A CLUSTER-BASED OPTIMIZATION APPROACH FOR THE VEHICLE ROUTING PROBLEM WITH TIME WINDOWS

Michael Bögl

Institute for Production and Logistics Management
Johannes Kepler University
Altenberger Straße 69
4040 Linz, Austria

michael.boegl@jku.at

ABSTRACT
In this paper we describe a simple cluster approach for solving the Vehicle Routing Problem with Time Windows (VRPTW). The idea is to combine heuristic and exact approaches to solve the problem. This is done by decomposing the problem into two sub-problems. The first step consists of building customer clusters, in the second step, those clusters are solved with an exact method. We present computational results for the Solomon benchmark problems.

Keywords: vehicle routing, time windows, VRPTW, metaheuristic, tabu search, multi-objective optimization

1. INTRODUCTION

Transportation problems arise in a lot of companies. In a typical supply chain these are the sourcing of raw materials, the in-company distribution, the distribution to the retailers and reverse logistics. Furthermore it is apparent in the supply of services like waste collection, public transport, mail distribution and others. Transportation not only causes a lot of cost but also pollutes the environment. The pressure of the market forces firms to cut cost down. Computerized planning of the distribution process have shown to produce considerable savings of about 5% to 20% (depending on the application) (Toth and Vigo 2001).

Research for the Vehicle Routing Problem (VRP) was started nearly 50 years ago by Dantzig and Ramser (1959). They tried to solve a route planning problem for gasoline distribution, where a set N of n station points (or customers respectively nodes) $p_1, p_2, ..., p_n$ are given and deliveries to them are made from node p_0, called terminal point or depot. A delivery vector is given, it specifies the demand q_i for every node i. A fleet K of k trucks is available; each truck has a capacity Q, where $Q > \max q_i$, i.e. the demand of every node is smaller than the truck capacity. And there is an arc set A, where $c_{ij} \in A$ are the travel cost from node p_i to p_j. They developed the first Linear Programming (LP) based approach for the VRP. At that time Integer Linear Programming (ILP) was in the beginning of its development, as noted in Dantzig and Ramser (1959).

The next milestone was set by Clarke and Wright (1964) with their savings heuristic. Their approach is simple and fast, which explains its popularity even nowadays (Laporte and Semet 2001). Many solution concepts for more complex problems (with a lot of different constraints) are based on that heuristic or use it to compute an initial solution.

Those two methods present two main research directions which were taken the last decades: exact versus heuristic approaches. A long time those two approaches were not combined. This has different causes. Among other things, by using heuristic approaches one spoils the chance to find the global optimum. On the other hand: exact approaches may take an inconsiderable long computation time (to find the optimum). For practitioners there is another important reason: most of the time it is not necessary to find the best solution, a solution which is better than current practice is often enough (Wren 1998).

From the first appearance of the VRP until now a lot of different generalizations evolved. Among others there are the VRP with multiple depots (MDVRP, MVRP), the VRP with stochastic demands (SDVRP, SVRP), Pickup and Delivery VRP (PDVRP), VRP with Backhauls (VRPB), or the VRP with Time Windows (VRPTW) (Dorronsoro 2007; Toth and Vigo 2001).

This work focuses on the VRPTW. It is NP-hard (cf. Cordeau et al. 2001) and consists of the following constraints: every node has a time window, i.e. a node p_i may only be serviced between its earliest arrival time (a_i) and latest arrival time (b_i). Every stop at a node takes a certain service time (s_i). If a vehicle arrives before a_i it must wait until then to begin service. After servicing the node the vehicle may leave and continue to service the next node or return to the depot. Every vehicle must return to the depot before a certain time, i.e. every route has a maximal time. All other constraints are the same as in the VRP (Cordeau et al. 2001). Before we can give a formal description of the VRPTW we need to define the set $V = N \cup \{p_0\}$ and the following variables: x_{ijk} define the flow of the

vehicles, i.e. x_{ijk}, $(i, j) \in A, k \in K$ is equal to 1 if arc (i, j) is serviced by vehicle k, 0 otherwise. Time variables w_{ik} define the time of service of vehicle k at node i.

Now the VRPTW can be formally described:

$$\min \sum_{k \in K} \sum_{(i,j) \in A} c_{ij} x_{ijk} \quad (1)$$

subject to

$$\sum_{k \in K} \sum_{j \in V} x_{ijk} = 1 \quad \forall i \in N \quad (2)$$

$$\sum_{j \in N} x_{0jk} = 1 \quad \forall k \in K \quad (3)$$

$$\sum_{i \in V} x_{ihk} - \sum_{\substack{j \in V \\ i \neq j}} x_{hjk} = 0 \quad \forall h \in N, \forall k \in K \quad (4)$$

$$\sum_{i \in N} x_{i0k} = 1 \quad \forall k \in K \quad (5)$$

$$\sum_{i \in N} q_i \sum_{j \in V} x_{ijk} \leq Q \quad \forall k \in K \quad (6)$$

$$w_{ik} + s_i + c_{ij} - M(1 - x_{ijk}) \leq w_{jk} \quad \forall i, j \in V, i \neq j, \forall k \in K \quad (7)$$

$$a_i \leq w_{ik} \leq b_i \quad \forall i \in V, \forall k \in K \quad (8)$$

$$x_{ijk} \in \{0,1\} \quad \forall (i, j) \in A, \forall k \in K \quad (9)$$

$$w_{ik} \geq 0 \quad \forall i \in N, \forall k \in K \quad (10)$$

The objective function (1) minimizes the length of the routes. Constraint (2) ensures that every node is visited exactly once. Constraints (3), (4) and (5) force every vehicle to leave the depot, move from one node to the next and finish the tour at the depot. Constraint (6) ensures that the capacity of every vehicle is not exceeded. Constraints (7) and (8) guarantee feasible schedules with respect to time windows. Constraint (9) enforces binary decision variables for the arcs. Finally, constraint (10) forces time variables to be positive. Please note: in this model the travel time equals the travel cost.

In this formal description the objective function minimizes the total route length. This may not always be the main goal. Another often used objective function is to minimize the vehicle fleet size and the total length of the routes. Then the VRPTW is a multi-objective optimization problem. But there are other goals one might want to achieve, e.g. the minimization of the waiting time (e.g. transportation of passengers) or the minimization of the total routing time (e.g. transportation of hazardous materials). In this work our primary optimization goal is the minimization of vehicles/routes and the secondary optimization goal is the minimization of the total length of the routes.

We will continue by giving a short literature review in section 2; section 3 motivates the research direction which this work is aimed at. Section 4 describes our solution method; section 5 shows results of computational experiments and at last section 6 gives an outlook on further ideas, refinements and research directions.

2. LITERATURE REVIEW

The VRPTW was subject of intensive research. Hence numerous different solution concepts were developed. The spectrum reaches from simple one stage heuristics (Solomon 1987), two stage heuristics (cluster-first, route-second; route-first, cluster-second) (Dorronsoro 2007; Cordeau et al. 2001), various local search methods (Bräysy, Hasle and Dullaert 2004), metaheuristics (Thangiah 1995; Potvin et al. 1996; Homberger and Gehring 2005) to exact approaches based on mathematical programming (Kallehauge, Larsen and Madsen 2001).

Solomon (1987) introduced three heuristics (I1, I2 and I3). They differ mainly in the weighting function. The I1 heuristic is nowadays the most popular. It is a sequential route construction heuristic and inserts customers into the current route according to a weighting function. It determines the best insertion position for all unrouted customers in the actual route by calculating the weighted sum of the detour and the shift of arrival time of the following customers. The customer for which the difference between the distance to the depot and the weighted sum of the best insertion position is maximized is inserted. This is repeated until all customers are serviced. If no customer can be inserted a new route is started.

A clustering metaheuristic was proposed by Thangiah (1995). He developed a system called GIDEON. The customers are sorted according to their polar coordinates (the depot is origin) and clustered into sectors. An insertion heuristic creates routes. The clusters are then optimized using a genetic algorithm.

One of the most successful metaheuristic approaches for large problem instances was developed by Homberger and Gehring (2005). It is a two phase metaheuristic. In the first phase an Evolution Strategy minimizes the number of routes, in the second phase a Tabu Search metaheuristic minimizes the total length of the routes.

3. MOTIVATION FOR A CLUSTER-BASED APPROACH

Even though instances up to 1000 nodes can be solved to optimality, benchmark problems with 100 customers are not yet solved to optimality (cf. Kallehauge, Larsen and Madsen 2001).

Only problems up to a few dozen of nodes can easily be solved to optimum nowadays (cf. Toth and Vigo 2001). For bigger instances (50 or more nodes) one possibility would be to decompose the problem into smaller sub-problems, which then can be solved faster by exact methods.

We think that sophisticated clustering methods for the VRPTW would contribute to the solution process of large scale problems. Because the problem size decreases, hence the effort to calculate the exact solution for a single cluster decreases too. Additionally, as nowadays the trend goes towards parallel computing (multi-core personal computers, GRID computing) clustering algorithms may contribute to the parallelization of solution concepts.

Here, an optimal clustering is a set of clusters, where every cluster contains the customers of the optimal routes. Due to the presence of time windows complex clusters arise. Figure 1 shows a solution for a 50 node benchmark problem (instance R101 of the Solomon (1987) benchmark problems; a short description is in section 5). The black rectangle in the middle is the depot; the Bezier curve in the upper left corner marks a simple cluster and the Bezier curve next to the depot marks a more complex cluster. Optimal solutions for bigger problems have a similar clustering (cf. Kallehauge, Larsen and Madsen (2001) for the optimal solution of a 100 customer problem).

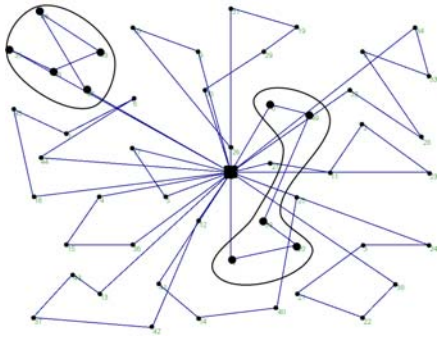

Figure 1: Solution of a 50 Nodes VRPTW Benchmark Problem of the Solomon (1987) Benchmarks

Recently a few researchers developed cluster algorithms for vehicle-routing like problems, e.g. Dondo and Cerdá (2007) propose a cluster based optimization approach for the multi-depot VRPTW.

4. SOLUTION CONCEPT

As mentioned earlier, we developed a cluster based optimization approach, i.e. our focus is to find clusters, from which optimal solutions can be derived. What we do is group customers into clusters and then calculate the route for every cluster. We start with an initial clustering and then iteratively change the customer clusters.

The general flow of the algorithm is depicted in figure 2. At first an initial clustering is calculated. Then the routing problem is solved for every cluster and the quality of the solution is evaluated. The variables *actual_solution* and *best_solution* are initialized with this solution. Now the main optimization loop is entered. A probabilistic Tabu Search metaheuristic optimizes the clusters using different neighborhood operators until a termination criterion is met.

Every solution cycle generates a certain predefined amount of neighbors probabilistically. The quality of these candidate solutions is evaluated and the best non-tabu solution is selected as the new *actual_solution* for the next iteration. If *actual_solution* is better than *best_solution*, *actual_solution* is assigned to *best_solution*. Basically, this is a standard Tabu Search algorithm.

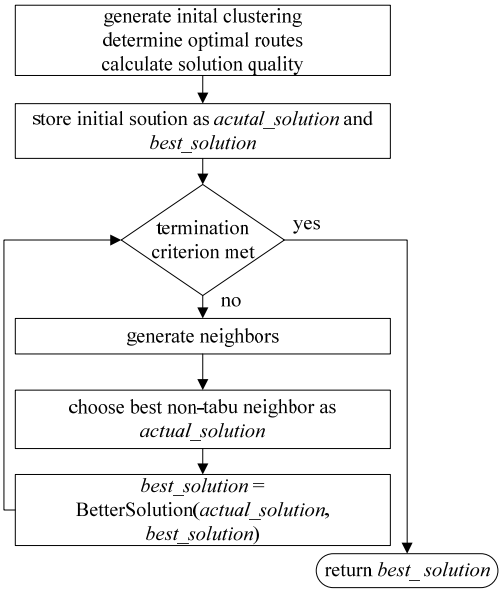

Figure 2: Tabu Search Flow Diagram

4.1. Solution Evaluation

A neighbor respectively a solution in this context is a set of clusters, where a cluster is again a set of customers. A cluster consists of customers who can feasibly be serviced by one vehicle, i.e. every cluster represents one tour.

We somehow must determine the quality of such a cluster. The following possibilities exist:

1. Use a lower bound method
2. Use an upper bound method
3. Use an exact method

Method 1 is fast, but e.g. bounds based on integer relaxation for the VRPTW are rather weak (cf. Cordeau et al. 2001) and we risk infeasible clusters, i.e. clusters for which no feasible route exists. Another fast approach would be to use an upper bound method. Even though one always gets a feasible solution, the disadvantage is that one can not say how good the solution is.

We decided to use an exact method and solve the routing problem to the global optimum. We use the MILP formulation given in section 1 and ILOG CPLEX 11 to solve it.

The disadvantage of this approach is that the execution time greatly depends on the problem instance. For example in Kallehauge, Larsen and Madsen (2001) the execution time for the 25 node problems of the Solomon (1987) benchmark problems varies between

0.1 second (instance R101) and 221087.1 seconds (instance RC207).

Because the calculation of the optimal solution can take a considerable amount of time, we cache results of solved routing problems in a so called cluster table. Each time a cluster is generated, it is looked up there. If it was already encountered, a table entry exists and the according value is taken. If no entry in the table exists, the MILP solver is called, the routing problem is solved and the result (length of the route) is stored in the cluster table.

This has the advantage that if we encounter a cluster again later in the search process, it is not necessary to optimize it again. Additionally we can discard special clusters, which we have not seen yet. Let's consider two clusters (i.e. two sets of customers) $y_1 = \{p_1, p_2, \ldots, p_m\}$ and $y_2 = \{p_1, p_2, \ldots, p_m, \ldots, p_n\}$, where $y_1 \subseteq y_2$. Then, if y_1 is infeasible (i.e. no route exists which satisfies all constraints), y_2 must be infeasible too.

A solution to the routing problem can take two values: either the length of the route or infeasible.

As the VRPTW here is seen as multi-objective optimization problem, it is necessary to keep in mind that our primary optimization goal is the minimization of the number of used vehicles (i.e. number of routes). The secondary optimization goal is the minimization of the length of the routes.

Objective function values are sorted lexicographically, i.e. a solution with two routes is always better than a solution with three routes regardless of the tour length.

4.2. Initial Clustering

Before we can start the main search process we need an initial clustering. We implemented two methods.

The first initialization method is based on the Solomon I1 heuristic. We partition the customers into clusters according to the routes generated by the I1 heuristic. For example if we solve 25 customer problem C104 with the Solomon algorithm we may get, depending on the parameters, the following three routes (0 is the depot, numbers 1 to 25 are the nodes; routes are written in a permutation like style, i.e. (0, 2, 5, 0) means the vehicle starts at node 0 (depot), visits node 2 then node 5 and finally returns to the depot): {(0, 13, 15, 17, 18, 19, 16, 14, 12, 0), (0, 20, 24, 25, 22, 11, 9, 23, 21, 10, 6, 2, 0), (0, 2, 7, 8, 4, 1, 3, 5, 0)} with a total length of 254.88. We now use these three customer sets as clusters and solve the according routing problem using a MILP solver. Every route is solved separately, i.e. in this example the MILP solver is called three times. The following three routes are returned: {(0, 13, 17, 18, 19, 15, 16, 14, 12, 0), (0, 24, 25, 2, 6, 11, 9, 10, 23, 22, 21, 20, 0), (0, 2, 7, 8, 4, 1, 3, 5, 0)} with a length of 233.65. Our initial solution has the quality (3, 233.65).

The second method we implemented is called one-node-per-cluster. As the name suggests, every customer builds a separate cluster. It is a very naïve method. The idea in using such a simple initialization is that if we encounter a lot of small infeasible clusters in the beginning of the search, the algorithm has gathered more knowledge about the problem and can discard more infeasible clusters later.

4.3. Neighborhood Operators

Neighborhood operators are used to produce new solutions and a crucial element of the Tabu Search metaheuristic (Gendreau and Potvin 2005).

Here the generation of a neighbor consists of two steps. At first the clusters are modified. The second step is the calculation of the optimal routes for the changed clusters. The length of the optimal route reflects the quality of the cluster.

Two different neighborhood operators were implemented: exchange and move. Those two types are typical operators used in the context of vehicle routing (cf. Homberger and Gehring 2005).

The move neighborhood operator takes the actual solution, selects two random clusters, e.g. $c_1 = \{p_a, p_b, p_c, \ldots, r, \ldots, p_k\}$ and $c_2 = \{p_n, p_o, p_q, p_r, \ldots, p_u\}$ and moves a random, non-forbidden node r from c_1 to c_2.

Now both new clusters are looked up in the cluster table. If it already contains an entry for both or either of the new clusters, the according quality value is taken. Missing values are calculated using the MILP solver.

If c_2 is infeasible the move is reversed, node r is temporary forbidden and another node of set c_1 is chosen. This step is repeated until a feasible solution is found or all nodes of c_1 are forbidden during this neighborhood operation. All encountered trial solutions are stored in the cluster table.

The move operator is able to decrease the number of routes in contrast to the exchange operator.

The exchange operator selects two random clusters $c_1 = \{p_a, p_b, \ldots, k, \ldots, p_i\}$ and $c_2 = \{p_s, p_t, \ldots, r, \ldots, p_u\}$. From cluster c_1 a random, non-forbidden node k is chosen, from cluster c_2 a random, non-forbidden node r is chosen. Those two nodes are exchanged, and yield to the following clusters: $c_1 = \{p_a, p_b, \ldots, p_i, r\}$, $c_2 = \{p_s, p_t, \ldots, p_u, k\}$. Again, both new clusters are looked up in the cluster table and in case of a missing value, the MILP solver is called. In case any of the clusters is infeasible the move is reversed and nodes k and r are not allowed to be exchanged during this neighborhood operation.

4.4. Tabu List

As the Tabu Search metaheuristic is a neighborhood based steepest descent search method it must somehow be ensured that the search process can escape from local optima. This is the purpose of the tabu list (or memory).

Generally, tabu lists are used to control the search process, i.e. short term memory may be used to prevent from cycling and intermediate and long term memory may be used to intensify or diversify the search process (Gendreau and Potvin 2005).

Here, we use short term memory to prevent cycling. No long term or intermediate memory is used.

We do not only store solution attributes but complete solutions. Those solutions must not be selected again.

Now the description of the developed solution concept is complete. In summary, it is a probabilistic Tabu Search with short term memory and two different neighborhood operators.

5. COMPUTATIONAL RESULTS

All computational experiments were done using the Solomon (1987) benchmark instances. They consist of different classes of problems: R, C and RC. Class R contains random node positions. The nodes in class C are clustered. Class RC contains a mixture of class R and class C. Every problem class contains two different sets of problem instances: 1 and 2. Class 1 contains problem instances with narrow time windows and class 2 contains problem instances with wide time windows. There are 6 different problem classes. R1, R2, C1, C2, RC1, RC2. Every problem class contains between 8 and 12 problem instances. Solomon (1987) created problem instances with 25, 50 and 100 customers. Distances are Euclidean and calculated using node coordinates.

We present 4 different test-runs. Each was run twice using another random seed. Table 1 shows the configuration of the 4 test-runs. A test-run was stopped after one of the termination criterions was met: either the iteration counter reached the iteration limit or execution time exceeded the maximal execution time. If more than one neighborhood operator is given, one is chosen randomly for every neighborhood generation.

Table 1: Configuration of the 4 Test Runs

	Test #1	Test #2	Test #3	Test #4
Iterations	1000	1000	1000	1000
Max. Execution Time	15 min.	15 min.	15 min.	15 min.
Neighbor-hood Size	20	20	20	20
Tabulist Length	1000	1000	1000	1000
Initialization	one-node-per-cluster	Solomon I1	one-node-per-cluster	Solomon I1
Neighbor-hood Operator	move	move	move, exchange	move, exchange

Table 2: Averaged (2 Runs) Solution Quality for all 25-Node-Solomon Instances

Label	Test #1		Test #2		Test #3		Test #4	
	V.	Length	V.	Length	V.	Length	V.	Length
C101	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*	*3*	215,57
C102	*3*	*190,74*	*3*	242,15	*3*	*190,74*	*3*	242,15
C103	*3*	203,62	*3*	*190,74*	*3*	208,46	*3*	202,40
C104	*3*	221,53	*3*	203,63	4	256,50	*3*	212,75
C105	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*
C106	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*	3,5	223,55
C107	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*
C108	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*	*3*	*191,81*
C109	*3*	*191,81*	*3*	213,07	*3*	*191,81*	*3*	225,85
C201	*2*	*215,54*	*2*	*215,54*	*2*	*215,54*	*2*	*215,54*
C202	*2*	*215,54*	*1*	*223,31*	*1,5*	*219,43*	*1*	*223,31*
C203	*2*	*215,54*	*1,5*	*241,14*	*2*	*215,54*	*1*	*224,46*
C204	2,5	235,50	*1*	*213,93*	2,5	240,19	*1*	214,35
C205	*1,5*	*256,50*	*1*	*297,45*	*1,5*	*256,50*	*1*	*297,45*
C206	*1,5*	*250,47*	*1*	*287,21*	*2*	*215,54*	*1*	*288,02*
C207	*2*	*215,34*	*1*	*274,78*	2,5	238,98	*1*	*274,78*
C208	*2*	*215,37*	*1*	*229,84*	*2*	*215,37*	*1*	*229,84*
R101	*8*	*618,33*	*8*	*618,33*	*8*	626,77	8,5	670,60
R102	*7*	558,12	*7*	*548,11*	*7*	*548,53*	*7*	*548,53*
R103	*4*	*473,39*	*4*	*495,02*	*5*	473,19	*4*	*490,56*
R104	*4*	425,18	5	498,16	*4*	424,53	4,5	463,89
R105	*5*	*559,84*	*5,5*	*547,35*	*5,5*	*544,13*	*5,5*	*558,90*
R106	*5*	*466,48*	*5*	*466,48*	*5*	*467,72*	*5*	*466,48*
R107	*4*	447,75	*4*	451,42	*4*	446,66	4,5	456,24
R108	*4*	*398,29*	*4*	442,72	*4*	406,47	*4*	443,43
R109	*4*	*467,28*	*4*	*460,52*	7	538,58	*4*	*460,52*
R110	*4*	*449,41*	*4*	*454,42*	*4*	*451,14*	*4*	*464,54*
R111	*4*	436,06	*4*	434,18	*4*	435,49	*4*	449,35
R112	*4*	404,15	*4*	404,10	*4*	407,58	*4*	406,40
R201	*2*	*525,25*	*2*	*525,25*	*2,5*	*502,80*	*2*	*528,93*
R202	*3,5*	*415,04*	*2*	*529,57*	*3,5*	*421,13*	*2*	*528,80*
R203	*3*	416,57	*2*	*447,01*	*3*	404,59	*2*	*474,28*
R204	*2*	*360,79*	*1*	*483,59*	*2,5*	*386,64*	*1*	*479,65*
R205	*2*	*412,81*	*2*	*405,98*	*2*	*419,24*	*2*	*419,20*
R206	*2*	*389,43*	*2*	*395,69*	*2*	*380,76*	*2*	*404,39*
R207	*2*	*372,45*	*1*	*408,60*	*2*	*374,32*	*1*	*408,60*
R208	2	335,69	*1*	*329,33*	2	336,07	*1*	*329,33*
R209	2,5	375,24	*2*	383,24	2,5	376,65	*2*	396,49
R210	*3*	411,61	*2*	*414,18*	*3*	422,58	*2*	*420,74*
R211	3	362,63	*2*	426,20	3	371,21	*2*	419,40
RC101	*4*	463,60	*4*	463,60	4,5	469,56	4,5	470,84
RC102	*3*	*352,74*	*3*	*352,74*	*3*	*352,74*	*3*	*352,74*
RC103	*3*	*333,92*	*3*	*333,92*	*3*	*333,92*	*3*	*333,92*
RC104	*3*	*307,14*	*3*	*307,14*	4	367,10	*3*	*307,14*
RC105	*4*	*412,38*	*4*	*412,38*	*4*	*412,38*	*4*	*412,38*
RC106	*3*	*346,51*	*3*	*346,51*	*3*	*346,51*	*3*	*346,51*
RC107	*3*	*298,95*	*3*	*298,95*	3,5	329,00	*3*	*298,95*
RC108	*3*	*294,99*	*3*	*294,99*	4	394,86	*3*	*294,99*
RC201	*2*	*432,30*	*2*	*450,43*	*2*	*433,26*	*2*	*468,56*
RC202	*3*	*338,82*	-	-	*3*	355,28	-	-
RC203	3	362,34	*1*	*521,51*	3,5	*357,09*	*1*	*506,84*
RC204	*3*	*300,23*	*1*	*344,01*	*3*	*355,04*	*1*	*342,70*
RC205	*3*	*338,93*	*2*	*512,49*	*2,5*	*391,75*	*2*	*529,22*
RC206	*3*	*325,10*	*2*	*450,84*	*2*	*361,47*	*2*	*473,13*
RC207	*3*	314,74	*2*	*427,36*	*3*	330,67	*2*	*474,82*
RC208	*3*	*312,71*	*2*	*393,47*	*4*	*396,23*	*2*	*458,37*

Computational experiments were done on an Intel Pentium 4 Mobile with 768 MB RAM. The solution concept was implemented in C# (.NET Framework 3.5). We used ILOG CPLEX 11 to solve the MILP sub-problem using the model presented in section 1.

Optimal solutions were taken from Kallehauge, Larsen and Madsen (2001). They cut off distances after the second decimal place. We use double precision. Their objective was the minimization of the tour length. They did not publish results for the following instances: R204, RC204 and RC208.

Table 2 shows the averaged results over two test-runs for all 25 customer problems of Solomon (1987). For every test-run, the column labeled V. contains the number of used vehicles (nr. of routes); the column labeled Length contains the total length of the solution. Italic entries mark solutions, which are closer than 0.5% to the given optimal quality (because of the difference in distance). If our solution to a problem instance uses fewer vehicles than the optimal solution, then the value is underlined (and both values are written in italic).

In columns Test #2 and Test #3 results for problem instance RC202 are missing, because the computer run out of memory when calculating the optimal routes for the initial solution.

Results for Test #1 show that our approach finds a solution which is at most 0.5% worse than the reference solution in 40 cases out of 56. In column Test #2 even 44 problems are within the 0.5% gap. In Test #3 we can solve 33 problems close to the optimum and in Test #4 39 problems. the Solomon I1 heuristic clustering seems to be better than the one-node-per-cluster initialization. Results for Test #3 and #4 (exchange and move operator randomly chosen) are inferior compared to Test #1 and #2 (only move operator).

6. CONCLUSIONS

Or approach is able to find near optimal solutions within reasonable time for small problems.

The main drawbacks of the developed solution concept are the lack of an appropriate diversification method and the weak MILP formulation for the VRPTW. The next step will be the development of an appropriate diversification strategy.

The use of exact methods would be the best choice, but the runtime heavily depends on the given problem instance. So we consider adapting an appropriate TSP heuristic method for the TSPTW (cf. Applegate, Bixby, Chvátal and Cook 2006).

REFERENCES

Applegate, D. E., Bixby, R. E., Chvátal, V., Cook, W. J., 2006. *The traveling salesman problem – a computational study*. Princeton:Princeton Series in Applied Mathematics.

Bräysy, O., Hasle, G., Dullaert, W., 2004. A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research* 159:586-605.

Clarke, G., Wright, J., 1964. Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research* 12:568-581.

Cordeau, J.-F., Desaulniers, G., Desrosiers, J., Solomon, M. M., Soumis, F., 2001. VRP with time windows. In: Toth, P., Vigo, D., eds. *The vehicle routing problem*. Philadelphia:SIAM Monographs on Discrete Mathematics and Applications, 157-186.

Dantzig, G. B., Ramser, J. H., 1959. The truck dispatching problem. *Management Science* 6:80-91.

Dondo, R., Cerdá, J., 2007. A cluster-based optimization approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research* 176:1478-1507.

Dorronsoro, B., 2007. *VRP Web*. Available from: http://neo.lcc.uma.es/radi-aeb/WebVRP/ [Accessed June 2008].

Gendreau, M., Potvin, J.-Y., 2005. Tabu search. In: Burke, E. K., Kendall, G., eds. *Search methodologies – introductory tutorials in optimization and decision support techniques*. New York:Springer Science+Business Media, 165-186.

Homberger, J., Gehring, H., 2005. A two-phase metaheuristic for the vehicle routing problem with time windows. *European Journal of Operational Research* 162:220-238.

Kallehauge, B., Larsen, J., Madsen, O., 2001. *Lagrangean duality applied on vehicle routing with time windows – experimental results*. Technical Report, IMM, Technical University of Denmark.

Laporte, G., Semet, F., 2001. Classical heuristics for the capacitated VRP. In: Toth, P., Vigo, D., eds. *The vehicle routing problem*. Philadelphia:SIAM Monographs on Discrete Mathematics and Applications, 109-126.

Potvin, J. Y., Kervahut, T., Garcia, B. L., Rousseau, J. M., 1996: The vehicle routing problem with time windows – part I: tabu search. *INFORMS Journal on Computing* 8:158-164.

Solomon, M. M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research* 35:254-273.

Thangiah, S. R., 1995: Vehicle routing with time windows using genetic algorithms. In: Chambers, L., eds. *Application handbook of genetic algorithms*. Florida:CRC Press, 253-277.

Toth, P., Vigo, D., 2001. An overview of vehicle routing problems. In: Toth, V., Vigo, D., eds. *The vehicle routing problem*. Philadelphia:SIAM Monographs on Discrete Mathematics and Applications, 1-23.

Wren, A., 1998. Heuristics ancient and modern: transport scheduling through the ages. *Journal of Heuristics* 4:87-100.