A Hybrid Genetic Algorithm for Solving the VRP with Pickup and Delivery in Rural Areas

Timo Stadler
OTH Regensburg
timo.stadler@oth-regensburg.de

Jonas Schrader
OTH Regensburg

Jan Dünnweber
OTH Regensburg

Abstract

In this paper, we present a new Hybrid Genetic Search (HGS) algorithm for solving the Capacitated Vehicle Routing Problem for Pickup and Delivery (CVRPPD) as it is required for public transport in rural areas. One of the biggest peculiarities here is that a large area has to be covered with as few vehicles as possible. The basic idea of this algorithm is based on a more general version of HGS, which we adopted to solve the CVRPPD in rural areas. It also implements improvements that lead to the acceleration of the algorithm and, thereby, to a faster generation of a fastest route. For example, we use a modified form of the 2Opt method. We tested the algorithm on real road data from Roding, a rural district in Bavaria, Germany. Moreover, we designed an API for converting data from the Openrouteservice, so that our algorithm can be applied on real world examples as well.

Keywords: VRP, Mobility-on-Demand, Pickup and Delivery, Genetic Algorithm, 2Opt

1. Introduction

Due to the changes in our society, there are increasingly complex route planning problems for applications such as home-delivery or mobility-on-demand applications. Especially in areas with insufficient public transport connections, such applications are urgently needed, because buses only run rarely. The presented research is a step towards improving mobility in rural areas and are developing a mobility-on-demand solution for rural areas. In our specific case, minibuses, called feeders, are supposed to pick up people close to their home and bring them to their desired destination in a maximum prescribed time.

Thus, we are facing a specific variant of the Vehicle Routing Problem (VRP) wherein we need to take pickup and delivery into account. A guest who is picked up by a vehicle must also be dropped off by the same vehicle. A precise mathematical definition is given in chapter 3.2. The VRP can be broken down into many more specific sub-problems. In our case, we consider the Capacitated VRP for Pickup and Delivery.

This particular version of the problem creates some constraints we had to cope with. In our case, these are the constraints that vehicle capacity and pickup/dropoff order should be respected. The key constraint in our problem is the correct order of pickup and dropoff nodes. If this is not observed, a route can be expected that is up to 200% as long, because a passenger has not yet to be picked up and therefore all further points of the route have to be repeated.

We are specifically looking at mobility-on-demand applications and ride-sharing options for rural areas. Specifically, all the test routes we selected for the evaluation of the algorithm are located in the area of the small town of Roding. The city of Roding has about 11500 inhabitants (as of 2008) spread over an area of $113km^2$. We additionally take into account the significantly less populated but larger neighbouring regions with a size of $674km^2$. For this purpose, 500 predetermined test points were randomly selected on the map and the best route was calculated by various algorithms. With this number, the entire district should be covered and there should be enough different points that can be approached. Different algorithms were used to calculate the best route. Our algorithm returned a solution closer to the optimum when tested against these other algorithms.

We solved this problem using hybrid genetic search (HGS). The HGS allows us to combine the advantages of a genetic algorithm with a local search (LS) and thus arrive at an optimised route more quickly. Here, an initial solution is assumed and its neighbouring solutions are examined, thus improving the solution iteratively (Groër et al., 2010).

In this paper, we describe the adjustments that need to be made to a HGS in order to make the CVRPPD solution possible. This is the basic requirement to be able to implement a mobility-on-demand solution in rural areas. Several vehicles with different capacities are taken into account and these solve the problem of pickup and delivery belonging together. This system is the best solution to provide the entire area with accessible public transport, especially because of the widely spaced stops.

In the next chapters, we will first look at related work and how the CVRPPD is solved here. This is followed by a description of the problem and the general functioning of the HGS. After that an explanation of the implementation of the algorithm, especially the extra steps that are necessary for solving the CVRPPD is done. Therafter we give a description of the use of the algorithm on real road data, and the use of the API to query routes for multiple vehicles. This is concluded in Section 6 by an outlook on further improvements, we plan to make to our algorithm.

2. Related Work

For our mobility-on-demand approach, we need buses with different capacities that can start from different depots. In Roding, as elsewhere in Germany however, there is a restriction that passengers can only be picked up at places that are marked as bus stops. Stadler et al., 2022 developed an algorithm in which new bus stops are determined by weighting a Voronoi diagram. The weighting is based on the size of the population and the number of points of interest within a polygon. These new points are combined with the existing stops as so-called virtual stops and can be used as depot or pickup and dropoff points. The number of points we have chosen is sufficient to cover the entire public transport space.

McKenna et al., 2019 defined a method that shows a dynamic route planning approach, based on real-time demand, is possible. For this, they completely deviate from the existing bus system with rigid routes and departure times and rely on a dynamic system. This system allows a user to call the bus via an app to a stop near him. As a result, the authors found that using dynamic buses emits 38.43% less Co_2 emissions while reducing travel time by 34.76%. This is mainly achieved by allowing the bus to travel shorter distances than on a static route and by not calling at stops without people. We also test our algorithm with real data and allow the user to address the service via an app.

There are many different methods for solving the VRP. Generally, a distinction is made between exact solution methods and the use of heuristics. For an exact solution, however, the path lengths of all possible paths along a graph would have to be calculated, which is why there are $(n-1)!/2$ different paths between a number of nodes (Laporte and Nobert, 1987). They are only used to calculate the optimum in order to obtain a benchmark for other algorithms later on, or for very small problems.

Over time, heuristics were found that led to a better solution more quickly. The best-known method is the LS method. For the sake of optimization, various improvement operators (so called *moves*) are used, which perform different tasks such as swapping two edges or removing an edge from a tour. An extension of the LS is the so-called tabu search. This adapts the LS in such a way that a list is created in order to avoid cycles when traversing the solution space (Fred W. Glover, 1997). By these conditions it represents an improvement of LS.

Another approach to solve the VRP is through an evolutionary approach; the genetic algorithm. Here, the evolution of a species is mimicked according to Darwin's "Survival of the Fittest" theory in order to achieve an optimal result. Baker and Ayechew, 2003 presented a genetic algorithm, which, however, performed worse than the tabu search. By incorporating a simple neighbourhood search, however, a significant improvement could be achieved. Thus, a hybrid algorithm was used. Unlike the Tabu search, for example, this Genetic Algorithm (GA) assumes more than one solution in each iteration step. Thus, an optimisation is searched for in several solutions in parallel, which expands the search space more quickly.

Another one of these hybrid approaches was presented by Vidal et al., 2012 with their hybrid genetic search for solving the capacitated VRP. Here, they combine a genetic algorithm with LS to form a new algorithm. In addition, they extend it with a population management that contributes to a higher performance level (Vidal et al., 2012). In chapter 3, the general functioning of the HGS is explained and how it can be used to solve the VRP.

Chapter 3 describes the general function of the HGS and explains how it can be used to solve the VRP.

3. Problem Statement

The problem to be solved is the capacitated vehicle routing problem for pickup and delivery (CVRPPD). In addition, other constraints, such as a maximum travel time will be considered.

3.1. Functionality of the HGS

The basic concept of a GA is to evolve a population of individuals that manage either a feasible or an infeasible solution (Holland, 1984). A population is a set of different solutions for the CVRPPD, which are gradually improved or replaced by new solutions with the help of evolution. Depending on which type of solution (feasible or infeasible) an individual manages, it is classified into the respective subpopulation.

The algorithm then applies a number of operators to select two parents among the individuals and combine them into a new individual, the offspring. This is first improved by applying LS methods (education and repair) and then divided into the correct subpopulation. This is referred to as a hybrid algorithm, since the algorithm is initially constructed according to the principles of GA, but is also optimised by LS.

A cost parameter is defined for each solution. This usually includes the sum of travel times for all routes and, for infeasible solutions, an additional penalty parameter. In addition, our approach to optimising Vidal et al., 2012 does not only consider the costs of a solution, but also the contribution a solution makes to the diversification of the gene pool. This provides a higher degree of diversity, allowing different solutions to be explored and increasing the performance of the algorithm.

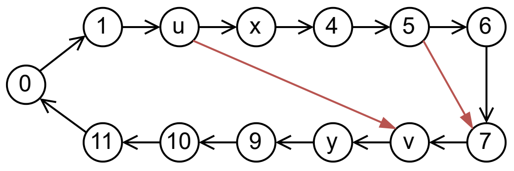

Figure 1. Structure of a single solution

A single solution of the population describes the route a bus must take to pick up all passengers. So in each solution, the depot is assigned to a customer as a departure or end point, each customer is recorded with his boarding point and the total tour that a bus has to take is defined. The tour is saved for each period of evolution. In addition, a cost parameter is set for each solution in the amount of the trip duration. In the case of infeasible solutions, this cost parameter is additionally assigned a penalty in order to mark them as infeasible and thus weaker in the evolution. Figure 1 shows the structure of a single solution of the HGS. The red arrows mark the pickup and dropoff pairs. For example, node u represents a pickup location and node v represents the corresponding dropoff point. For readability reasons, only 2 node pairs are used here, whereas all other nodes also represent either pickup or dropoff pairs.

A solution is called infeasible if it does not comply with the conditions of the algorithm. In the case of the CVRP, the maximum journey time is too long or the maximum capacity of a vehicle is exceeded. However, it turned out that it makes sense to penalise infeasible solutions for their costs and to initiate a controlled exploration of the infeasible subpopulation (Glover and Hao, 2010).

During each iteration, evolution occurs through the selection of parent solutions. If no improvements are made to the best solution for a fixed number of iterations, the entire population is diversified. This prevents the too early convergence of a solution, which occurs when only good solutions are kept in each evolution. Survivor selection tries to keep the best solutions of a subpopulation and at the same time keep solutions that contribute to the diversification of the gene pool, no matter how high their costs are. All other solutions are discarded because they are either clones of solutions that have been retained or do not contribute sufficiently to the diversification of the gene pool. The algorithm is terminated either after a certain number of iterations or when a fixed time has elapsed.

During each iteration, a mutation of the solution can occur with a certain fixed probability. This is not randomised in HGS, but is based on LS methods. In this phase of route improvement (RI), the edges of the graph are iterated randomly and each of the optimisation moves is evaluated. These are methods that correspond to inserting a node into a route, swapping the positions of two nodes. A final move that can be executed is the 2-opt-move. Here, edges between nodes are swapped in such a way that a crossover-free graph can be created (Croes, 1958). From these moves, a random one is chosen until one leads to an improvement of the route. The RI phase is terminated when all moves have been tested one after the other and none of them leads to an improvement of the route.

In addition, a so-called repair phase takes place for 50% of the infeasible solutions. Here, an attempt is made to generate a feasible solution from an infeasible solution. Depending on the result of the repair, the solution is divided into the corresponding subpopulation.

In the following, the application of the HGS to the CVRPPD is described and it is shown which further constraints apply to this problem in the context of rural transportation. Moreover which moves of the RI cannot be implemented or which new moves can be applied instead.

3.2. Necessary adjustments for Pickup and Delivery

Let $G = (V, E)$ be a directed graph where $V = v_0, \ldots, v_n$ is a set of nodes representing locations and $E = (v_i, v_j)|v_i, v_j \in V, i \neq j$ is a set of edges representing the connections between the locations. In addition to the edges there is a distance matrix (d_{ij}). The vertex v_0 represents a depot from which a fleet of m vehicles start their tours. The remaining vertices correspond to n geographically distributed locations, which are either pickup or dropoff points of a customer. Each vehicle has a capacity Q and a maximum driving distance D. The goal of the CVRPPD is designing a set of at most m routes such that

1. each route starts and ends at the depot
2. each location is visited exactly once by exactly one vehicle
3. the vehicle load at any point of a route does not exceed the capacity of the vehicle assigned to it
4. the total distance of each route does not exceed the preset limit D
5. the total routing cost is minimized

The CVRPPD can be defined mathematically as follows and is an extension of Christofides' VRP formulation. Thereby following decision variables exist:

- x_{ijk} is a binary variable indicating whether vehicle k traversed the edge i, j ($x_{ijk} = 1$ means that the edge has been traversed)
- y_{ijk} is the load of the vehicle k while traversing the edge i, j
- b_{ij} is a decision variable indicating whether vertex i is visited before vertex j ($b_{ij} = 1$ means that vertex i is visited before vertex j
- s_{ijk} is a decision variable indicating whether the vertices i and j are visited by the vehicle k ($s_{ijk} =$ 1 means that the vertices are on the same route)

And following objective function Z has to be minimized:

$$f \sum_{k=1}^{m} \sum_{j=1}^{n} x_{0jk} + g \sum_{i=0}^{n} \sum_{j=1}^{n+1} \sum_{k=1}^{m} d_{ij} x_{ijk} \tag{1}$$

Subject to

$$\sum_{i=0}^{n} \sum_{k=1}^{m} x_{ijk} = 1 \text{ for } 1 \leq j \leq n \tag{2}$$

$$\sum_{j=0}^{n} x_{ijk} = \sum_{j=1}^{m} x_{ijk} \quad \text{for } 1 \leq i \leq n, 1 \leq k \leq m \tag{3}$$

$$\sum_{j=1}^{n} x_{0jk} \leq 1 \text{ for } 1 \leq k \leq m \tag{4}$$

$$y_{ijk} \leq x_{ijk} Q \quad \text{for } 0 \leq i \leq n, 1 \leq j \leq n+1, 1 \leq k \leq m \tag{5}$$

$$\sum_{j=1}^{n} y_{0jk} = \sum_{j=1}^{n} q_j \sum_{i=0}^{n} x_{ijk} \quad \text{for } 1 \leq k \leq m \tag{6}$$

$$x_{ijk} \in 0, 1 \quad \text{for } 0 \leq i \leq n, 1 \leq j \leq n+1, 1 \leq k \leq m \tag{7}$$

$$y_{ijk} \geq 0 \quad \text{for } 0 \leq i \leq n, 1 \leq j \leq n+1, 1 \leq k \leq m \tag{8}$$

W.l.o.g let v_i be the pickup location to the dropoff location v_j:

$$b_{ij} = 1 \text{ for } 1 \leq i < j \leq n \tag{9}$$

$$\exists! k \in K : s_{ijk} = 1 \text{ for } 1 \leq i < j \leq n \tag{10}$$

Each vehicle must include both the pickup and the dropoff for each passenger in its route. Thus, passengers cannot split their pickup and dropoff among the routes of two different vehicles, i.e. one always gets off the bus they got on before. In addition, the pickup must always precede its associated dropoff in the route for the route to be feasible. This condition is not given for VRP instances where goods instead of passengers are transported, since they are identical and only the capacity of the vehicles plays a role in the delivery. In our case, however, we are considering the transport of different individuals.

A solution is considered infeasible even if this condition is not met. Furthermore, some of the moves listed in 3.1 automatically lead to an infeasible solution. Especially swap moves that exchange a single point between the routes of two vehicles automatically create an infeasible solution.

In order to solve the problem, a matrix is first needed that determines which pickup point belongs to which dropoff point at any given time. The moves for an insertion and swap must be adapted so that they also function meaningfully for the CVRPPD. We developed a new *move* that makes it possible to relocate a pair of pickup and delivery points within a route. Furthermore, we found a method to ensure that the order of pickup and delivery pairs is maintained. In addition, we introduce the *2kOpt* method, to speed up the process of finding a solution for the CVRPPD.

4. Implementation of the Algorithm for solving the CVRPPD

In order for the HGS to be applied to the CVRPPD in conjunction with the LS, both compoents must be adjusted to take into account the pickup and delivery problem. Contrary to other VRP instances, our problem does not allow for delivery nodes placed in front of a pickup node or split pickup and delivery into two different routes.

4.1. Assignment of a pickup and dropoff to each user

The first point to implement is the algorithm's handling of input and output. In the generic VRP, each node can be handled on its own and has no relationship to other nodes. Thus, besides the change in capacity at that point, only the coordinate points of the node need to be recorded. In CVRPPD, two points always have a relationship that a node always needs a reference to the associated pickup or delivery node. In our format, this is made possible by additionally passing the node its own id, the id of its corresponding counterpart, as well as a label indicating whether it is a pickup or dropoff node. The only exception for nodes that do not have an associated point is the position of the depot at the start and end of a route. In addition, when the input is passed, it is already checked whether a delivery point is also assigned to each pickup. If this is not the case, the algorithm terminates. With this data, the algorithm can determine whether the additional conditions of the CVRPPD are fulfilled.

4.2. Consideration of distance matrices

To enable the algorithm to work on real data, the new parameter *real* was introduced by us for the type of edge weight. This states that all points that are passed to the algorithm are real coordinates. This makes a difference for the calculation, as very precise points have to be calculated in some cases and no improvements are detected in part due to rounding errors. If this parameter is passed, a distance or cost matrix is created in which the travel time/distance from each node to all other nodes of the graph is recorded in an $N : N$ representation. The route-data is generated from a map using the *Openrouteservice* (Neis and Zipf, 2008). In addition, the polyline of the route is determined for each route. This polyline represents the exact course of a route and can be used for later visualisation.

4.3. Adaptation of standard moves for RI

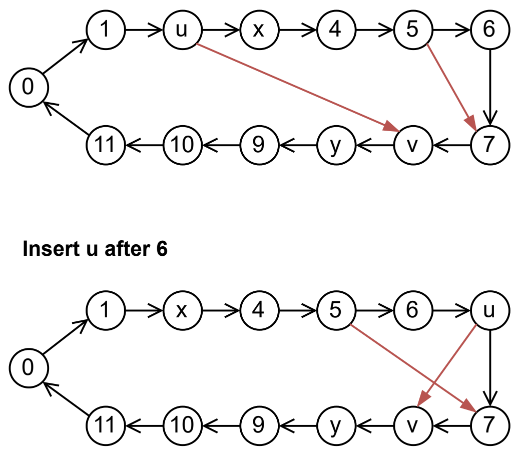

Figure 2. Individual after insert move

With the RI of the generic VRP, infeasible solutions would occur far too often, so that after a short time almost all generated solutions would be infeasible and the algorithm would not be able to find a solution. To prevent this, the *insert* and *swap* moves of the RI were adapted accordingly. In general, it should be avoided that a node would be relocated from one route to another or that the resulting route violates the precedence of a pickup and dropoff node pair. Therefore, the corresponding pickup/dropoff node should always be handled additionally. Before each *insert* move, it is checked whether the node is relocated to another route or within the same route. If it is transferred to the route of another vehicle, this move is not carried out. In the figure 2 an example insert move is shown. At the top of the image, the single solution already presented in figure 1 can be seen again. In the illustrated move, node u has been inserted after node 6. This is possible because node u remains in the same route and is inserted before its dedicated dropoff node. In doing so, all original dependencies in the tour must be reset and a new fitness value of the individual is created. As dependencies counts for example the deletion of the connection $u \rightarrow x$ and the addition of the connection $6 \rightarrow u$.

A similar procedure is used for the *swap* moves and can be seen in figure 3. Here the tour created by the previous insert move was taken and node u was swapped with node 5. This move is also possible because it is again completed inside the same tour and no pickup node is placed after its corresponding dropoff node, as

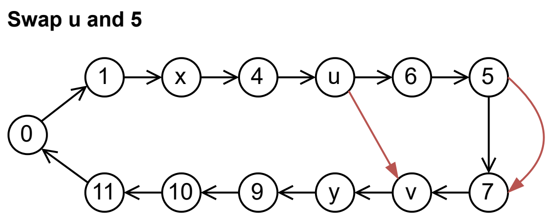

Figure 3. Individual after swap move

can be seen from the red arrows.

Since the previously known moves often fail due to the additional conditions, we developed two new moves that can be used in any case. These are the moves *relocatePair* and *swapPair*.

The swapPair move is similar to the standard swap move, but here not only one node is swapped with another, but the respective pickup and dropoff pairs are swapped with each other in any route. The pickup node is swapped with the other pickup node and the corresponding dropoff nodes are swapped analogously. Since this move does not change the order of pickup and dropoff nodes to each other, it cannot result in infeasible routes. Thus these checks can be ommited in contrast to the standard swap move.

relocatePair instead moves a pair of nodes so that in the new route both nodes are still in the correct order and the dropoff point is at most 3 nodes after the pickup point in the route. The calculation of the costs for this move is done according to the methods defined in Pacheco et al., 2021.

To minimize the computation time in each iteration, this move can only be applied on 3 separate nodes. No significantly better result would be obtained if this were calculated for all nodes. Like the *swapPair* move, this move also doesn't violate the order of pickup and dropoff pairs.

In addition to these modified versions of the standard moves for LS, moves have also been developed that can be applied specifically to the CVRPPD. These moves ensure that the individual routes are shortened by removing detours from the route and avoiding having to travel the same route several times.

4.4. One Point Crossover and 2kOpt-Move Implementation

Two other adaptations that we would like to discuss in more detail here are the One Point Crossover and the 2kOpt move.

The one point crossover is responsible for creating a new individual c from two existing ones, the parents p_1, p_2. The parents are selected for this in a binary tournament selection based on their fitness value. Then, with uniform probability, a cutting point $s \in \{1, \dots n\}$ is randomly chosen in the giant tour of p_1. The nodes $(\pi_1^{p_1}, \dots, \pi_s^{p_1})$ of the first parent are inserted in the new giant tour of the offspring. After that, the second parent is traversed twice from the beginning to the end for adding the missing nodes to the new individual. During the first traversal, all missing dropoff nodes are added to the already existing pickup nodes of c in the same order as in p_2. In the final traversal, all remaining nodes are added to the offspring. All in all, a new individual has been created where the order of pickup and dropoff nodes has not been compromised (Homsi et al., 2020).

The 2Opt move belongs to one of the often used LS strategies. Here 2 edges are exchanged by 2 new ones and the order of the nodes between them is reversed. If one would transfer this methodology 1 to 1 to the CVRPPD, the probability would be high to produce an infeasible route, since thereby pickup and dropoff nodes could be traversed in the wrong order. To avoid this problem, the so-called 2kOpt is used here, which is created by applying nested 2Opt moves.

In the 2kOpt move first 2 nodes u and y are selected between which the order of the nodes is to be reversed. W.l.o.g. let u be before y. Then a pointer is set to $u \rightarrow next$, called the start pointer, and to $y \rightarrow prev$, called the end pointer, which traverse the intermediate sequence of nodes in the opposite direction until they arrive at the same node. For each pair of nodes to which the pointers point, a check is made to see if they can be swapped. This check depends on the type of node and is performed first at the start pointer node and then at the end pointer node. Here it is distinguished as follows whether a node is *OK* or *NOK* for the exchange with the current counterpart node.

A node is OK exactly when:

- node is pickup node and the corresponding dropoff node is behind the current end pointer
- node is dropoff node and the corresponding pickup node is before the current start pointer

A node is NOK exactly when:

- node is pickup node and the corresponding dropoff node is before or equal to the current end pointer
- node is dropoff node and the corresponding pickup node is behind or equal the current start pointer

Thus the following cases of node states can occur:

- start node is NOK and end node $\in \{OK, NOK\}$

- end node is NOK and start node $\in \{OK, NOK\}$
- start and end node are OK

If the start node is NOK, the start pointer is shifted by one to the next node and the check is repeated until a node exists that is OK or start and end pointers point to the same node. Only when a start node with the status OK has been found, the process continues at the end pointer. For the end node, the procedure is analogous, except that the pointer is shifted in the opposite direction. If both nodes are OK, the two pointers are moved forward by one or backward by one and the two previously checked nodes are swapped.

The figure 4 shows the individual steps for performing a simple 2kOpt move. Here the red arrows visualize the pickup and dropoff node pairs, where the initiating node represents the pickup node and the end node is the corresponding dropoff node. In step 1 a swap of the nodes x with v takes place because both have the status OK. In the second step no swap can take place because the check for the new start node failed and the node is marked as NOK. Now node 5 is checked and gets the status OK. After that Node 7 is also checked positively and the two nodes can be swapped. Now start and end pointer are both on node 6 and so the 2kOpt move terminates. The first END diagram shows the new order of the nodes with additional arrows for the pickup and dropoff node pairs. In the second END diagram the placement of the nodes corresponds to the old order and the green arrows show the new order caused by the move.

By using these two algorithms, we can more quickly achieve the best possible solution to the CVRPPD through new recombinations.

Figure 5 shows a route calculated by our algorithm. The colour-coded lines represent the routes taken by different vehicles. The red dots are stops where people can be picked up or dropped off.

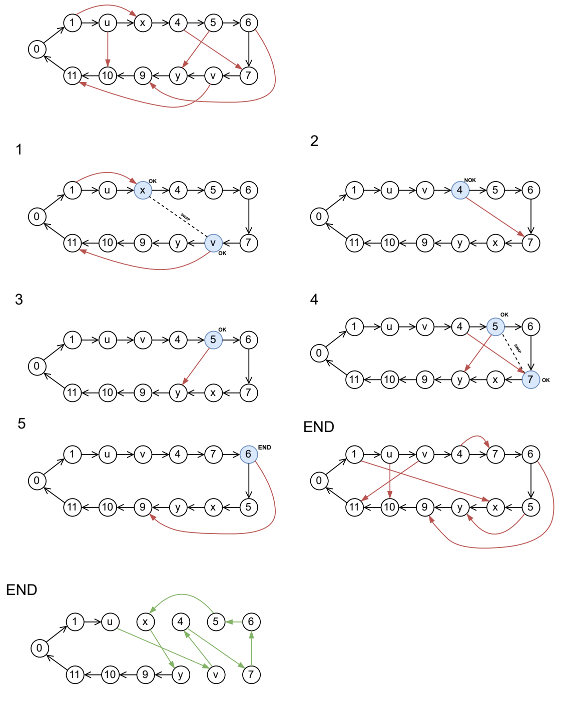

Figure 4. Implementation of the 2kOpt Move

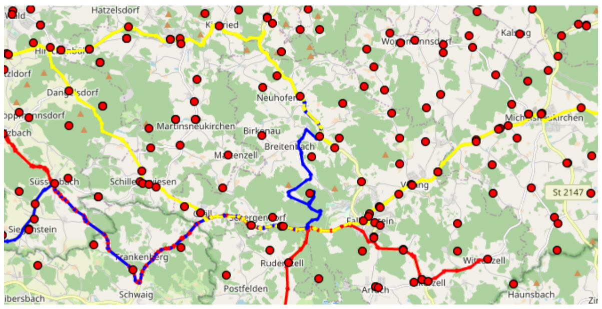

Figure 5. Visualization of a Solution calculated by our algorithm

5. Evaluation

In order to be able to integrate our algorithms into our application, we also implemented a REST API that makes it easier to address the algorithm. In total, there are three different functions that can be invoked via different addresses. These are: Solving the CVRPPD with artificial data, solving the CVRPPD for real traffic data and the generation of a data set of test routes, as well as the calculation of all test routes. These test routes can later be used to generate a large number of cases that are to represent specific test cases of the dynamic mobility solution. The API was developed using the Node.js runtime environment and thus allows us to execute our code platform-independently.

If artificial routes are to be calculated by the algorithm, the edge weight type *2D-eucl* must be passed to the algorithm as an option within a JSON file. This specifies that all coordinates are defined as points on a 2D surface and have a distance to the other points that is defined as the Euclidean distance. The definition of Lin and Chou, 2012 is used as the basis for the definition of the Euclidean distance.

We worked with integers when testing the algorithm in order to be able to understand and visualise the result of the algorithm. Figure 6 shows an example of the generated route from our algorithm for simulated data. In addition, each transport request that is to be processed must be passed within the JSON file. Here, in addition to the coordinates in longitude and latitude, it must also be specified whether the point is a new pickup, or to which pickup point the dropoff belongs.

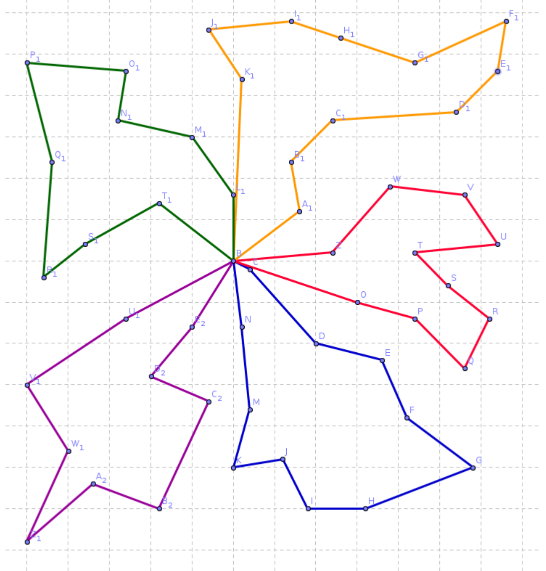

Figure 6. Visualisation of the algorithm for artificial selected points with Euclidean distance

In contrast to the artificial routes, real coordinates in longitude and latitude format must be passed here. With the help of these coordinates, an instance of the *Openrouteservice* is called (Neis and Zipf, 2008). It calculates the distance matrix with the real distances between all points and the actual travel time between the points. The output is also a JSON file that contains all vehicles and the stops there are to approach. In addition, the route to be travelled by each vehicle is transferred as a polyline, which contains the exact route for each vehicle and can be visualised in a map display. Figure 5 shows the route calculated by our algorithm on real data. The individual points to be approached are marked with Arabic letters. The subscripts have been used for numbering purposes only and do not indicate any relationship between two points. The coloured routes here represent the polyline.

The last option of addressing the API is to generate a test data set of pickup and delivery problems and get the solution of each problem after calculation. This feature was very useful as it allowed us to quickly check the functioning of the algorithm.

As already mentioned, the data of a real route can also be visualised. For this purpose, a frontend developed by us can make the requests to the HGS-CVRPPD and receives back all stops to be served as well as the polyline that is to be travelled. In real use, this application is an app via which users can call a dynamic call bus to take them to their destination. This app is a prototype that is not yet publicly available. The user sees the route to be taken and the total journey time until they reach their desired destination.

We adapted the VROOM algorithm by Coupey et al., 2021 to generate the required test data for our special version of the CVRPPD. There is no standard test data set or format to be used for this problem so far. So, using our REST interface, we created our own dataset of 100 routes in the area of the small town of Roding. This was generated from the stops provided to us by the local transport companies. The positions of these stops are now available on Google Maps. In order to determine the longitude and latitude of the stops the map material of *Geofabrik* from 01-02-2022 was used and the partial routes were calculated with the *Openrouteservice* version 6.7.0.

The only other open source application we found that implements our problem is VROOM (Coupey, 2021). VROOM uses a tabu search to plan the routes. We used a parser to convert our data set into a format that VROOM could understand. As parameters for the maximum duration, we set a time of $100ms$, $200ms$ and $300ms$ for the HGS. VROOM had an average runtime of $263ms$ on our test system.

As a result, we found that our algorithm had a deviation of 3.7% from the optimal result at a duration of 100ms. At $200ms$ the deviation was 1.9% and at $300ms$ a deviation of 1.4%. VROOM achieved an average deviation of 1.9%. Thus, it can be assumed that our algorithm produces usable results even at lower runtimes.

6. Conclusion

In this paper we presented our extended HGS for solving the CVRPPD. This is the first approach to solving this particular sub-problem of the VRP based on a HGS. It allows us to test many different solutions by using the genetic algorithm and thus not to end up in a local optimum in the route finding. By using the LS, we were able to achieve optimal results in a short period of time. The quality of the algorithm is determined by the calculated distance covered by all vehicles. In our evaluation, we found that our algorithm achieves a result that deviates only 1.8% from the optimum with a runtime of $200ms$. Our algorithm thus outperforms previously used methods such as the tabu search in solving the CVRPPD by 0.1% in the quality of the solution and 24% in the required calculation time.

In particular, our newly developed methods for adapting the LS to solve the CVRPPD, as well as our new methods to solve our problem by a genetic algorithm (One Point Crossover, 2kOpt), represent the most important contribution of this paper.

Our algorithm was developed based on an HGS for solving the VRP. Although the basic idea for solving the VRP is still the same, this algorithm can only be used effectively for solving the CVRPPD. For a different or less specific problem, the algorithm performs significantly worse than comparable methods due to the many constraints set.

In order to integrate our on-demand system with dynamic buses into the public transport system, all static routes must be known to the system. In the next step, we do not have to consider a simple transport from A to B, but rather an optimal integration of our on-demand system into the static routes as so-called feeder buses. This requires further adjustments to the algorithm as well as the integration of the timetables of static bus routes to enable fast transfers.

We plan to make our HGS available as open-source software so that it can be used as a benchmark algorithm for the CVRPPD in the future. We also want to make the benchmark dataset we use available so that it can be easily compared with our algorithm.

We want to develop the algorithm further by adjusting the parameters of the genetic algorithm accordingly and developing further moves for LS, to get an even better performance.

References

Baker, B. M., & Ayechew, M. (2003). A genetic algorithm for the vehicle routing problem. *Computers &amp Operations Research*, *30*(5), 787–800.

Coupey, J. (2021). New features for our route optimization api. https://blog.verso-optim.com/2022/05/31/solving-problems-better-and-faster/

Croes, G. A. (1958). A method for solving traveling-salesman problems. *Oper. Res.*, *6*(6), 791–812. https://doi.org/10.1287/opre.6.6.791

Fred W. Glover, M. L. (1997). *Tabu search*. Springer US.

Glover, F., & Hao, J. K. (2010). Efficient evaluations for solving large 0-1 unconstrained quadratic optimisation problems. *International Journal of Metaheuristics*, *1*(1), 3. https://doi.org/10.1504/ijmheur.2010.033120

Groër, C., Golden, B., & Wasil, E. (2010). A library of local search heuristics for the vehicle routing problem. *Mathematical Programming Computation*, *2*(2), 79–101. https://doi.org/10.1007/s12532-010-0013-5

Holland, J. H. (1984). Genetic algorithms and adaptation. *Adaptive control of ill-defined systems* (pp. 317–333). Springer US. https://doi.org/10.1007/978-1-4684-8941-5_21

Homsi, G., Martinelli, R., Vidal, T., & Fagerholt, K. (2020). Industrial and tramp ship routing problems: Closing the gap for real-scale instances. *European Journal of Operational Research*, *283*(3), 972–990. https://doi.org/https://doi.org/10.1016/j.ejor.2019.11.068

Laporte, G., & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. *Surveys in combinatorial optimization* (pp. 147–184). Elsevier. https://doi.org/10.1016/s0304-0208(08)73235-3

Lin, J.-H., & Chou, T.-C. (2012). A geo-aware and vrp-based public bicycle redistribution system. *International Journal of Vehicular Technology*, *2012*.

McKenna, C., Clarke, S., & Golpayegani, F. (2019). Floating buses: Dynamic route planning and passenger allocation based on real-time demand. *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*, 2203–2207. https://doi.org/10.1109/ICCC47050.2019.9064471

Neis, P., & Zipf, A. (2008). Openrouteservice. org is three times "open": Combining opensource, openls and openstreetmaps. *GIS Research UK (GISRUK 08). Manchester*.

Pacheco, T., Martinelli, R., Subramanian, A., Toffolo, T. A. M., & Vidal, T. (2021). Exponential-size neighborhoods for the pickup-and-delivery traveling salesman problem.

Stadler, T., Hofmeister, S., & Dünnweber, J. (2022). A method for the optimized placement of bus stops based on voronoi diagrams. *Proceedings of the Annual Hawaii International Conference on System Sciences*. https://doi.org/10.24251/hicss.2022.694

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., & Rei, W. (2012). A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, *60*(3), 611–624. https://doi.org/10.1287/opre.1120.1048