An effective solution method for the time-dependent vehicle routing problem

Paul Weinhold

Zittau/Görlitz University of Applied Sciences, Görlitz 02826, Germany,
s3pawein@stud.hszg.de

Abstract. The TDVRP is an extension of the classical VRP with time-dependent speed models. In this paper, the problem is divided into two subproblems. The first subproblem is to find valid routes and is solved by a genetic algorithm. We present a simple individual representation, which can be converted to optimal routes by a splitting algorithm. The second part of the problem is to find optimal departure times and is solved with a greedy-heuristic. For the evaluation of the performance, benchmark instances of Solomon [18] and the speed of Figliozzi [9] models were used. We achieved on average 7.63% time savings with 3.63% vehicle additional effort compared to Figliozzis solution method.

1 Introduction

The vehicle routing problem with time windows (VRPTW) models many practical optimization problems in logistics or maintenance of geographically distributed organizations. In practice, the classical model is often not adequate because constant travel times between locations are assumed. Time-varying factors, such as traffic conditions or weather have a significant impact on the actual travel time. Therefore, the travel time between two locations depends on the specific departure time. To represent these external influences, the VRPTW is extended to a time-dependent vehicle routing problem with time windows (TDVRP). In this case the changing driving time is represented by a time-dependent function.

Our use case for the TDVRP, is the maintenance of offshore wind farms by boats. The challenge is to create operational plans for service boats under changing and partly predictable weather conditions. The travel time between two turbines depends on the wind speed and direction for the respective point in time of travel. We use weather forecasts to include this into the planning. A boat can either wait at a wind turbine until conditions are more favorable, or navigate with higher costs directly to the next location. Considering all this, the goal is to find a schedule for each boat, so that the total costs are minimized while performing all necessary maintenance tasks in time.

We divide a TDVRP into two subproblems (Section 3). First, an assignment of customers/jobs to vehicles and a valid order of customers for each vehicle must be found such that all customers can be served in the given time windows

(routing problem). The second subproblem is to find an optimal schedule for each route (scheduling problem). Hashimoto et al. [11] has proven that the scheduling problem in general is also NP-hard.

The present research results are limited to a consideration of TDVRP with hard time windows. This means, the service must necessarily start in the given time window. It follows that an early arrival or later departure is feasible. For evaluation of the results, we consider recently published time-dependent problem instances of Figliozzi [9], where the classical Solomon [18] instances are combined with time-dependent speed models. Our contribution to this line of research is:

- Adaptation of a genetic algorithm [17] for the search of near optimal routes (routing problem) while minimizing the total travel time of a given TDVRP (Section 4). We have found a chromosome representation of the TDVRP, which can be converted in optimal routes by a special splitting algorithm (Section 4.1).
- A simple, practically oriented heuristic for the optimization of schedules (scheduling problem), which are created from the identified routes (Section 5).
- Achieve shorter travel times than [9] by a more complex optimization process without loss of generality.

2 Problem definition

The TDVRP with time windows can be described as a directed graph $G = (V, E)$ with $V = \{0, 1, \ldots, n\}$ and $E = \{(i, j) \mid i \neq j \wedge i, j \in V\}$. In the vertex set V, vertex 0 is the depot and the others are customers. Each customer $i > 0$ is associated with:

- a demand $q_i \geq 0$,
- a service time $s_i \geq 0$ and
- a time window $[e_i, l_i]$ in that the customer i is reachable

The time window $[e_0, l_0]$ is the opening time of the depot. Because the depot is not associated with any service time or demand, $q_i = 0$ and $s_i = 0$. The set $K = \{1 \ldots n\}$ specifies $n \geq 1$ available vehicles. Moreover, there is a fixed maximum capacity $q_{max} \geq 0$ for each vehicle. In contrast to classical VRP we consider additionally that the travel time between vertex i and vertex j is determined by $c(t, i, j)$, where t is the departure time at vertex i. It cannot be assumed that $c(t, i, j)$ is symmetrical for a given t.

For each vehicle/route k we define n decision variables x_n^k, where x_i^k is the ith vertex of route k. In addition we define that the first and last vertex of the route is the depot ($x_0^k = 0$ and $x_n^k = 0$). Further applies that $x_i^k = 0$ for all $i > n$ of route k.

In a classical VRP, arrival and departure times can be calculated with the order of the customer, because the first in, first out (FIFO) [12] property holds.

In TDVRP also the FIFO property holds, but with dynamic speed model so that the arrival and departure times can not be determined exclusively by the order of route k [11]. Therefore, we define for each vehicle k a decision variable d_i^k, which represents the departure time of the ith vertex for route k. Now we can compute the resulting arrival time a_i^k at vertex i using the departure time d_{i-1}^k at the previous for each vehicle k:

$$a_i^k = \begin{cases} d_{i-1}^k + c(d_{i-1}^k, x_{i-1}^k, x_i^k) & \text{if } i > 0 \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Figure 1 illustrates the relationship between arrival, start and departure time for a sample route. The arrival times a_i^k and the start times $(\max(a_i^k, e_{x_i^k}))$ can be each derived from the decision variables d_i^k and x_i^k.

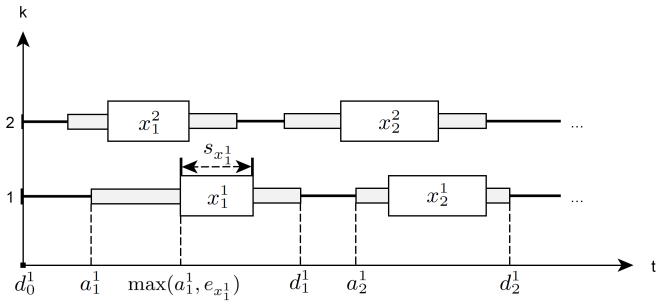

Fig. 1. Arrival, start and departure times for a sample route

Now we can determine the number of customers per route $n(k)$ by using the decision variables x_i^k as follows:

$$n(k) = \sum_{i=1}^{|V|-1} p(k,i), \text{ with } p(k,i) = \begin{cases} 1 & \text{if } x_i^k > 0 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Furthermore, the total travel time $t(k)$ for each vehicle k can be determined as follows:

$$t(k) = \sum_{i=0}^{n(k)} c(d_i^k, x_i^k, x_{i+1}^k) \tag{3}$$

In addition, we need for the objective function the number of solution routes/vehicles. A route is only a part of the solution, if at least one customer is assigned to it. Hence, r is determined as follows:

$$r = \sum_{k \in K} p(k), \text{ with } p(k) = \begin{cases} 1 & \text{if } n(k) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The objective function (5) is primarily to minimize the number of vehicles used and the second objective is to minimize the total travel time. Therefore, the objective function is defined by addition of the number of required vehicles r and the ratio of travel time and maximum possible travel time. Given the above definitions, the TDVRP can be formulated as follows:

$$\text{Minimize } \; r + \sum_{k \in K} \frac{t(k)}{r \cdot l_0} \tag{5}$$

subject to

$$\bigcup_{k \in K} \bigcup_{i \in I} \{x_i^k\} = V \quad I = \{1, \ldots, |V| - 1\} \tag{6}$$

$$x_0^k = 0 \quad \forall k \in K \tag{7}$$

$$x_i^k = 0 \quad \forall k \in K, \forall i = n(k), \ldots, |V| \tag{8}$$

$$x_i^k \in V \setminus \{0\} \quad \forall k \in K, \forall i = 1, \ldots, n(k) \tag{9}$$

$$i \neq j \Rightarrow x_i^k \neq x_j^k \quad \forall k \in K, \; 0 < i \leq n(k), 0 < j \leq n(k) \tag{10}$$

$$\sum_{i=1}^{n(k)} q_{x_i^k} \leq q_{max} \quad \forall k \in K \tag{11}$$

$$a_i^k \leq d_i^k \quad \forall k \in K, \forall i = 1, \ldots, n(k) \tag{12}$$

$$d_i^k \leq d_{i+1}^k \quad \forall k \in K, \forall i = 0, \ldots, n(k) \tag{13}$$

$$d_i^k - \max(e_{x_i^k}, a_i^k) \geq s_{x_i^k} \quad \forall k \in K, \forall i = 0, \ldots, n(k) \tag{14}$$

$$d_i^k \geq e_{x_i^k} \quad \forall k \in K, \forall i = 1, \ldots, n(k) + 1 \tag{15}$$

$$a_i^k \leq l_{x_i^k} \quad \forall k \in K, \forall i = 1, \ldots, n(k) + 1 \tag{16}$$

Constraint (6) guarantees that every customer is achieved by a vehicle. Constraints (7) and (8) shall ensure that each route starts and ends at the depot. Constraint (9) ensures that no invalid vertex occur within a route. Constraint (10) ensures that each customer is served exactly once. Compliance with the maximum capacity of each vehicle is guaranteed by constraint (11). Constraints (12) to (16) are related to time windows and guarantee the feasibility of the schedule for each vehicle. In particular, constraints (12) and (13) guarantee the temporal order of each route. Constraint (14) ensures that the time window for the customer i between earliest possible start and departure time is sufficient for the specific service time s_i. Finally, the last constraints (15) and (16) ensure that each customer i is served in the specific time window $[e_i, l_i]$. A lower bound on the time of arrival is not defined. It follows that a vehicle can reach a customer i before the earliest time e_i, but it can not be started earlier with the service.

3 Solution approach

Due to the time-dependent speeds, it is not easily possible to indicate a common meta-heuristic for this problem. We therefore split the NP-hard TDVRP into two subproblems and search schedules for each vehicle as near as possible to optimum of the objective function. The subproblems are derived from the two decision variables defined above:

1. *Routing problem*: Finding feasible possible optimal routes for each vehicle k.
2. *Scheduling problem*: Finding possible optimal valid departure times d_i^k for each customer i and vehicle k.

Because routes and departure times are coupled together by the constraints (14) to (16), these subproblems can not be solved separately without restrictions. Hence, we assume that for a practical solution, optimizing the routes has a larger influence than optimizing the schedules. We presuppose realistic speed models for the travel time function.

When routing, the optimization of schedules is neglected. It then applies to the departure time $\tilde{\mathbf{d}}_i^k$ and arrival time $\tilde{\mathbf{a}}_i^k$ of a vehicle k following:

$$\tilde{\mathbf{d}}_0^k = 0 \tag{17}$$

$$\tilde{\mathbf{d}}_i^k = \tilde{\mathbf{a}}_i^k + \max(e_{x_i^k}, \tilde{\mathbf{a}}_i^k) + s_{x_i^k} \quad \forall i = 1, \ldots, n(k) \tag{18}$$

$$\tilde{\mathbf{a}}_i^k = \tilde{\mathbf{d}}_{i-1}^k + c(\tilde{\mathbf{d}}_{i-1}^k, x_{i-1}^k, x_i^k) \quad \forall i = 1, \ldots, n(k) \tag{19}$$

Thus, the vehicle travels as in classical VRP immediately after the service to the next customer. We can do this simplification without violating the constraints, because the FIFO principle is applied in the model. The thus separated routing problem can now be easily optimized using a meta-heuristic. For this purpose, a genetic algorithm was developed, which is described in section 4.

In the second part of the problem, schedules for fixed routes now have to be optimized in terms of total travel time. Because the travel time between two locations depends on the departure time, the total travel time of each route can be minimized by skillfully delaying departure times. For this purpose, a simple heuristic H_s (Section 5) is applied as a final step.

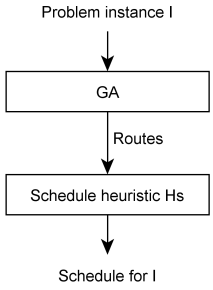

Fig. 2. Solution process

Finally, the process $s(I) = H_s(\mathrm{GA}(I))$ shown in figure 2 are used for the optimization of the problem instance I.

4 Solution of the routing problem with GA

In the following the search heuristic for the optimization of the routes will be considered in more detail. A particular challenge is the representation of the routes in the chromosome, which is explained in section 4.1.

4.1 Routes representation with chromosome

In typical GA's a chromosome is modeled as a sequence of positive integers so that many well-known genetic operations can be easily applied. If each integer value is assigned to a customer, the chromosome can be interpreted directly as a route for a travel salesman problem (TSP). In a VRP, this is not sufficient, because multiple routes must be modeled as a solution. Hence, we interpret the sequence $S = (c_1, \ldots, c_n)$ simply as a order in which n customers must be served. Each vehicle is assigned to a partial route of this sequence. For example, the subroutes $(4, 1)$ and $(3, 2, 5)$ can be derived from the sequence $(4, 1, 3, 2, 5)$, but not $(4, 2, 5)$ and $(1, 3)$.

The goal is to generate optimal valid routes from a given sequence of all customers, where the constraints (7) to (16) must be considered for each subroute. For this purpose, we adapt the splitting procedure by Prins [17] for the TDVRP to determine the best solution for the routing problem for a given chromosome (see algorithm 1).

Algorithm 1 split

Require: $S_c = (c_1, \ldots, c_n)$ for a given chromosome c
  $E \leftarrow$ feasibleRoutes(S_c)
  $Q \leftarrow$ shortestPaths(E)
  $shortestTravelTime \leftarrow \infty$
  $optimalSplitting \leftarrow \{\}$
  for all $q \in Q$ **do**
    if travelTime(q) $<$ $shortestTravelTime$ **then**
      $optimalSplitting \leftarrow q$
      $shortestTravelTime \leftarrow$ travelTime(q)
    end if
  end for
  return $optimalSplitting$

First, we determine, in accordance with the constraints for a given chromosome, the set E of all feasible routes. A route is defined by a 2-tuple $(i-1, j)$, where i is the starting index and j is the ending index of the route in the chromosome sequence S.

Then we define a auxiliary directed acyclic graph (DAG) $G = (V, E)$ with $V = \{0, 1, \ldots, n\}$, $E = \{(i-1, j) \mid \text{feasible}(i, j, S) \wedge i, j \in \{1, \ldots, n\}\}$ and $n = |S|$. Now we determine the set Q of shortest paths from vertex v_0 to v_n. Shortest path algorithms for DAG's with $\mathcal{O}(|V| + |E|)$ time complexity are well known (eg. Cormen et al. [7]).

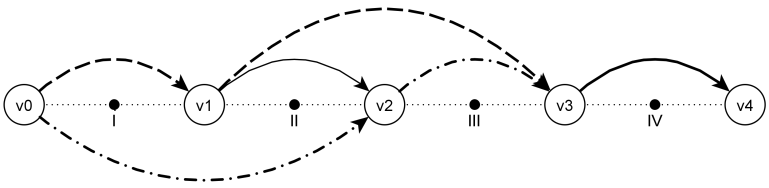

Fig. 3. Auxiliary DAG G

Figure 3 illustrates an example of a possible DAG for a chromosome with the sequence $S = (\mathrm{I}, \mathrm{II}, \mathrm{III}, \mathrm{IV})$. In this DAG, we find two different shortest paths $Q = \{(v_0, v_2, v_3, v_4), (v_0, v_1, v_3, v_4)\}$. In relation to the routing problem, this means that $s_1 = \{(\mathrm{I}, \mathrm{II}), (\mathrm{III}), (\mathrm{IV})\}$ and $s_2 = \{(\mathrm{I}), (\mathrm{II}, \mathrm{III}), (\mathrm{IV})\}$ are respectively

possible solutions. Therefore, the optimal splitting for this chromosome results in exactly three routes.

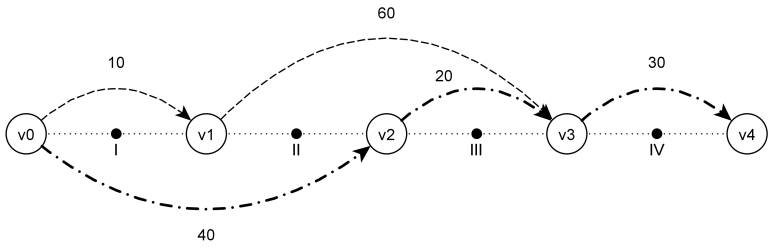

Fig. 4. Auxiliary DAG G with travel time weights

Finally, the best route splitting must be found for the final solution. For this purpose, the travel time of the edges is considered and the path of the set Q with the lowest total travel time is selected. In the example, therefore, s_2 is selected as the final solution with a total travel time of 90 (See figure 4).

The fitness value of an individual can now be calculated by means of splitting of the chromosome c as follows:

$$\text{fitness}(S_c) = \left(|R| + \sum_{r \in R} \frac{\text{travelTime}(r)}{|R| \cdot l_0} \right)^{-1}, \text{ with } R = \text{split}(S_c) \tag{20}$$

With the above presented route-first, cluster-second method always valid routes can be produced from any sequence. Thus, all genetic operations may be performed without consideration of the problem-specific constraints. Hence, the validity of the final solution depends only on the number of the resulting routes and must not exceed the number of available vehicles. Since the number of routes is incrementally minimized by the GA iterations, a validation of this constraint is only necessary at the end.

4.2 Population

The population P is modeled by a set of chromosomes, with a fixed population size p_{max}. In each iteration step a subset P' of the population P is selected with the fitness-proportionate roulette-wheel selection (RWS), also called stochastic sampling with replacement [4]. The number of selected individuals depends on the fixed rate selection s_r. After the execution of different genetic operations, P' and P are combined and reduced to the fixed size p_{max} (replacement phase).

Before in each iteration of the GA this process can be performed, a start population must be generated at the initial phase. In the traditional GA individuals are randomly generated. Because this technique ensures a large search

space, it often also requires a longer period of time to achieve good quality results. In contrast, when only good individuals constitute the initial population, the probability is very high that the GA converges quickly in a local optima, since the diversity of the population at the beginning is very low.

In order to combine the advantages of both methods, there are some hybrid population seeding techniques [16]. For the current use case, a simple hybrid method has been implemented. At the beginning, four solutions are computed using the heuristics $H_1, \ldots, H_4$ from Clarke and Wright [6], Solomon [18], Ioannou et al. [13], and a own developed parallel nearest neighbor heuristic. The routes of each solution are concatenated into a chromosome. Subsequently the population is filled with generated chromosome by a pure random based heuristic H_0, so that the required population size is achieved. The use of different heuristics and a high proportion of randomly generated individuals, guarantees an evolutionary leap while maintaining a high diversity in the initial population.

4.3 Crossover operators

For the reproduction process of GA recombinations play a crucial role to generate new individuals. Two parents from the set of selected individuals are combined into two child individuals. For further reproduction processes only the fittest of the two children is maintained. In the literature, a number of crossover operators for different problems are described. For the TDVRP, the following three suitable crossovers have been implemented:

- Cyclic crossover (CX) by Oliver et al. [15]
- Ordered crossover (OX) by Davis [8]
- Non-wrapping order crossover (NWOX) by Cicirello [5]

We summarize the three crossovers together in the set $X = \{\text{CX}, \text{OX}, \text{NWOX}\}$. Each selected individual is crossed with another randomly selected individual. The corresponding crossover operator is randomly chosen from X. We have studied different combinations of crossover operations and even a single operator and found that a random mix of all three operators leads to a more robust genetic algorithm for the TDVRP.

4.4 Mutation operators

Because the GA can come to better solution by using mutation, after recombination, the new individuals are mutated with a probability p_m. The following three simple operators are used, which have been analyzed by Abdoun et al. [1]:

- Twors mutation
- Centre inverse mutation (CIM)
- Reverse sequence mutation (RSM)

All mutation operators are often used for the TSP. We assume that these are also suitable for the TDVRP what still needs to be studied separately. These mutations are collected in the set $M = \{\text{TWORS}, \text{CIM}, \text{RSM}\}$, wherein a specific mutation of this set is randomly selected for each individual. All of these operators can be performed directly on the sequence S_c of a chromosome.

4.5 Replacement

An impact on the quality and performance of the solution, also has the replacement strategy after each iteration. In this phase, the objective is to replace the best possible individuals by poorer, but at the same time keep the diversity of the population. This goal can be achieved by restricted tournament replacement (RTR) [10]. For the realization we have also studied other replacement strategies [14] and found that RTR leads to better results and faster convergence for the TDVRP. In RTR, each individual x of a new population P' is incorporated in the original population P using the following procedure:

1. Let y be the individual from population P that is most similar to x, in terms of genotypic distance.
2. Replace y with x, if x is better, otherwise discard x.

Because we represent a chromosome by a sequence of integers, we can understand simplified the genotypic distance of two chromosomes by the Hamming distance between the two corresponding sequences.

4.6 GA workflow

In general, the GA is an iteration with stepwise approximation to the optimum. It must be ensured that the process stops after a determinable time while a solution near the optimum results. For this it is necessary to detect during the iteration, whether the process converges. If the optimal value of the objective function can be estimated or even known, then this value can be simply used as the termination criterion. Due to the complexity of the TDVRP an estimate of the objective function is difficult to perform.

Similar to [17], we introduce an upper bound α_{max} for the number of iterations of the GA's. Because in most cases the GA converges before reaching the upper limit, a further stopping criterion is needed. Ideally, each iteration resulting in an improvement at least one individual of the population. However, that is not always the case because often worsening is necessary to leave a local optimum. Therefore we define as an additional stopping criterion, a maximum number of iterations β_{max} without improvement of any individual. Once one of these limits is exceeded, stops the GA and the so far fittest known individual is considered as the final solution of the problem instance.

The algorithm 2 illustrates the general process of the implemented GA. The starting population P of individuals is created in line 5 to 8, wherein the individuals produced by the heuristics H_0 to H_4 for the given problem instance I.

Algorithm 2 General workflow of the GA

Require: $I = (G, K, c)$ as problem instance

$\alpha \leftarrow \alpha_{max}$
$\beta \leftarrow 0$
$largestFitness \leftarrow -\infty$
$fittest \leftarrow ()$
$P \leftarrow \{H_1(I), H_2(I), H_3(I), H_4(I)\}$
while $|P| < p_{max}$ **do**
  $P \leftarrow P \cup H_0(I)$
end while
while $\alpha > 0 \wedge \beta < \beta_{max}$ **do**
  $\alpha \leftarrow \alpha - 1$
  $P' \leftarrow \mathrm{RWS}(P, s_r)$
  $P'' \leftarrow \{\}$
  for all $p_1 \in P'$ **do**
    $p_2 \leftarrow p \in P' \setminus \{p_1\}$ select at random
    $c \leftarrow \mathrm{cross}(p_1, p_2)$ {cross $\in X$ select at random}
    if random $\leq p_m$ **then**
      $c \leftarrow \mathrm{mutate}(c)$ {mutate $\in M$ select at random}
    end if
    $P'' \leftarrow P'' \cup \{c\}$
  end for
  $P \leftarrow \mathrm{RTR}(P, P'')$
  for all $c \in P$ **do**
    if $\mathrm{fitness}(c) > largestFitness$ **then**
      $largestFitness \leftarrow \mathrm{fitness}(c)$
      $fittest \leftarrow c$
      $\beta \leftarrow 0$
    else
      $\beta \leftarrow \beta + 1$
    end if
  end for
end while
return split($fittest$)

Then follows the main loop of GA's with the above-stated stop criteria. In line 11, a sub-population P' is selected by the roulette-wheel selection method and the rate s_r. On line 14 and 15, each element of the set P' is recombined with a randomly selected other element of the set P' by a crossover operator of the set X. Then in line 16 to 18, each recombinant child is mutated with probability p with any mutation operator of the set M. All child individuals are collected in the auxiliary set P''. In line 18, the parent generation P is replaced by the child generation P'' by restricted tournament replacement. Finally, the fittest individual of the current set P with the best-known individual is compared in each iteration. The best-known individual is reset in the event of an improvement. The result of GA's is the set of all solution routes (line 32).

4.7 GA parameters

For the setting of the GA parameters, we compared 36 constellations for different instances of SOLOMON [18]. For a robust GA with good performance following parameter settings were found:

$$\begin{aligned}
\textit{Population size} \quad & p_{max} = 30 \\
\textit{Selection rate} \quad & s_r = 0.4 \\
\textit{Mutation probability} \quad & p_m = 0.3 \\
\textit{Max rounds} \quad & \alpha_{max} = 100000 \\
\textit{Max rounds without improving} \quad & \beta_{max} = 2000 \\
\textit{Crossover operators} \quad & X = \{\mathrm{CX}, \mathrm{OX}, \mathrm{NWOX}\} \\
\textit{Mutation operators} \quad & M = \{\mathrm{TWORS}, \mathrm{CIM}, \mathrm{RSM}\}
\end{aligned}$$

5 Schedules improvement

After the execution of GA's the decision variables x_i^k are determined. Now the actual routes, so the order of the customer for each vehicle are known. The second part of the problem is to determine the actual departure times at each node, so that the total travel time is minimized.

Because this problem in general case is NP-hard (given by the not necessarily convex travel time function $c(t, i, j)$), we have developed a greedy-heuristic H_i for determining the departure times d_i^k. In this heuristic is assumed that the distance between two customers still be has a major impact on the travel time. For two customers is assumed: The further away, the greater is the travel time between these two vertices. The idea for saving time is now to choose the departure times for long distances so that a minimum travel time results. It is considered that minimizing the travel time of great distances at a realistic travel time function $c(t, i, j)$ has the greatest potential.

For the improvement heuristic H_i, we discretize the time units and search in single time steps for good departure times. The algorithm 3 shows the improvement heuristic for a given route r. A route is a single tuple of the result set of the previous performed GA.

Algorithm 3 Improvement heuristic H_i

Require: $r = (c_1, \ldots, c_n)$ {a single route with n customers}
  $Q \leftarrow \{\}$
  $i \leftarrow 0$
  for $j = 1$ **to** n **do**
    $Q \leftarrow Q \cup \{(i, r_j)\}$
    $d_i \leftarrow$ earliestDepartureTime(i)
    $i \leftarrow r_j$
  end for
  while $|Q| > 0$ **do**
    $s_{max} \leftarrow -\infty$
    for all $q \in Q$ **do**
      if distance$(q_1, q_2) > s_{max}$ **then**
        $q' \leftarrow q$
        $s_{max} \leftarrow$ distance(q_1, q_2)
      end if
    end for
    $Q \leftarrow Q \setminus q'$
    $d' \leftarrow d_{q'_1}$
    $t_{min} \leftarrow \infty$
    while $d_{q'_1} \leq$ latestDepartureTime(q'_1) **do**
      $t, i \leftarrow 0$
      for $j = 1$ **to** n **do**
        $t \leftarrow t + c(i, r_j, d_i)$
        $i \leftarrow r_j$
      end for
      if $t < t_{min}$ **then**
        $d' \leftarrow d_{q'_1}$
        $t_{min} \leftarrow t$
      end if
      $d_{q'_1} \leftarrow d_{q'_1} + 1$
    end while
    $d_{q'_1} \leftarrow d'$
    $Q \leftarrow Q \setminus q'$
  end while
  return $(d_0, \ldots, d_{n-1})$

Finally, the complete solution of a problem instance can be constructed with the schedule construction algorithm H_s. For this purpose, all routes of the GA-result will be evaluated by the heuristic H_i. The result is a set of pairs which are

each composed from the route and the corresponding departure times. Algorithm 4 illustrates the construction of the schedule.

Algorithm 4 Schedule construction H_s

Require: $R = (r_1, \ldots, r_n)$ {n solution routes}
  $S \leftarrow \{\}$
  for all $r \in R$ **do**
    $S \leftarrow S \cup \{(r, H_i(r))\}$
  end for
  return S {schedule}

5.1 Evaluation of improvement heuristic

We have analyzed the improvements based on all 56 SOLOMON [18] instances and the speed model *TD1* type *b* [9]. The figure 5 shows the average travel times with and without improvement heuristic for 50 and 100 customers. At 100 customers per instance a time saving of 1.6% was achieved and for 50 customers 2.6% were achieved.

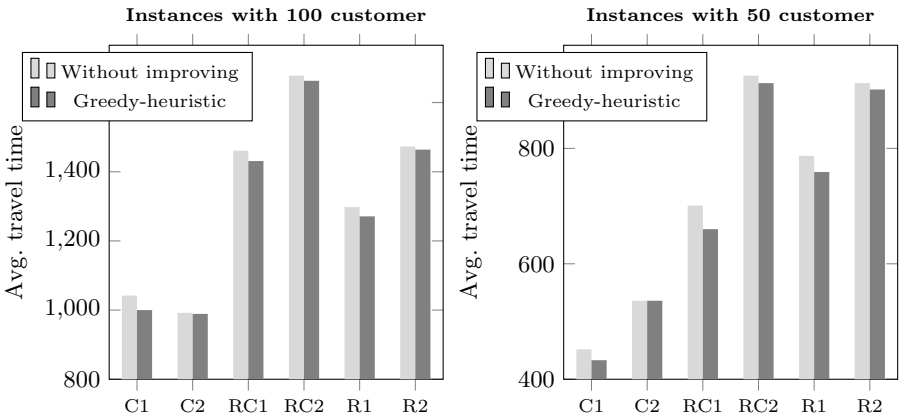

Fig. 5. Impact of the greedy heuristic on the travel times compared to "without improvement"

As expected, the improvement of the departure times compared to routing only a minor influence on the total driving time. Especially for routes with little timing margin improvements are minimal. However, we consider this problem as a post-processing step after execution of the GA. Due to this problem reduction has been able to reduce the complexity of the GA's.

6 Computational results

For the evaluation of the developed solution several benchmarks were performed for all 56 Solomon [18] instances. We compare the presented solution approach (GA) with the Iterative Route Construction and Improvement (IRCI) method developed by Figliozzi [9]. Based on the algorithm of [12] we can create the travel time function $c(t, i, j)$ for each speed model *TD1-TD4* of Figliozzi.

For each problem instance (C1, C2, R1, R2, RC1 and RC2) now the average number of vehicles, distance and travel time for all 13 speed models (constant speed + 12 dynamic models by Figliozzi) was measured. Table 1 shows the results of the GA in comparison to IRCI. The complete results with all speed models are listed in the appendix A.

Table 1. Averaged results for each instance class

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	11.37	2.70	10.00	3.00	11.14	3.36
GA	11.83	2.96	10.00	3.05	11.66	3.12
Average distance						
IRCI	1276	1223	867	671	1421	1424
GA	1280	1074	862	605	1448	1290
Average travel time						
IRCI	959	916	666	511	1067	1066
GA	935	770	631	460	1055	939

The results show that the GA achieved for clustered instances better results with the same number of vehicles. In randomly scattered instances the GA achieved also shorter travel times, but needs a little more vehicles. In the folling the average of all measured values is represented:

	IRCI	**GA**	%
Average number of vehicles	6.89	7.14	+3.63
Average distance	1147	1093	−4.71
Average travel time	864	798	−7.64

On average, 7.63% time savings with 3.63% vehicle additional effort can be achieved compared to IRCI for all Solomon instances.

7 Related work

Our research focuses on the results of Figliozzi work [9]. He defined a TDVRP with a set of speed models for hard and soft time windows. The speed models divide the period into five periods between depot opening and closing time, which are simulated typical scenarios in urban traffic. He developed an iterative

solution method (IRCI) which he evaluated by using of benchmark problems by Solomon [18]. Solomon examined different heuristics for solving a VRP with time windows and constant speeds. Type 1 of the insertion heuritic described by him was used in the current work to create initial individuals of the GA. In this context, we have studied more suitable heuristics for the VRP in the literature. Ioannou et al. [13] developed a greedy look-ahead heuristic for the VRP with time windows, which is based on the heuristic of Atkinson [3] and has a good performance for standard test problems.

For the current work, Ichoua et al. [12] provided a relevant mathematical basis for problem definition of a VRP with time-dependent travel times. They formulated a model for the TDVRP with a FIFO property. The FIFO property guarantees that if a vehicle leaves a vertex i for a vertex j at a given time, any identical vehicle leaving vertex i for vertex j at a later time will arrive later at vertex j. Hashimoto et al. [11] divided the TDVRP in the optimal start time problem and the finding of visiting orders. They have proven that in general case the start time problem is NP-hard.

In the literature there are only a few genetic algorithms for the TDVRP, so our algorithm is only based on researches on GA's for classical VRP's. Alba and Dorronsoro [2] developed a cellular genetic algorithm for the VRP where the route separation is directly encoded in the chromosomes. In contrast, Prins [17] investigated a GA with an indirect routes representation. He developed a procedure, which transforms an ordered sequence of customers in an optimal route splitting. We use this idea in our solution method and implement a splitting procedure for time-dependent routes.

8 Conclusions

In this research, we have developed a method for solving time-dependent VRP's with hard time windows. By a not necessarily convex travel time function, it is possible to model any influencing factors of the travel time between two customers at a given time. Thus, realistic time-dependent processes are simulated. We solve the problem with the unknown departure times regardless of finding the routes, so that the more relevant routing can be solved easily. Compared to [9] shorter travel times can be achieved, but more vehicles are also sometimes required.

We suppose that the developed solution method compared to a strong problem-optimized heuristics is very robust and for any other instances and speed models give good results. This assumption needs to be examined in future work. Furthermore, the influence of other suitable genetic operations on the number of vehicles for a solution must be studied.

References

1. O. Abdoun, J. Abouchabaka, C. Tajani, Analyzing the Performance of Mutation Operators to Solve the Travelling Salesman Problem, in Wotic, 2011

2. E. Alba, B. Dorronsoro, Solving the Vehicle Routing Problem by Using Cellular Genetic Algorithms, Evolutionary Computation in Combinatorial Optimization, Lecture Notes in Computer Science, Volume 3004, 2004, Pages 11-20
3. J. B. Atkinson, A Greedy Look-Ahead Heuristic for Combinatorial Optimization: An Application to Vehicle Scheduling with Time Windows, The Journal of the Operational Research Society, Volume 45, Issue 6, June 1994, Pages 673-684
4. J. E. Baker, Reducing Bias and Inefficiency in the Selection Algorithm, In Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, NJ, USA, 1987, Pages 14-21
5. V. A. Cicirello, Non-wrapping order crossover: an order preserving crossover operator that respects absolute position, Genetic and Evolutionary Computation Conference, 2006, Pages 1125-1132
6. G. Clarke, J. W. Wright, Scheduling of vehicles from a central depot to a number of delivery points, Operations Research, Volume 12, 1964, Pages 568-581
7. T. H. Cormen, C. E. Leiserson, R. Rivset, C. Stein, Introduction to Algorithms, Section 24.2, Single-source shortest paths in directed acyclic graphs, 2001, Pages 592-595
8. L. Davis, Applying Adaptive Algorithms to Epistactic Domains, Proceedings of the Int. Joint Conf. on Artificial Intelligence, IEEE Computer Society Press, Los Angeles, 1985, Pages 162-166
9. M. A. Figliozzi, The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics, Transportation Research Part E: Logistics and Transportation Review, Volume 48, Issue 3, May 2012, Pages 616-636
10. G. R. Harik, Finding multimodal solutions using restricted tournament selection., Proceedings of the Sixth International Conference on Genetic Algorithms, 1995, Pages 24-31
11. H. Hashimoto, T. Ibaraki, S. Imahori, M. Yagiura, The vehicle routing problem with flexible time windows and traveling times, Discrete Applied Mathematics, Volume 154, Issue 16, 1 November 2006, Pages 2271-2290
12. S. Ichoua, M. Gendreau, J-Y Potvin, Vehicle Dispatching With Time-Dependent Travel Times, European Journal of Operational Research, Volume 144, Issue 2, 2003, Pages 379-396
13. G. Ioannou, M. Kritikos, G. Prastacos, A greedy look-ahead heuristic for the vehicle routing problem with time windows, J Oper Res Soc, Volume 52, Issue 5, 2001, Pages 523-537
14. M. Lozano, F. Herrera, C. J. Ramn, Replacement strategies to preserve useful diversity in steady-state genetic algorithms, Information Sciences, Volume 178, Issue 23, 1 December 2008, Pages 4421-4433
15. I. M. Oliver, D. J. Smith, J. R. C. Holland, A study of permutation crossover operators on the traveling salesman problem. In Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application, NJ, USA, 1987, Pages 224-230.
16. P. P. Paul, A. Ramalingam, R. Baskaran, P. Dhavachelvan, K. Vivekanandan, R. Subramanian, A new population seeding technique for permutation-coded Genetic Algorithm: Service transfer approach, Journal of Computational Science, Volume 5, Issue 2, March 2014, Pages 277-297
17. C. Prins, A simple and effective evolutionary algorithm for the vehicle routing problem, Computer and Operations Research, Volume 31, 2004, Pages 1985-2002

18. M. M. Solomon, Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints, Operations Research, Volume 35, Issue 2, Mar.-Apr. 1987, Pages 254-265

A Concrete computational results

A.1 Constant speed

Table 2. Results for classical Solomon instances - constant speed

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
Best ever	11.92	2.73	10.00	3.00	11.50	3.25
IRCI	12.58	3.00	10.00	3.00	12.12	3.38
GA	13.00	3.36	10.00	3.00	12.88	3.75
Average distance						
Best ever	1210	952	828	590	1384	1119
IRCI	1248	1124	841	626	1466	1308
GA	1307	1028	838	596	1503	1226

A.2 Time-dependent speeds

Table 3. Results for travel time distribution 1 - type (a)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	11.67	2.82	10.00	3.00	11.38	3.25
GA	12.17	3.00	10.00	3.00	11.63	3.38
Average distance						
IRCI	1295	1216	859	657	1405	1444
GA	1288	1088	843	605	1436	1255
Average travel time						
IRCI	1080	990	729	563	1164	1177
GA	1047	841	693	515	1158	975

Table 4. Results for travel time distribution 2 - type (a)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	10.75	2.55	10.00	3.00	10.50	2.88
GA	11.00	2.82	10.00	3.00	10.88	3.13
Average distance						
IRCI	1258	1244	864	654	1395	1454
GA	1267	1076	863	615	1446	1274
Average travel time						
IRCI	897	861	644	495	989	993
GA	859	680	630	457	974	817

Table 5. Results for travel time distribution 3 - type (a)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	9.92	2.27	10.00	3.00	10.00	2.75
GA	10.33	2.73	10.00	3.13	10.38	3.00
Average distance						
IRCI	1237	1269	880	697	1362	1434
GA	1271	1069	865	603	1433	1284
Average travel time						
IRCI	793	774	608	485	860	867
GA	745	584	577	405	825	717

Table 6. Results for travel time distribution 1 - type (b)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	12.42	3.00	10.00	3.00	12.13	3.38
GA	13.08	3.27	10.00	3.13	13.25	3.75
Average distance						
IRCI	1289	1212	880	670	1454	1403
GA	1273	1060	865	600	1483	1284
Average travel time						
IRCI	1064	1027	608	545	1180	1200
GA	1022	865	577	492	1160	1081

Table 7. Results for travel time distribution 2 - type (b)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	11.50	2.73	10.00	3.00	11.25	3.25
GA	11.92	3.00	10.00	3.00	11.75	3.38
Average distance						
IRCI	1279	1218	883	667	1429	1433
GA	1289	1064	879	608	1468	1292
Average travel time						
IRCI	905	893	650	467	1010	1053
GA	877	724	539	428	977	919

Table 8. Results for travel time distribution 3 - type (b)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	11.42	2.73	10.00	3.00	11.00	3.00
GA	11.67	2.90	10.00	3.00	11.38	3.38
Average distance						
IRCI	1265	1245	866	714	1442	1483
GA	1303	1157	892	601	1503	1368
Average travel time						
IRCI	808	831	584	446	916	981
GA	780	687	473	386	885	849

Table 9. Results for travel time distribution 1 - type (c)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	11.67	2.73	10.00	3.00	11.50	3.25
GA	12.33	3.00	10.00	3.13	11.88	3.38
Average distance						
IRCI	1302	1245	865	683	1435	1407
GA	1282	1072	869	601	1435	1266
Average travel time						
IRCI	1066	1003	697	573	1186	1147
GA	1030	838	697	502	1164	1014

Table 10. Results for travel time distribution 2 - type (c)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	10.83	2.55	10.00	3.00	10.75	2.75
GA	10.92	2.73	10.00	3.00	11.25	3.25
Average distance						
IRCI	1266	1238	863	658	1413	1472
GA	1266	1118	857	627	1412	1243
Average travel time						
IRCI	881	843	618	483	1012	1027
GA	857	728	609	463	975	839

Table 11. Results for travel time distribution 3 - type (c)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	10.17	2.36	10.00	3.00	10.13	2.75
GA	10.83	2.64	10.00	3.25	10.88	3.00
Average distance						
IRCI	1272	1243	862	665	1409	1438
GA	1255	1078	870	606	1396	1365
Average travel time						
IRCI	801	760	565	451	904	886
GA	738	614	550	401	857	811

Table 12. Results for travel time distribution 1 - type (d)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	12.25	3.00	10.00	3.00	12.00	3.38
GA	12.67	3.09	10.00	3.00	12.50	3.50
Average distance						
IRCI	1311	1218	872	666	1425	1394
GA	1289	1069	844	610	1440	1297
Average travel time						
IRCI	1114	1045	731	552	1192	1192
GA	1099	920	708	505	1205	1102

Table 13. Results for travel time distribution 2 - type (d)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	11.58	2.73	10.00	3.00	11.25	3.38
GA	12.08	3.00	10.00	3.00	11.63	3.25
Average distance						
IRCI	1272	1216	856	679	1404	1412
GA	1267	1048	850	597	1431	1334
Average travel time						
IRCI	943	915	652	494	1035	1053
GA	932	787	645	432	1051	990

Table 14. Results for travel time distribution 3 - type (d)

Method	R1	R2	C1	C2	RC1	RC2
Average number of vehicles						
IRCI	11.08	2.64	10.00	3.00	10.75	3.25
GA	11.83	3.00	10.00	3.00	11.38	3.38
Average distance						
IRCI	1293	1215	867	690	1436	1424
GA	1286	1034	864	599	1440	1285
Average travel time						
IRCI	871	846	612	461	964	975
GA	865	707	608	397	961	861