    lastCustomer(currentTour) )
7.      feasible=TRUE
8.      c11 = TimeMatrix[getnumber(iCustomer)][getnumber(uCustomer)]
              + TimeMatrix[getnumber(uCustomer)][getnumber(jCustomer)]
              - MY * TimeMatrix[getnumber(uCustomer)][getnumber(jCustomer)]
9.      tempDeliveryTime= [getnumber(iCustomer)][getnumber(uCustomer)] +
                                              getDeliveryTime(iCustomer)
10.     if (tempDeliveryTime < getstarttime(uCustomer))
11.         tempDeliveryTime = getstarttime(uCustomer)
12.     else
13.         if (tempDeliveryTime > getendtime(uCustomer))
14.             feasible = FALSE
15.         end if
16.     end if

17.     tempDeliveryTime =
           TimeMatrix[getnumber(uCustomer)][getnumber(jCustomer)]
           +   tempDeliveryTime
18.     if (TempDeliveryTime > getDeliveryTime (jCustomer))
            c12=TempDeliveryTime – getDeliveryTime(jCustomer)
19.     else
20.         c12=0
```

```
    end if

    c1 = ALPHA1 * c11 + ALPHA2 * c12

    if (c1 > currentBestC1) AND (feasibility=TRUE)
        for (fc_Customer=jCustomer to ENDOFTOUR)
            TempDeliveryTime=getdeliverytime(fc_Customer)+c12

            if (getdeliverytime(fcCustomer) > getendtime(fcCustomer))
                feasible = FALSE
            end if

            if (getdeliverytime(fcCustomer) < getstarttime(fcCustomer))
                feasible = FALSE
            end if

            if (feasible = TRUE)
                currentBestC1=c1
                PostionToInsert=jCustomer
            end if
        end for
    end if
end for
return c1, PostionToInsert
```

CalculateC2 (`uCustomer, c1`)

```
double c2
c2 = LAMBDA * TimeMatrix[Depot][getnumber(uCustomer)]  - c1
return c2
```

The following constants and functions are used in the algorithm:

- ***ENDOFTOUR*** represents the last customer of the tour.

- ***ALPHA1***, ***ALPHA2***, ***MY*** and ***LAMBDA*** are taken for the constants α_1, α_2, μ and λ that are used by Solomon [40] for the calculation of c1 and c2. The tests are based on different combinations of these values namely (α_1=1, α_2=0, μ=1, λ=1), (α_1=0, α_2=1, μ=1, λ=1), (α_1=1, α_2=0, μ=1, λ=2) and (α_1=0, α_2=1, μ=1, λ=2).

- The function ***getnumber*** returns the node ID of the customer that is used as parameter.

- Return value of ***getstarttime*** for the customer that is handed over is the beginning of the time window for the given customer.

- **getDeliveryTime** evaluates the delivery time that is considered for the customer passed to the function as parameter.

- The functions ***firstCustomer***, ***secondCustomer***, ***penultimateCustomer*** and ***lastCustomer*** are used to get a pointer to the respective customer in the tour that is handed over to the functions.

A-4 VRPTW – OR-Opt

The following algorithm shows how it was implemented in detail. ***Or_Opt_AllTours*** is the parent procedure to control the OR-opt considering a sequence of one, two and three adjacent customers and ***Or_Opt_SingleMove*** performs an Or-Opt-Move.

Or_Opt_AllTours (Tourlist)

1. boolean ImprovementFound
2. list <virtual customer> currentTour

3. for currentTour = all tours from Tourlist
4.   ImprovementFound=TRUE
5.   while (ImprovementFound)
6.   ImprovementFound, currentTour = ***Or_Opt_SingleMove*** (currentTour,1)

[40] [SOL01]

```
     end while

    ImprovementFound=TRUE
    while (ImprovementFound)
        ImprovementFound, currentTour = Or_Opt_SingleMove (currentTour,2)
    end while

    ImprovementFound=TRUE
    while (ImprovementFound)
        ImprovementFound, currentTour = Or_Opt_SingleMove (currentTour,3)
    end while

    Tourlist = updateTourlist (currentTour)
end for
```

The function ***Or_Opt_SingleMove*** performs an Or-Opt-Move. Parameters handed over to the function are the Tour that should be improved and the NumberOfElements that should be moved to another position. If an improvement is found, the function returns TRUE and the improved Tour, otherwise it returns FALSE.

Or_Opt_SingleMove (Tour, NumberOfElements)

```
boolean ImprovementFound
int i, j, k
pointer <virtual customer> iCustomer,  jCustomer,  kCustomer
list <virtual customer> moveElements
 list <virtual customer> TempTour, TempTour1
double InitTourCost

TempTour=Tour
rv = FALSE
InitTourCost=getTourCost(Tour)
if (NumberOfElements+1 < getToursize(Tour))
```

```
for i = 1 to (getToursize(Tour) - NumberOfElements)
                         j = i + NumberOfElements to getToursize(Tour)
    iCustomer=getCustomer(Tour,i)
    jCustomer=getCustomer(Tour,j)
    moveElements, TempTour = SplitTour(TempTour,iCustomer,jCustomer)
    for k = 1 to getToursize(TempTour)
        TempTour1=TempTour
        kCustomer =getCustomer(TempTour,k)
        TempTour1=CombineTour(TempTour1, moveElement)
        if  (TempTour1 = {} && (getTourCost(TempTour1) < InitTourCost))
            ImprovementFound=TRUE
            Tour=TempTour1
            goto EndOfForLoops
        endif
    end for
end for
end if
EndOfForLoops  // Position where the goto instruction jumps to.
return ImprovementFound, Tour
```

The following functions are used in the procedure ***Or_Opt_SingleMove***:

- ***getToursize*** returns the number of elements that contains the list.

- ***CombineTour*** inserts the Tour that is handed over as second parameter into the one that is passed as first parameter, recalculates the delivery time of all elements in the tour. If errors occur during that calculation due to violations of the time window restrictions, it is an infeasible solution and the return value is an empty tour.

- The function ***SplitTour*** removes the sequence starting at customer i and ending at customer j from the tour handed over as first parameter. The changed tour, as well as a second tour including the removed elements, are the return values for the function.

- ***getCustomer*** returns a pointer to the element, defined by second parameter, from the tour, which is represented by the first parameter.

- ***getTourCost*** calculates the cost for the tour that is handed over as first parameter.

A-5 VRPTW – VNS

The following algorithm shows how the VNS and the Shaking are implemented. The initial solution is built by the Construction heuristic described prior in this document and the OR-opt was also performed. The stopping criteria that was chosen for that algorithm is MAXITERATIONS without improvement. The algorithm was tested with various values for MAXITERATIONS.

VNS (Tourlist)

```
int i, j, k
int iteration
TempTour

iteration=0
k=1
while (iteration < MAXITERATION)
    TempTourList=copyTour(TourList)
    TourList=Shaking(Tourlist, k)
    TourList=OrOpt(Tourlist)
    if (getTotalCost(TempTourList) < getTotalCost(TourList))
        TourList=copyTour(TempTourList)
        if (k <= KN)
            k = k + 1
        endif
    else
        k=1
        iteration=0
    end if
```

19. end while
20. return Tourlist

Below an overview of the functions and parameters used in the VNS procedure can be found, excluding ***getTotalCost*** and ***copyTour*** which were already explained for the OrOpt.

- The constant ***MAXITERATION*** defines the number of iterations without improvement- the stopping criterion of the function.

- ***KN*** is the parameter used for the highest possible neighbourhood.

- The function ***Shaking*** returns a feasible, random neighbour where the highest neighbourhood is defined by the parameter k. While an infeasible neighbour is found by during the selection procedure, this step is repeated.

  The random neighbour is found by the following decisions.
    - Choose randomly, from how many tours elements will be moved to another tour. Possible results are one or two tours, for performing either a move or a cross. This value depends on the highest possible neighbourhood defined by k. If k is smaller or equal than three it is automatically one.

    - Select randomly the tours that are affected by the neighbourhood moves. After the first tour is selected, the range for selecting the second tour is limited to those tours that are from the same release, in order to avoid that too many infeasible solutions are selected.

    - Within each tour the starting position for the move or the cross is randomly defined.

    - For each tour the number of nodes to be moved is evaluated randomly – the maximum of possible nodes is limited by the highest neighbourhood that was handed over via parameter k or if the end of the tour is reached.

- ***OrOpt*** is running in that procedure only for the tours affected in the shaking step.

Appendix B Detailed Results of the Analysis

B-1 Construction Heuristics

μ	λ	α 1	α 2	Number of tours	Total waiting time	Total travel time	Total tour ime	Average waiting time	Average travel time	Average quantity	Minimal quantity	Maximal quantity	Average Customers in tour
Scenario 1: with stepwise release													
1	1	1	0	72,00	23,49	4179,00	4202,49	0,33	58,04	3333,33	485,00	7955,00	8,39
1	1	0	1	71,00	3,43	4178,42	4181,85	0,05	58,85	3380,28	80,00	7537,00	8,48
1	2	1	0	74,00	50,12	4315,02	4365,14	0,68	58,31	3243,24	340,00	7872,00	8,22
1	2	0	1	75,00	41,75	4362,63	4404,38	0,56	58,17	3200,00	340,00	7627,00	8,13
Scenario 2: without stepwise release													
1	1	1	0	30,00	2845,01	3377,22	6222,23	91,77	108,94	7741,94	3233,00	8000,00	16,84
1	1	0	1	31,00	2638,80	3963,96	6602,76	85,12	127,87	7741,94	2465,00	8000,00	16,84
1	2	1	0	31,00	3077,87	3494,06	6571,93	99,29	112,71	7741,94	3450,00	8000,00	16,84
1	2	0	1	31,00	2410,29	3952,18	6362,47	77,75	127,49	7741,94	5058,00	8000,00	16,84
Scenario 3: stepwise release - extend time window by 30 minutes													
1	1	1	0	47,00	12,55	3991,34	4003,89	0,27	84,92	5106,38	1120,00	7992,00	11,79
1	1	0	1	47,00	12,15	4024,96	4037,11	0,26	85,64	5106,38	1135,00	7992,00	11,79
1	2	1	0	50,00	24,43	4247,34	4271,77	0,49	84,95	4800,00	170,00	7997,00	11,20
1	2	0	1	49,00	24,43	4228,49	4252,92	0,50	86,30	4894,49	280,00	7997,00	11,35
Scenario 3 (2) stepwise release - extend time window by 60 minutes													
1	1	1	0	40,00	22,80	4341,69	4364,49	0,57	108,54	6000,00	1380,00	7995,00	13,50
1	1	0	1	39,00	8,63	4237,07	4245,70	0,22	108,64	6153,85	2300,00	8000,00	13,79
1	2	1	0	41,00	18,01	4368,21	4386,22	0,44	106,54	5853,66	1660,00	7997,00	13,22
1	2	0	1	41,00	4,06	4407,10	4411,16	0,10	107,49	5853,66	2300,00	7997,00	13,22
Scenario 4: stepwise release - increase vehicle capacity by 2000													
1	1	1	0	71,00	23,49	4167,44	4190,93	0,33	58,70	3380,28	485,00	9605,00	8,48
1	1	0	1	71,00	3,43	4173,12	4176,55	0,05	58,78	3380,28	80,00	9077,00	8,48
1	2	1	0	73,00	33,43	4281,39	4314,82	0,46	58,65	3287,67	250,00	9698,00	8,30
1	2	0	1	74,00	41,75	4350,67	4392,42	0,56	58,79	3243,24	340,00	8745,00	8,22
Scenario 4 (2) stepwise release - increase vehicle capacity by 8000													
1	1	1	0	71,00	23,49	4167,44	4190,93	0,33	58,70	3380,28	485,00	9605,00	8,48
1	1	0	1	71,00	3,43	4173,12	4176,55	0,05	58,78	3380,28	80,00	10823,00	8,48
1	2	1	0	72,00	33,43	4282,54	4315,97	0,46	59,48	3333,33	250,00	10480,00	8,39
1	2	0	1	74,00	41,75	4340,64	4382,39	0,56	58,66	3243,24	340,00	12062,00	8,22
Scenario 5: with stepwise release - extend the time window of the first period to 120 minutes													
1	1	1	0	64,00	23,49	4402,95	4426,44	0,37	68,80	3750,00	485,00	7995,00	9,19
1	1	0	1	64,00	3,43	4296,60	4300,03	0,05	67,13	3750,00	80,00	8000,00	9,19
1	2	1	0	65,00	50,12	4441,20	4491,32	0,77	68,33	3692,31	340,00	7997,00	9,08
1	2	0	1	66,00	41,75	4466,36	4508,11	0,63	67,67	3636,36	340,00	7997,00	8,97

Figure 35: Results for various scenarios - Construction heuristic

	μ	λ	α 1	α 2	1st release cycle	2nd release cycle	3rd release cycle	4th release cycle	Number of Tours
Minimum Vehicles needed - with stepwise release									
Min Vehicles					15	4	7	5	31
Minimum Vehicles needed - without stepwise release									
Min Vehicles					30	0	0	0	30
Scenario 1: with stepwise release									
	1	1	1	0	25	7	27	13	72
	1	1	0	1	25	7	26	13	71
	1	2	1	0	27	7	27	13	74
	1	2	0	1	27	8	27	13	75
Scenario 2: without stepwise release									
	1	1	1	0	31	0	0	0	31
	1	1	0	1	31	0	0	0	31
	1	2	1	0	31	0	0	0	31
	1	2	0	1	31	0	0	0	31
Scenario 3: stepwise release - extend time window by 30 minutes									
	1	1	1	0	19	5	14	9	47
	1	1	0	1	19	5	14	9	47
	1	2	1	0	20	5	15	10	50
	1	2	0	1	20	5	15	9	49
Scenario 3 (2) stepwise release - extend time window by 60 minutes									
	1	1	1	0	17	4	11	8	40
	1	1	0	1	18	4	10	7	39
	1	2	1	0	18	4	11	8	41
	1	2	0	1	18	4	11	8	41
Scenario 4: stepwise release - increase vehicle capacity by 2000									
	1	1	1	0	24	7	27	13	71
	1	1	0	1	25	7	26	13	71
	1	2	1	0	26	7	27	13	73
	1	2	0	1	26	8	27	13	74
Scenario 4 (2) stepwise release - increase vehicle capacity by 8000									
	1	1	1	0	24	7	27	13	71
	1	1	0	1	25	7	26	13	71
	1	2	1	0	25	7	27	13	72
	1	2	0	1	26	8	27	13	74
Scenario 5: with stepwise release - extend the time window of the first period to 120									
	1	1	1	0	17	7	27	13	64
	1	1	0	1	18	7	26	13	64
	1	2	1	0	18	7	27	13	65
	1	2	0	1	18	8	27	13	66

Figure 36: Number of tours needed - Construction Heuristic

	μ	λ	α_1	α_2	Number of tours		Total travel time		Average quantity		Average Customers in tour	
Scenario 1												
	1	1	1	0	72,00	2	4179,00	2	3333,33	2	8,39	2
	1	1	0	1	71,00	1	4178,42	1	3380,28	1	8,48	1
	1	2	1	0	74,00	3	4315,02	3	3243,24	3	8,22	3
	1	2	0	1	75,00	4	4362,63	4	3200,00	4	8,13	4
Scenario 2												
	1	1	1	0	30,00	4	3377,22	1	7741,94	1	16,84	1
	1	1	0	1	31,00	1	3963,96	4	7741,94	1	16,84	1
	1	2	1	0	31,00	1	3494,06	2	7741,94	1	16,84	1
	1	2	0	1	31,00	1	3952,18	3	7741,94	1	16,84	1
Scenario 3												
	1	1	1	0	47,00	1	3991,34	1	5106,38	1	11,79	1
	1	1	0	1	47,00	1	4024,96	2	5106,38	1	11,79	1
	1	2	1	0	50,00	4	4247,34	4	4800,00	4	11,20	4
	1	2	0	1	49,00	3	4228,49	3	4894,49	3	11,35	3
Scenario 3 (2)												
	1	1	1	0	40,00	2	4341,69	2	6000,00	2	13,50	2
	1	1	0	1	39,00	1	4237,07	1	6153,85	1	13,79	1
	1	2	1	0	41,00	3	4368,21	3	5853,66	3	13,22	3
	1	2	0	1	41,00	3	4407,10	4	5853,66	3	13,22	3

	μ	λ	α_1	α_2	Number of tours		Total travel time		Average quantity		Average Customers in tour	
Scenario 4												
	1	1	1	0	71,00	1	4167,44	1	3380,28	1	8,48	1
	1	1	0	1	71,00	1	4173,12	2	3380,28	1	8,48	1
	1	2	1	0	73,00	3	4281,39	3	3287,67	3	8,30	3
	1	2	0	1	74,00	4	4350,67	4	3243,24	4	8,22	4
Scenario 4 (2)												
	1	1	1	0	72,00	2	4167,44	1	3380,28	1	8,48	1
	1	1	0	1	71,00	1	4173,12	2	3380,28	1	8,48	1
	1	2	1	0	72,00	2	4282,54	3	3333,33	3	8,39	1
	1	2	0	1	74,00	4	4340,64	4	3243,24	4	8,22	2
Scenario 5												
	1	1	1	0	64,00	1	4402,95	2	3750,00	1	9,19	1
	1	1	0	1	64,00	1	4296,60	1	3750,00	1	9,19	1
	1	2	1	0	65,00	3	4441,20	3	3692,31	3	9,08	3
	1	2	0	1	66,00	4	4466,36	4	3636,36	4	8,97	4

Figure 37: Construction Heuristic - Ranking Parametersets of all Scenarios

B-2 OR-opt

	μ	λ	α_1	α_2	Number of tours		Total travel time		Average quantity		Average Customers in tour	
Scenario 1												
	1	1	1	0	72,00	2	3562,99	2	3333,33	2	8,39	2
	1	1	0	1	71,00	1	3549,34	1	3380,28	1	8,48	1
	1	2	1	0	74,00	3	3628,73	3	3243,24	3	8,22	3
	1	2	0	1	75,00	4	3666,35	4	3200,00	4	8,13	4
Scenario 2												
	1	1	1	0	30,00	4	2656,10	1	7794,23	1	17,30	1
	1	1	0	1	31,00	1	3282,32	4	7741,94	2	16,84	2
	1	2	1	0	31,00	1	2660,57	2	7741,94	2	16,84	2
	1	2	0	1	31,00	1	3201,27	3	7741,94	2	16,84	2
Scenario 3												
	1	1	1	0	47,00	1	2917,59	1	5106,38	1	11,79	1
	1	1	0	1	47,00	1	2961,08	2	5106,38	1	11,79	1
	1	2	1	0	50,00	4	3051,48	4	4800,00	4	11,20	4
	1	2	0	1	49,00	3	3022,38	3	4897,96	3	11,39	3
Scenario 3 (2)												
	1	1	1	0	40,00	2	2742,39	1	6000,00	2	13,50	2
	1	1	0	1	39,00	1	2756,63	2	6153,85	1	13,79	1
	1	2	1	0	41,00	3	2771,73	3	5853,66	3	13,22	3
	1	2	0	1	41,00	3	2845,31	4	5853,66	3	13,22	3

	μ	λ	α_1	α_2	Number of tours		Total travel time		Average quantity		Average Customers in tour	
Scenario 4												
	1	1	1	0	71,00	1	3543,08	1	3380,28	1	8,48	1
	1	1	0	1	71,00	1	3547,59	2	3380,28	1	8,48	1
	1	2	1	0	73,00	3	3602,32	3	3287,67	3	8,30	3
	1	2	0	1	74,00	4	3652,63	4	3243,24	4	8,22	4
Scenario 4 (2)												
	1	1	1	0	72,00	2	3543,08	1	3380,28	1	8,48	1
	1	1	0	1	71,00	1	3547,59	2	3380,28	1	8,48	1
	1	2	1	0	72,00	2	3590,64	3	3333,33	3	8,39	1
	1	2	0	1	74,00	4	3647,59	4	3243,24	4	8,22	2
Scenario 5												
	1	1	1	0	64,00	1	3390,37	2	3750,00	1	9,19	1
	1	1	0	1	64,00	1	3386,12	1	3750,00	1	9,19	1
	1	2	1	0	65,00	3	3442,24	3	3692,31	3	9,08	3
	1	2	0	1	66,00	4	3475,90	4	3636,36	4	8,97	4

Figure 38: Ranking results Or-Opt

μ	λ	α_1	α_2	Number of tours	Total waiting time	Total travel time	Total tour ime	Average waiting time	Average travel time	Average quantity	Minimal quantity	Maximal quantity	Average Customers in tour
Scenario 1: with stepwise release													
1	1	1	0	72,00	55,11	3562,99	3618,10	0,77	49,49	3333,33	485,00	7955,00	8,39
1	1	0	1	71,00	28,76	3549,34	3578,10	0,41	49,99	3380,28	80,00	7537,00	8,48
1	2	1	0	74,00	72,21	3628,73	3700,94	0,98	49,04	3243,24	340,00	7872,00	8,22
1	2	0	1	75,00	59,36	3666,35	3725,71	0,79	48,88	3200,00	340,00	7627,00	8,13
Scenario 2: without stepwise release													
1	1	1	0	30,00	3432,03	2656,10	6088,13	114,40	88,54	7794,23	3233,00	8000,00	17,30
1	1	0	1	31,00	3231,50	3282,32	6513,82	104,24	105,88	7741,94	2465,00	8000,00	16,84
1	2	1	0	31,00	3795,32	2660,57	6455,89	122,43	85,82	7741,94	3450,00	8000,00	16,84
1	2	0	1	31,00	3026,98	3201,27	6228,25	97,64	103,27	7741,94	5058,00	8000,00	16,84
Scenario 3: stepwise release - extend time window by 30 minutes													
1	1	1	0	47,00	84,53	2917,59	3002,12	1,80	62,08	5106,38	1120,00	7992,00	11,79
1	1	0	1	47,00	29,51	2961,08	2990,59	0,63	63,00	5106,38	1135,00	7992,00	11,79
1	2	1	0	50,00	56,10	3051,48	3107,58	1,12	61,03	4800,00	170,00	7997,00	11,20
1	2	0	1	49,00	45,09	3022,38	3067,47	0,92	61,68	4897,96	280,00	7979,00	11,39
Scenario 4: stepwise release - extend time window by 60 minutes													
1	1	1	0	40,00	56,00	2742,39	2798,39	1,40	68,56	6000,00	1380,00	7995,00	13,50
1	1	0	1	39,00	34,87	2756,63	2791,50	0,89	70,68	6153,85	2300,00	8000,00	13,79
1	2	1	0	41,00	57,86	2771,73	2829,59	1,41	67,60	5853,66	1660,00	7997,00	13,22
1	2	0	1	41,00	19,57	2845,31	2864,88	0,48	69,40	5853,66	2300,00	7997,00	13,22
Scenario 5: stepwise release - increase vehicle capacity by 2000													
1	1	1	0	71,00	55,11	3543,08	3598,19	0,78	49,90	3380,28	485,00	9605,00	8,48
1	1	0	1	71,00	28,76	3547,59	3576,35	0,41	49,97	3380,28	80,00	9077,00	8,48
1	2	1	0	73,00	54,08	3602,32	3656,40	0,74	49,35	3287,67	250,00	9698,00	8,30
1	2	0	1	74,00	59,36	3652,63	3711,99	0,80	49,36	3243,24	340,00	8745,00	8,22
Scenario 6: stepwise release - increase vehicle capacity by 8000													
1	1	1	0	72,00	55,11	3543,08	3598,19	0,78	49,90	3380,28	485,00	9605,00	8,48
1	1	0	1	71,00	28,76	3547,59	3576,35	0,41	49,97	3380,28	80,00	10823,00	8,48
1	2	1	0	72,00	54,08	3590,64	3644,72	0,75	49,87	3333,33	250,00	10480,00	8,39
1	2	0	1	74,00	59,36	3647,59	3706,95	0,80	49,29	3243,24	340,00	12062,00	8,22
Scenario 7: with stepwise release - extend the time window of the first period to 120 minutes													
1	1	1	0	64,00	55,11	3390,37	3445,48	0,86	52,97	3750,00	485,00	7995,00	9,19
1	1	0	1	64,00	28,76	3386,12	3414,88	0,45	52,91	3750,00	80,00	8000,00	9,19
1	2	1	0	65,00	72,21	3442,24	3514,45	1,11	52,96	3692,31	340,00	7997,00	9,08
1	2	0	1	66,00	59,36	3475,90	3535,26	0,90	52,67	3636,36	340,00	7997,00	8,97

Figure 39: Results OR-Opt (detailed view)

μ	λ	$α_1$	$α_2$	Total travel time			Total tour time			Average travel time		
				before Or Opt	after Or Opt	Change in percent	before Or Opt	after Or Opt	Change in percent	before Or Opt	after Or Opt	Change in percent
Scenario 1: with stepwise release												
1	1	1	0	4179,00	3562,99	-15%	4202,49	3618,10	-14%	58,04	49,49	-15%
1	1	0	1	4178,42	3549,34	-15%	4181,85	3578,10	-14%	58,85	49,99	-15%
1	2	1	0	4315,02	3628,73	-16%	4365,14	3700,94	-15%	58,31	49,04	-16%
1	2	0	1	4362,63	3666,35	-16%	4404,38	3725,71	-15%	58,17	48,88	-16%
Scenario 2: without stepwise release												
1	1	1	0	3377,22	2656,10	-21%	6222,23	6088,13	-2%	108,94	88,54	-19%
1	1	0	1	3963,96	3282,32	-17%	6602,76	6513,82	-1%	127,87	105,88	-17%
1	2	1	0	3494,06	2660,57	-24%	6571,93	6455,89	-2%	112,71	85,82	-24%
1	2	0	1	3952,18	3201,27	-19%	6362,47	6228,25	-2%	127,49	103,27	-19%
Scenario 3: stepwise release - extend time window by 30 minutes												
1	1	1	0	3991,34	2917,59	-27%	4003,89	3002,12	-25%	84,92	62,08	-27%
1	1	0	1	4024,96	2961,08	-26%	4037,11	2990,59	-26%	85,64	63,00	-26%
1	2	1	0	4247,34	3051,48	-28%	4271,77	3107,58	-27%	84,95	61,03	-28%
1	2	0	1	4228,49	3022,38	-29%	4252,92	3067,47	-28%	86,30	61,68	-29%
Scenario 3 (2) stepwise release - extend time window by 60 minutes												
1	1	1	0	4341,69	2742,39	-37%	4364,49	2798,39	-36%	108,54	68,56	-37%
1	1	0	1	4237,07	2756,63	-35%	4245,70	2791,50	-34%	108,64	70,68	-35%
1	2	1	0	4368,21	2771,73	-37%	4386,22	2829,59	-35%	106,54	67,60	-37%
1	2	0	1	4407,10	2845,31	-35%	4411,16	2864,88	-35%	107,49	69,40	-35%
Scenario 4: stepwise release - increase vehicle capacity by 2000												
1	1	1	0	4167,44	3543,08	-15%	4190,93	3598,19	-14%	58,70	49,90	-15%
1	1	0	1	4173,12	3547,59	-15%	4176,55	3576,35	-14%	58,78	49,97	-15%
1	2	1	0	4281,39	3602,32	-16%	4314,82	3656,40	-15%	58,65	49,35	-16%
1	2	0	1	4350,67	3652,63	-16%	4392,42	3711,99	-15%	58,79	49,36	-16%
Scenario 4 (2): stepwise release - increase vehicle capacity by 8000												
1	1	1	0	4167,44	3543,08	-15%	4190,93	3598,19	-14%	58,70	49,90	-15%
1	1	0	1	4173,12	3547,59	-15%	4176,55	3576,35	-14%	58,78	49,97	-15%
1	2	1	0	4282,54	3590,64	-16%	4315,97	3644,72	-16%	59,48	49,87	-16%
1	2	0	1	4340,64	3647,59	-16%	4382,39	3706,95	-15%	58,66	49,29	-16%
Scenario 5: with stepwise release - extend the time window of the first period to 120 minutes												
1	1	1	0	4402,95	3390,37	-23%	4426,44	3445,48	-22%	68,80	52,97	-23%
1	1	0	1	4296,60	3386,12	-21%	4300,03	3414,88	-21%	67,13	52,91	-21%
1	2	1	0	4441,20	3442,24	-22%	4491,32	3514,45	-22%	68,33	52,96	-22%
1	2	0	1	4466,36	3475,90	-22%	4508,11	3535,26	-22%	67,67	52,67	-22%

Figure 40: Comparison results construction heuristic and Or-Opt

B-3 VNS

10 iterations without improvement

μ	λ	α_1	α_2	constr. heuristic	After 3-opt	Cost Avg	Cost Min	Cost Max	Improve-ment	Time (s) Avg	Time (s) Min	Time (s) Max	Iterations Avg	Iterations Min	Iterations Max
Scenario 1: with stepwise release															
1	1	1	0	4179,00	3562,99	3552,65	3543,71	3560,28	0,54%	0,00	0,00	0,00	29,80	10,00	62,00
1	1	0	1	4178,42	3549,34	3542,96	3536,64	3549,34	0,36%	0,00	0,00	1,00	18,20	10,00	95,00
1	2	1	0	4315,02	3628,73	3620,30	3608,16	3628,52	0,57%	0,00	0,00	0,00	21,30	11,00	39,00
1	2	0	1	4362,63	3666,35	3639,41	3619,08	3664,90	1,29%	0,10	0,00	1,00	45,60	16,00	95,00
Scenario 3: stepwise release - extend time window by 30 minutes															
1	1	1	0	3991,34	2917,59	2914,31	2911,09	2917,59	0,22%	0,00	0,00	1,00	18,50	10,00	41,00
1	1	0	1	4024,96	2961,08	2957,33	2952,35	2961,08	0,29%	0,40	0,00	1,00	20,30	10,00	30,00
1	2	1	0	4247,34	3051,48	3046,56	3035,47	3051,25	0,52%	0,20	0,00	1,00	18,80	11,00	41,00
1	2	0	1	4228,49	3022,38	3018,17	3006,67	3022,38	0,52%	0,10	0,00	1,00	18,20	10,00	32,00
Scenario 3 (2) stepwise release - extend time window by 60 minutes															
1	1	1	0	4341,69	2742,39	2733,59	2715,31	2740,87	0,99%	0,50	0,00	1,00	23,80	11,00	41,00
1	1	0	1	4237,07	2756,63	2748,91	2730,86	2756,63	0,93%	0,70	0,00	1,00	25,20	10,00	51,00
1	2	1	0	4368,21	2771,73	2767,30	2758,87	2771,73	0,46%	0,20	0,00	1,00	17,60	10,00	29,00
1	2	0	1	4407,10	2845,31	2838,14	2825,71	2845,31	0,69%	0,10	0,00	1,00	22,50	10,00	46,00

Figure 41: VNS with 10 iterations

100 iterations

μ	λ	α_1	α_2	constr. heuristic	After 3-opt	Cost Avg	Cost Min	Cost Max	Improve-ment	Time (s) Avg	Time (s) Min	Time (s) Max	Iterations Avg	Iterations Min	Iterations Max
Scenario 1: with stepwise release															
1	1	1	0	4179,00	3562,99	3486,25	3466,47	3536,13	2,71%	3,50	1,00	6,00	843,70	176,00	1266,00
1	1	0	1	4178,42	3549,34	3457,18	3446,11	3478,79	2,91%	3,60	1,00	32,00	935,90	176,00	2030,00
1	2	1	0	4315,02	3628,73	3553,84	3532,52	3572,36	2,65%	3,50	2,00	6,00	900,40	580,00	1365,00
1	2	0	1	4362,63	3666,35	3565,76	3556,32	3578,62	3,00%	2,70	1,00	4,00	809,80	486,00	996,00
Scenario 3: stepwise release - extend time window by 30 minutes															
1	1	1	0	3991,34	2917,59	2822,94	2772,22	2893,83	4,98%	7,00	3,00	32,00	938,60	372,00	2030,00
1	1	0	1	4024,96	2961,08	2856,69	2809,28	2895,23	5,13%	7,70	5,00	15,00	1062,00	609,00	1593,00
1	2	1	0	4247,34	3051,48	2950,10	2921,23	2999,91	4,27%	6,60	4,00	10,00	988,60	531,00	1352,00
1	2	0	1	4228,49	3022,38	2905,36	2887,45	2929,23	4,46%	8,80	6,00	13,00	1251,90	934,00	1688,00
Scenario 3 (2) stepwise release - extend time window by 60 minutes															
1	1	1	0	4341,69	2742,39	2645,87	2610,65	2680,43	4,80%	13,30	7,00	32,00	1055,70	602,00	2030,00
1	1	0	1	4237,07	2756,63	2626,32	2591,16	2661,79	6,00%	14,70	7,00	23,00	1068,90	527,00	1598,00
1	2	1	0	4368,21	2771,73	2668,87	2618,35	2728,68	5,53%	13,50	4,00	26,00	949,10	262,00	1536,00
1	2	0	1	4407,10	2845,31	2715,74	2675,00	2748,90	5,99%	15,50	10,00	23,00	1169,30	622,00	1690,00

Figure 42: VNS with 100 iterations

1.000 iterations without improvement

μ	λ	α_1	α_2	constr. heuristic	After 3-opt	Cost Avg	Cost Min	Cost Max	Improve-ment	Time (s) Avg	Time (s) Min	Time (s) Max	Iterations Avg	Iterations Min	Iterations Max
Scenario 1: with stepwise release															
1	1	1	0	4179,00	3562,99	3450,23	3440,20	3456,07	3,45%	23,10	15,00	34,00	5291,70	3592,00	8147,00
1	1	0	1	4178,42	3549,34	3428,21	3418,47	3447,03	3,69%	26,60	15,00	247,00	6133,20	3592,00	18068,00
1	2	1	0	4315,02	3628,73	3509,75	3487,70	3523,40	3,89%	28,70	18,00	37,00	7401,10	4443,00	10380,00
1	2	0	1	4362,63	3666,35	3515,39	3501,41	3525,18	4,50%	24,70	15,00	35,00	6816,20	4242,00	9372,00
Scenario 3: stepwise release - extend time window by 30 minutes															
1	1	1	0	3991,34	2917,59	2722,72	2697,48	2746,81	7,54%	71,90	53,00	246,00	9501,50	6529,00	17774,00
1	1	0	1	4024,96	2961,08	2741,72	2694,95	2763,61	8,99%	82,50	49,00	134,00	10448,20	5960,00	17976,00
1	2	1	0	4247,34	3051,48	2825,60	2795,72	2879,78	8,38%	88,90	50,00	129,00	11897,10	7498,00	16701,00
1	2	0	1	4228,49	3022,38	2803,93	2727,17	2846,73	9,77%	90,20	47,00	161,00	11596,70	5710,00	18068,00
Scenario 3 (2) stepwise release - extend time window by 60 minutes															
1	1	1	0	4341,69	2742,39	2531,07	2490,20	2549,95	9,20%	136,10	72,00	246,00	10904,50	6529,00	17774,00
1	1	0	1	4237,07	2756,63	2523,92	2495,54	2552,19	9,47%	142,30	75,00	195,00	10354,00	6505,00	14803,00
1	2	1	0	4368,21	2771,73	2537,77	2506,70	2558,17	9,56%	154,30	89,00	198,00	12436,30	7381,00	17168,00
1	2	0	1	4407,10	2845,31	2596,96	2563,91	2626,28	9,89%	173,20	95,00	247,00	12664,20	7010,00	16637,00

Figure 43: VNS with 1000 iterations

10.000 iterations without improvement

μ	λ	α_1	α_2	constr. heuristic	After 3-opt	Cost Avg	Cost Min	Cost Max	Improve-ment	Time (s) Avg	Time (s) Min	Time (s) Max	Iterations Avg	Iterations Min	Iterations Max
Scenario 1: with stepwise release															
1	1	1	0	4179,00	3562,99	3443,94	3435,49	3454,42	3,58%	117,70	85,00	160,00	27169,90	19179,00	33965,00
1	1	0	1	4178,42	3549,34	3414,44	3402,53	3427,66	4,14%	112,00	61,00	1373,00	29224,80	16824,00	117335,00
1	2	1	0	4315,02	3628,73	3499,13	3474,95	3515,00	4,24%	111,20	83,00	160,00	29108,50	21374,00	42516,00
1	2	0	1	4362,63	3666,35	3503,83	3489,20	3519,78	4,83%	114,50	76,00	151,00	32441,60	21338,00	45557,00
Scenario 3 stepwise release - extend time window by 30 minutes															
1	1	1	0	3991,34	2917,59	2693,25	2659,79	2726,93	8,84%	332,70	231,00	1373,00	45711,10	30094,00	117335,00
1	1	0	1	4024,96	2961,08	2713,91	2661,56	2742,74	10,12%	388,20	209,00	790,00	52822,50	33408,00	96765,00
1	2	1	0	4247,34	3051,48	2785,84	2754,48	2804,58	9,73%	502,90	347,00	723,00	67995,40	49795,00	92289,00
1	2	0	1	4228,49	3022,38	2761,57	2731,84	2787,04	9,61%	460,20	302,00	667,00	62929,60	40591,00	95619,00
Scenario 3 (2) stepwise release - extend time window by 60 minutes															
1	1	1	0	4341,69	2742,39	2492,27	2465,75	2539,39	10,09%	763,80	493,00	1373,00	67894,40	47851,00	117335,00
1	1	0	1	4237,07	2756,63	2510,45	2480,35	2542,25	10,02%	586,10	330,00	951,00	47319,80	29651,00	76208,00
1	2	1	0	4368,21	2771,73	2507,95	2467,81	2532,81	10,96%	793,50	403,00	1161,00	65486,30	32054,00	95620,00
1	2	0	1	4407,10	2845,31	2559,83	2521,97	2585,94	11,36%	665,80	313,00	1068,00	58508,40	28374,00	78937,00

Figure 44: VNS with 10.000 iterations

100.000 iterations without improvement

	μ	λ	α_1	α_2	constr. heuristic	After 3-opt	Cost Avg	Cost Min	Cost Max	Improve-ment	Time (s) Avg	Time (s) Min	Time (s) Max	Iterations Avg	Iterations Min	Iterations Max
Scenario 1: with stepwise release																
	1	1	1	0	4179,00	3562,99	3439,08	3431,19	3449,86	3,70%	512,38	441,00	750,00	119446,25	105795,00	144128,00
	1	1	0	1	4178,42	3549,34	3419,55	3395,38	3440,50	4,34%	500,00	380,00	8249,00	116641,50	105795,00	494152,00
	1	2	1	0	4315,02	3628,73	3497,87	3487,03	3506,79	3,90%	520,88	387,00	930,00	134958,00	106759,00	218818,00
	1	2	0	1	4362,63	3666,35	3509,39	3495,97	3522,91	4,65%	466,29	380,00	570,00	131264,57	113132,00	171346,00
Scenario 3: stepwise release - extend time window by 30 minutes																
	1	1	1	0	3991,34	2917,59	2673,83	2647,25	2707,63	9,27%	1826,75	1240,00	4956,00	228630,25	157420,00	407134,00
	1	1	0	1	4024,96	2961,08	2700,33	2662,47	2733,83	10,08%	1376,75	948,00	2303,00	190594,00	131698,00	331042,00
	1	2	1	0	4247,34	3051,48	2779,06	2754,66	2799,49	9,73%	2107,25	1229,00	3272,00	268763,75	163058,00	383928,00
	1	2	0	1	4228,49	3022,38	2769,19	2746,57	2785,90	9,13%	2065,00	1217,00	3404,00	282569,00	181464,00	393710,00
Scenario 3 (2) stepwise release - extend time window by 60 minutes																
	1	1	1	0	4341,69	2742,39	2480,05	2439,18	2510,06	11,06%	3384,00	2390,00	4956,00	279379,75	157420,00	407134,00
	1	1	0	1	4237,07	2756,63	2510,92	2490,36	2529,93	9,66%	3460,75	1684,00	4525,00	261796,00	143295,00	358366,00
	1	2	1	0	4368,21	2771,73	2516,34	2492,07	2551,19	10,09%	3067,13	1980,00	4208,00	255586,75	168219,00	338950,00
	1	2	0	1	4407,10	2845,31	2550,36	2520,01	2583,01	11,43%	3762,71	1828,00	8249,00	305955,14	182404,00	494152,00

Figure 45: VNS with 100.000 iterations

μ	λ	α_1	α_2	Difference Or-Opt and VNS in %					Avg. Iterations					Avg. Time				
				10	100	1.000	10.000	100.000	10	100	1.000	10.000	100.000	10	100	1.000	10.000	100.000
Scenario 1: with stepwise release																		
1	1	1	0	0,54%	2,71%	3,45%	3,58%	3,70%	29,80	843,70	5.291,70	27.169,90	119.446,25	0,00	3,50	23,10	117,70	512,38
1	1	0	1	0,36%	2,91%	3,69%	4,14%	4,34%	18,20	935,90	6.133,20	29.224,80	116.641,50	0,00	3,60	26,60	112,00	500,00
1	2	1	0	0,57%	2,65%	3,89%	4,24%	3,90%	21,30	900,40	7.401,10	29.108,50	134.958,00	0,00	3,50	28,70	111,20	520,88
1	2	0	1	1,29%	3,00%	4,50%	4,83%	4,65%	45,60	809,80	6.816,20	32.441,60	131.264,57	0,10	2,70	24,70	114,50	466,29
Scenario 3: stepwise release - extend time window by 30 minutes																		
1	1	1	0	0,22%	4,98%	7,54%	8,84%	9,27%	18,50	938,60	9.501,50	45.711,10	228.630,25	0,00	7,00	71,90	332,70	1826,75
1	1	0	1	0,29%	5,13%	8,99%	10,12%	10,08%	20,30	1.062,00	10.448,20	52.822,50	190.594,00	0,40	7,70	82,50	388,20	1376,75
1	2	1	0	0,52%	4,27%	8,38%	9,73%	9,73%	18,80	988,60	11.897,10	67.995,40	268.763,75	0,20	6,60	88,90	502,90	2107,25
1	2	0	1	0,52%	4,46%	9,77%	9,61%	9,13%	18,20	1.251,90	11.596,70	62.929,60	282.569,00	0,10	8,80	90,20	460,20	2065,00
Scenario 3 (2): stepwise release - extend time window by 60 minutes																		
1	1	1	0	0,99%	4,80%	9,20%	10,09%	11,06%	23,80	1.055,70	10.904,50	67.894,40	279.379,75	0,50	13,30	136,10	763,80	3384,00
1	1	0	1	0,93%	6,00%	9,47%	10,02%	9,66%	25,20	1.068,90	10.354,00	47.319,80	261.796,00	0,70	14,70	142,30	586,10	3460,75
1	2	1	0	0,46%	5,53%	9,56%	10,96%	10,09%	17,60	949,10	12.436,30	65.486,30	255.586,75	0,20	13,50	154,30	793,50	3067,13
1	2	0	1	0,69%	5,99%	9,89%	11,36%	11,36%	22,50	1.169,30	12.664,20	58.508,40	305.955,14	0,10	15,50	173,20	665,80	3762,71

Figure 46: Overview - all VNS runs

Appendix C Abstract

In dieser Diplomarbeit wird ein auf Heuristiken basierter Lösungsansatz für das Free Newspaper Delivery Problem präsentiert. Das Free Newspaper Delivery Problem ist eine Variante des Vehicle Routing Problems, bei dem der Produzent Entscheidungen, über Menge, Lieferzeitpunkt und Routenplanung für die Verteilung aller an einem Tag produzierten Zeitungen an Strassenbahn und U-Bahn-Stationen, trifft. Dabei ist zu beachten, dass im Planungszeitraum die Anzahl der Zeitungen in den Stationen mindestens Eins und an dessen Ende Null sein muss. Grundlage für die Construction Heuristik ist eine Dekompositionsmethode, die die Lösung der vielfältigen Entscheidungen in zwei Phasen teilt. Zuerst wird ein Lieferplan erstellt, danach erfolgt die Erstellung der Routenpläne. Als Improvement Heuristiken für dieses Problem wurden OR-Opt und Variable Neighbourhood Search verwendet. Der Fokus der Arbeit liegt in der Erstellung einer Lösungsmethode und deren Test und Analyse anhand einer Testinstanz.

Appendix D Lebenslauf

Ing. Sabine Krenbek Bakk.

Lebenslauf

geboren am 10.6.1977 in Wien

E-Mail: sabine.krenbek@gmx.net

Berufspraxis

	B.I.T. Blue IT- Services Ges.m.b.H. (an IBM-Company)
Seit 07/2008	Technical Solution Manager
10/2005 – 07/2008	Teamleiter SAP Operation
12/2004 - 09/2005	Senior SAP Basis Consultant
	METROPOLITAN Datenservice Ges.m.b.H.
07/1998 – 11/2004	SAP Basis Consultant
10/1997 - 06/1998	Buchhaltungssachbearbeiter

Ausbildung

Seit 2007	**Universität Wien** Betriebswirtschaft - Masterstudium Schwerpunkte: Finanzdienstleistungen, Transportation Logistics
1996 – 2006	**Universität Wien** Betriebswirtschaft Abschluss Bakkalaureat:15.11.2006
1991 - 1996	**Höhere Technische Bundeslehranstalt Wien 22** Fachrichtung Elektronische Datenverarbeitung und Organisation