Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics and Operations Research

Optimizing the exchange locations in the driver and vehicle routing problem.

Beilly Zhu (611700)

Supervisor:	Dr. Twan Dollevoet
Second assessor:	Danny Jia Hui Zhu
Date final version:	1st July 2024

Abstract

This paper addresses the Driver and Vehicle Routing Problem (DVRP), a routing problem with two depots. Each driver must start and end at the same depot to return home, while each vehicle must begin and end at different depots for maintenance checks. Therefore, the drivers must switch vehicles at designated exchange locations. Previously, an efficient heuristic for the DVRP was developed in the literature. However, this heuristic only assumed one arbitrary exchange location. We present an improved version of the heuristic that allows multiple exchange locations and incorporates the locations as a decision. We use a multi-armed bandit approach to find the best locations. This extension consistently yields better results than when one arbitrary exchange location is considered. Additionally, our implementation does not compromise on computation time. This shows that, in practical applications, installing more exchange locations strategically is beneficial.

1 Introduction

Vehicle routing problems (VRP) are studied extensively because of their relevance in society. Transportation plays a big part in our daily lives, e.g., public transport and postal services. Therefore, it is in the best interest of a transportation company to effectively and efficiently identify routes for its vehicles such that its customers are served and the costs are minimized. VRP problems commonly assume that a driver stays with one vehicle throughout their entire route.

The Driver and Vehicle Routing Problem (DVRP), proposed in Domínguez-Martín et al. (2018a), takes a different view in which the drivers and the vehicles have different routes. Realistically, the drivers wish to return home after a long time on the road. Vehicles do not have this requirement. This variant of the VRP includes two depots, each housing a homogeneous fleet of vehicles and a homogeneous crew of drivers. Each driver leaving from a depot must return to the same depot at the end of their route. On the other hand, each vehicle departing from a depot must end at the other depot. This is possible by having drivers exchange vehicles while on duty. This swap can only take place in specified exchange locations. A vehicle can only move when accompanied by a driver. Drivers can only move in a vehicle. All driver-vehicle pairs must collectively visit all customers. The duration of a route is defined as the time between the departure from a depot and the arrival to a depot. Under protective labor laws, the duration of a driver's route must not exceed a given time limit. Vehicles, in contrast, can be on the road indefinitely.

This problem was originally inspired by a case in air transportation in the Canary Islands (Salazar-González, 2014). In this problem, a set of scheduled flights between airports must performed daily by aircraft and flight crew. Two major airports played the role of the depots, Tenerife North and Las Palmas. The crew leaving from each base must return by the end of the day to avoid overnight hotel costs. Each aircraft must undergo maintenance every other day, and maintenance is only possible in Las Palmas. As this airport has limited capacity, it is required that each aircraft departing from Las Palmas ends in Tenerife North, while aircraft departing from Tenerife North must fly to Las Palmas for maintenance.

For the DVRP, an effective heuristic is constructed in Domínguez-Martín et al. (2023). The heuristic has two phases. The first phase generates the routes of drivers while ensuring that the duration does not exceed the allowed limit and every customer is visited. The second phase constructs the vehicle routes accordingly. However, that heuristic assumes one exchange location which is selected arbitrarily. This setting is restrictive and costly. Therefore, the main contribution of our paper is to extend the heuristic to incorporate the exchange location as a decision and to allow multiple exchange locations. To find a balance between exploring new exchange locations and exploiting familiar exchange locations, this problem is solved as a multi-armed bandit problem. This extension has many cost benefits. As the exchange of vehicles is mandatory, the exchange location will be visited many times. Having an exchange location close to the depots will reduce the costs associated with travel distance. Furthermore, having multiple exchange locations intuitively cuts costs by implicitly clustering the customer locations based on distance. In practical applications, the DVRP is a frequently repeated problem (e.g., daily in the Canary Islands). The amount of costs saved will accumulate. Therefore, it is beneficial

to invest and install more facilities that support exchanging vehicles.

The remainder of the paper is structured as follows. Section 2 provides a summary of the related literature. Section 3 provides the framework for the DVRP with multiple exchange locations and provides insight into the data instances. Section 4 and 5 constitute the methodology of the paper. In the former section, the heuristic algorithm is explained, while the latter section proposes modifications to determine good exchange locations. Section 6 and 7 form the numerical results. The former section discusses the issues with replicating the heuristic algorithm. The latter section focuses on the results of the proposed extensions. Lastly, the conclusions are gathered in Section 8.

2 Literature review

A VRP classically includes one depot which all vehicles depart from and return to (see Mor & Speranza, 2022, for a recent survey). The idea to expand the problem to multiple depots followed soon, with Laporte (1984) providing a branch-and-bound algorithm to find the optimal value. In the standard multi-depot VRP, every vehicle must start and end at the same depot. This allows efficient heuristics that first assign customer locations to depots, and then construct the routes in a second phase (e.g., Renaud et al., 1996). To augment flexibility and practicality, Crevier et al. (2007) describes the Multi-Depot VRP with Inter-depot routes (MDVRPI). In this problem, vehicles can end their route in any depot that is convenient, adding a layer of complexity. Our DVRP takes a unique approach with the depots. Multiple depots are present but strict rules apply on which depots to visit.

Traditionally, no distinction was made between the driver and the vehicle. Groër et al. (2009) emphasizes the relationship between the drivers and customers. Customer satisfaction is heightened when a driver consistently visits the same customers. Another paper that values the driver consistency is Spliet & Dekker (2016). The authors first assign customers to drivers when demand is unknown, and then create the vehicle routes when the demand is revealed. Despite the focus on the driver, a driver stays with one vehicle from start to finish. In terms of mathematical formulation, the driver and vehicle are still one entity.

Plenty of papers view the crew and the vehicle as separate factors. One approach is to view the route construction and crew scheduling as two separate problems. Examples of the sequential approach are Ball et al. (1983); Darby-Dowman et al. (1988). While commonly the routes of the vehicles are constructed first and the crew is assigned second, we reverse the order for the DVRP in this paper. This helps shift the focus from the vehicles to the crew. Another approach to the vehicle and crew scheduling problem is to integrate vehicle routing and crew scheduling. One of the earliest formulations for the single depot case is proposed in Freling et al. (1999). Cordeau et al. (2001) solves the same type of problem in air transportation. The integrated vehicle and crew scheduling problem is extended to multi-depot in Huisman et al. (2005). Hollis et al. (2006) discusses a similar problem in the context of postal services. The simultaneous approach received much more attention lately due to the many benefits that it offers. It easier to respect the working hours of the crew. Integration also generally leads to lower costs because the costs attributed to personnel often outweigh the costs attributed to vehicles. There is an increased focus on the crew because a large turnover has become a serious

problem, especially in the truckload industry (Vergara & Root, 2013). Srinivas & Gajanand (2017) presents a qualitative survey on the factors that influence driver behavior, especially in cases where the driver deviates from planning.

The DVRP was first defined in Domínguez-Martín et al. (2018a). In that paper, the authors created an Integer Linear Programming formulation that integrates the driver and vehicle constraints. Furthermore, they developed a branch-and-cut algorithm that can solve to optimality for instances with a size of up to 30 nodes. Domínguez-Martín et al. (2018b) subsequently modeled a two-phase heuristic for the problem with one exchange location. The first phase constructs driver routes using an Integer Linear Programming formulation, and the second phase constructs the corresponding vehicle routes. This approach solves instances with sizes up to 50. An improved version of the two-phase heuristic is described in Domínguez-Martín et al. (2023). In this method, the first phase of the heuristic is replaced with a multi-start loop. This multi-start heuristic regularly finds optimal solutions for small instances with a size of up to 30 nodes. The method also finds solutions of high quality in a short time for bigger instances with a size of up to 1000 nodes.

3 Problem description

The DVRP is defined on a complete directed graph $G = (V, A)$. The node set $V = D \cup V_c$ consists of the set of customer locations $V_c = \{1, ..., n\}$, and the set of depots $D = \{0, n + 1\}$. All customer locations must be visited by a vehicle, led by a driver. Therefore the number of nodes in the graph is $n + 2$. The set of arcs is $A = \{(i, j) : i, j \in V, i \neq j\}$. When traversing arc (i, j), a cost c_{ij} is incurred, and the time t_{ij} is needed. The total travel time of a driver must not exceed a given time limit T. The vehicle routes are not limited by time. Let K_d be the set of drivers available at each depot $d \in D$. All drivers must start and end their route at the same depot. Vehicles, on the other hand, must end in the other depot than the starting depot. This is possible when drivers switch vehicles in exchange locations. The set $E \subseteq V_c$ is the set of exchange locations. It is only possible for a driver to switch vehicles in these nodes. Multiple drivers can visit an exchange location. An exchange location must be visited at least once, while the other customer locations must be visited exactly once. To make the interactions at the exchange locations possible, the routes must be time-synchronized. This means that drivers must have consistent departure and arrival times at the exchange locations. However, when we assert that every driver and vehicle visit exactly one exchange location, time synchronization is no longer a decision because drivers and vehicles are free to leave when convenient. This assertion is reasonable because visiting multiple exchange locations leads to unnecessary costs. Furthermore, to guarantee the compatibility of driver routes and vehicle routes, we impose that at every exchange location, the number of visiting drivers from one depot equals the number of visiting drivers from the other depot. This guarantee is proven in Section 3.2.

To represent a driver route as a walk in the graph G, we introduce R^{l_d} as a sequence of nodes that the driver $l_d \in K_d$ leaving from depot $d \in D$ will visit. The goal of the DVRP is to find feasible routes for drivers and vehicles in G such that all customers are visited while minimizing the total costs. The exact mathematical formulation is described in Domínguez-Martín et al. (2018a).

3.1 Data

The data consists of three sets of randomly generated instances. Each of these instances specifies customer n to be the exchange location. The first set (Class I) appeared first in Domínguez-Martín et al. (2018a). The instances have size $n+2 \in \{10, 15, 20, 25, 30\}$ and the node coordinates are in the square $[0, 100] \times [0, 100]$. All depots, exchange locations, and other customer locations are uniformly distributed over the area. There are five instances for each size, resulting in 25 instances in Class I. The second set (Class II) appeared first in Domínguez-Martín et al. (2018b). The instances have size $n + 2 = 50$. The nodes have more structure. One depot is randomly placed in the area $[0, 20] \times [0, 100]$, and the other depot in $[80, 100] \times [0, 100]$. The exchange location is placed in $[40, 60] \times [0, 100]$. This configuration attempts to represent the common situation where the two depots are located far from each other, and the exchange location is placed at a central location between them. There are 16 instances of this class. Lastly, the third set of instances (Class III) is the set of the largest instances, first introduced in Domínguez-Martín et al. (2023). The instances have size $n + 2 \in \{100, 200, 300, 400, 500, 600, 800, 1000\}$. These instances are generated as in Class II, with two depots located far from each other and the exchange location between them. Each size also has five instances. All of these 81 instances are available at data.mendeley.com/datasets/w5sbtwy8y9/4.

3.2 Time synchronization

In the DVRP with one exchange location, Domínguez-Martín et al. (2018b) claims that when the number of drivers leaving from each depot is equal, then there exist compatible vehicle routes corresponding to the driver routes. While this sounds trivial, we present a formal proof. Let $G' = (V, A') = \bigcup_{d \in D} \bigcup_{l_d \in K_d} R^{l_d}$ be a directed multigraph, which is the union of all driver routes. Let $e \in V$ be the exchange location. Then every driver route is a cycle that visits e and either 0 or $n + 1$. Additionally, the in- and out-degree of a node $v \in V$ is denoted as $\delta^-(v)$ and $\delta^+(v)$ respectively.

Theorem 1. *If $\delta^+(0) = \delta^+(n + 1) = m, m \in \mathbb{N}$ and there is only one exchange location, then there exist compatible vehicle routes corresponding to G'.*

Proof. We prove this by induction.
Base step: $m = 1$. This means that there are two cycles that only share the node e. Let $(0, v_1, ..., v_{p-1}, e, v_{p+1}, ..., 0)$ be the cycle of the driver leaving from depot 0, with e being the p^{th} visit number. Let $(n+1, u_1, ..., u_{q-1}, e, u_{q+1}, ..., n+1)$ be the cycle of the other driver, with e being the q^{th} visit number. All u_i and v_i are different. The first vehicle route is the path $(0, v_1, ..., v_p - 1, e, u_{q+1}, ..., n + 1)$. The second vehicle route is is the path $(n + 1, u_1, ..., u_{q-1}, e, v_{p+1}..., 0)$. Because the union of these paths equals G', no driver moves without a vehicle and no vehicle moves without a driver.
Induction hypothesis: Let $M \in \mathbb{N}$ and let G'_M be a graph containing the driver cycles. Assume that there exist compatible vehicle routes for G'_M.
Inductive step: Let G'_{M+1} be a graph of driver routes such that $\delta^+(0) = \delta^+(n + 1) = M + 1$. We select an arbitrary driver $l_0 \in K_0$ and an arbitrary driver $l_{n+1} \in K_{n+1}$. From the two corresponding cycles, we construct two vehicle paths in the same manner as described in the

base step. Denote the paths as P_0 and P_{n+1}. Then, we consider the subgraph $G'_M \subset G'_{M+1}$ where all arcs and non-exchange customer nodes in the two vehicle paths are removed. This eliminates l_0 and l_{n+1} from the resulting problem. In G'_M, the out-degrees of both depots will be M, because exactly one outgoing arc is removed from each depot. By the induction hypothesis, G'_M has compatible vehicle routes for $(K_0 \cup K_{n+1}) \setminus \{l_0, l_n + 1\}$. Taking the union of P_0, P_{n+1}, and the vehicle routes from G'_M, we obtain a set of vehicle routes that ensure that no driver moves without a vehicle and no vehicle moves without a driver. □

We can extend Theorem 1 to multiple exchange locations. We only require the condition that the number of drivers from one depot that visit some exchange location must be equal to the number of drivers from the other depot that visit the exchange location.

Corollary 2. *If $\forall e \in E$: the number of cycles which include e and 0 equals the number of cycles which include e and $n + 1$, then there exist compatible vehicle routes corresponding to G'.*

Proof. Let $G'_e \subset G'$ be a subgraph that contains all the driver cycles that visit exchange location $e \in E$. For illustration, $\bigcup_{e \in E} G'_e = G'$ and $\bigcap_{e \in E} G'_e = \{0, n + 1\}$. For every G'_e, the graph has one exchange location. In the directed multigraph G'_e, the number of cycles that visit e and 0 equals $\delta^+(0)$ and the number of cycles that visit e and $n + 1$ equals $\delta^+(n + 1)$. Therefore, by Theorem 1, every subgraph G'_e has compatible vehicle routes. The union of all the vehicle routes equals G'. Therefore, no driver moves without a vehicle and no vehicle moves without a driver. □

4 Heuristic algorithm

In this section, we describe the heuristic algorithm to solve the DVRP. Domínguez-Martín et al. (2023) originally defined the heuristic as a two-phase algorithm, where the first phase creates the driver routes, and the second phase creates the vehicle routes. For the remainder of the paper, we omit the second phase. It has an insignificant impact on the computation time. Theorem 1 and Corollary 2 also ensure the existence of the vehicle routes. Domínguez-Martín et al. (2018b) describes an effective algorithm to construct the vehicle routes given the driver routes. Algorithm 1 describes the multi-start heuristic for the construction of driver routes. While the time limit is not reached, the heuristic tries finding a solution with a fixed number of drivers $nDrivers$ leaving each depot, starting with $nDrivers = 1$. It searches for a solution using a multi-start loop, which is described in lines 6 to 14 in the algorithm. In the loop, a new solution is repeatedly constructed and improved upon. The lowest-cost feasible solution is kept. Infeasible solutions are disposed of. A solution is feasible if every customer is served and no driver route has a duration exceeding T. This loop is repeated a specified number of times $maxIter$. If no solution is found after this number of iterations, $nDrivers$ is increased, and the multi-start loop repeats.

4.1 Initial construction of driver routes

Let $G' = \bigcup_{d \in D} \bigcup_{l_d \in K_d} R^{l_d}$ be the set of all driver routes. As every route must visit an exchange location, the algorithm initializes $R^{l_d} = (d, e, d)$ for a given $e \in E$. The strategy to choose e is

Algorithm 1 Multi-start heuristic for the DVRP, adapted from Domínguez-Martín et al. (2023)

Input: Instance data and parameters `timeLim` and `maxIter`
Output: Drivers' routes S^* and solution value f^*

$f^* \leftarrow \infty$
$S^* \leftarrow \emptyset$
$nDrivers \leftarrow 1$
while $time \leq timeLim$ **and** no feasible solution S^* found **and** $nDrivers \leq maxDrivers$ **do**
  $nIter \leftarrow 1$
  while $time \leq timeLim$ **and** $nIter \leq maxIter$ **do**
    $S \leftarrow$ `ConstructGreedySol`$(nDrivers)$
    $S \leftarrow$ `LocalSearch`(S)
    if S is feasible **and** $f(S) < f^*$ **then**
      $S^* \leftarrow S$
      $f^* \leftarrow f(S)$
    end if
    $nIter \leftarrow nIter + 1$
  end while
  if not feasible solution S^* found **then**
    $nDrivers \leftarrow nDrivers + 1$
  end if
end while
return S^*, and f^*

described in Section 5. It continues by repeatedly inserting a customer in a route until G' covers all nodes. In every iteration, a customer $i \in V_c$ that is not in any route is randomly selected and is added to a route using the cheapest insertion strategy. This means that we search for two consecutive nodes u, v in any route in G' such that $c_{ui} + c_{iv} - c_{uv}$ is minimal. The customer is inserted in the selected route if the route duration does not exceed T. If it does, then the customer i is inserted into the route with the shortest duration, following again the cheapest insertion strategy.

To inspect the time complexity, we specify $r = 2 * nDrivers$ as the number of routes. In one multi-start iteration, the greedy construction is $O(n^2)$. The first node to be inserted must consider $2r$ number of positions. For each route, it must be inserted either right before or after the exchange node. The second node must consider $2r + 1$ positions. Continuing this way, the i^{th} node must consider $2r + i - 1$ positions. In total, using the arithmetic series formula, we have $n\frac{2r+2r+n-1}{2}$ options to consider, which is $O(n^2)$.

4.2 Improvements of driver routes

The improvement procedure consists of two parts. First, an inter-route customer relocation local search is repeated until there is no feasible improvement. Second, a 2-opt arc exchange local search is repeated until a local minimum is reached. This simple design is demonstrated to be very effective in Domínguez-Martín et al. (2023).

The inter-route operator randomly chooses a customer in G' that is not an exchange location. The drawn customer is removed from its current route and is inserted into another route using

the cheapest insertion strategy. Every other route is examined for an insertion that would lead to the biggest reduction in costs. The insertion must be time-feasible. This means that the relocation is not considered if it causes a route's duration to exceed the time limit T. This may cause difficulty in escaping a local minimum. The insertion must also lead to lower costs. A disadvantage is that the heuristic might fail to find a feasible solution, although it exists. This can occur when there is an imbalance in the lengths of the driver routes. Suppose that there is a short route and a long route that exceeds the time limit. Relocating a customer from the long route to the short route possibly makes the solution feasible. However, this option is neglected if it increases the costs. When no feasible relocation leads to lower costs, the next customer is considered. When every customer is examined and no customer relocation leads to lower costs, we move to the 2-opt local search.

In the 2-opt operator, for every route, we consider all non-adjacent pairs of arcs. For every pair, we calculate the change in costs if we were to delete the arcs and reconnect the route. The pair of arcs that leads to the largest decrease in costs is swapped in that manner. This procedure is repeated as long as a cost-decreasing arc exchange is found. It is unnecessary to take the time feasibility into account, as the arc exchanges cannot increase the duration of the route.

The running time of the inter-route and the 2-opt local search depends on the number of iterations. The number of iterations depends on whether improvements are found, and numerical experiments are conducted to inspect this. For one iteration of the inter-route operator, the best-case time is $O(n)$. In this case, a customer i considers all positions outside its route to relocate to and finds a feasible improvement. If π is the fraction of nodes in the graph that are in the same route as i, then $(1-\pi)n$ positions must be considered, which is $O(n)$. The worst-case running time of one iteration is $O(n^2)$, which happens at the termination of the inter-route local search. When a customer fails to find a feasible improvement, the next customer is considered. The local search terminates when all customers are inspected and none yield a desirable relocation. Therefore, n customers examine $O(n)$ positions, which is $O(n^2)$. Under the assumption that the number of iterations is uncorrelated with n, the whole inter-route relocation would be $O(n^2)$. In this case, the last iteration, which is $O(n^2)$, dominates all the other iterations. This assumption is investigated. One iteration of the 2-opt local search is $O(n^2)$. For one route, let π be the fraction of nodes in that route. One route is a cycle, which means the number of arcs equals the number of nodes. Every pair of non-adjacent arcs must be examined to find the best exchange. The number of pairs has the upper bound $\binom{\pi n}{2}$, which is $O(n^2)$.

5 Finding optimal exchange locations

Domínguez-Martín et al. (2023) considered the restrictive scenario of the DVRP where only one exchange location is present. Furthermore, the exchange location is arbitrarily placed, especially in Class I instances. This section describes the strategy to find multiple alternative exchange locations.

5.1 Candidate exchange locations

To inspect the characteristics of favorable exchange locations, we solve the DVRP using alternative exchange locations. We first define what makes a customer node a *candidate* exchange location. A customer location is a *candidate* if the highest distance of the location to a depot does not exceed the distance between the two depots. We further introduce C as the set of *candidate* locations.

We solve the DVRP with one alternative exchange node. Two different criteria are inspected to select the exchange location. The first criterion is the minimum distance to one depot, under the constraint that the location is a *candidate*. Therefore, we select a location as close as possible to one depot without being too far from the other. In a graph with no *candidates* locations, which happens when two depots are very close to each other, we do not inspect this criterion. The second criterion is the sum of the distances of the location to the two depots. Therefore, we are searching for a location in the middle of the two depots, irrespective of the location being a *candidate*. We compare the results with the original exchange locations described in Section 3.1.

5.2 Deciding on good exchange locations

To satisfy the conditions of Corollary 2, we decide the exchange location for each pair of drivers coming from different depots. This decision will take place exclusively in the initial greedy construction. It is not effective to incorporate switching the exchange location as part of the solution improvement. Switching leads very unlikely to lower costs because of the cheapest insertion strategy during the greedy construction. Due to the multi-start nature of the algorithm, we approach the greedy construction as a multi-armed bandit problem. We employ the epsilon-decreasing strategy (Sutton & Barto, 2018); for a varying ϵ, the best currently known exchange locations are chosen with a probability of $1 - \epsilon$. Otherwise, with probability ϵ, a random set of exchange locations is chosen. With a decreasing ϵ, exploration is emphasized at the start, and exploitation is emphasized near the end of the algorithm. We use the simple linear function $\epsilon = 1 - \frac{nIter}{maxIter}$. More complex strategies for the multi-armed bandit are left for future research. When deciding on the set of exchange locations, we consider a k-combination with repetitions of the set of *candidate* locations, with $k = nDrivers$. If $k = 1$, the heuristic simply optimizes the exchange location. If there are no *candidates*, which happens when the two depots are close to each other, then the set C contains one node for which the sum of the distances to the depots is minimal. Let $|C|$ be the number of *candidate* locations. The number of possible combinations is $\binom{|C|+k-1}{k}$. If this value exceeds the maximum number of multi-start iterations $maxIter$, then the exploration of *candidate* locations is limited. The time complexity of the greedy construction is unaffected as all the multi-armed bandit operations are constant time.

6 Results of replication

The algorithm in Section 4 was coded in Java. All experiments were run on a computer with an Intel(R) Core(TM) i7-11800H CPU @ 2.30 GHz, 16 GB RAM, and Windows 11 Home. We replicate Domínguez-Martín et al. (2023) for the values of the parameters, as their configuration

showed an appropriate balance between solution quality and running time. Throughout all instances of data, the number of drivers available $maxDrivers$ at each depot is three. The costs c_{ij} equal the Euclidean distance between location i and j. The time required to traverse arc (i, j) is defined as $t_{ij} = c_{ij}/60 + 0.5$. The maximum number of multi-start iterations $maxIter$ is set to $100,000$. The parameter T, the upper bound of the duration the driver routes, takes different values for different instances and is reported in the tables below. For Class I and Class III instances, we inspect small values of $T = T_A$ that still allow feasible solutions and large values of $T = T_B$ [1] that generate cheap solutions with one driver. The maximum computation time $timeLim$ is set to 2 hours, which is the same limit as in Domínguez-Martín et al. (2018a). This value is chosen in contrast to the 10 minutes in Domínguez-Martín et al. (2018b) as our experiments failed to find feasible solutions for large datasets. We employ these settings throughout this entire section and Section 7.

6.1 Computation time

The computation time of the algorithm depends on three major factors. The first is the size of the instance $n + 2$. The second is the bound of driver route duration T. Smaller values of T allow few feasible solutions, requiring the algorithm to search longer and inspect multiple values of $nDrivers$. The third is the number of multi-start iterations $maxIter$. However, the value of this parameter is fixed at 100,000 throughout the paper. Therefore we do not consider this factor.

With the time limit set to 2 hours, we do not consider data instances with a size of 500 or higher. The heuristic finds a feasible solution only up to size 400 within the time limit. This contradicts Domínguez-Martín et al. (2023), in which their heuristic always found a solution within 10 minutes for all datasets, including the largest with size 1000. Their algorithm was coded in C++ using a desktop computer with an Intel(R) Core(TM) i7-10750H CPU @ 2.60 GHz, 16 GB RAM, and Windows 10 Home. According to the single-thread rates available at www.cpubenchmark.net, our hardware contradictorily has a better performance. The discrepancy in computation times is too large to attribute to the difference in software. A possible explanation is the difference in interpretation of the algorithm. The authors were not transparent in their approach. In that paper, they state that they repeat the inter-route and 2-opt operator while the total costs decrease. This has an ambiguous meaning. Our algorithm stops the inter-route operator when no customer has an improving relocation. A faster alternative would be to terminate when the randomly chosen customer has no improving relocation, so other customers would not be considered. Furthermore, our algorithm finds the best improvement among all possible arc exchanges. A faster alternative would be to perform the first improvement. In this case, not every pair of non-adjacent arcs must be examined every iteration. These faster alternatives do not search for improvements as thoroughly as our implementation. Therefore, we would expect solutions of worse quality if these alternatives are employed.

To dissect the computation times of our algorithm, we time how much each of the three operators contributes to the total running time. The three operators are the greedy construction, inter-route relocation, and 2-opt. We inspect one instance of each size $n + 2 \in$

[1] The values of T_A and T_B correspond to T_A and T_D in Domínguez-Martín et al. (2023), respectively

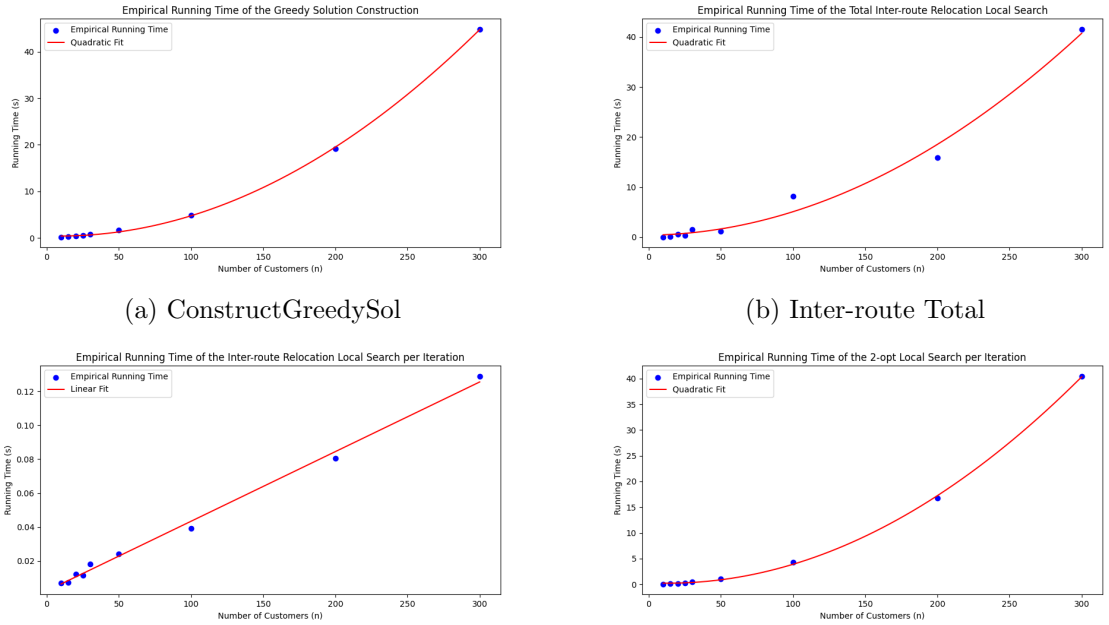
$\{10, 15, 20, 25, 30, 50, 100, 200, 300\}$. We solve the DVRP with $T = T_B$ for a fair comparison, as the algorithm always finds a feasible solution with one driver per depot. Furthermore, we select the exchange location specified in the data instance, as in Domínguez-Martín et al. (2023). We additionally track the average number of inter-route iterations and 2-opt iterations across multi-start iterations. This means we calculate the average number of customer relocations and arc exchanges in one multi-start iteration. The results are reported in Table 1.

Name	T	k	**Total time**	**Greedy**	**Inter-route**		**2-opt**	
					Time	**Iterations**	**Time**	**Iterations**
n10-1	10	1	0.42s	0.19s	0.05s	7.02	0.18s	2.22
n15-1	12	1	0.68s	0.24s	0.12s	15.49	0.32s	2.44
n20-1	14	1	1.34s	0.36s	0.59s	47.70	0.40s	2.62
n25-1	16	1	1.61s	0.50s	0.31s	26.67	0.80s	2.70
n30-1	18	1	3.74s	0.78s	1.51s	83.07	1.45s	3.21
n50-1	18	1	8.87s	1.70s	1.13s	47.00	6.04s	5.66
n100-1	40	1	48.75s	4.85s	8.14s	208.13	35.76s	8.32
n200-1	60	1	293.33s	19.16s	15.87s	197.00	258.29s	15.40
n300-1	90	1	945.92s	44.87s	41.57s	322.23	859.48s	21.23

Table 1: Summary of Computation Times and Iterations

Evidently, the running time of each operator increases as the size increases. The running time of 2-opt grows the fastest and requires the majority of the time. Between the greedy construction and inter-route operator, there is not one that is consistently faster. The number of iterations in the inter-route relocation has a volatile increasing trend, while for 2-opt a much more stable increasing trend is visible. When inspecting the number of iterations, we find surprisingly that the inter-route relocation has many more than the 2-opt. This means that the inter-route finds improvements more frequently. Despite that the 2-opt exchanges happen less often, they still require more time.

Figure 1 shows evidence in favor of the theoretical running times explained in Section 4. The horizontal axis shows the number of customers n, and the vertical axis shows the time in seconds. The blue dots are the observed running times and the red line is the fitted curve an^2+bn+c. The coefficient parameters a, b, and c are estimated using Ordinary Least Squares. Figure 1a shows the growth of greedy construction time. Figure 1b shows the evolution of the total computation time of the inter-route local search. Both resemble a quadratic parabola, giving evidence that both operators are $O(n^2)$. However, the inter-route operator has a worse fit. This is likely because the number of iterations has an increasing trend. The number of iterations must be uncorrelated with n for the local search to be $O(n^2)$. Figures 1c and 1d shows the average time per iteration of the inter-route relocation and 2-opt, respectively. The times are calculated by dividing the time attributed to the operator by the average number of iterations. For the inter-route relocation, the running times show a linear pattern. This implies that the best case, where an immediate improvement is found, happens the most often. Therefore, most of the total computation time is attributed to the few dominating worst cases that are $O(n^2)$. Because the 2-opt has a quadratic growth in every iteration, the growth in computation time is the fastest, contributing the most to the total computation time of the heuristic. For the bigger datasets with a size of 100 or higher, the average running times per iteration of 2-opt are close to the



(a) ConstructGreedySol

(b) Inter-route Total

(c) Inter-route per Iteration

(d) 2-opt per Iteration

Figure 1: Running time of parts of the heuristic

total running of the greedy construction and inter-route relocation. This gives further indication that the time complexities are $O(n^2)$.

6.2 Solution quality

Next to discrepancies in computation time, there are also some sizable gaps in the solution values. The results of the multi-start heuristic taken from Domínguez-Martín et al. (2023) are associated with the symbol H_2, which is used in that paper. The columns under e_n are the results of our attempt to replicate the heuristic. This symbol signifies that the n^{th} customer serves as the exchange location. Despite that the H_2 also follows this setting, the reason for choosing e_n will be apparent in Section 7.1. Furthermore, the column T_A or T_B sets the maximum duration of a driver route, with T_A being a small value and T_B being a bigger value. The column sol is the objective value of the solution, the total costs of all routes. The parameter k is the number of drivers per depot. Lastly, the Gap is the percentage deviation in solution value between the original H_2 heuristic and the replicated e_n heuristic. The value is defined as $100*((\textit{replication solution value})$ - $(\textit{original solution value}))$ / $(\textit{original solution value})$. A positive gap means that our replication has higher costs.

For the sake of brevity and ease of reading, not all instances are displayed. The presented datasets represent their size in terms of running time and solution quality. All results of the replication are available in Appendix A. Table 2 and 3 show a comparison for Class I and Class II instances, respectively. The gaps between the results are small for the two instance classes. The number of drivers required always stays the same. The differences in solution value are likely explained by different precision levels of decimal numbers. Furthermore, many solutions are optimal, indicated with an asterisk * in the tables. The optimality is confirmed using the mathematical formulation of the DVRP (Domínguez-Martín et al., 2018a). An anomalous

observation is the instance n15-3 with $T = T_A$. The absolute gap is too large to be justified by the difference in rounding.

Name	T_A	H_2		e_n			T_B	H_2		e_n		
		sol	k	sol	k	Gap		sol	k	sol	k	Gap
n10-1	6	652*	2	657	2	0.73	10	369*	1	371	1	0.51
n10-2	5	486*	2	486	2	0.08	10	292*	1	293	1	0.44
n10-3	5	987*	3	987	3	-0.04	10	383*	1	383	1	-0.08
n15-1	6	454*	2	456	2	0.34	12	302*	1	302	1	-0.04
n15-2	6	746*	2	746	2	-0.06	12	406*	1	405	1	-0.25
n15-3	6	723	2	660	2	-8.70	12	388*	1	387	1	-0.28
n20-1	7	1268	3	1275	3	0.52	14	520*	1	521	1	0.22
n20-2	7	666*	2	667	2	0.10	14	399*	1	402	1	0.70
n20-3	7	845*	2	846	2	0.14	14	507*	1	508	1	0.17
n25-1	8	718*	2	720	2	0.31	16	483*	1	483	1	0.01
n25-2	8	801	2	805	2	0.56	16	483*	1	486	1	0.68
n25-3	8	603*	2	601	2	-0.32	16	405*	1	402	1	-0.62
n30-1	9	897	2	896	2	-0.07	18	581*	1	582	1	0.25
n30-2	9	854	2	861	2	0.08	18	552*	1	553	1	0.26
n30-3	9	812	2	812	2	0.01	18	485*	1	487	1	0.34

Table 2: Replication results for Class I instances

Name	T	H_2		e_n		
		sol	k	sol	k	Gap
n50-1	18	606*	1	608	1	0.38
n50-2	18	592	1	591	1	-0.16
n50-3	18	585	1	586	1	0.12
n50-4	18	593*	1	597	1	0.69
n50-5	18	613	1	617	1	0.60

Table 3: Replication results for Class II intances

Table 4 shows a comparison for Class III instances. The number of drivers required is consistent. The gap remains close to zero when $T = T_B$. The differences are likely again due to precision. However, when $T = T_A$, our replication consistently performs better. The gaps are always negative and are substantial on some occasions. This indicates that our implementation of the multi-start heuristic finds better solutions at the cost of higher computation times than Domínguez-Martín et al. (2023). This means that the authors most probably applied the faster alternatives to the local search operators as described in Section 6.1.

Name	T_A	H_2		e_n			T_B	H_2		e_n		
		sol	k	sol	k	Gap		sol	k	sol	k	Gap
n100-1	15	1130	3	1129	3	-0.07	40	738	1	741	1	0.40
n100-2	15	1246	3	1238	3	-0.63	40	787	1	788	1	0.16
n200-1	25	1721	3	1575	3	-8.50	60	1120	1	1141	1	1.85
n200-2	25	1865	3	1687	3	-9.54	60	1139	1	1140	1	0.12
n300-1	35	1816	3	1781	3	-1.95	90	1375	1	1380	1	0.39
n300-2	35	2189	3	2047	3	-6.51	90	1340	1	1335	1	-0.34
n400-1	45	2229	3	2109	3	-5.38	115	1573	1	1551	1	-1.40
n400-2	45	2007	3	1974	3	-1.67	115	1536	1	1531	1	-0.35

Table 4: Replication results for Class III intances

7 Results of optimizing exchange locations

This section provides the results of the extensions of Section 5. The modifications were implemented in Java as well. The hardware and the value of all parameters are the same as specified in Section 6.

7.1 Candidate exchange locations

This section investigates the characteristics of favorable exchange locations. In each of the following tables, e_n refers to the solution where the customer n is arbitrarily chosen as the exchange location (Section 3.1). On the other hand, e_C and e_M show solutions where the exchange location is selected using the first and second criterion respectively. The columns under e_C show the results when the *candidate* location closest to a depot serves as the exchange node. The solutions under e_M appoint the customer with the minimal sum of distances to the two depots as the exchange node. Similarly to the previous section, T_A and T_B set the maximum duration of a route, sol refers to the solution value and k is the number of drivers. Additionally, Time refers to the computing time in seconds. The Gap is now the percentage deviation in solution value between using customer n and the alternative as the exchange location. The value is calculated as $100*((\textit{alternative solution value})$ - $(e_n$ *solution value*$))$ / $(e_n$ *solution value*$)$. A negative gap means that the alternative solution has lower costs than the original. Lastly, for every table, we report the averages of the solution values and gaps. The purpose of these values is to give a superficial impression of the results.

Not all results are presented. The reported averages are the averages of the displayed instances. The full results are available in Appendix A. Tables 5 and 6 show the results for Class I with low and high values for T respectively. In the datasets n10-3, n25-2, and n30-2, no customer was a *candidate* because the depots were close to each other. Using alternative exchange locations proved to decrease the costs significantly. Both the first and second criteria often lead to lower objective values. The second criterion provides better solutions than the first criterion more frequently. The lower average gap also indicates that the second criterion is superior.

Interestingly, if the original exchange location n is not a *candidate*, switching to the second criterion location always leads to improvement. If the original is a *candidate*, the improvement is arbitrary. Some instances show better results, some do not. The gap in solution value is generally bigger with strict T values. Specifically, in n10-3 and n20-1, a solution is found using one driver fewer, reducing the costs immensely. As fewer possibilities for the number of drivers are examined, the computation time is also reduced.

Name	T_A	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n10-1	6	656.79	2	0.75	543.91	2	0.79	-17.19	462.47	2	0.64	-29.59
n10-2	5	486.40	2	0.56	506.18	2	0.56	4.07	521.52	2	0.61	7.22
n10-3	5	986.58	3	0.88	-				546.33	2	0.57	-44.62
n15-1	6	455.55	2	1.28	446.85	2	1.29	-1.91	446.85	2	1.29	-1.91
n15-2	6	745.57	2	1.09	598.56	2	1.2	-19.72	664.07	2	1.21	-10.93
n15-3	6	660.07	2	1.12	556.13	2	1.21	-15.75	529.55	2	1.38	-19.77
n20-1	7	1274.56	3	3.14	705.37	2	1.96	-44.66	705.37	2	1.96	-44.66
n20-2	7	666.67	2	1.96	682.87	2	1.94	2.43	596.18	2	2.06	-10.57
n20-3	7	846.16	2	1.94	615.65	2	2.08	-27.24	615.65	2	2.08	-27.24
n25-1	8	720.21	2	3.08	805.67	2	3.00	11.87	878.05	2	2.96	21.92
n25-2	8	805.47	2	2.87	-				743.89	2	3.11	-7.65
n25-3	8	601.05	2	3.37	694.03	2	3.57	15.47	621.04	2	3.60	3.33
n30-1	9	896.36	2	4.23	680.35	2	4.84	-24.10	680.35	2	4.84	-24.10
n30-2	9	860.85	2	4.50	-				697.57	2	4.78	-18.97
n30-3	9	812.05	2	5.37	653.59	2	4.94	-19.51	598.21	2	5.06	-26.33
Average	-	764.96	-	-	624.10	-	-	-11.35	620.47	-	-	-15.59

Table 5: Results for Class I instances with tight T values with one exchange location

Name	T_B	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n10-1	10	370.89	1	0.27	371.62	1	0.24	0.20	340.17	1	0.25	-8.28
n10-2	10	293.28	1	0.19	298.16	1	0.29	1.66	292.82	1	0.26	-0.16
n10-3	10	382.70	1	0.23	-				337.15	1	0.25	-11.90
n15-1	12	301.87	1	0.60	305.09	1	0.52	1.07	305.09	1	0.52	1.07
n15-2	12	405.00	1	0.48	329.41	1	0.60	-18.66	318.26	1	0.77	-21.42
n15-3	12	386.90	1	0.48	341.84	1	0.66	-11.65	338.67	1	0.56	-12.47
n20-1	14	521.12	1	1.27	465.61	1	1.41	-10.65	465.61	1	1.41	-10.65
n20-2	14	401.81	1	0.99	428.65	1	1.18	6.68	385.82	1	1.18	-3.98
n20-3	14	507.87	1	1.19	462.19	1	1.36	-8.99	462.19	1	1.36	-8.99
n25-1	16	483.04	1	1.42	511.09	1	2.04	5.81	541.43	1	2.11	12.09
n25-2	16	486.29	1	1.91	-				462.62	1	2.26	-4.87
n25-3	16	402.49	1	1.74	443.50	1	1.60	10.19	415.52	1	1.82	3.24
n30-1	18	582.43	1	3.18	539.50	1	3.04	-7.37	539.50	1	3.04	-7.37
n30-2	18	553.43	1	2.50	-				534.53	1	3.41	-3.42
n30-3	18	486.64	1	3.28	436.27	1	3.30	-10.35	434.64	1	3.03	-10.69
Average	-	437.72	-	-	411.08	-	-	-3.51	411.60	-	-	-5.85

Table 6: Results for Class I instances with loose T values with one exchange location

Table 7 shows the results of Class II instances, where $T = 18$. In contrast to Class I, changing exchange locations rarely leads to cost reduction. Similar to a prior observation, when

the exchange node is a *candidate*, the improvements are arbitrary. In this class, the initial exchange location n always is a *candidate*. Therefore, few improvements are found. Among the five samples, no solution with an alternative exchange location has lower costs. The algorithm frequently needs two drivers instead of one to find a feasible solution, causing gaps in solution value of at least 15%. The computation time is also affected negatively.

Name	T	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n50-1	18	608.29	1	8.45	695.26	2	21.05	14.30	708.7	2	21.51	16.51
n50-2	18	591.06	1	6.83	687.62	2	19.63	16.34	592.2	1	7.68	0.19
n50-3	18	585.70	1	9.69	707.24	2	19.75	20.75	693.64	2	21.14	18.43
n50-4	18	597.10	1	7.49	675.07	2	20.90	13.06	615.53	1	7.59	3.09
n50-5	18	616.66	1	9.07	741.87	2	22.59	20.30	750.57	2	23.73	21.72
Average	-	599.76	-	-	701.41	-	-	16.95	672.13	-	-	11.99

Table 7: Results for Class II instances with one exchange location

Table 8 and 9 show the results for Class III instances with a low and high value of T, respectively. A similar pattern is visible as in Class II. The alternative exchange locations do not consistently improve the solutions, due to the characteristics of the exchange location n.

Name	T_A	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n100-1	15	1129.24	3	100.70	1393.97	3	87.56	23.44	1393.97	3	87.57	23.44
n100-2	15	1238.10	3	98.26	1262.44	3	91.16	1.97	1219.55	3	91.99	-1.50
n200-1	25	1574.77	3	536.97	1760.87	3	505.66	11.82	1553.33	3	512.76	-1.36
n200-2	25	1687.01	3	527.93	1745.22	3	494.98	3.45	1745.22	3	494.98	3.45
n300-1	35	1780.61	3	1626.38	2143.58	3	1459.08	20.38	1902.87	3	1504.76	6.87
n300-2	35	2046.51	3	1582.45	2086.63	3	1481.98	1.96	1826.19	3	1572.64	-10.77
n400-1	45	2109.17	3	3683.42	2375.76	3	3315.56	12.64	2206.67	3	3538.23	4.62
n400-2	45	1973.55	3	3709.12	2313.91	3	3343.26	17.25	1985.88	3	3510.73	0.62
Average	-	1692.37	-	-	1885.2975	-	-	11.61	1729.21	-	-	3.17

Table 8: Results for Class III instances with tight T values with one exchange location

Name	T_B	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n100-1	40	740.94	1	49.13	869.22	1	59.32	17.31	869.22	1	55.53	17.31
n100-2	40	788.22	1	48.38	788.72	1	57.89	0.06	779.28	1	55.41	-1.13
n200-1	60	1140.77	1	308.41	1193.78	2	296.86	4.65	1130.96	1	304.54	-0.86
n200-2	60	1140.32	1	306.47	1208.67	2	675.19	5.99	1208.67	2	675.19	5.99
n300-1	90	1380.37	1	980.23	1406.87	2	1964.33	1.92	1391.18	1	965.20	0.78
n300-2	90	1335.43	1	1014.02	1366.26	2	1972.34	2.31	1336.74	1	966.22	0.10
n400-1	115	1551.02	1	2354.91	1563.50	2	4590.02	0.80	1548.95	1	2305.22	-0.13
n400-2	115	1530.55	1	2282.85	1577.96	2	4445.27	3.10	1546.47	1	2303.86	1.04
Average	-	1200.95	-	-	1246.87	-	-	4.52	1226.43	-	-	2.89

Table 9: Results for Class III instances with loose T values with one exchange location

Generally, the second criterion exchange location performs better than the first criterion.

Throughout all three classes of instances, the average gap in solution value is lower for the second criterion. In summary, if the exchange location is too far from a depot (is not a *candidate*), then it is highly beneficial to consider relocating it to the center of the two depots. If the exchange location is sufficiently close to both depots (is a *candidate*), then there is no clear measure to find better alternatives.

As established, the running time depends on the size of the data and the number of drivers that are needed for a feasible solution. This is visible throughout all datasets. A different exchange location only influences the running time of the algorithm through the number of drivers. For any solution where the different exchange locations yield solutions with the same number of drivers, the running times are the same. Any disparity is caused by randomness. When a solution is found with one driver fewer, as commonly in Class I, the computation time decreases significantly. Oppositely, if there is one driver more, as commonly in Class II and III, the computation time increases significantly.

7.2 Deciding on good exchange locations

In this section, we inspect whether allowing multiple exchange locations shows significant improvement over only using exchange location n. If only one exchange location is used, then we inspect whether exploring various exchange locations is beneficial. In the following tables, e_n again refers to the solutions with exchange location n. The columns under E show the results when multiple exchange locations are used. All other columns have the same definition as in the previous section. The average solution value and average gap of the displayed instances are also reported. For the sake of brevity, the running times of the algorithms are not reported but can be found in Appendix B. The strategy to select new exchange locations does not contribute visibly to the running time. Similar to the previous section, the computation time is only affected if a solution is found with a different number of drivers.

Table 10 shows a sample of the results of Class I instances. All results are available in Appendix B. In all of the 25 instances, solving the DVRP with a tight value $T = T_A$ leads to cost improvement. Furthermore, the absolute gap is also always bigger than or equal to the gap in the e_n solutions with one exchange location. There is more cost reduction when $T = T_A$ than when $T = T_B$. In the latter case, a solution is always found with one driver from each depot. This means that only one exchange location is used, neglecting the advantages of using multiple exchange locations. Therefore, there is not always a cost improvement when the duration of the driver route is large.

Table 11 shows results of Class II instances with one value of T. There is frequently a small improvement in costs. The size of the improvements is limited because only one driver per depot is involved in most solutions. This shows that searching for good exchange locations as a multi-armed bandit problem consistently leads to good solutions without the expense of longer computation times.

Name	T_A	e_n		E			T_B	e_n		E		
		sol	k	sol	k	Gap		sol	k	sol	k	Gap
n10-1	6	656.79	2	436.79	2	-33.50	10	370.89	1	340.17	1	-8.28
n10-2	5	486.40	2	463.89	2	-4.63	10	293.28	1	281.56	1	-4.00
n10-3	5	986.58	3	546.33	2	-44.62	10	382.70	1	337.15	1	-11.90
n15-1	6	455.55	2	444.57	2	-2.41	12	301.87	1	305.09	1	1.07
n15-2	6	745.57	2	580.66	2	-22.12	12	405.00	1	318.26	1	-21.42
n15-3	6	660.07	2	512.11	2	-22.42	12	386.90	1	338.67	1	-12.47
n20-1	7	1274.56	3	705.37	2	-44.66	14	521.12	1	465.61	1	-10.65
n20-2	7	666.67	2	583.09	2	-12.54	14	401.81	1	413.72	1	2.96
n20-3	7	846.16	2	586.99	2	-30.63	14	507.87	1	457.29	1	-9.96
n25-1	8	720.21	2	688.56	2	-4.39	16	483.04	1	482.20	1	-0.17
n25-2	8	805.47	2	732.46	2	-9.06	16	486.29	1	462.62	1	-4.87
n25-3	8	601.05	2	530.72	2	-11.70	16	402.49	1	402.49	1	0.00
n30-1	9	896.36	2	634.25	2	-29.24	18	582.43	1	531.56	1	-8.73
n30-2	9	860.85	2	696.70	2	-19.07	18	553.43	1	534.53	1	-3.42
n30-3	9	812.05	2	611.97	2	-24.64	18	486.64	1	435.51	1	-10.51
Average	-	764.96	-	583.63	-	-21.04	-	437.72	-	407.10	-	-6.82

Table 10: Results for Class I instances with multiple exchange locations

Name	T	e_n		E		
		sol	k	sol	k	Gap
n50-1	18	608.29	1	591.62	1	-2.74
n50-2	18	591.06	1	587.65	1	-0.58
n50-3	18	585.70	1	581.59	1	-0.70
n50-4	18	597.10	1	594.31	1	-0.47
n50-5	18	616.66	1	613.62	1	-0.49
Average	-	599.76	-	593.76	-	-1.00

Table 11: Results for Class II instances with multiple exchange locations

Table 12 shows the results of Class III instances. Similarly to Class I, when $T = T_A$, there is always a sizable negative gap. However, when only one driver per depot is needed to satisfy the driver route constraint, the absolute size of the gap diminishes and improvements are less frequent. Instance n100-1 has 86 *candidate* locations. With three drivers from each depot, the number of possible combinations of exchange locations is 109,736. Instance n200-1 has 115 *candidates*. With the same number of drivers, the number of combinations exceeds a million. Details on the number of *candidates* for all the other datasets can be found in Table 22 in Appendix B. With the maximum number of multi-start iterations fixed at 100,000, the exploration of new exchange locations is limited for Class III instances. Despite this, the algorithm finds solutions of high quality with three drivers from every depot.

Name	T_A	e_n		E			T_B	e_n		E		
		sol	k	sol	k	Gap		sol	k	sol	k	Gap
n100-1	15	1129.24	3	964.68	3	-14.57	40	740.94	1	738.89	1	-0.28
n100-2	15	1238.10	3	1019.96	3	-17.62	40	788.22	1	779.27	1	-1.14
n200-1	25	1574.77	3	1312.97	3	-16.62	60	1140.77	1	1116.88	1	-2.09
n200-2	25	1687.01	3	1288.77	3	-23.61	60	1140.32	1	1119.05	1	-1.87
n300-1	35	1780.61	3	1500.17	3	-15.75	90	1380.37	1	1376.54	1	-0.28
n300-2	35	2046.51	3	1479.32	3	-27.71	90	1335.43	1	1334.85	1	-0.04
n400-1	45	2109.17	3	1695.25	3	-19.62	115	1551.02	1	1556.80	1	0.37
n400-2	45	1973.55	3	1672.92	3	-15.23	115	1530.55	1	1536.15	1	0.37
Average	-	1692.37	-	1366.76	-	-18.84	-	1200.95	-	1194.80	-	-0.62

Table 12: Results for Class III instances with multiple exchange locations

8 Conclusion

In this paper, we studied the Driver and Vehicle Routing problem where two depots are present. The drivers must return to their starting depot and the vehicles must end in the other depot than their starting depot. To make this possible, the drivers must switch their vehicles at designated exchange locations. We extended an existing multi-start heuristic such that it can be applied to multiple exchange locations.

When inspecting favorable exchange locations, our results suggest that the location should not be further from a depot than the distance between the depots. If this requirement is satisfied, there is no precise measure to determine the best exchange location. Therefore, when incorporating the exchange locations as a decision, we confine the potential exchange nodes to locations that satisfy this requirement. Our results prove that exploring potential exchange locations as a multi-armed bandit problem consistently yields improvements for all of our datasets. The improvements are especially reliable when multiple drivers are required to serve all customers.

The computation time of the heuristic took unexpectedly long. This is speculated to be caused by implementation choices in the route improvement part of the heuristic. More investigation on the proposed faster alternatives would be helpful. Additionally, all of the data was provided from a previous paper. Further robustness checks using newly generated data is another suggestion for future research. Using instances with various sizes might provide clearer insights into the time complexity. Lastly, the multi-armed bandit problem uses a simple exploration-exploitation scheme. Using complex strategies such as a Markov Decision Process seems the most prospective suggestion for future work.

References

Ball, M., Bodin, L., & Dial, R. (1983). A matching based heuristic for scheduling mass transit crews and vehicles. *Transportation Science*, *17*(1), 4–31.

Cordeau, J.-F., Stojković, G., Soumis, F., & Desrosiers, J. (2001). Benders decomposition for simultaneous aircraft routing and crew scheduling. *Transportation Science*, *35*(4), 375-388. Retrieved from https://doi.org/10.1287/trsc.35.4.375.10432 doi: 10.1287/trsc.35.4.375.10432

Crevier, B., Cordeau, J.-F., & Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, *176*(2), 756-773. Retrieved from https://www.sciencedirect.com/science/article/pii/S0377221705006983 doi: https://doi.org/10.1016/j.ejor.2005.08.015

Darby-Dowman, K., Jachnik, J. K., Lewis, R. L., & Mitra, G. (1988). Integrated decision support systems for urban transport scheduling: Discussion of implementation and experience. In J. R. Daduna & A. Wren (Eds.), *Computer-aided transit scheduling* (pp. 226–239). Berlin, Heidelberg: Springer Berlin Heidelberg.

Domínguez-Martín, B., Rodríguez-Martín, I., & Salazar-González, J.-J. (2018a). The driver and vehicle routing problem. *Computers & Operations Research*, *92*, 56-64. Retrieved from https://www.sciencedirect.com/science/article/pii/S030505481730309X doi: https://doi.org/10.1016/j.cor.2017.12.010

Domínguez-Martín, B., Rodríguez-Martín, I., & Salazar-González, J.-J. (2018b). A heuristic approach to the driver and vehicle routing problem. In R. Cerulli, A. Raiconi, & S. Voß (Eds.), *Computational logistics* (pp. 295–305). Cham: Springer International Publishing.

Domínguez-Martín, B., Rodríguez-Martín, I., & Salazar-González, J.-J. (2023). An efficient multistart heuristic for the driver and vehicle routing problem. *Computers & Operations Research*, *150*, 106076. Retrieved from https://www.sciencedirect.com/science/article/pii/S0305054822003069 doi: https://doi.org/10.1016/j.cor.2022.106076

Freling, R., Wagelmans, A. P. M., & Paixão, J. M. P. (1999). An overview of models and techniques for integrating vehicle and crew scheduling. In N. H. M. Wilson (Ed.), *Computer-aided transit scheduling* (pp. 441–460). Berlin, Heidelberg: Springer Berlin Heidelberg.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & Service Operations Management*, *11*(4), 630-643. Retrieved from https://doi.org/10.1287/msom.1080.0243 doi: 10.1287/msom.1080.0243

Hollis, B., Forbes, M., & Douglas, B. (2006). Vehicle routing and crew scheduling for metropolitan mail distribution at australia post. *European Journal of Operational Research*, *173*(1), 133-150. Retrieved from https://www.sciencedirect.com/science/article/pii/S0377221705000329 doi: https://doi.org/10.1016/j.ejor.2005.01.005

Huisman, D., Freling, R., & Wagelmans, A. P. M. (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, *39*(4), 491-502. Retrieved from https://doi.org/10.1287/trsc.1040.0104 doi: 10.1287/trsc.1040.0104

Laporte, G. (1984). Optimal solutions to capacitated multidepot vehicle routing problems. *Congressus Nemerantium*, *4*, 283-292. Retrieved from https://cir.nii.ac.jp/crid/1573105975386160000

Mor, A., & Speranza, M. G. (2022). Vehicle routing problems over time: a survey. *Annals of Operations Research*, *314*(1), 255–275.

Renaud, J., Laporte, G., & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, *23*(3), 229-235. Retrieved from https://www.sciencedirect.com/science/article/pii/030505489500026P doi: https://doi.org/10.1016/0305-0548(95)O0026-P

Salazar-González, J.-J. (2014). Approaches to solve the fleet-assignment, aircraft-routing, crew-pairing and crew-rostering problems of a regional carrier. *Omega*, *43*, 71-82. Retrieved from https://www.sciencedirect.com/science/article/pii/S0305048313000674 doi: https://doi.org/10.1016/j.omega.2013.06.006

Spliet, R., & Dekker, R. (2016). The driver assignment vehicle routing problem. *Networks*, *68*(3), 212-223. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/net.21694 doi: https://doi.org/10.1002/net.21694

Srinivas, S. S., & Gajanand, M. S. (2017). Vehicle routing problem and driver behaviour: a review and framework for analysis. *Transport Reviews*, *37*(5), 590–611. Retrieved from https://doi.org/10.1080/01441647.2016.1273276 doi: 10.1080/01441647.2016.1273276

Sutton, R., & Barto, A. (2018). *Reinforcement learning, second edition: An introduction*. MIT Press. Retrieved from https://books.google.be/books?id=uWV0DwAAQBAJ

Vergara, H. A., & Root, S. (2013). Mixed fleet dispatching in truckload relay network design optimization. *Transportation Research Part E: Logistics and Transportation Review*, *54*, 32-49. Retrieved from https://www.sciencedirect.com/science/article/pii/S1366554513000616 doi: https://doi.org/10.1016/j.tre.2013.04.001

A Candidate exchange locations

Name	T_A	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n10-1	6	656.79	2	0.75	543.91	2	0.79	-17.19	462.47	2	0.64	-29.59
n10-2	5	486.40	2	0.56	506.18	2	0.56	4.07	521.52	2	0.61	7.22
n10-3	5	986.58	3	0.88	-			-100.00	546.33	2	0.57	-44.62
n10-4	5	610.55	2	0.50	610.55	2	0.65	0.00	610.55	2	0.65	0.00
n10-5	5	594.32	2	0.51	448.03	2	0.59	-24.61	379.37	2	0.59	-36.17
n15-1	6	455.55	2	1.28	446.85	2	1.29	-1.91	446.85	2	1.29	-1.91
n15-2	6	745.57	2	1.09	598.56	2	1.20	-19.72	664.07	2	1.21	-10.93
n15-3	6	660.07	2	1.12	556.13	2	1.21	-15.75	529.55	2	1.38	-19.77
n15-4	6	1093.02	3	1.88	1344.28	3	1.73	22.99	622.62	2	1.17	-43.04
n15-5	6	788.47	2	1.15	573.19	2	1.23	-27.30	540.50	2	1.23	-31.45
n20-1	7	1274.56	3	3.14	705.37	2	1.96	-44.66	705.37	2	1.96	-44.66
n20-2	7	666.67	2	1.96	682.87	2	1.94	2.43	596.18	2	2.06	-10.57
n20-3	7	846.16	2	1.94	615.65	2	2.08	-27.24	615.65	2	2.08	-27.24
n20-4	7	601.70	2	2.10	626.30	2	2.00	4.09	626.30	2	2.00	4.09
n20-5	7	689.25	2	2.04	567.71	2	2.19	-17.63	529.65	2	2.18	-23.16
n25-1	8	720.21	2	3.08	805.67	2	3.00	11.87	878.05	2	2.96	21.92
n25-2	8	805.47	2	2.87	-			-100.00	743.89	2	3.11	-7.65
n25-3	8	601.05	2	3.37	694.03	2	3.57	15.47	621.04	2	3.60	3.33
n25-4	8	685.64	2	3.35	677.76	2	3.28	-1.15	603.10	2	3.59	-12.04
n25-5	8	724.48	2	3.21	663.78	2	3.39	-8.38	565.27	2	3.46	-21.98
n30-1	9	896.36	2	4.23	680.35	2	4.84	-24.10	680.35	2	4.84	-24.10
n30-2	9	860.85	2	4.50	-			-100.00	697.57	2	4.78	-18.97
n30-3	9	812.05	2	5.37	653.59	2	4.94	-19.51	598.21	2	5.06	-26.33
n30-4	9	720.83	2	4.33	771.13	2	4.09	6.98	892.01	2	4.06	23.75
n30-5	9	750.20	2	4.63	625.39	2	4.92	-16.64	583.70	2	4.98	-22.19
Average	-	749.31	-	-	654.42	-	-	-9.00	610.41	-	-	-15.84

Table 13: Results for Class I instances with tight T values with one exchange location

Name	T_B	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n10-1	10	370.89	1	0.27	371.62	1	0.24	0.20	340.17	1	0.25	-8.28
n10-2	10	293.28	1	0.19	298.16	1	0.29	1.66	292.82	1	0.26	-0.16
n10-3	10	382.70	1	0.23	-				337.15	1	0.25	-11.90
n10-4	10	382.72	1	0.19	382.72	1	0.20	0.00	382.72	1	0.20	0.00
n10-5	10	350.92	1	0.21	290.62	1	0.23	-17.18	270.02	1	0.24	-23.05
n15-1	12	301.87	1	0.60	305.09	1	0.52	1.07	305.09	1	0.52	1.07
n15-2	12	405.00	1	0.48	329.41	1	0.60	-18.66	318.26	1	0.77	-21.42
n15-3	12	386.90	1	0.48	341.84	1	0.66	-11.65	338.67	1	0.56	-12.47
n15-4	12	460.23	1	0.59	518.81	1	0.64	12.73	441.03	1	0.55	-4.17
n15-5	12	468.96	1	0.57	404.47	1	0.57	-13.75	389.40	1	0.63	-16.97
n20-1	14	521.12	1	1.27	465.61	1	1.41	-10.65	465.61	1	1.41	-10.65
n20-2	14	401.81	1	0.99	428.65	1	1.18	6.68	385.82	1	1.18	-3.98
n20-3	14	507.87	1	1.19	462.19	1	1.36	-8.99	462.19	1	1.36	-8.99
n20-4	14	416.76	1	0.91	420.71	1	1.17	0.95	420.71	1	1.17	0.95
n20-5	14	408.71	1	1.19	381.09	1	0.97	-6.76	372.62	1	0.98	-8.83
n25-1	16	483.04	1	1.42	511.09	1	2.04	5.81	541.43	1	2.11	12.09
n25-2	16	486.29	1	1.91	-				462.62	1	2.26	-4.87
n25-3	16	402.49	1	1.74	443.50	1	1.60	10.19	415.52	1	1.82	3.24
n25-4	16	456.06	1	1.56	454.71	1	1.86	-0.30	443.93	1	1.67	-2.66
n25-5	16	453.99	1	1.71	437.37	1	1.92	-3.66	428.74	1	2.23	-5.56
n30-1	18	582.43	1	3.18	539.50	1	3.04	-7.37	539.50	1	3.04	-7.37
n30-2	18	553.43	1	2.50	-				534.53	1	3.41	-3.42
n30-3	18	486.64	1	3.28	436.27	1	3.30	-10.35	434.64	1	3.03	-10.69
n30-4	18	495.30	1	2.70	506.63	1	2.75	2.29	551.75	1	3.08	11.40
n30-5	18	491.64	1	2.42	473.46	1	3.36	-3.70	461.10	1	2.96	-6.21
Average	-	438.04	-	-	418.342	-	-	-3.25	413.44	-	-	-5.72

Table 14: Results for Class I instances with loose T values with one exchange location

Name	T	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n50-1	18	608.29	1	8.45	695.26	2	21.05	14.30	708.70	2.00	21.51	16.51
n50-2	18	591.06	1	6.83	687.62	2	19.63	16.34	592.20	1.00	7.68	0.19
n50-3	18	585.70	1	9.69	707.24	2	19.75	20.75	693.64	2.00	21.14	18.43
n50-4	18	597.10	1	7.49	675.07	2	20.90	13.06	615.53	1.00	7.59	3.09
n50-5	18	616.66	1	9.07	741.87	2	22.59	20.30	750.57	2.00	23.73	21.72
n50-6	18	584.58	1	7.68	706.39	2	18.24	20.84	695.98	2.00	18.88	19.06
n50-7	18	551.73	1	7.41	649.16	2	19.36	17.66	674.88	2.00	17.66	22.32
n50-8	18	590.76	1	7.46	662.10	2	19.83	12.08	604.29	1.00	7.71	2.29
n50-9	18	607.38	1	7.50	795.74	2	22.07	31.01	600.84	1.00	7.48	-1.08
n50-10	18	602.78	1	9.18	762.09	2	20.52	26.43	762.09	2.00	20.52	26.43
n50-11	18	602.58	1	8.26	719.44	2	20.80	19.39	593.02	1.00	9.14	-1.59
n50-12	18	628.36	1	8.15	738.23	2	19.80	17.49	793.61	2.00	21.06	26.30
n50-13	18	785.73	2	18.03	778.48	2	19.62	-0.92	798.79	2.00	19.80	1.66
n50-14	18	618.02	1	7.93	692.16	2	24.24	12.00	723.97	2.00	22.05	17.14
n50-15	18	812.87	2	16.90	733.79	2	23.06	-9.73	738.86	2.00	23.66	-9.10
n50-16	18	614.92	1	8.13	743.21	2	23.38	20.86	732.40	2.00	22.38	19.10
Average	-	624.91	-	-	717.99	-	-	15.74	692.46	-	-	11.40

Table 15: Results for Class II instances with one exchange location

Name	T_A	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n100-1	15	1129.24	3	100.70	1393.97	3	87.56	23.44	1393.97	3	87.57	23.44
n100-2	15	1238.10	3	98.26	1262.44	3	91.16	1.97	1219.55	3	91.99	-1.50
n100-3	15	1159.73	3	92.01	1320.92	3	83.86	13.90	1284.31	3	87.88	10.74
n100-4	15	1206.03	3	95.69	1418.71	3	90.73	17.63	1299.60	3	94.38	7.76
n100-5	15	1277.14	3	94.49	1450.35	3	87.79	13.56	1366.84	3	86.92	7.02
n200-1	25	1574.77	3	536.97	1760.87	3	505.66	11.82	1553.33	3	512.76	-1.36
n200-2	25	1687.01	3	527.93	1745.22	3	494.98	3.45	1745.22	3	494.98	3.45
n200-3	25	1558.62	3	533.83	1808.41	3	484.42	16.03	1785.29	3	472.58	14.54
n200-4	25	1436.11	3	543.59	1812.97	3	470.06	26.24	1645.56	3	488.58	14.58
n200-5	25	1476.02	3	535.52	1807.36	3	486.54	22.45	1498.36	3	510.42	1.51
n300-1	35	1780.61	3	1626.38	2143.58	3	1459.08	20.38	1902.87	3	1504.76	6.87
n300-2	35	2046.51	3	1582.45	2086.63	3	1481.98	1.96	1826.19	3	1572.64	-10.77
n300-3	35	1780.55	3	1681.32	2329.78	3	1407.9	30.85	2211.39	3	1432.98	24.20
n300-4	35	1859.17	3	1616.87	2219.31	3	1439.74	19.37	1832.90	3	1500.91	-1.41
n300-5	35	1601.47	3	1645.45	2102.78	3	1491.45	31.30	1724.51	3	1540.10	7.68
n400-1	45	2109.17	3	3683.42	2375.76	3	3315.56	12.64	2206.67	3	3538.23	4.62
n400-2	45	1973.55	3	3709.12	2313.91	3	3343.26	17.25	1985.88	3	3510.73	0.62
n400-3	45	1942.33	3	3561.48	2434.66	3	3266.69	25.35	2071.19	3	3367.01	6.63
n400-4	45	2393.49	3	3405.18	2429.62	3	3224.33	1.51	2311.45	3	3379.86	-3.43
n400-5	45	2288.83	3	3435.85	2429.52	3	3230.22	6.15	2218.68	3	3246.42	-3.06
Average	-	1675.92	-	-	1932.34	-	-	15.86	1754.19	-	-	5.61

Table 16: Results for Class III instances with tight T values with one exchange location

Name	T_B	e_n			e_C				e_M			
		sol	k	Time	sol	k	Time	Gap	sol	k	Time	Gap
n100-1	40	740.94	1	49.13	869.22	1	59.32	17.31	869.22	1	55.53	17.31
n100-2	40	788.22	1	48.38	788.72	1	57.89	0.06	779.28	1	55.41	-1.13
n100-3	40	767.04	1	47.49	785.21	1	56.23	2.36	774.81	1	57.96	1.01
n100-4	40	762.67	1	45.70	883.33	1	58.54	15.82	763.22	1	57.49	0.07
n100-5	40	786.12	1	54.13	799.88	1	57.38	1.75	790.27	1	61.56	0.52
n200-1	60	1140.77	1	308.41	1193.78	2	296.86	4.64	1130.96	1	304.54	-0.86
n200-2	60	1140.32	1	306.47	1208.67	2	675.19	5.99	1208.67	2	675.19	5.99
n200-3	60	1144.46	1	319.14	1189.36	2	601.19	3.92	1181.74	2	607.92	3.25
n200-4	60	1138.27	1	298.60	1232.96	2	631.06	8.32	1140.19	1	299.84	0.17
n200-5	60	1136.13	1	314.80	1218.76	2	609.58	7.27	1126.56	1	303.86	-0.84
n300-1	90	1380.37	1	980.23	1406.87	2	1964.33	1.92	1391.18	1	965.20	0.78
n300-2	90	1335.43	1	1014.02	1366.26	2	1972.34	2.31	1336.74	1	966.22	0.10
n300-3	90	1328.21	1	968.36	1433.73	2	2263.61	7.94	1371.49	1	951.27	3.26
n300-4	90	1406.39	1	1001.15	1610.04	1	927.09	14.48	1398.07	1	963.36	-0.59
n300-5	90	1330.07	1	1010.96	1707.64	1	942.54	28.39	1341.52	1	957.32	0.86
n400-1	115	1551.02	1	2354.91	1563.50	2	4590.02	0.80	1548.95	1	2305.22	-0.13
n400-2	115	1530.55	1	2282.85	1577.96	2	4445.27	3.10	1546.47	1	2303.86	1.04
n400-3	115	1583.78	1	2318.07	1600.13	2	4323.72	1.04	1575.64	1	2260.13	-0.51
n400-4	115	1562.59	1	2307.30	1642.09	2	4854.67	5.09	1571.94	1	2283.64	0.60
n400-5	115	1601.82	1	2238.69	1589.68	2	4424.49	-0.76	1620.55	1	2166.35	1.17
Average	-	1207.76	-	-	1283.39	-	-	6.59	1223.37	-	-	1.60

Table 17: Results for Class III instances with loose T values with one exchange location

B Deciding on good exchange locations

Name	T_A	e_n		E			T_B	e_n		E		
		sol	k	sol	k	Gap		sol	k	sol	k	Gap
n10-1	6	656.79	2	436.79	2	-33.50	10	370.89	1	340.17	1	-8.28
n10-2	5	486.40	2	463.89	2	-4.63	10	293.28	1	281.56	1	-4.00
n10-3	5	986.58	3	546.33	2	-44.62	10	382.70	1	337.15	1	-11.90
n10-4	5	610.55	2	590.93	2	-3.21	10	382.72	1	374.85	1	-2.06
n10-5	5	594.32	2	379.37	2	-36.17	10	350.92	1	270.02	1	-23.05
n15-1	6	455.55	2	444.57	2	-2.41	12	301.87	1	305.09	1	1.07
n15-2	6	745.57	2	580.66	2	-22.12	12	405.00	1	318.26	1	-21.42
n15-3	6	660.07	2	512.11	2	-22.42	12	386.90	1	338.67	1	-12.47
n15-4	6	1093.02	3	575.95	2	-47.31	12	460.23	1	441.03	1	-4.17
n15-5	6	788.47	2	515.39	2	-34.63	12	468.96	1	389.40	1	-16.97
n20-1	7	1274.56	3	705.37	2	-44.66	14	521.12	1	465.61	1	-10.65
n20-2	7	666.67	2	583.09	2	-12.54	14	401.81	1	413.72	1	2.96
n20-3	7	846.16	2	586.99	2	-30.63	14	507.87	1	457.29	1	-9.96
n20-4	7	601.70	2	540.84	2	-10.11	14	416.76	1	416.76	1	0.00
n20-5	7	689.25	2	500.70	2	-27.36	14	408.71	1	366.28	1	-10.38
n25-1	8	720.21	2	688.56	2	-4.39	16	483.04	1	482.20	1	-0.17
n25-2	8	805.47	2	732.46	2	-9.06	16	486.29	1	462.62	1	-4.87
n25-3	8	601.05	2	530.72	2	-11.70	16	402.49	1	402.49	1	0.00
n25-4	8	685.64	2	547.69	2	-20.12	16	456.06	1	432.36	1	-5.20
n25-5	8	724.48	2	563.75	2	-22.19	16	453.99	1	435.29	1	-4.12
n30-1	9	896.36	2	634.25	2	-29.24	18	582.43	1	531.56	1	-8.73
n30-2	9	860.85	2	696.70	2	-19.07	18	553.43	1	534.53	1	-3.42
n30-3	9	812.05	2	611.97	2	-24.64	18	486.64	1	435.51	1	-10.51
n30-4	9	720.83	2	644.82	2	-10.54	18	495.30	1	486.24	1	-1.83
n30-5	9	750.20	2	546.77	2	-27.12	18	491.64	1	451.30	1	-8.21
Average	-	749.312	-	566.43	-	-22.18	-	438.042	-	406.80	-	-7.13

Table 18: Results for Class I instances with multiple exchange locations

Name	T	e_n		E		
		sol	k	sol	k	Gap
n50-1	18	608.29	1	591.62	1	-2.74
n50-2	18	591.06	1	587.65	1	-0.58
n50-3	18	585.70	1	581.59	1	-0.70
n50-4	18	597.10	1	594.31	1	-0.47
n50-5	18	616.66	1	613.62	1	-0.49
n50-6	18	584.58	1	584.58	1	0.00
n50-7	18	551.73	1	534.69	1	-3.09
n50-8	18	590.76	1	584.81	1	-1.01
n50-9	18	607.38	1	585.34	1	-3.63
n50-10	18	602.78	1	609.47	1	1.11
n50-11	18	602.58	1	586.55	1	-2.66
n50-12	18	628.36	1	605.95	1	-3.57
n50-13	18	785.73	2	602.63	1	-23.30
n50-14	18	618.02	1	605.21	1	-2.07
n50-15	18	812.87	2	705.10	2	-13.26
n50-16	18	614.92	1	612.38	1	-0.41
Average	-	624.91	-	599.09	-	-3.55

Table 19: Results for Class II instances with multiple exchange locations

Name	T_A	e_n		E			T_B	e_n		E		
		sol	k	sol	k	Gap		sol	k	sol	k	Gap
n100-1	15	1129.24	3	964.68	3	-14.57	40	740.94	1	738.89	1	-0.28
n100-2	15	1238.10	3	1019.96	3	-17.62	40	788.22	1	779.27	1	-1.14
n100-3	15	1159.73	3	914.72	3	-21.13	40	767.04	1	760.54	1	-0.85
n100-4	15	1206.03	3	1088.24	3	-9.77	40	762.67	1	763.85	1	0.15
n100-5	15	1277.14	3	1242.01	3	-2.75	40	786.12	1	776.94	1	-1.17
n200-1	25	1574.77	3	1312.97	3	-16.62	60	1140.77	1	1116.88	1	-2.09
n200-2	25	1687.01	3	1288.77	3	-23.61	60	1140.32	1	1119.05	1	-1.87
n200-3	25	1558.62	3	1297.58	3	-16.75	60	1144.46	1	1155.68	1	0.98
n200-4	25	1436.11	3	1335.54	3	-7.00	60	1138.27	1	1157.06	1	1.65
n200-5	25	1476.02	3	1306.64	3	-11.48	60	1136.13	1	1141.17	1	0.44
n300-1	35	1780.61	3	1500.17	3	-15.75	90	1380.37	1	1376.54	1	-0.28
n300-2	35	2046.51	3	1479.32	3	-27.71	90	1335.43	1	1334.85	1	-0.04
n300-3	35	1780.55	3	1945.44	3	9.26	90	1328.21	1	1334.30	1	0.46
n300-4	35	1859.17	3	1559.22	3	-16.13	90	1406.39	1	1395.77	1	-0.76
n300-5	35	1601.47	3	1460.25	3	-8.82	90	1330.07	1	1338.27	1	0.62
n400-1	45	2109.17	3	1695.25	3	-19.62	115	1551.02	1	1556.80	1	0.37
n400-2	45	1973.55	3	1672.92	3	-15.23	115	1530.55	1	1536.15	1	0.37
n400-3	45	1942.33	3	1705.38	3	-12.20	115	1583.78	1	1578.11	1	-0.36
n400-4	45	2393.49	3	1801.46	3	-24.74	115	1562.59	1	1581.72	1	1.22
n400-5	45	2288.83	3	1707.32	3	-25.41	115	1601.82	1	1579.33	1	-1.40
Average	-	1675.92	-	1414.89	-	-14.88	-	1207.76	-	1206.06	-	-0.20

Table 20: Results for Class III instances with multiple exchange locations

Class I	T_A		T_B		Class II	T		Class III	T_A		T_B	
Name	k	Time	k	Time	Name	k	Time	Name	k	Time	k	Time
n10-1	2	0.67	1	0.27	n50-1	1	8.06	n100-1	3	95.29	1	55.59
n10-2	2	0.50	1	0.19	n50-2	1	7.19	n100-2	3	95.55	1	55.47
n10-3	2	0.50	1	0.23	n50-3	1	8.78	n100-3	3	92.01	1	50.51
n10-4	2	0.45	1	0.19	n50-4	1	7.59	n100-4	3	94.08	1	55.55
n10-5	2	0.49	1	0.21	n50-5	1	9.55	n100-5	3	89.47	1	57.63
n15-1	2	1.12	1	0.60	n50-6	1	7.73	n200-1	3	527.08	1	302.46
n15-2	2	1.05	1	0.48	n50-7	1	7.15	n200-2	3	533.07	1	308.07
n15-3	2	1.08	1	0.48	n50-8	1	7.29	n200-3	3	510.63	1	289.64
n15-4	2	1.02	1	0.59	n50-9	1	7.34	n200-4	3	512.46	1	296.72
n15-5	2	1.14	1	0.57	n50-10	1	7.96	n200-5	3	516.03	1	297.95
n20-1	2	1.82	1	1.27	n50-11	1	8.43	n300-1	3	1599.76	1	979.05
n20-2	2	1.80	1	0.99	n50-12	1	7.84	n300-2	3	1606.19	1	972.57
n20-3	2	2.02	1	1.19	n50-13	1	7.86	n300-3	3	1486.08	1	938.93
n20-4	2	1.95	1	0.91	n50-14	1	8.10	n300-4	3	1547.98	1	953.13
n20-5	2	1.98	1	1.19	n50-15	2	15.77	n300-5	3	1566.09	1	953.16
n25-1	2	2.97	1	1.42	n50-16	1	8.36	n400-1	3	3576.43	1	2283.63
n25-2	2	2.95	1	1.91				n400-2	3	3592.27	1	2329.32
n25-3	2	3.12	1	1.74				n400-3	3	3493.03	1	2263.58
n25-4	2	3.15	1	1.56				n400-4	3	3441.49	1	2264.12
n25-5	2	3.22	1	1.71				n400-5	3	3425.54	1	2149.53
n30-1	2	4.51	1	3.18								
n30-2	2	4.41	1	2.50								
n30-3	2	4.71	1	3.28								
n30-4	2	4.23	1	2.70								
n30-5	2	4.77	1	2.42								

Table 21: Running times in seconds of all instances with multiple exchange locations

Class I				Class II				Class III			
Name	$\|C\|$	k=2	k=3	Name	$\|C\|$	k=2	k=3	Name	$\|C\|$	k=2	k=3
n10-1	2	3	4	n50-1	22	253	2024	n100-1	86	3741	109736
n10-2	6	21	56	n50-2	38	741	9880	n100-2	62	1953	41664
n10-3	0	1	1	n50-3	38	741	9880	n100-3	56	1596	30856
n10-4	3	6	10	n50-4	29	435	4495	n100-4	68	2346	54740
n10-5	2	3	4	n50-5	32	528	5984	n100-5	40	820	11480
n15-1	2	3	4	n50-6	46	1081	17296	n200-1	187	17578	1107414
n15-2	2	3	4	n50-7	43	946	14190	n200-2	115	6670	260130
n15-3	7	28	84	n50-8	34	595	7140	n200-3	174	15225	893200
n15-4	8	36	120	n50-9	21	231	1771	n200-4	166	13861	776216
n15-5	3	6	10	n50-10	47	1128	18424	n200-5	184	17020	1055240
n20-1	1	1	1	n50-11	39	780	10660	n300-1	216	23436	1703016
n20-2	6	21	56	n50-12	45	1035	16215	n300-2	251	31626	2667126
n20-3	4	10	20	n50-13	47	1128	18424	n300-3	138	9591	447580
n20-4	12	78	364	n50-14	37	703	9139	n300-4	166	13861	776216
n20-5	15	120	680	n50-15	39	780	10660	n300-5	247	30628	2542124
n25-1	21	231	1771	n50-16	32	528	5984	n400-1	339	57630	6550610
n25-2	0	1	1					n400-2	306	46971	4822356
n25-3	16	136	816					n400-3	352	62128	7331104
n25-4	17	153	969					n400-4	240	28920	2332880
n25-5	3	6	10					n400-5	243	29646	2421090
n30-1	8	36	120								
n30-2	0	1	1								
n30-3	3	6	10								
n30-4	17	153	969								
n30-5	7	28	84								

Table 22: Number of candidates for every instance

C Programming code

The `src` directory contains all the code. The `Main` class is the control panel where all the configurations are set and runs are performed. The other classes are explained in detail using Javadoc comments.

There are three configuration files, one for each instance class. The name of the file is `configClass`i`.csv`, with i being the class. Their purpose is to easily iterate over the datasets. The maximum route durations are also collected.

The `diagnostics` directory contains the results of the computation times and the number of iterations for each part of the heuristic. The computation times are acquired with the `extractRunningTimes()` method in the `Main` class. The code to obtain the number of iterations has unfortunately been lost as counting the iterations greatly slows the code down. To obtain the numbers, a modified version of the `performReplication()` method in the `Main` class is used. The results are reported in Table 1 and Figure 1.

The `outputReplicate` directory contains the results of the replication part. There are

summarizing `.csv` files containing the most relevant information of every instance. The summary files are separated by class and by route time limit. There are additionally more detailed `.dat` files for every instance that show the routes that constitute the solution. These results are obtained using the `performReplication()` method in the `Main` class. The results are scattered among all tables in Sections 7.1 and 7.2 under the column e_n.

The `outputExchange` directory contains the results for when one alternative exchange location is considered. Only the summary `.csv` files are stored. There would be too many `.dat` files to save because we need four different variations of one instance. There are two route time limits and two exchange locations to consider. These results are obtained using the `iterateExchange()` method in the `Main` class. The results are reported in the Tables 5, 6, 7, 8, and 9. They also appear in Appendix A.

The outputMulti directory contains the results for when the exchange locations are selected using a multi-armed bandit approach. This directory contains the same type of files as `outputReplicate`. These results are obtained using the `iterateMulti()` method in the `Main` class. The results are reported in Tables 10, 11, and 12. They also appear in Appendix B.

Lastly, the number of *candidates* in every dataset, found in Table 22, is obtained using the `testExchangeCriterion()` method in the `Main` class.