Optimizing the Home Care Service

Project 2 IT3708 Spring 2025

Groups Allowed? Groups are allowed, up to 4 students. You can keep the same group from Project1 and for the future project. This is up to your preference. All students in the group must be present during the demo day. All members of a group should sign up for the same time slot on demo day. For this project, you will do a conference-style presentation as described at the end of the document.
Guidance: Questions will be answered during lab hours or on the Blackboard forum.
Deadline: March 10th 2025

Introduction

In this project, you will implement a genetic algorithm (GA) to solve a simplified version of a real-life optimization problem. It involves creating routes for home-care nurses in order to visit and care for patients while minimizing time spent driving. Solving this problem has a great social impact, as nurses have more time for patient care and spend less time planning.

Visma Resolve is involved as a guest project host. We currently deliver a meta-heuristic solver being used in the daily operations of the home-care services in a growing number of municipalities in Norway. The problem we solve is naturally more complex than the problem you will solve in this project, but the basic nature of the problem is identical. You can read more about us in a later section in this document.

Resolve will host a guest lecture and project kick-off on February 6th. Here, we will tell you more about how we have implemented our algorithm, how we deliver it as a service, and kick off the project. We hope to see you here!

Timeline

Tuesday, February 4th: Project description is available.
Thursday, February 6th (12:15 - 14:00): Kick-off with Visma Resolve @auditorium F2.
February 4th to March 10th: Work on the project.
Deadline, March 10th: Demo Day.

The Home-Care Vehicle Routing Problem Variant

Background

Vehicle routing problems (VRPs) are classical combinatorial optimization problems that have received much attention in recent years [3,4,5]. This is due to their computational complexity, wide applicability, and economic importance. VRP formulations are used to model an extremely broad range of issues in many application fields: transportation, supply chain management, production planning, and telecommunication, to name a few. In a large number of practical situations, additional or revised constraints are defined to satisfy real-life scenarios, yielding many different variants of the VRP.

A typical VRP can be stated as follows [3,4,5]:

- A set of geographically dispersed customers with known demands are to be serviced by a (often homogenous) fleet of vehicles with limited capacity.
- Each customer is to be fully serviced exactly once.
- Several vehicles are assigned to a depot.
- A vehicle starts at a depot and has to return to the same depot.
- The objective is to minimize or maximize some goal. An example objective is to minimize the total time spent travelling for all vehicles.

In this project, you will solve a vehicle routing problem with time windows. This can be applied to many logistic cases, but in this project we will adapt it to the routing of home-care nurses to patients. This way, it becomes a simplified version of the problem Visma Resolve solves in the home-care service in Norwegian municipalities.

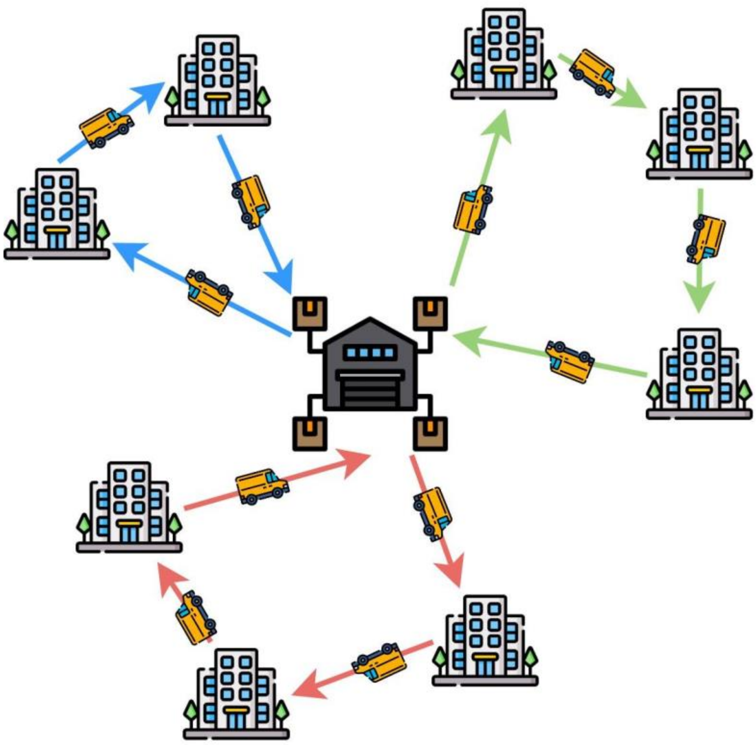

Figure 1: Hypothetical VRP instance with one depot, three vehicles and eight customers.

Problem Description

The following paragraphs describe the problem constraints and objective in detail. These details will be crucial to your implementation of the problem, so please read them carefully.

A set of *nurses* (equivalent to vehicles) cares for a set of *patients* (equivalent to customers). There is one *depot*, and each patient must be visited exactly once. Each nurse starts at the depot, visits an arbitrary subset of the patients to provide care, and returns to the depot. This is called a *route*. Each nurse must be back at the depot before the *return time*.

Each nurse has a *capacity* and each patient has a *demand*. The total demand of the patients in a nurse's route must be less than or equal to their capacity. All nurses in a problem instance have the same capacity, but the patients' demands vary. Practically, the demand can be viewed as the strain on the nurse to perform the patient care. The nurse's capacity is then how much strain a nurse can handle during a route.

Each patient has a *care time* and a *time window*. The care time is the time it takes to care for the patient. The time window has a start and end time. The *start time* defines the earliest time a nurse can start caring for the patient. The *end time* defines the latest time a nurse must be finished with the care. The time windows are strict, meaning that a task cannot begin before the start time and must be completed before the end time. If a nurse arrives at a patient before the start time, they *wait* until the start time before starting the patient care.

The above can be summarized with the following constraints:

1. Each route starts at the depot on time 0.
2. Each route ends at the depot and must arrive before the given depot return time.
3. The total demand on a route must be less than or equal to the nurse's capacity.
4. Each patient visit on a route must be within the respective time windows.
5. Each patient is visited on exactly one route.

The objective is to minimize the *total travel time*, i.e., the sum of the *travel time* of all routes. Note that this *does not* include the care time or the potential waiting time!

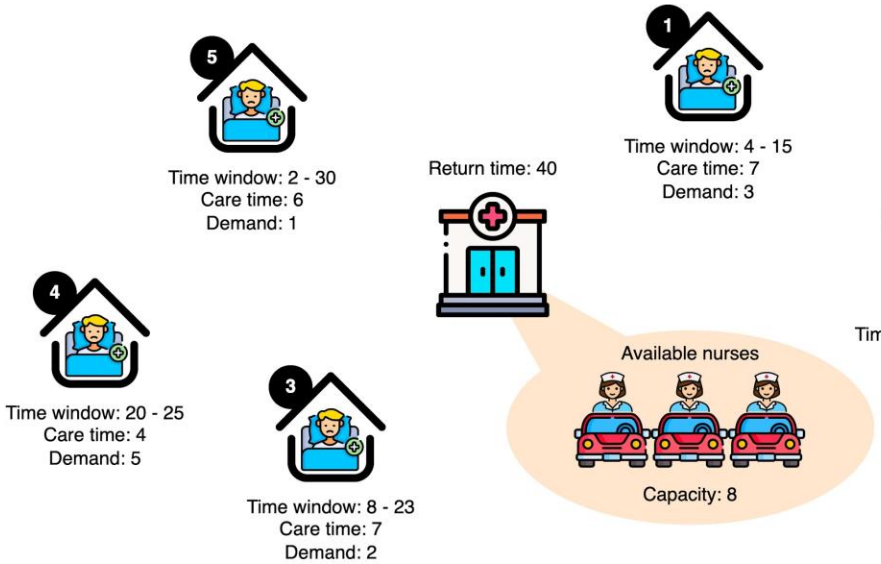

Figure 2: Example of a problem.

Solution Example

A solution to the problem is a route for each nurse. There are many ways to represent a solution. Figure 3 shows a solution example to the problem in Figure 2.

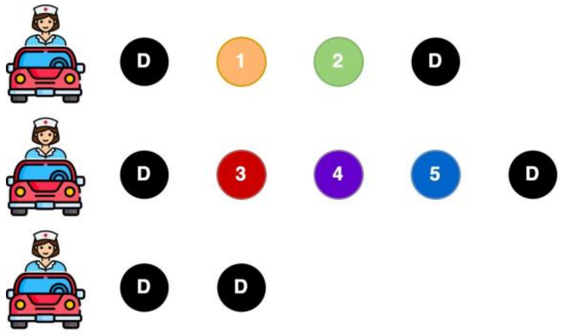

Figure 3: Example of a solution to the problem introduced in Figure 2, not using nurse 3.

This can be translated to a solution representation you can use in your implementation. For example, you could use a list of lists, where each inner list is a route and its elements are the patients that are visited on the route. An empty inner list would represent a nurse not being used. You don't need to include the depot in the inner lists, as all routes start and end there.

Figure 4 breaks down the calculation of a route's patient visit times and duration and how these should be matched with the time windows and return time. The travel and care times are given by the problem instance. Recall that the wait time is only greater than zero if a nurse arrives at a patient before the start time. Note that a route always starts at time 0.

It is important that you remember to distinguish between a route's *duration* and *travel time*. The duration ("Time" in the orange boxes in Figure 4) includes travel, wait, and care time and should be the time that you match with the time windows and return time. The travel time naturally only includes the time spent travelling between patients, and this is the objective value of a solution. Don't forget to include the travel times from and to the depot!

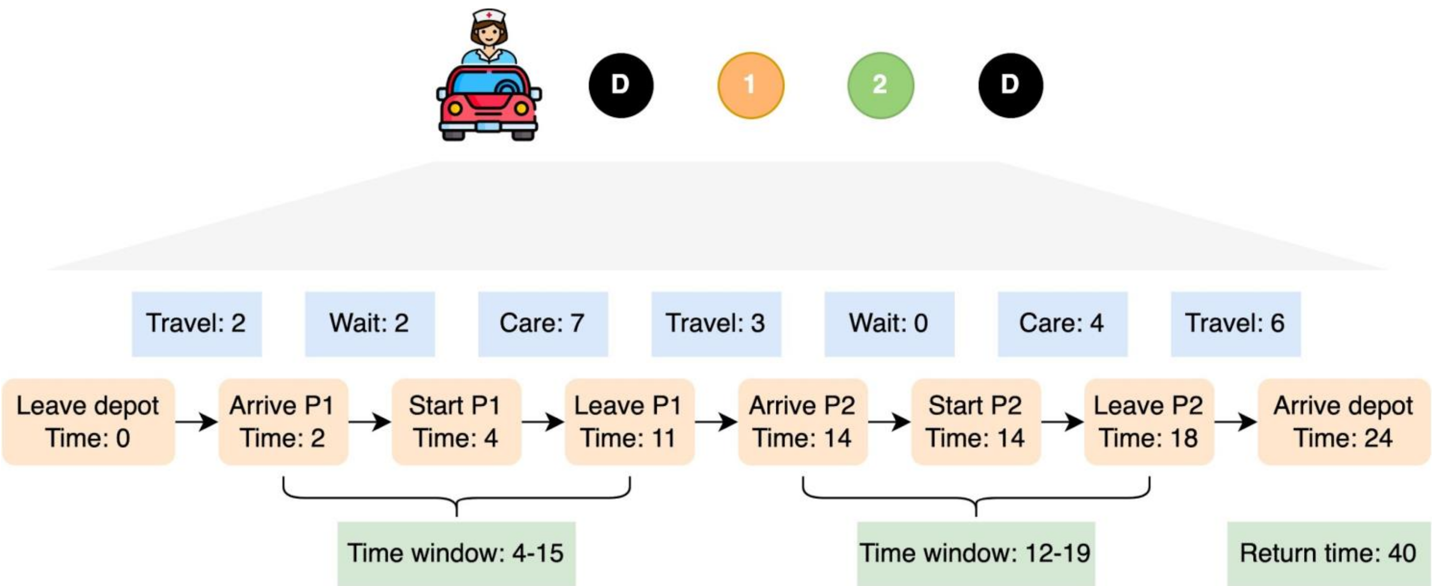

Figure 4: Breakdown of route checkpoints (orange boxes). The travel, wait, and care times are written in the blue boxes, while the green boxes show the time windows and return time.

Description of the Problem Instances

You are given ten instances of the problem outlined in the Problem Description section. The instances are provided as JSON files containing the following fields:

- `instance_name`: The name of the instance.
- `nbr_nurses`: The number of nurses available.
- `capacity_nurse`: The capacity of a nurse.
- `benchmark`: The benchmark objective value for the instance.
- `depot`:
  - `return_time`: The time all nurses must be back at the depot.
  - `x_coord`: The x-coordinate of the depot's location (only needed for plot).
  - `y_coord`: The y-coordinate of the depot's location (only needed for plot).
- `patients`: Patient-related information for each patient.
  - `demand`: The patient's demand (strain on nurse performing care).
  - `start_time`: The start time of the patient's time window.
  - `end_time`: The end time of the patient's time window.
  - `care_time`: The time it takes for a nurse to care for the patient.
  - `x_coord`: The x-coordinate of the patient's location (only needed for plot).
  - `y_coord`: The y-coordinate of the patient's location (only needed for plot).
- `travel_times`: The travel time matrix, see details below.

You are advised to test your genetic algorithm implementation on **all** of these instances as you are developing it. The difficulty of the provided instances reflects the difficulty of the unseen instances given on the demo.

The travel time matrix contains travel times between all nodes in the problem instance. This means that it contains the travel times between depot and all the patients, and the travel times from each patient to all other patients. The depot is the first row/column, patient 1 the second row/column, and so on. For example, index (2, 3) and (3, 2) in Figure 5 gives the distance between patient 1 and 2. The travel times are floats, so **do not** round them.

$$
\begin{array}{r c}
 & \begin{array}{cccccc} \text{Depot} & \text{Patient 1} & \text{Patient 2} & \text{Patient 3} & \text{Patient 4} & \text{Patient 5} \end{array} \\
\begin{array}{r} \text{Depot} \\ \text{Patient 1} \\ \text{Patient 2} \\ \text{Patient 3} \\ \text{Patient 4} \\ \text{Patient 5} \end{array} &
\begin{bmatrix} 0 & 2 & 6 & 2 & 7 & 1 \\ 2 & 0 & 3 & 9 & 2 & 3 \\ 6 & 3 & 0 & 1 & 5 & 2 \\ 2 & 9 & 1 & 0 & 7 & 3 \\ 7 & 2 & 5 & 7 & 0 & 4 \\ 1 & 3 & 2 & 3 & 4 & 0 \end{bmatrix}
\end{array}
$$

Figure 5: Example of a travel time matrix. Some of the values were used in Figure 2.

Solution Method: Genetic Algorithm

The problem you will solve in this project is NP-hard, which roughly means that an efficient (polynomial time) algorithm for solving all problem instances to optimality is likely unavailable. For this reason, meta-heuristic algorithms are good choices to solve the problem. In this project, you will implement a genetic algorithm (GA) [1,2,3], as you did in Project 1. It might also be beneficial to test whether elitism, local search, and/or parallelization gives better solutions faster, and perhaps also consult the other pointers among the references and lecture materials if needed.

To Do

Mandatory

1. Implement a genetic algorithm that solves the home-care routing problem.
Implement a GA that reads a problem instance and finds increasingly good solutions to that instance. It should have some logging as it runs, and output the best solution found according to the format described in the next section.

It is **not** recommended to implement this algorithm in Python. We encourage everyone to use a faster language like Julia, Java, C++ or Rust, as navigating the search space can be computationally expensive.

To test your GA during development, we have provided 10 problem instances and *objective benchmarks*. The benchmarks are objective values that should be obtainable, even though they might be tough to reach. They are provided to give you an idea of what “good” solutions are to each instance.

2. Implement code for plotting the solutions.
You must also write code for plotting the solutions of your algorithm. See the next section for details on the plot format. You are free to use any plotting library you want. Python has many easy-to-use plotting tools like matplotlib, plotly, and seaborn. If you use these, you would implement the GA in one of the above-mentioned languages (or a similar, fast one), and do the plotting in Python.

Optional

The optional part of the project is to participate in the competition (description below) hosted by Visma Resolve. This has nothing to do with your grade, the point system, or the demo, and is only meant to be fun and motivational. There will be nice prizes to win :)

Demo

The Demo is virtually divided into two parts:

- Demo part 1: Report and presentation of your work
- Demo part 2: Performance on the test datasets

You can get a **total of 35 points for this assignment**, where max 10 points are available from the report and presentation (Demo part 1) and max 25 points are available from the performance achieved on the test data (Demo part 2). The whole Demo should take max 10 minutes per each team/group.
Everything demo- and score-related will be handled by the course staff and the student assistants.

Demo part 1: Report and presentation (10p)

You will prepare a conference-style presentation for this. During or after the presentation, you might be asked questions regarding the behaviour and implementation of your code. In these discussions, you may also be asked to explain certain concepts from the theory of this course. The exact questions will not be given in advance, but typical questions could be:

- Describe and show different parts of the code, such as the chromosome representation, mutation operation(s), and crossover operation(s). Also, be ready to discuss alternatives and the effects of your choices for example with regards to the exploration-exploitation trade-off.
- Explain related theory concepts from the course's curriculum.

Demo part 2: Performance on the test datasets (25p)

In this part of the demo, you will show us your results on the test data and comparison to the benchmarks. The test data will be given to you *a few days before the Demo Day*. You will run your implemented GA on the 3 given test datasets, save and present your results during your Demo day time. You might get some questions here as well.

The point distribution is as follows: Testing three problem instances (25p ≈ 8.33p x 3). For each problem instance, you can get full or partial scores as follows:

- If your value is within 5% of the benchmark value, or better than the benchmark value, you will get 8.33 points (full score).
- If your value is within 10% of the benchmark value, you will get 6.5 points.
- If your value is within 20% of the benchmark value, you will get 4 points.
- If your value is within 30% of the benchmark value, you will get 2 points.
- Otherwise, you will get 0 points.

A plot and a solution text output for the best solution your GA finds must be shown for each of the test instances. The next section describes the expected format of the plot and solution output.

Expected Formats

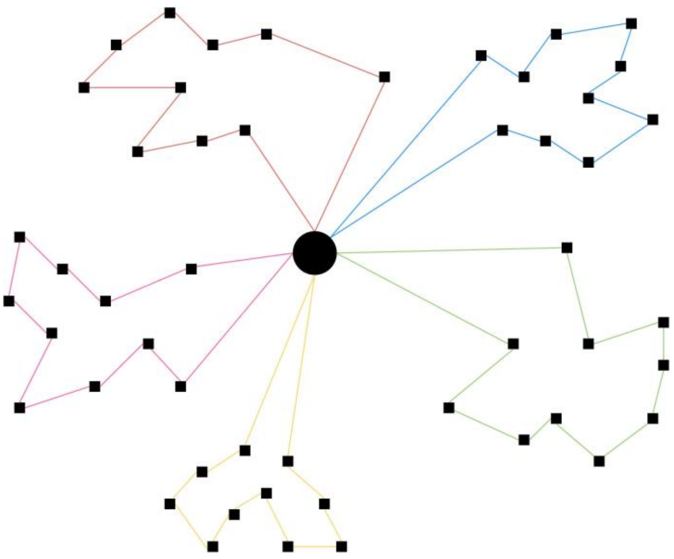

Figure 6: Expected plot format. The most important thing is that you make sure to use different colors for each route. The plot must only be shown for the best-found solution.

Nurse capacity

Depot return time

Nurse 1 (N1)	Route duration N1	Covered demand N1	Patient sequence N1
Nurse 2 (N2)	Route duration N2	Covered demand N2	Patient sequence N2
⋮	⋮	⋮	⋮
Nurse N (NN)	Route duration NN	Covered demand NN	Patient sequence NN

Objective value (total duration)

Figure 7: Expected solution output format. Can be written to the console or saved as a file. The solution output needs only to be shown for the best-found solution.

Nurse capacity: 30

Depot return time: 100

Nurse 1	80.63	27	D (0) → 17 (5.14-9.14) [4-15] → ··· → D (80.63)
⋮	⋮	⋮	⋮

Objective value (total duration): 593.17

Figure 8: Example of a solution output. Patient 17 is visited (among others, not specified in the example), the visit time is shown within (), and the time window is shown within [].

Delivery Method and Deadline

You should hand in **2 deliverables in Blackboard** (a report, and your code), and give a **brief presentation** during the **demo day**:

- **Code**: you should deliver a zip file with your code on BlackBoard.
- **Report**: you should deliver a group report (no more than 8 pages), summarizing the methodology, your experimental setup, the results of your experiments, your findings and your conclusions. Remember to include your references and **cite them properly**.

The submission system will be closed on **March 10th 2025**. The **demo day** for Project 2 is also **March 10th 2025**. A signup schedule will be announced one week before. Please follow Blackboard for details. **Every student must submit a copy of their jointly developed code and report. You must attend the demo individually on the scheduled demo date. No early or late submission or demo will be entertained except in case of an emergency.**

Instances and Benchmarks

Table 1 presents the benchmark objective values (travel time) for the training instances found in the zip file on Blackboard. Three similar instances will be provided on the demo day. To get all five points for an instance, the objective value of the best found solution must be within 5% of the benchmark.

Note that these benchmark values are not necessarily the optimal known values for each problem, but rather provide a benchmark that you might be able to reach. If they are set too strict, the course staff may adjust them after the demo.

Instance	Benchmark	5%	10%	20%	30%
train_0	827	868	910	992	1075
train_1	589	618	648	707	766
train_2	1258	1321	1384	1510	1635
train_3	1132	1189	1245	1358	1472
train_4	1261	1324	1387	1513	1639
train_5	1092	1147	1201	1310	1420
train_6	924	970	1016	1109	1201
train_7	870	914	957	1044	1131
train_8	731	768	804	877	950
train_9	855	898	941	1026	1112

Table 1: Benchmark objective values for all training instances.

Validation and Competition by Resolve

We have set up a service in order for you to (1) validate that you have implemented the constraints and objective calculation correctly, and (2) participate in a competition as you develop your GA. The service is available at https://it3708.resolve.visma.com/.

Validation

In the validation part of the service, you can paste in a solution to an instance that you choose and get as output whether the solution is valid, i.e. not violating any constraints, and the objective value of the solution. In order for everything to work, you must follow the expected solution format. This is a list of lists where each inner list is a nurse's route. The elements are the patient IDs of the instance.

Competition

In order to increase the motivation of working with this assignment, we are hosting a fun and friendly competition. In the competition, you can use your algorithm to compete against your classmates and Resolve's Operational Route Planner (ORP). In order to compete, run your algorithm on instance ***train_9*** on your local machine and submit the solution on the website. On the website you will see a highscore list showing other student's scores. You choose your username yourself, and you can submit a solution as many times as you want. The highscore board will only be updated when you beat your previous best highscore.

The student or a group with the best score wins a gift card for komplett.no worth **1000 NOK**! If there are ties, the two or more winners will get a new instance and potentially a cap on the run time. This will go on until we have one student or a group as the winner.

KOMPLETT®.no

About Visma Resolve

In short, Visma is the number one software provider in the Nordics. It is a big organization with more than 14 000 employees and 200 unique software products. It has been growing fast and steady for the previous 20 years, showing no signs of slowing down any time soon.

Resolve is a centralized team in Visma consisting of optimization, machine learning, and infrastructure experts. We love challenging problems, and we love to solve them. We have customers who struggle with some of the most challenging and complex mission-critical problems every single day. Our job is to find those customers, identify their problems and solve them. By dedicating ourselves to the latest research within artificial intelligence, particularly machine learning and optimization, we seek to significantly improve the operations of our customers within healthcare, education and the private sector.

By empowering clients with user-friendly, analytical tools in the cloud, we radically transform their manual processes into competitive advantages, enabling them to make powerful and informed decisions.

We believe that the success of applied optimization and machine learning depends on domain knowledge and a deep understanding of the underlying business. That is why we take part in the whole value chain of the project - from business process improvement to mathematical modelling, algorithm construction, full-stack development, and application hosting. We take pride in being a close-knit group of international, curious and devoted tech talents.

To stay updated on the latest technologies and research, we have frequent knowledge sharing and learning sessions. We also love to arrange hackathons to maintain the team's innovative spirit and drive.

Read more about our team on our website.

References

[1] A. E. Eiben and J. E. Smith. “Introduction to Evolutionary Computing,” 2nd Edition, Springer 2015, pages 67 - 70 (permutation representation) & pages 203 - 211 (constraint handling).

[2] D. Simon. “Evolutionary Optimization Algorithms,” Wiley 2013, pages 449 - 478 (combinatorial optimization incl. TSP).

[3] B. Ombuki-Berman and F. T. Hanshar. “Using Genetic Algorithms for Multi-depot Vehicle Routing.” In: F. B. Pereira and J. Tavares (eds). “Bio-inspired Algorithms for the Vehicle Routing Problem.” Studies in Computational Intelligence, vol 161. Springer 2009. https://doi.org/10.1007/978-3-540-85152-3_4

[4] J. R. Montoya-Torres, J. L. Franco, S. N. Isaza, H. F. Jiménez, N. Herazo-Padilla. “A literature review on the vehicle routing problem with multiple depots.” Computers & Industrial Engineering, Volume 79, 2015, pages 115-129, https://doi.org/10.1016/j.cie.2014.10.029.

[5] K. Braekers, K. Ramaekers, I. V. Nieuwenhuyse. “The vehicle routing problem: State of the art classification and review.” Computers & Industrial Engineering, Volume 99, 2016, pages 300-313, https://doi.org/10.1016/j.cie.2015.12.007.