Erasmus University Rotterdam
Erasmus School of Economics
Bachelor Thesis Econometrics & Operational Research

A Heuristic Approach for Identifying Locally Optimal Exchange and Depot Locations in the Driver and Vehicle Routing Problem

Femke Leentjens (604874)

Supervisor:	Danny Zhu
Second assessor:	D. Huisman
Date final version:	1st July 2024

Abstract

Previous research has introduced the Driver and Vehicle Routing Problem (DVRP). In routing problems it is generally assumed that drivers use one vehicle for their entire route. The DVRP is different in the fact that drivers return to their home depot at the end of the day, while the vehicles should end at another depot. Solving this problem to optimality can only be done for small instances and therefore previous research has implemented a multistart heuristic that is able to find optimal solutions efficiently. This paper aims to find a heuristic that is not only able to find a quick solution for the Driver and Vehicle Routing Problem (DVRP), but can also improve the exchange and depot locations. These improved exchange and depot locations are customer locations that give lower total costs when they are used as exchange and depot locations compared to the original exchange and depot locations. First, the multistart heuristic is implemented. Then, an extra step in the heuristic is introduced that can find better exchange and depot locations. The results show that the objective values can be significantly decreased while barely adding to the computation times for most of the instances. Furthermore, two different starting exchange locations were used for the adjusted heuristic. However, while these starting exchange locations often give different solutions, not one of them consistently outperforms the other when looking at the computation times or the objective values.

1 Introduction

The scheduling of drivers and vehicles has been a longstanding issue that has multiple different applications. First of all, there is the well-known case of packages that need to be delivered to customers. Then, there are also other applications such as transporting big loads using trucks or the scheduling of planes and crews for air transport. All these cases have slightly different characteristics and therefore need different formulations and heuristics. The problem that we will discuss in this paper is the Driver and Vehicle Routing Problem (DVRP). This problem differs from the classical Vehicle Routing Problem (VRP). In the VRP, there is only one depot and it is assumed that the vehicles' routes start and end at the single depot. Furthermore, each vehicle is driven by only one driver for the entire route.

The DVRP was introduced by Domínguez-Martín, Rodríguez-Martín and Salazar-González (2018a) and is defined as follows. There are two depots and a set of customers. At each of the depots, a given number of drivers and vehicles is based. The drivers must start and end their routes at the same depot, while vehicles must start their routes at one depot and end at the other depot. A vehicle must always be driven by a driver and a driver needs a vehicle to get from one location to another. This can be as driver or as passenger. When there are multiple drivers in one vehicle, any one of them can drive the vehicle. The routes of the driver may not exceed a given time duration. The duration of a drivers' route is the time between the departure from the base depot and the arrival at the base depot and this includes both the time as a driver and as a passengers. Since the drivers have to return to the same depot they left from, but vehicles have to go to the other depot, there has to be a location where drivers can switch vehicles. This is the exchange location, and these are the only customer locations that can be visited more than once. All the other customer locations must be visited exactly once. The objective of the DVRP is to find a feasible set of drivers' and vehicles' routes such that the total cost is minimized.

The DVRP is based on different problems considered in real life. It takes its main inspiration from the air transportation in the Canary Islands (Salazar-González, 2014). The Canary Islands have two main airports, Tenerife North and Las Palmas, and multiple smaller airports. There are flights in between all airports, but the crews of the aircrafts must return to their base airport in order to avoid costs of hotels. The aircrafts, however, have to end their route at another airport than they started from because of maintenance operations. These maintenance operations can only be performed at Las Palmes and take place every other day.

The DVRP can also be used in long-distance ground transportation. Traditionally, drivers would transport their cargo from origin to destination. However, because of this, they were often away from home for quite a long time and this causes a high rate of driver turnover as mentioned by Üster and Kewcharoenwong (2011) and Vergara and Root (2013). Therefore, they propose a dispatching method with relay points. These relay points are locations where drivers can switch vehicles. This allows them to get home sooner, while their cargo still reaches the destination.

This paper builds upon the paper of Domínguez-Martín, Rodríguez-Martín and Salazar-González (2023) who proposed a heuristic for the DVRP. This heuristic considers the case where there is one exchange location and is able to solve the DVRP for instances with up to 1000 locations (998 customers and 2 depots). In the first phase of the heuristic, the algorithm tries to find drivers' routes for a given set of drivers leaving from each depot. The second phase of the algorithm then creates vehicles' routes that are compatible with the best found drivers' routes.

A major drawback of the paper of Domínguez-Martín et al. (2023) is that they generate all locations randomly. This also includes the depots and the exchange location. As a result, the exchange location and the depots might not be located in the optimal locations. This can results in costs that are higher than in the optimal situation. Therefore, it would be interesting to research if another exchange location and depot locations can result in better objective values. In this paper, an extra step in the heuristic is proposed that is able to find a better exchange location and depot locations while barely increasing the calculation times. This can help organizations with locating better locations for the exchange and depot locations and thus reducing their total costs. The new exchange and depot locations are chosen from the set of customer locations. Results show that for approximately 93% of the instances, the solution can be improved and that the total costs can be decreased up to 56.53%. Furthermore, for instances with up to problem size 400, better solutions can be found in a couple of seconds. For larger problem sizes, finding better exchange and depot locations can result in significantly higher computation times.

The remainder of this paper is organized as follows. Section 2 discusses the related literature. Then, the problem description is given in Section 3, while the methodology is described in Section 4. The numerical results are given in Section 5. Lastly, the conclusion and ideas for further research are given in Section 6.

2 Literature Review

The DVRP is a routing problem with more than one depot. It was first introduced by Domínguez-Martín et al. (2018a). In this paper, the authors give a mathematical formulation of the problem. Furthermore, they introduce a branch-and-cut algorithm that is able to solve the problem to optimality. However, this is only possible for small instances up to 30 locations (28 passengers and 2 depots). Domínguez-Martín, Rodríguez-Martín and Salazar-González (2018b) propose a heuristic to solve the problem in the case that there is one exchange location. In this particular case we can assume that drivers and vehicles arrive at the exchange location simultaneously since drivers and vehicles can leave and enter a depot when it is convenient for them. This heuristic consists of two phases. In the first phase, the problem of planning the routes of the drivers is modeled as an Integer Linear Programming problem. This is then solved using a branch-and-cut algorithm. Afterwards, in phase two the routes for the vehicles are made based on the routes that were found for the drivers. This heuristic is able to find solutions for instances with a maximum of 50 locations (48 customers and 2 depots).

Another heuristic for the DVRP is introduced by Domínguez-Martín et al. (2023). In this paper, a heuristic algorithm is proposed that is more efficient than the heuristic of Domínguez-Martín et al. (2018b). Similar to that heuristic, this heuristic algorithm first finds feasible routes for the drivers in phase one and then finds the routes for the vehicles and considers the case where there is only one exchange location. However this is no longer done using a Integer Linear Programming and a branch-and-cut algorithm. Domínguez-Martín et al. (2023) find feasible routes for the drivers by building the routes of the drivers iteratively using a random procedure in an inner multistart loop. These routes are improved using local search and the best feasible set of routes is kept. Phase two then finds the vehicle routes based on the best found solution. This heuristic is able to find solutions for instances with up to 1000 locations (998 customers and 2 depots).

Because the DVRP considers two depots, it is closely related to the Multi-Depot Vehicle Routing Problem (MDVRP). Since in the DVRP, the vehicles have to end at another depot, it is especially similar to the MDVRP with inter-depot routes that was first introduced by Crevier, Cordeau and Laporte (2007). In this case, vehicles are allowed to start and end at the same depot, but also to start and end at different depots. The MDVRP is a very complex problem and therefore there have been multiple heuristics proposed. There have for example been multiple papers that discuss genetic algorithms to solve the MDVRP. Ho, Ho, Ji and Lau (2008) introduce two genetic algorithms. The first algorithm generates the initial solutions randomly, while the second algorithm incorporates the Clarke and Wright saving method and the nearest neighbour heuristic for the initialization procedure. In their paper, they prove that the second algorithm is superior to the first one. The paper of Surekha and Sumathi (2011) also considers a genetic algorithm. In their algorithm, they first group customers based on the distance to their nearest depot. Then, the customers are routed using the Clarke and Wright saving method. An elaborate research of genetic algorithms for the MDVRP has been done by Karakatič and Podgorelec (2015).

Finding the optimal exchange and depot locations in the MDVRP has never been investigated before. Since it also not a part of other vehicle routing problems, it has a closer resemblance

with the (uncapacitated) facility location problem. The facility location problem is the problem of placing a facility in a location such that the total cost is minimized. The total cost consists of the costs of opening the facilities and serving all the customers from the open facilities. A hybrid multistart heuristic for this problem is introduced by Resende and Werneck (2006). This heuristic is able to find solutions that are either optimal or very close to the optimal value. Avella, Boccia, Sforza and Vasil'ev (2009) also propose a heuristic to solve the facility location problem. Their heuristic is based on Lagrangean relaxation which is used to find a subset of variables that form the core problem. Then a branch-and-cut algorithm is used to solve this core problem. This heuristic is also able to find optimal solutions or solutions that are very close to the optimal value.

3 Problem Description

The DVRP can be defined on a complete directed graph $G = (V, A)$ where $V = \{0, ..., n+1\}$ is the set of locations. This set V is defined as $D \cup C$, where $D = \{0, n+1\}$ is the set of depots and $C = \{1, ..., n\}$ is the set of customers. The exchange location is denoted by e and is customer n. This is the only customer location where vehicles can be exchanged and it must be visited by all the drivers and vehicles. All the other customer locations must be visited exactly once. The set A is the set of arcs and is defined as $A = \{(i, j) : i, j \in V, i \neq j\}$. For any subset of locations $S \subseteq V$, $\delta^+(S) = \{(i, j) \in A : i \in S, j \notin S\}$ and $\delta^-(S) = \{(i, j) \in A : i \notin S, j \in S\}$. Furthermore, K_d is the set of available drivers at depot d and L_d is the set of available vehicles at depot d. The cost of an arc $(i, j) \in A$ is denoted by c_{ij} and the time needed to traverse this arc is given by t_{ij}. The total time of the route of a driver cannot exceed a given time limit T. There is no maximum time for the route of a vehicle. All vehicles start at one depot and end at the other depot, while drivers must start and end at the same depot. Furthermore, all vehicles must be driven by a driver, and a driver needs a vehicle to get from one place to another. The objective of the DVRP is to find feasible routes for both the drivers and vehicles such that all customers are visited and the total cost is minimized. A complete mathematical formulation of the DVRP can be found in the paper of Domínguez-Martín et al. (2018a).

4 Methodology

In this section, the methodology will be discussed. First of all, Section 4.1 will discuss the Multistart Heuristic Algorithm as proposed by Domínguez-Martín et al. (2023). Then, an extra step in the heuristic is introduced in Section 4.2 that is able to find better exchange and depot locations.

4.1 Multistart Heuristic Algorithm

The heuristic as introduced by Domínguez-Martín et al. (2023) can be found in Algorithm 1. The DVRP with one exchange location can be solved using this heuristic. The heuristic algorithm consists of two phases. In the first phase, the drivers' routes are generated using a multistart procedure. In each iteration, routes for the drivers are generated using an iterative procedure.

These routes are then improved using local search. In the second phase, the routes for the vehicles are generated based on the best found drivers' routes. The general algorithm works as follows. The algorithm tries to find a solution within a given time limit. The number of drivers k leaving from each depot is initialized to 1. The algorithm then tries to find a feasible solution for the drivers' routes and tries to improve this using local search. The best solution that is also feasible is kept. If the algorithm is not able to find a feasible solution, the number of drivers k at each depot and the number of multistart iterations are increased by 1. Then it restarts the multistart loop. This continues until a set of feasible drivers' routes has been found, until the time limit has been passed or until the number of drivers leaving from each depot exceeds the maximum number of drivers allowed. When the best feasible drivers' routes have been constructed, the routes of the vehicles are created in such a way that they are compatible with the drivers' routes. Important to note is that in this heuristic, we assume that the number of drivers leaving from the depots is the same and that there is a unique exchange location. As a result the total cost of the problem is equal to the total distance driven by the drivers. Finding the routes for the vehicles is therefore also straightforward after the best feasible drivers' routes have been found.

Algorithm 1 Multistart heuristic for the DVRP

Input: instance data and parameters $timeLim$ and $maxIter$
Output: drivers' routes S^*, vehicles' routes R^*, and solution value f^*
$f^* \leftarrow \infty, S^* \leftarrow \emptyset, nDrivers \leftarrow 1$
while $time \leq timeLim$ & not feasible solution S^* found & $nDrivers \leq maxDrivers$ **do**
  $nIter \leftarrow 1$
  while $time \leq timeLim$ & $nIter \leq maxIter$ **do**
    {multistart loop}
    $S \leftarrow$ ConstructGreedySol($nDrivers$)
    $S \leftarrow$ LocalSearch(S)
    if S is feasible & $f(S) < f^*$ **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
if $S^* \neq \emptyset$ **then**
  $R^* \leftarrow$ ConstructVehicleRoutesFromDriverRoutes(S^*)
end if
return S^*, R^*, and f^*

4.1.1 Construction of the Drivers' Routes

In this section the algorithm that builds the drivers' routes is explained. These routes are made for a fixed number of drivers k from each depot, where $k \leq |K_d|$ for all $d \in D$. The sequence of locations (nodes) that define the route of driver l whom departs from depot d is denoted by S_d^l where $l \in \{1, ..., k\}$ and $d \in D$. S is denoted as the set of all drivers' routes in a DRVP solution, thus $S = \cup_{d \in D} \cup_{l-1}^{k} S_d^l$. All routes in S are initialized as $S_d^l = \{d, e, d\}$, since the routes of a driver start and end at the same depot and they have to visit the exchange location at some point in the route so they can switch vehicles. Then the routes are expanded in an iterative way until all customers are included in S. In each iteration, a random customer $i \in V$ that is not yet included in S is inserted into one of the routes in S. This is done by using a cheapest insertion strategy. In this strategy, we look for two consecutive locations (nodes) u, v in a driver route in S. If $c_{ui} + c_{iv} - c_{uv}$ is minimal and the duration bound T is not exceeded, customer i is inserted in between customers u and v. Otherwise, customer i is inserted in the shortest route. This is again done by using the cheapest insertion criterion.

4.1.2 Improvement of the Drivers' Routes

After creating the drivers' routes, the drivers' routes are improved by first using inter-route customer relocation local search en then using a 2-opt local search.

The inter-route customer relocation local search works as follows. First, a random customer in the solution S is chosen and removed from its current driver route. This customer is then inserted in a different driver route using the cheapest insertion criterion. This means that all other routes in S are examined. The route where the customer can be inserted while producing the minimum cost increment is the route where the customer is inserted. This is applied iteratively as long as the solution S improves. This thus ends in a local minimum.

Then the 2-opt local search method is applied. This method is applied to every driver route in S and works as follows. In each route, two arcs are deleted and two other arcs are added in order to reconnect it. This is repeated as long as the total cost decreases.

4.1.3 Stopping Criterion and Checking Feasibility

The most important part of the heuristic is the loop that generates the drivers' routes. These routes are created for a given number of drivers that leave from each depot and this number is initialized to 1. This loop keeps on repeating until a feasible set of routes for the drivers are found, the time limit is reached, or the maximum number of drivers that is allowed at each depot is reached. In each iteration of the main loop, an inner multistart loop is executed which produces the drivers' routes, improves these drivers' routes and checks the feasibility of these drivers' routes. A set S of drivers' routes is feasible if the duration of every route in S does not exceed the duration limit T. The best set S that is also feasible is used to build the routes of the vehicles and returned. If at the end of the multistart loop, no feasible set S has been found, the number of drivers leaving from each depot is increased by one and the main loop is performed again.

4.1.4 Generating Vehicles' Routes

After the best set S of feasible drivers' routes has been found, the routes for the vehicles are created. This is done in the same was as described in Domínguez-Martín et al. (2018b). If a driver $k \in K_d$ traverses arc $(d, i) \in \delta^+(d)$, then a vehicle departs from depot d using that arc. This vehicle then follows the route of the driver until it reaches the exchange location. From the exchange location, the vehicle follows the route of a driver $k' \in K_{d'}$ to the other depot d'. When all arcs in $\delta^+(d)$ have been processed, it needs to be checked if the number of vehicles that depart from depot d is equal to the number of different arcs that are used by drivers to enter d'. If this is equal to each other, then this means that no more vehicles need to leave from depot d. Otherwise, it means that there is a driver $k' \in K_{d'}$ going to depot d' from the exchange location without a vehicle. In order to correct this, an extra vehicle should be taken to the exchange location from d following the route of an existing driver that is currently not leading any other vehicle. From the exchange location on, it follows the route of driver k' to depot d'. After doing the same for the arcs $\delta^+(d')$, we have a set of feasible vehicles' routes that is compatible with the best set of drivers' routes S.

4.2 Exchange Location and Depot Heuristic

In this section, we will introduce extra steps in the heuristic that are able to find locations for the exchange location and the depots that result in a better objective value. First, the algorithm as described in Section 4.1 is used until the point is reached where the vehicles' routes are constructed. Before the routes of the vehicles are constructed, we will first try to find a better exchange location. If a better exchange location is found, the exchange location, the drivers' routes and the objective value are changed accordingly. The method that will be used for improving the exchange location is described in Section 4.2.1. Afterwards, a better location for d_1 and d_2 are found using the method described in Section 4.2.2. This is first done for d_1 and then for d_2, where d_1 is the first depot that was generated (location 0) and d_2 is the second depot that was generated (location $n + 1$). The exchange location and depot locations are adjusted in this order for as long as at least one of them is adjusted. When there is no longer an improvement in the objective value, the loop stops and the vehicles' routes are constructed such that they are compatible with the best drivers' routes found. The adjusted heuristic can be found in Algorithm 2.

Algorithm 2 Adjusted Multistart heuristic for the DVRP with exchange location and depots

Input: instance data and parameters $timeLim$, $maxIter$ and p

Output: exchange location e^*, depot 1 d_1^*, depot 2 d_2^*, drivers' routes S^*, vehicles' routes R^*, and solution value f^*

$f^* \leftarrow \infty, S^* \leftarrow \emptyset, nDrivers \leftarrow 1$, $e^* \leftarrow$ starting exchange location, $d_1^* \leftarrow d_1$, $d_2^* \leftarrow d_2$

while $time \leq timeLim$ & not feasible solution S^* found & $nDrivers \leq maxDrivers$ **do**

  $nIter \leftarrow 1$

  while $time \leq timeLim$ & $nIter \leq maxIter$ **do**

    {multistart loop}

    $S \leftarrow$ ConstructGreedySol($nDrivers$)

    $S \leftarrow$ LocalSearch(S)

    if S is feasible & $f(S) < f^*$ **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

if $S^* \neq \emptyset$ **then**

  while Solution improves **do**

    $f^*, S^*, e^* \leftarrow$ FindBetterExchangeLocation(S^*)

    $f^*, S^*, d_1^* \leftarrow$ FindBetterDepotLocation(S^*, d_1^*)

    $f^*, S^*, d_2^* \leftarrow$ FindBetterDepotLocation(S^*, d_2^*)

  end while

  $R^* \leftarrow$ ConstructVehicleRoutesFromDriverRoutes(S^*)

end if

return $S^*, R^*, f^*, e^*, d_1^*$ and d_2^*

4.2.1 Improvement of the Exchange Location

In order to improve the exchange location, a random customer location that is at most distance p from the current exchange location is assigned to be the new exchange location. Then, all the constructed drivers' routes are adjusted. The route that already contains the new exchange location stays the same. In all of the other routes the old exchange location is removed and the new exchange location is inserted using the cheapest insertion criterion. Afterwards, the local search method as described in Section 4.1.2 is used again to improve the driver's routes. If this results in a better objective value than for the drivers' routes with the old exchange location and the new routes are feasible, the new exchange location is assigned to be the current exchange location. Then we will look again for a new exchange location that is at most distance p from this newly assigned exchange location and has not been considered before as exchange location.

Local search will then be used again to improve these routes and the objective value is compared to the old objective value. If there are no more available customer locations, the loop stops.

4.2.2 Improvement of the Depot Location

Improving the depot locations is done in a similar way as improving the exchange location. First, a random customer location that is at most distance p from the current depot is assigned to be the new depot. Then, all the drivers' routes are adjusted. The new depot location is removed from its current route and replaces the old depot at the beginning and end of its routes. The old depot is then inserted in a route using the cheapest insertion criterion. The new routes are then improved using the local search method.

If this results in a better objective value than for the drivers' routes with the old depot location and the new routes are feasible, the new depot location is assigned to be the current depot location. Then we will look again for a new depot location that is at most distance p from this newly assigned depot location and has not been considered before as depot location. Local search will then be used again to improve these routes and the objective value is compared to the old objective value. The loop stops if there are no more available customer locations.

5 Numerical Results

The algorithms described in Section 4 were programmed in Java and executed on a laptop with an AMD Ryzen 5 5500U, 2.10 GHz, 16 GB RAM and Windows 11 Home. Three different sets of instances are used for the computational experiments. These are the same sets that were also used in the paper of Domínguez-Martín et al. (2023). For all of these instances, the costs c_{ij} represent the Euclidean distance between location i and location j. The time needed to travel from location i to location j is defined as $t_{ij} = \frac{c_{ij}}{60} + 0.5$.

The first set (*Class I*) consists of randomly generated instances with coordinates of the locations in the square $[0, 100] \times [0, 100]$. The size of these instances are $n+2 \in \{10, 15, 20, 25, 30\}$. The first and last generated locations are d_1 and d_2 respectively. Locations 1 to n are customers locations and location n is also the exchange location. There are 25 instances in total, 5 for every problem size. Furthermore, the parameter T takes four different values denoted by T_A, T_B, T_C and T_D, where T_A is generally the tightest bound that allows a feasible solution and T_D is the largest bound.

The second set (*Class II*) consists of 16 instances with each 50 locations. The exchange location is randomly generated in the $[40, 60] \times [0, 100]$, while d_1 and d_2 are randomly generated in $[0, 20] \times [0, 100]$ and $[80, 100] \times [0, 100]$ respectively. All the other customer locations are randomly generated in $[0, 100] \times [0, 100]$. The time bound of the drivers' routes is set to $T = 18$ for all of these instances.

The last set of instances (*Class III*) consists of large instances with
$n + 2 \in \{100, 200, 300, 400, 500, 600, 800, 1000\}$. Five instances were generated for each of these problem sizes. All of the locations are randomly generated in the same way as *Class II*. Just like for *Class I*, the drivers' time bound takes four different increasing values which are again denoted by T_A, T_B, T_C and T_D.

The heuristic algorithm was run with a time limit of 600 seconds on every instance (parameter *timeLim* in Algorithm 1). This is the same time limit as was used in Domínguez-Martín et al. (2023). However, because the algorithm was not able to find a feasible solution for some of the *Class III* instances within this time limit, we have continued running the algorithm for these instances until a feasible solution was found. Furthermore, we have set the parameter $maxIter$ equal to 100,000 which is the same as in the paper of Domínguez-Martín et al. (2023).

Section 5.1 will give the results for the multistart heuristic as described in Section 4.1. The results for the adjusted heuristic algorithm that was described in Section 4.2 can be found in Sections 5.2. The data that are displayed in the columns are:

- *Name*: Instance name.
- $n + 2$: Instance size.
- T: Drivers' routes time limit.
- Sol: Objective value.
- Drivers: Number of drivers leaving from each depot.
- Time: Computing time in seconds.

5.1 Multistart Heuristic

The results for *Class I* for the multistart heuristic can be found in Table 1. Bigger instances generally result in higher objective values and higher computation times. Furthermore, as the time bound of the drivers' routes gets bigger, the computation time generally decreases. A reason for this, is that with larger time bounds for the drivers' routes, drivers are able to visit more customer and fewer drivers are needed. As a result, the heuristic algorithm performs less iterations and this results in a shorter computation time. Moreover, because fewer drivers are needed, the objective value also decreases when the drivers' time bound increases. These results are very similar to the results in Domínguez-Martín et al. (2023). However, sometimes the objective values differ slightly and our computation times are higher.

Name	$n+2$	T_A				T_B				T_C				T_D			
		T	Sol	Drivers	Time	T	Sol	Drivers	Time	T	Sol	Drivers	Time	T	Sol	Drivers	Time
n10-1	10	6	654	2	3.43	7	443	1	1.86	8	411	1	2.72	10	371	1	1.27
n10-2	10	5	486	2	3.45	6	293	1	0.63	7	293	1	0.65	10	293	1	1.01
n10-3	10	5	987	3	4.58	6	646	2	2.07	7	390	1	0.98	10	383	1	0.79
n10-4	10	5	611	2	2.94	6	533	2	2.22	7	384	1	0.70	10	383	1	0.61
n10-5	10	5	594	2	3.04	6	366	1	0.65	7	355	1	1.15	10	351	1	0.71
n15-1	15	6	456	2	5.71	8	349	1	2.64	9	349	1	3.90	12	302	1	2.35
n15-2	15	6	746	2	4.19	8	412	1	1.29	9	405	1	1.64	12	405	1	1.47
n15-3	15	6	660	2	4.69	8	441	1	1.26	9	387	1	1.69	12	387	1	1.59
n15-4	15	6	1093	3	8.94	8	715	2	5.06	9	498	1	1.30	12	460	1	2.47
n15-5	15	6	788	2	3.99	8	749	2	4.11	9	473	1	1.50	12	469	1	1.50
n20-1	20	7	1262	3	13.64	9	844	2	10.06	10	559	1	3.44	14	521	1	5.77
n20-2	20	7	667	2	6.64	9	613	2	8.78	10	441	1	3.28	14	402	1	3.49
n20-3	20	7	846	2	4.87	9	757	2	9.09	10	540	1	2.41	14	508	1	4.18
n20-4	20	7	602	2	8.39	9	582	2	8.31	10	448	1	3.10	14	417	1	2.64
n20-5	20	7	651	2	7.20	9	538	2	8.82	10	434	1	3.09	14	409	1	4.72
n25-1	25	8	719	2	12.90	10	708	2	13.98	11	501	1	4.23	16	483	1	4.18
n25-2	25	8	796	2	7.97	10	699	2	14.27	11	510	1	3.43	16	486	1	6.54
n25-3	25	8	601	2	13.71	10	553	2	12.76	11	436	1	3.47	16	402	1	3.83
n25-4	25	8	686	2	13.19	10	681	2	12.82	11	470	1	3.48	16	456	1	3.54
n25-5	25	8	724	2	11.57	10	692	2	12.70	11	494	1	3.43	16	454	1	3.59
n30-1	30	9	887	2	12.02	12	816	2	21.69	13	628	1	4.78	18	582	1	10.75
n30-2	30	9	852	2	15.61	12	827	2	20.49	13	561	1	4.51	18	553	1	7.05
n30-3	30	9	812	2	18.37	12	797	2	19.40	13	507	1	4.51	18	487	1	3.88
n30-4	30	9	721	2	12.74	12	663	2	21.80	13	536	1	4.40	18	495	1	9.43
n30-5	30	9	750	2	18.12	12	732	2	20.50	13	495	1	4.86	18	492	1	8.73

Table 1: Results for *Class I* instances

Table 2 shows the results for the instances of *Class II*. These results show even more clearly that when the number of drivers increases, so does the computation time. Furthermore, when the number of drivers leaving from each depot is equal to one, the objective value is either high in the 500s or low in the 600s. However, when the number of drivers increases to two, the objective value is either high in the 700 or low in the 800. This clearly shows that an increase in drivers leads to a higher objective value and thus more costs. Similar to the results of *Class I*, the obtained results show a lot of resemblance with the results of Domínguez-Martín et al. (2023), except for the fact that our computation times are longer. Furthermore, while Domínguez-Martín et al. (2023) is able to find solutions with one driver per depot for n50-5 and n50-10, our heuristic is not able to do this. This results in a higher objective value and longer computation times. However, this is possible because of the randomness in the heuristic. Running the heuristic again or increasing the number of iterations could result in similar objective values.

Name	$n+2$	Sol	Drivers	Time
n50-1	50	608	1	11.26
n50-2	50	591	1	10.59
n50-3	50	593	1	10.62
n50-4	50	597	1	10.76
n50-5	50	757	2	60.42
n50-6	50	585	1	10.86
n50-7	50	551	1	11.05
n50-8	50	591	1	10.89
n50-9	50	607	1	11.00
n50-10	50	788	2	44.46
n50-11	50	603	1	10.82
n50-12	50	628	1	10.75
n50-13	50	786	2	60.94
n50-14	50	618	1	11.41
n50-15	50	813	2	44.66
n50-16	50	615	1	11.38

Table 2: Results for *Class II* instances with $T = 18$

The results for *Class III* can be found in Table 3. Just like for the results for *Class I*, it is clear that the computation times decrease when the drivers' time bound increases. Furthermore, while the algorithm is able to find optimal solutions within the time limit for instances with a problem size of 100 and for a few instances with a problem size of 200, it is not able to do this for larger instances, because the time limit is reached. When the problem sizes are 300 or larger and the time bound for the drivers' routes is tight, the heuristic is not able to find feasible solutions within the time limit. In this case the first feasible solution is returned and therefore these results can vary quite a bit from the results from Domínguez-Martín et al. (2023)

Name	$n+2$	T_A				T_B				T_C				T_D			
		T	Sol	Drivers	Time	T	Sol	Drivers	Time	T	Sol	Drivers	Time	T	Sol	Drivers	Time
n100-1	100	15	1091	3	343.50	20	888	2	239.49	30	884	2	152.41	40	745	1	79.81
n100-2	100	15	1232	3	324.71	20	969	2	221.01	30	960	2	169.48	40	795	1	73.84
n100-3	100	15	1054	3	350.44	20	890	2	248.63	30	848	2	272.18	40	770	1	108.84
n100-4	100	15	1173	3	336.88	20	942	2	232.44	30	934	2	207.09	40	766	1	64.43
n100-5	100	15	1152	3	347.50	20	950	2	240.92	30	915	2	312.35	40	788	1	109.50
n200-1	200	25	1474	3	600.00	35	1251	2	600.00	50	1197	2	600.00	60	1150	1	220.73
n200-2	200	25	1470	3	600.00	35	1299	2	600.00	50	1222	2	600.00	60	1154	1	220.29
n200-3	200	25	1443	3	600.00	35	1248	2	600.00	50	1208	2	600.00	60	1196	2	600
n200-4	200	25	1431	3	600.00	35	1227	2	600.00	50	1223	2	600.00	60	1145	1	213.28
n200-5	200	25	1426	3	600.00	35	1237	2	600.00	50	1213	2	600.00	60	1158	1	218.79
n300-1	300	35	1885	3	1147.94	45	1797	3	1155.03	70	1588	2	600.00	90	1407	1	600.00
n300-2	300	35	2007	3	1163.06	45	1919	3	1154.88	70	1746	2	600.00	90	1359	1	600.00
n300-3	300	35	1819	3	1161.87	45	1937	3	1150.26	70	1517	2	600.00	90	1360	1	600.00
n300-4	300	35	1957	3	1161.85	45	1938	3	1154.73	70	1566	2	600.00	90	1424	1	600.00
n300-5	300	35	1921	3	1162.44	45	1782	3	1154.91	70	1634	2	600.00	90	1355	1	600.00
n400-1	400	45	2168	3	1767.60	55	2127	3	1788.65	90	1804	2	600.00	115	1599	1	600.00
n400-2	400	45	2235	3	1768.21	55	2130	3	1770.01	90	1844	2	600.00	115	1569	1	600.00
n400-3	400	45	2215	3	1765.48	55	2124	3	1762.25	90	1809	2	600.00	115	1603	1	600.00
n400-4	400	45	2277	3	1774.11	55	2282	3	1765.75	90	1962	2	600.00	115	1605	1	600.00
n400-5	400	45	2285	3	1778.92	55	2214	3	1798.10	90	1956	2	600.00	115	1637	1	600.00
n500-1	500	55	2501	3	2924.48	65	2299	3	2923.74	110	1887	2	600.00	140	1917	2	600.00
n500-2	500	55	2580	3	2953.46	65	2386	3	2923.79	110	2219	2	600.00	140	2058	2	600.00
n500-3	500	55	2419	3	2930.89	65	2375	3	2922.18	110	2057	2	600.00	140	2189	2	600.00
n500-4	500	55	2466	3	2953.77	65	2432	3	2925.96	110	2251	2	600.00	140	2108	2	600.00
n500-5	500	55	2312	3	2958.01	65	2357	3	2928.73	110	2217	2	600.00	140	1915	2	600.00
n600-1	600	65	2525	3	5127.86	115	2344	2	600.00	155	2203	2	600.00	205	1982	1	600.00
n600-2	600	65	2608	3	5116.94	115	2186	2	600.00	155	2296	2	600.00	205	1961	1	600.00
n600-3	600	65	2772	3	5163.11	115	2331	2	600.00	155	2075	2	600.00	205	1976	1	600.00
n600-4	600	65	2579	3	5121.32	115	2447	2	600.00	155	2042	2	600.00	205	1922	1	600.00
n600-5	600	65	2358	3	5145.57	115	2263	2	600.00	155	2173	2	600.00	205	2020	1	600.00
n800-1	800	85	2788	3	11868.64	135	2477	2	600.00	175	2478	2	600.00	225	2308	1	600.00
n800-2	800	85	2977	3	12113.01	135	2787	2	600.00	175	2695	2	600.00	225	2303	1	600.00
n800-3	800	85	2817	3	12243.97	135	2495	2	600.00	175	2392	2	600.00	225	2306	1	600.00
n800-4	800	85	2980	3	12078.07	135	2465	2	600.00	175	2412	2	600.00	225	2298	1	600.00
n800-5	800	85	2865	3	12146.54	135	2657	2	600.00	175	2352	2	600.00	225	2258	1	600.00
n1000-1	1000	105	3270	3	22890.15	155	3030	2	600.00	205	2766	2	600.00	275	2580	1	600.00
n1000-2	1000	105	3587	3	23526.60	155	2781	2	600.00	205	2808	2	600.00	275	2606	1	600.00
n1000-3	1000	105	3427	3	22903.68	155	2950	2	600.00	205	2957	2	600.00	275	2537	1	600.00
n1000-4	1000	105	3246	3	22859.86	155	2840	2	600.00	205	2827	2	600.00	275	2533	1	600.00
n1000-5	1000	105	3144	3	25031.49	155	2846	2	600.00	205	3057	2	600.00	275	2547	1	600.00

Table 3: Results for *Class III* instances

5.2 Exchange Location and Depot Heuristic

In this section the results for the adjusted heuristic that tries to find better exchange and depot locations are given. First, this was tried for the instances in *Class I* for three different p values, two time bounds for the drivers T and the two starting exchange locations which will be denoted by e. The different p values that will be used are 10, 25 and 50. These were chosen to assess the difference in the objective values and the difference in the computation times. The time bounds for the drivers' routes that will be used are T_A and T_D that were also used in Table 1.

There are two locations that will be used as starting exchange location. First of all, the exchange locations as created by Domínguez-Martín et al. (2023) will be used as starting exchange location and is denoted by *original*. Secondly, the location that is exactly in between the two depots is calculated. Then, the customer location that is the closest to this location is chosen to be the starting exchange location. If multiple customer locations are located at an

equal distance from this location, one of these customer locations is picked at random as the starting exchange location. This starting exchange location is denoted by *middle*.

The tables will now contain some extra information. The extra data that are displayed in the columns are:

- Gap: Percentage deviation between the adjusted heuristic objective value and the multistart heuristic objective value. It is calculated as $\frac{Sol\ adjusted\ heuristic - Sol\ multistart\ heuristic}{Sol\ multistart\ heuristic} \times 100$.
- e^*: Best found exchange location.
- d_1^*: Best found location for depot d_1.
- d_2^*: Best found location for depot d_2.

A negative value for gap means that the adjusted heuristic is able to find a better exchange location and/or depot locations that result in a better objective value. When the gap is equal to zero, the adjusted heuristic is not able to find a better exchange location and depot locations and this means that we have found a (local) optimum. It is also possible for the gap to be positive. This can happen when the best solution found after running the multistart heuristic part of the adjusted heuristic is different from the best solution found when running the multistart heuristic. A reason for this is that there is randomness in the heuristic and because the heuristic stops when it reaches the time limit. It can also happen when the starting exchange location is different from the one used in Domínguez-Martín et al. (2023) because the best found drivers' routes might be less efficient.

The results for $T = T_A$ and $e = middle$ can be found in Table 4, for $T = T_A$ and $e = original$ in Table 5, for $T = T_D$ and $e = middle$ in Table 6 and the results for $T = T_D$ and $e = original$ can be found in Table 7. These results show that the adjusted heuristic is able to find a better solution for almost all the instances. The only instances for which no improvement is possible is for n20-4 and n25-3 when the drivers' time bound is equal to T_D. The column *Gap* in the tables show that there is generally more improvement possible when the time bound is tighter. This is likely because when the time bound for the drivers' routes is bigger, it is easier to insert a location into a route at the best place. This causes these objective values to already be lower than for a tighter bound, even before searching for better exchange and depot locations. Furthermore, when the instance size increases, there is generally less improvement in the objective value. This is especially the case for a looser time bound for the drivers' routes.

It is also possible for the different starting locations to give different locally optimal exchange and depot locations. For example, when $p = 10$, having *middle* as starting exchange location is more likely to give better results. A reason for this could be that the original exchange and depot locations are generated completely random in a square of $[0, 100] \times [0, 100]$. It is therefore possible that the exchange location is close to the border with no other location nearby. It is therefore stuck in that location and not able to find a better solution. Increasing the distance p increases the chance of finding a better location and is therefore more likely to improve the solution. If the starting exchange location is in between the two depots, it is more likely that it is located in a more efficient location and this can therefore improve the solution more easily. However, because the starting exchange location differs from the one used in the multistart

heuristic, it is also possible to get a solution that is worse than the original solution found. This can for example be seen for n10-2, n20-2, n20-4, n25-1 and n30-4 in Table 4. Nevertheless, when the distance p increases, the heuristic is able to find a solution that is at least as good as the solution found by the multistart heuristic. When starting from the *original* exchange location" you will always be able to find a solution that is at least as good as the solution found for the multistart heuristic for the instances in *Class I* . In general, when the distance p increases both starting exchange locations have a better chance of finding the optimal locations for the exchange and depot locations. In 28% of the cases both starting exchange locations give the same results, *middle* outperforms *original* 42% of the time and *original* performs better than *middle* for 30% of the instances. The computation times for both starting exchange locations are approximately the same as for the multistart heuristic. Therefore none of the starting exchange locations consistently outperforms the other.

Name	$n+2$	T	$p=10$							$p=25$							$p=50$						
			Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n10-1	10	6	462	2	5	0	9	1.91	-29.36	449	2	5	0	4	1.92	-31.35	449	2	5	0	4	1.91	-31.35
n10-2	10	5	488	2	3	0	5	1.67	0.41	373	2	8	6	3	1.65	-23.25	373	2	8	6	3	1.67	-23.25
n10-3	10	5	546	2	5	0	9	2.00	-44.68	546	2	5	0	9	1.99	-44.68	429	2	4	6	9	1.99	-56.53
n10-4	10	5	611	2	8	0	9	1.71	0.00	611	2	8	0	9	1.72	0.00	555	2	8	2	1	1.71	-9.17
n10-5	10	5	379	2	6	0	9	2.01	-36.20	379	2	6	0	9	2.01	-36.20	379	2	6	0	9	2.02	-36.20
n15-1	15	6	447	2	10	0	14	5.35	-1.97	440	2	10	0	8	4.18	-3.51	440	2	10	0	8	4.20	-3.51
n15-2	15	6	664	2	3	0	14	2.95	-10.99	652	2	3	0	12	2.95	-12.60	451	2	10	6	1	2.94	-39.54
n15-3	15	6	530	2	3	0	14	4.28	-19.70	491	2	3	0	8	4.23	-25.61	491	2	3	0	8	4.24	-25.61
n15-4	15	6	621	2	10	0	14	3.20	-43.18	604	2	10	3	14	3.20	-44.74	604	2	10	3	14	3.20	-44.74
n15-5	15	6	545	2	4	0	14	4.41	-30.84	540	2	6	0	14	4.43	-31.47	540	2	6	0	14	4.44	-31.47
n20-1	20	7	705	2	15	0	19	7.34	-44.14	705	2	15	0	19	5.95	-44.14	578	2	12	20	2	5.95	-54.20
n20-2	20	7	671	2	16	0	1	6.74	0.60	604	2	1	16	19	6.80	-9.45	570	2	18	19	9	6.86	-14.54
n20-3	20	7	616	2	1	0	19	9.06	-27.19	616	2	1	0	19	9.05	-27.19	616	2	1	0	19	9.09	-27.19
n20-4	20	7	604	2	11	8	1	9.58	0.33	587	2	18	11	19	9.68	-2.49	596	2	12	0	19	9.61	-1.00
n20-5	20	7	530	2	8	0	19	10.09	-18.59	485	2	8	0	12	9.88	-25.50	463	2	8	2	12	9.62	-28.88
n25-1	25	8	720	2	18	0	24	10.99	0.14	643	2	14	17	23	11.11	-10.57	618	2	14	22	5	10.89	-14.05
n25-2	25	8	733	2	2	0	24	11.47	-7.91	606	2	8	2	15	11.36	-23.87	595	2	7	8	9	10.84	-25.25
n25-3	25	8	621	2	4	0	24	15.61	3.33	572	2	23	15	14	14.59	-4.83	591	2	19	4	24	14.76	-1.66
n25-4	25	8	594	2	9	0	24	14.82	-13.41	563	2	15	14	24	14.38	-17.93	529	2	15	9	14	14.21	-22.89
n25-5	25	8	648	2	12	0	19	14.60	-10.50	619	2	12	17	19	14.67	-14.50	564	2	19	24	8	14.16	-22.10
n30-1	30	9	680	2	7	0	29	19.28	-23.34	669	2	18	17	29	18.73	-24.58	665	2	18	9	29	18.72	-25.03
n30-2	30	9	618	2	12	21	5	18.42	-27.46	635	2	21	12	5	17.64	-25.47	629	2	21	12	5	17.78	-26.17
n30-3	30	9	616	2	3	0	18	22.96	-24.14	523	2	11	29	12	22.37	-35.59	549	2	18	16	29	22.40	-32.39
n30-4	30	9	772	2	19	27	29	14.10	7.07	655	2	18	14	29	14.31	-9.15	666	2	14	28	18	14.01	-7.63
n30-5	30	9	569	2	15	1	20	21.01	-24.13	557	2	20	15	29	21.01	-25.73	557	2	20	15	29	21.10	-25.73

Table 4: Results for *Class I* instances with $T = T_A$ and $e = middle$

Name	$n+2$	T	$p=10$							$p=25$							$p=50$						
			Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n10-1	10	6	654	2	8	0	9	2.04	0.00	622	2	8	2	5	2.08	-4.89	449	2	5	0	4	2.06	-31.35
n10-2	10	5	389	2	8	0	6	1.94	-19.96	373	2	8	6	3	1.93	-23.25	380	2	8	6	2	1.95	-21.81
n10-3	10	5	987	3	8	0	9	3.13	0.00	987	3	8	0	9	3.14	0.00	560	3	4	3	6	3.13	-43.26
n10-4	10	5	611	2	8	0	9	1.72	0.00	611	2	8	0	9	1.70	0.00	555	2	8	2	1	1.75	-9.17
n10-5	10	5	594	2	8	0	9	1.88	0.00	594	2	8	0	9	1.91	0.00	380	2	6	5	9	1.86	-36.03
n15-1	15	6	456	2	13	0	14	4.26	0.00	440	2	10	0	8	4.25	-3.51	440	2	10	0	8	4.28	-3.51
n15-2	15	6	746	2	13	0	14	2.80	0.00	642	2	13	5	2	2.79	-13.94	451	2	10	6	1	2.79	-39.54
n15-3	15	6	660	2	13	0	14	3.03	0.00	491	2	3	0	8	3.03	-25.61	491	2	3	0	8	3.03	-25.61
n15-4	15	6	1093	3	13	0	14	6.88	0.00	1071	2	13	3	14	6.86	-2.01	701	3	10	9	11	6.90	-35.86
n15-5	15	6	785	2	13	6	11	2.75	-0.38	785	2	13	6	11	2.77	-0.38	540	2	6	0	14	2.77	-31.47
n20-1	20	7	1039	3	13	0	18	13.17	-17.67	919	3	16	11	15	13.94	-27.18	637	3	10	12	2	13.77	-49.52
n20-2	20	7	667	2	18	0	19	7.63	0.00	619	2	18	9	19	7.68	-7.20	527	2	1	16	19	7.56	-20.99
n20-3	20	7	846	2	18	0	19	5.43	0.00	734	2	5	1	19	5.60	-13.24	585	2	13	8	19	5.59	-30.85
n20-4	20	7	598	2	18	8	1	9.39	-0.66	581	2	18	11	1	9.39	-3.49	596	2	12	0	19	9.29	-1.00
n20-5	20	7	632	2	14	0	19	8.34	-2.92	476	2	8	0	12	8.29	-26.88	449	2	8	2	12	8.23	-31.03
n25-1	25	8	719	2	23	0	24	11.75	0.00	675	2	14	0	18	12.04	-6.12	616	2	14	22	23	12.14	-14.33
n25-2	25	8	796	2	23	0	24	8.87	0.00	647	2	15	8	18	8.68	-18.72	647	2	15	8	18	8.74	-18.72
n25-3	25	8	601	2	23	0	24	15.16	0.00	593	2	23	15	24	14.34	-1.33	566	2	19	14	24	15.13	-5.82
n25-4	25	8	668	2	13	0	2	15.08	-2.62	575	2	17	14	24	14.99	-16.18	528	2	9	14	17	14.89	-23.03
n25-5	25	8	724	2	23	0	24	13.07	0.00	643	2	8	0	24	13.09	-11.19	592	2	24	15	19	12.82	-18.23
n30-1	30	9	879	2	28	0	29	11.25	-0.90	853	2	28	29	11	11.28	-3.83	718	2	7	21	29	11.25	-19.05
n30-2	30	9	778	2	28	5	12	15.87	-8.69	705	2	16	23	18	17.20	-17.52	675	2	5	21	12	16.69	-20.77
n30-3	30	9	812	2	28	0	29	20.28	0.00	549	2	18	12	29	20.48	-32.39	550	2	18	12	29	20.18	-32.27
n30-4	30	9	690	2	28	10	29	13.96	-4.30	691	2	28	10	29	14.17	-4.16	666	2	14	28	18	14.19	-7.63
n30-5	30	9	652	2	28	16	29	19.88	-13.07	652	2	28	16	29	20.49	-13.07	564	2	20	15	29	19.95	-24.80

Table 5: Results for *Class I* instances with $T = T_A$ and $e = original$

Name	$n+2$	T	$p=10$							$p=25$							$p=50$						
			Solution	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Solution	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Solution	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n10-1	10	10	340	1	5	0	9	1.10	-8.36	329	1	4	0	5	0.68	-11.32	321	1	4	0	6	0.63	-13.48
n10-2	10	10	298	1	3	0	9	0.63	1.71	298	1	3	0	9	0.56	1.71	298	1	3	0	9	0.63	1.71
n10-3	10	10	337	1	5	0	9	0.68	-12.01	337	1	5	0	9	0.63	-12.01	298	1	6	5	8	0.63	-22.19
n10-4	10	10	383	1	8	0	9	0.60	0.00	383	1	8	0	9	0.59	0.00	375	1	3	0	9	0.59	-2.09
n10-5	10	10	270	1	6	0	9	0.62	-23.08	270	1	6	0	9	0.61	-23.08	270	1	6	0	9	0.62	-23.08
n15-1	15	12	305	1	10	0	14	1.37	0.99	302	1	13	0	5	1.34	0.00	302	1	13	0	5	1.34	0.00
n15-2	15	12	318	1	3	0	14	1.26	-21.48	318	1	3	0	14	1.27	-21.48	318	1	3	0	14	1.27	-21.48
n15-3	15	12	339	1	3	0	14	1.28	-12.40	339	1	3	0	14	1.29	-12.40	339	1	3	0	14	1.29	-12.40
n15-4	15	12	451	1	10	0	14	1.18	-1.96	451	1	10	0	14	1.17	-1.96	447	1	9	5	14	1.17	-2.83
n15-5	15	12	402	1	4	0	14	1.50	-14.29	389	1	6	0	14	1.49	-17.06	389	1	6	0	14	1.51	-17.06
n20-1	20	14	466	1	15	0	19	4.65	-10.56	466	1	15	0	19	4.64	-10.56	439	1	10	5	2	4.65	-15.74
n20-2	20	14	414	1	12	0	19	2.81	2.99	414	1	12	0	19	2.80	2.99	414	1	12	0	19	2.81	2.99
n20-3	20	14	462	1	1	0	19	3.80	-9.06	462	1	1	0	19	3.83	-9.06	457	1	13	1	19	3.82	-10.04
n20-4	20	14	435	1	11	0	19	2.14	4.32	417	1	18	0	19	2.14	0.00	417	1	18	8	19	2.14	0.00
n20-5	20	14	373	1	8	0	19	2.41	-8.80	373	1	8	0	19	2.41	-8.80	373	1	8	0	19	2.42	-8.80
n25-1	25	16	482	1	18	0	24	4.60	-0.21	482	1	18	0	24	4.65	-0.21	482	1	18	0	24	4.61	-0.21
n25-2	25	16	463	1	2	0	24	8.61	-4.73	446	1	15	24	18	8.57	-8.23	447	1	7	2	18	8.59	-8.02
n25-3	25	16	416	1	4	0	24	5.37	3.48	402	1	23	0	19	5.32	0.00	402	1	23	10	24	5.31	0.00
n25-4	25	16	437	1	15	0	24	3.77	-4.17	437	1	15	10	24	3.82	-4.17	432	1	16	22	24	3.73	-5.26
n25-5	25	16	435	1	12	3	24	4.95	-4.19	435	1	12	3	24	4.92	-4.19	424	1	0	3	24	4.94	-6.61
n30-1	30	18	539	1	7	0	29	9.08	-7.39	532	1	18	0	13	9.44	-8.59	532	1	18	0	29	9.49	-8.59
n30-2	30	18	535	1	3	0	29	12.72	-3.25	517	1	12	21	3	12.71	-6.51	493	1	28	19	3	12.78	-10.85
n30-3	30	18	436	1	3	0	29	12.06	-10.47	435	1	18	0	29	12.06	-10.68	435	1	18	17	29	12.32	-10.68
n30-4	30	18	495	1	19	0	29	6.56	0.00	495	1	19	0	29	6.61	0.00	486	1	14	0	3	6.63	-1.82
n30-5	30	18	461	1	15	0	29	11.39	-6.30	448	1	14	0	29	11.39	-8.94	442	1	29	0	10	11.32	-10.16

Table 6: Results for *Class I* instances with $T = T_D$ and $e = middle$

Name	$n+2$	T	$p=10$							$p=25$							$p=50$						
			Solution	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Solution	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Solution	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n10-1	10	10	371	1	8	0	9	0.69	0.00	353	1	8	2	9	0.65	-4.85	308	1	2	8	4	0.65	-16.98
n10-2	10	10	293	1	8	0	9	0.70	0.00	293	1	8	0	9	0.73	0.00	293	1	8	0	9	0.71	0.00
n10-3	10	10	383	1	8	0	9	0.94	0.00	383	1	8	0	9	0.92	0.00	293	1	3	7	9	0.94	-23.50
n10-4	10	10	383	1	8	0	9	0.73	0.00	383	1	8	0	9	0.71	0.00	359	1	9	2	4	0.72	-6.27
n10-5	10	10	351	1	8	0	9	0.77	0.00	351	1	8	0	9	0.77	0.00	270	1	6	0	9	0.78	-23.08
n15-1	15	12	302	1	13	0	14	1.76	0.00	302	1	13	0	14	1.75	0.00	302	1	13	0	14	1.76	0.00
n15-2	15	12	405	1	13	0	14	1.60	0.00	398	1	5	0	14	1.61	-1.73	337	1	7	8	10	1.62	-16.79
n15-3	15	12	387	1	13	0	14	1.67	0.00	339	1	3	0	14	1.66	-12.40	339	1	3	4	14	1.67	-12.40
n15-4	15	12	460	1	13	0	14	2.69	0.00	460	1	13	0	14	2.77	0.00	441	1	9	5	11	2.71	-4.13
n15-5	15	12	469	1	13	0	14	1.64	0.00	469	1	13	0	14	1.64	0.00	390	1	6	3	14	1.67	-16.84
n20-1	20	14	516	1	13	0	19	5.27	-0.96	461	1	16	0	11	5.27	-11.52	471	1	15	4	19	5.30	-9.60
n20-2	20	14	384	1	9	0	18	3.75	-4.48	384	1	9	0	18	3.82	-4.48	380	1	9	0	15	3.96	-5.47
n20-3	20	14	508	1	18	0	19	4.61	0.00	478	1	7	9	19	4.70	-5.91	471	1	13	0	19	4.80	-7.28
n20-4	20	14	417	1	18	0	19	2.69	0.00	417	1	18	0	19	2.71	0.00	417	1	18	0	19	2.87	0.00
n20-5	20	14	398	1	14	10	19	5.38	-2.69	369	1	16	0	15	5.36	-9.78	369	1	16	7	19	5.49	-9.78
n25-1	25	16	483	1	23	0	24	4.04	0.00	483	1	3	0	16	4.05	0.00	483	1	3	9	24	4.04	0.00
n25-2	25	16	486	1	23	0	24	7.93	0.00	461	1	8	0	24	7.78	-5.14	461	1	8	0	24	7.78	-5.14
n25-3	25	16	402	1	23	0	24	4.46	0.00	402	1	23	0	24	4.53	0.00	402	1	23	0	24	4.48	0.00
n25-4	25	16	448	1	13	0	24	4.30	-1.75	436	1	16	22	24	4.29	-4.39	432	1	16	0	12	4.29	-5.26
n25-5	25	16	454	1	23	0	24	4.17	0.00	452	1	20	0	24	4.26	-0.44	442	1	7	23	24	4.17	-2.64
n30-1	30	18	582	1	28	0	29	11.77	0.00	580	1	28	1	8	11.77	-0.34	527	1	29	1	19	11.83	-9.45
n30-2	30	18	536	1	28	0	5	8.39	-3.07	545	1	16	5	0	8.34	-1.45	534	1	16	3	21	8.37	-3.44
n30-3	30	18	487	1	28	0	29	4.56	0.00	470	1	9	4	18	4.56	-3.49	469	1	9	16	12	4.59	-3.70
n30-4	30	18	492	1	10	0	29	11.32	-0.61	486	1	14	27	3	11.32	-1.82	486	1	14	0	3	11.42	-1.82
n30-5	30	18	492	1	28	0	29	9.95	0.00	492	1	28	0	29	9.96	0.00	459	1	14	1	29	10.01	-6.71

Table 7: Results for *Class I* instances with $T = T_D$ and $e = original$

Table 8 displays the results for the adjusted heuristic for *Class II* where $T = 18$ and $e = middle$ and Table 9 shows the results for the adjusted heuristic for *Class II* where $T = 18$ and $e = original$. Because the results for *Class I* show that $p = 10$ does not give as good results as $p = 25$ and $p = 50$ and the computation times barely increase when p is increased, the *Class II* instances were run for just $p = 25$ and $p = 50$.

Just like for the instances for *Class I*, a higher p gives better results in most cases while again barely increasing the computation times. While the adjusted heuristic was able to find improvements for almost all instances of *Class I*, this is not the case for *Class II*. The results show that there are only real improvements for n50-5, n50-10, n50-13 and n50-15. These are the instances that gave for both the multistart heuristic and the adjusted heuristic solutions with two drivers. The other instances show at most an improvement of 2.63% or no improvement at all. This shows again that when the time bound for the drivers' routes is loose and less drivers are needed, less improvement is possible when adjusting the exchange and depot locations.

The results for the column *Gap* show that there are also multiple quite large positive numbers. This happens when $e = middle$, since this is accompanied by the fact that 2 drivers are needed while the original solution using the multistart heuristic needed only 1 driver. When extra drivers are needed, more costs are made and this therefore results in a higher objective value.

For these instances, starting exchange location *original* outperforms *middle* in 75% of the cases. There are no big differences in computation times and thus starting exchange location *original* seems to perform better for these instances.

Name	$n+2$	$p=25$							$p=50$						
		Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n50-1	50	678	2	31	37	1	39.00	11.51	678	2	31	37	12	38.67	11.51
n50-2	50	591	1	23	0	49	9.80	0.00	591	1	23	0	22	10.03	0.00
n50-3	50	592	1	14	32	19	10.03	-0.17	589	1	8	0	49	10.24	-0.67
n50-4	50	716	2	16	22	49	45.92	19.93	662	2	38	13	35	46.41	10.89
n50-5	50	684	2	33	9	20	51.55	-9.64	711	2	25	27	31	51.29	-6.08
n50-6	50	592	1	20	0	30	9.83	1.20	592	1	20	0	49	9.89	1.20
n50-7	50	542	1	39	0	49	10.04	-1.63	542	1	39	0	49	10.15	-1.63
n50-8	50	596	1	39	0	49	9.97	0.85	593	1	34	0	49	10.04	0.34
n50-9	50	607	1	48	0	49	10.14	0.00	595	1	23	27	49	10.12	-1.98
n50-10	50	677	2	35	45	29	38.94	-14.09	674	2	1	38	29	39.11	-14.47
n50-11	50	603	1	48	0	49	9.82	0.00	603	1	48	4	49	9.89	0.00
n50-12	50	794	2	24	0	43	43.98	26.43	714	2	12	48	38	44.24	13.69
n50-13	50	725	2	44	9	46	36.57	-7.76	740	2	26	25	42	36.92	-5.85
n50-14	50	616	1	12	0	49	10.00	-0.32	614	1	31	37	4	10.32	-0.65
n50-15	50	671	2	5	0	29	39.30	-17.47	644	2	24	44	29	39.63	-20.79
n50-16	50	617	1	21	0	49	10.08	0.33	617	1	21	0	4	10.20	0.33

Table 8: Results for *Class II* instances with $T = 18$ and $e = middle$

Name	$n+2$	$p=25$							$p=50$						
		Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n50-1	50	608	1	48	0	49	10.05	0.00	592	1	17	0	49	10.04	-2.63
n50-2	50	591	1	48	0	49	9.93	0.00	591	1	48	0	49	9.95	0.00
n50-3	50	590	1	8	32	49	9.99	-0.51	588	1	8	30	49	10.20	-0.84
n50-4	50	597	1	48	0	49	9.98	0.00	597	1	48	0	49	10.07	0.00
n50-5	50	689	2	27	2	25	53.84	-8.98	676	2	27	26	25	53.98	-10.70
n50-6	50	585	1	48	0	49	9.96	0.00	585	1	48	0	49	10.03	0.00
n50-7	50	542	1	39	0	49	10.08	-1.63	544	1	39	11	49	10.09	-1.27
n50-8	50	585	1	34	43	49	10.10	-1.02	585	1	34	24	49	10.08	-1.02
n50-9	50	607	1	48	0	49	10.23	0.00	595	1	23	13	45	10.18	-1.98
n50-10	50	728	2	26	31	49	40.02	-7.61	668	2	39	42	3	40.10	-15.23
n50-11	50	592	1	2	0	49	10.05	-1.82	603	1	48	4	44	9.93	0.00
n50-12	50	628	1	48	0	49	12.01	0.00	617	1	46	25	49	12.08	-1.75
n50-13	50	734	2	36	0	48	53.23	-6.62	690	2	36	13	48	53.53	-12.21
n50-14	50	605	1	43	0	49	10.24	-2.10	605	1	43	6	49	10.20	-2.10
n50-15	50	737	2	38	6	19	39.54	-9.35	731	2	38	6	19	39.51	-10.09
n50-16	50	615	1	48	29	49	10.04	0.00	615	1	48	29	49	10.28	0.00

Table 9: Results for *Class II* instances with $T = 18$ and $e = original$

The results for *Class III* can be found in the Appendix in Section A. Table 10 shows the results for $e = middle$ and Table 11 displays the results for $e = original$. The drivers' time bound T is equal to T_C for both tables. The results show that the objective values can be decreased with up to 20.10%. All originally found solutions can be improved and for 60% of the instances, $e = middle$ outperforms $e = while$ while for the remaining 40% it is the other way around. The different values for p barely influence the objective values. However, from $n + 2 = 200$ onwards, $p = 25$ is significantly faster than $p = 50$ for nearly all of the instances. This shows that having a smaller p is better for larger instances.

6 Conclusion

This paper discusses a heuristic that is able to find locally optimal exchange and depot locations for the Driver and Vehicle Routing Problem. In this problem, there are a set of customers and

2 depots. One of the customers is also an exchange location. While drivers have to return to their home depot at the end of their shift, the vehicles have to end at the other depot. One of the customer locations is also an exchange location where drivers can switch vehicles. Our heuristic builds upon the multistart heuristic that was introduced by Domínguez-Martín et al. (2023). The multistart heuristic first tries to find the routes for the drivers for a given set of drivers leaving from each depot. Then the vehicles' routes are created in such a way that they are compatible with the best found driver's routes.

First, the multistart heuristic was implemented. The results show that our results are very similar to the results of Domínguez-Martín et al. (2023). Secondly, we have implemented an extra step in the heuristic that is able to find better locations for the exchange and depot locations for most of the instances. In order to do this, we look for customer locations that are within distance p from the current best exchange or depot location and try to improve the solution. If the solution improves, we continue with the new customer location as exchange or depot location. This continues until no further improvements can be found.

Results show that increasing the distance p does not significantly increase the running time for instances with up to 100 locations, while it does allow for better solutions. For the instances with 200 locations or more, increasing p does result in significantly higher computation times while it does not result in objective values that are significantly better.

Furthermore, two different exchange locations were used to construct the first set of drivers' routes which are then later on improved by finding other exchange and depot locations. The first exchange location used is the same location as was used in Domínguez-Martín et al. (2023). The second exchange location is the customer location that is the closest to the point that is exactly in between the two depots. While both starting exchange locations do sometimes give the same solution, this is not always the case. However, not one of the starting exchange locations consistently outperforms the other when looking at the objective values and the computation times.

Moreover, the results show that if the time bound for the drivers' routes is tighter, more drivers are needed and this generally also increases the chance that the solution can be improved by locating better exchange and depot locations. When the time bound for the drivers' routes is loose and only one driver per depot is needed, the adjusted algorithm is not able to find solutions that are significantly better. This is especially the case for instances with bigger problem sizes.

For further research, it would be interesting to develop a heuristic that is able to find globally optimal exchange and depot locations while barely increasing computation times. This could help even further in reducing the total costs. Another interesting direction for further research would be use multiple exchange locations. It would then be interesting to see what this does to the optimal exchange and depot locations and to the total costs.

References

Avella, P., Boccia, M., Sforza, A. & Vasil'ev, I. (2009). An effective heuristic for large-scale capacitated facility location problems. *Journal of Heuristics*, *15*, 597–615.

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.

Domínguez-Martín, B., Rodríguez-Martín, I. & Salazar-González, J.-J. (2018a). The driver and vehicle routing problem. *Computers & Operations Research*, *92*, 56–64.

Domínguez-Martín, B., Rodríguez-Martín, I. & Salazar-González, J.-J. (2018b). A heuristic approach to the driver and vehicle routing problem. In *International conference on computational logistics* (pp. 295–305).

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.

Ho, W., Ho, G. T., Ji, P. & Lau, H. C. (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering applications of artificial intelligence*, *21*(4), 548–557.

Karakatič, S. & Podgorelec, V. (2015). A survey of genetic algorithms for solving multi depot vehicle routing problem. *Applied Soft Computing*, *27*, 519–532.

Resende, M. G. & Werneck, R. F. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, *174*(1), 54–68.

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.

Surekha, P. & Sumathi, S. (2011). Solution to multi-depot vehicle routing problem using genetic algorithms. *World Applied Programming*, *1*(3), 118–131.

Üster, H. & Kewcharoenwong, P. (2011). Strategic design and analysis of a relay network in truckload transportation. *Transportation Science*, *45*(4), 505–523.

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.

A Extra Tables

Name	$n+2$	$p=25$							$p=50$						
		Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n100-1	100	854	2	45	83	5	188.73	-3.39	857	2	45	56	88	151.19	-3.05
n100-2	100	882	2	17	45	72	255.59	-8.13	884	2	17	45	72	199.96	-7.92
n100-3	100	855	2	57	17	1	213.94	0.83	854	2	57	17	1	168.09	0.71
n100-4	100	841	2	94	86	82	243.51	-9.96	844	2	94	86	82	194.39	-9.64
n100-5	100	848	2	41	30	23	318.75	-7.32	846	2	73	81	76	254.70	-7.54
n200-1	200	1181	2	75	160	199	603.23	-1.34	1168	2	148	68	127	609.61	-2.42
n200-2	200	1214	2	144	0	199	602.19	-0.65	1200	2	4	0	199	604.11	-1.80
n200-3	200	1231	2	194	147	91	604.49	1.90	1228	2	194	192	91	612.00	1.66
n200-4	200	1209	2	189	3	86	604.59	-1.14	1211	2	189	62	180	608.14	-0.98
n200-5	200	1183	2	96	97	174	603.15	-2.47	1188	2	96	97	171	605.73	-2.06
n300-1	300	1452	2	88	35	109	618.65	-8.56	1436	2	220	34	150	631.89	-9.57
n300-2	300	1403	2	266	112	47	607.37	-19.64	1395	2	236	39	47	619.26	-20.10
n300-3	300	1513	2	249	154	98	615.85	-0.26	1531	2	121	154	277	614.84	0.92
n300-4	300	1467	2	167	92	272	606.06	-6.32	1466	2	211	159	253	646.92	-6.39
n300-5	300	1379	2	15	172	273	621.61	-15.60	1395	2	139	269	273	622.54	-14.63
n400-1	400	1607	2	259	349	328	684.39	-10.92	1607	2	267	349	328	692.22	-10.92
n400-2	400	1602	2	345	151	367	623.03	-13.12	1618	2	7	364	352	684.39	-12.26
n400-3	400	1651	2	90	0	38	622.90	-8.73	1636	2	135	0	93	665.70	-9.56
n400-4	400	1671	2	223	9	184	639.38	-14.83	1661	2	223	32	276	668.80	-15.34
n400-5	400	1632	2	339	221	399	634.89	-16.56	1636	2	339	169	399	653.05	-16.36
n500-1	500	1959	2	323	213	230	673.16	3.82	1923	2	2	392	114	754.73	1.91
n500-2	500	1874	2	155	0	281	658.74	-15.55	1852	2	449	496	366	857.29	-16.54
n500-3	500	1858	2	469	147	39	662.89	-9.67	1847	2	106	205	133	812.38	-10.21
n500-4	500	1866	2	471	324	475	675.49	-17.10	1841	2	243	386	475	770.58	-18.21
n500-5	500	1829	2	101	115	125	754.84	-17.50	1820	2	101	0	420	735.30	-17.91
n600-1	600	2001	2	412	244	202	849.77	-9.17	2003	2	530	241	202	1209.47	-9.02
n600-2	600	1972	2	293	415	351	884.63	-14.11	1979	2	293	415	118	1075.66	-13.81
n600-3	600	2015	2	463	564	599	906.44	-2.89	2030	2	463	165	599	878.31	-2.17
n600-4	600	1957	2	578	484	356	761.10	-4.16	1936	2	264	75	466	1256.85	-5.19
n600-5	600	2040	2	537	0	316	752.65	-6.12	2035	2	237	218	594	984.15	-6.35
n800-1	800	2397	2	706	0	799	853.14	-3.27	2359	2	511	791	191	2861.38	-4.80
n800-2	800	2332	2	316	720	84	1167.96	-13.47	2333	2	609	656	84	1838.29	-13.43
n800-3	800	2490	2	468	239	672	1575.15	4.10	2430	2	648	689	205	2095.14	1.59
n800-4	800	2299	2	33	155	218	2086.90	-4.68	2347	2	355	570	442	2375.49	-2.69
n800-5	800	2301	2	684	386	513	1789.47	-2.17	2314	2	321	690	37	2477.08	-1.62
n1000-1	1000	2585	2	721	965	815	2423.37	-6.54	2596	2	307	198	510	3567.49	-6.15
n1000-2	1000	2604	2	208	146	229	3106.67	-7.26	2600	2	27	515	229	4144.39	-7.41
n1000-3	1000	2536	2	954	545	777	4378.06	-14.24	2517	2	5	706	901	5967.27	-14.88
n1000-4	1000	2583	2	27	938	414	4140.62	-8.63	2601	2	27	779	999	3836.21	-7.99
n1000-5	1000	2555	2	13	973	450	2870.42	-16.42	2564	2	684	973	82	6041.29	-16.13

Table 10: Results for *Class III* instances with $T = T_C$ and $e = middle$

Name	$n+2$	$p = 25$							$p = 50$						
		Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap	Sol	Drivers	e^*	d_1^*	d_2^*	Time	Gap
n100-1	100	861	2	26	98	86	139.80	-2.60	868	2	26	1	7	133.14	-1.81
n100-2	100	899	2	66	45	30	154.39	-6.35	909	2	37	45	74	147.76	-5.31
n100-3	100	827	2	64	17	99	206.78	-2.48	820	2	20	17	60	196.24	-3.30
n100-4	100	867	2	53	86	90	147.61	-7.17	871	2	53	86	90	139.93	-6.75
n100-5	100	877	2	23	43	80	222.33	-4.15	871	2	23	77	80	211.98	-4.81
n200-1	200	1173	2	68	160	199	602.47	-2.01	1180	2	195	157	199	605.71	-1.42
n200-2	200	1210	2	144	0	18	603.11	-0.98	1192	2	4	168	199	604.30	-2.45
n200-3	200	1199	2	198	147	165	604.77	-0.75	1177	2	81	147	99	612.17	-2.57
n200-4	200	1211	2	156	157	95	603.71	-0.98	1207	2	22	157	95	607.28	-1.31
n200-5	200	1203	2	198	120	174	603.07	-0.82	1212	2	198	97	119	608.43	-0.08
n300-1	300	1416	2	148	253	143	614.12	-10.83	1437	2	143	34	31	624.96	-9.51
n300-2	300	1403	2	156	132	100	609.50	-19.64	1419	2	97	112	47	627.20	-18.73
n300-3	300	1409	2	252	293	23	613.92	-7.12	1405	2	47	102	23	635.81	-7.45
n300-4	300	1447	2	168	92	255	615.79	-7.60	1449	2	168	80	269	626.67	-7.47
n300-5	300	1397	2	35	169	273	610.79	-14.50	1398	2	298	118	194	619.69	-14.44
n400-1	400	1620	2	333	349	208	638.94	-10.20	1616	2	307	349	243	658.77	-10.42
n400-2	400	1614	2	398	155	188	661.88	-12.47	1615	2	381	155	188	737.42	-12.42
n400-3	400	1640	2	52	152	347	625.11	-9.34	1640	2	398	0	347	668.07	-9.34
n400-4	400	1707	2	200	269	273	634.34	-13.00	1712	2	93	10	184	672.69	-12.74
n400-5	400	1676	2	328	273	399	612.52	-14.31	1686	2	59	273	207	663.69	-13.80
n500-1	500	1800	2	498	472	224	669.20	-4.61	1802	2	498	426	114	939.69	-4.50
n500-2	500	1875	2	394	71	39	662.98	-15.50	1856	2	171	190	379	815.14	-16.36
n500-3	500	1879	2	102	0	361	654.55	-8.65	1883	2	469	429	499	742.94	-8.46
n500-4	500	1907	2	92	318	475	703.53	-15.28	1897	2	340	178	475	925.25	-15.73
n500-5	500	1839	2	76	201	132	687.83	-17.05	1831	2	463	0	493	782.72	-17.41
n600-1	600	1990	2	582	518	599	680.99	-9.67	2003	2	204	507	243	914.31	-9.02
n600-2	600	1982	2	410	291	599	688.95	-13.68	1982	2	243	0	599	941.99	-13.68
n600-3	600	2014	2	105	442	599	809.63	-2.94	1994	2	226	541	599	1381.85	-3.90
n600-4	600	1976	2	587	75	599	671.97	-3.23	1971	2	173	379	599	1004.79	-3.48
n600-5	600	2043	2	598	440	284	793.38	-5.98	2020	2	519	285	316	1307.53	-7.04
n800-1	800	2312	2	798	659	526	1609.98	-6.70	2302	2	219	552	712	3006.64	-7.10
n800-2	800	2347	2	431	4	710	1619.75	-12.91	2338	2	497	0	289	2194.37	-13.25
n800-3	800	2292	2	334	46	716	2812.87	-4.18	2293	2	683	46	693	3108.39	-4.14
n800-4	800	2305	2	217	570	382	1774.32	-4.52	2309	2	411	662	393	3069.82	-4.35
n800-5	800	2298	2	798	690	578	1809.93	-2.30	2300	2	305	0	578	2949.09	-2.21
n1000-1	1000	2582	2	998	356	815	2332.80	-6.65	2585	2	126	0	815	5271.27	-6.54
n1000-2	1000	2627	2	650	870	229	1678.22	-6.45	2618	2	650	0	229	4977.84	-6.77
n1000-3	1000	2545	2	572	530	901	1417.22	-13.93	2574	2	685	729	901	5157.18	-12.95
n1000-4	1000	2535	2	899	938	812	1945.80	-10.33	2553	2	386	938	999	5082.05	-9.69
n1000-5	1000	2558	2	101	269	404	2182.77	-16.32	2560	2	419	973	470	10156.44	-16.26

Table 11: Results for *Class III* instances with $T = T_C$ and $e = original$

B Programming Code

All of the algorithms were programmed in Java. In order to have some overview, ten different classes were created to perform the heuristic. All of these classes will be briefly discussed in the sections.

B.1 Main

The first class is the "Main" class. This is the class that is run and calls all the other classes. Furthermore, this is the class that prints the results. It contains two methods. The first method is "read" and calling this method performs the multistart heuristic and prints the result. The second method is "readWithExchangeLocation". This method performs the adjusted heuristic and prints those results. When running either of these methods for a specified problem size n, the algorithm is run for all instances with problem size n.

B.2 StopWatch

The "StopWatch" class is the class that keeps track of the computation times.

B.3 Location

The class "Location" defines a location of a customer or one of the depots.

B.4 Route

The "Route" class represents a drivers' route or a vehicles' route. It is implemented as an ArrayList of instances of "Location".

B.5 ReadFile

The class "ReadFile" reads the data from the file and creates instances of the "Location" class.

B.6 DistanceCalculator

The "DistanceCalculator" class calculates the distances between all the points and puts them in a matrix.

B.7 TimeCalculator

The "TimeCalculator" class calculates the distances between all the points and puts them in a matrix.

B.8 MultistartHeuristic

The class "MultistartHeuristic" performs the multistart heuristic as is described in Section 4.1

B.9 HeuristicExchangeAndDepot

The class "HeuristicExchangeAndDepot" performs the adjusted heuristic as is described in Section 4.2

B.10 CheapestInsertion

The "CheapestInsertion" class finds the best location in a route to insert a customer location.