Solution to CVRP problem with the Ant Colony optimization algorithm

Przemyslaw Chojecki
STUDENT NO. 298814

2022.04.04

Report submitted for course
Artificial Intelligence Methods 2 at
The Faculty of Mathematics and Information Science,
Warsaw University of Technology

Abstract

Solution to CVRP problem with
the Ant Colony optimization algorithm

Capacitated Vehicle Routing Problem (CVRP) is a widely encountered problem in business. Transportation costs are a big part of costs for many companies. We consider the version of the problem, where a fixed number of vehicles have to serve several customers distributed on 2D space. Every customer has a fixed order, and every vehicle has a fixed capacity. Moreover, every vehicle has a maximum distance possible to do in a given day, which represents the time of a shift of a specific driver.

In this work, we considered the Ant Colony algorithm to solve this problem. We proposed some modifications to the primary Ant Colony algorithm and compared them with the greedy algorithm.

We propose the Reduced Graph modification, where the driver is only allowed to go to the portion of the closest neighbours of the vertex. We also propose the Divide Graph modification, where the customers are clustered with the 'kmeans' algorithm at the beginning, and then every cluster is solved independently. In most of our tests, we found that the Divided Graph modification was the best of the considered algorithms.

We developed software that can be easily used for solving such problems. We know this is not the definitive solution to the CVRP, but we believe this is a big step towards it.

Keywords: CVRP, Ant Colony, Artificial Intelligence

Contents

1 Context of this Report

This document is the Report of the work for the project in class **Artificial Intelligence Methods 2**. The project consists of the Conspectus, the solution's code, and the Report. The code developed for the project is available at GitHub [2].

1.1 This paper

This Report will describe the results of the experiments on the proposed solution for the CVRP. We will determine the rightness of the hypotheses introduced in the Conspectus. We will also formulate the conclusions on the work and show possible future development paths.

This Report assumes the reader is familiarized with the Conspectus. If it is not the cast, then it is available in [2] in the location:
'MSI2/project1/documents/Conspectus.pdf'.

The most recent versions of this Report is available in [2] in the location:
'MSI2/project1/documents/Report.pdf'.

2 Hypotheses

In this section, the hypotheses for the work are stated as in the Conspectus. Those hypotheses are tested in the work.

1. The **basic version** will faster find better solutions than the Reduced graph and the Divided graph modifications for **smallest** graphs (up to 50 nodes).

2. The **Reduced graph** modification will faster find better solutions than the basic version and the Divided graph modification for **medium** graphs (from 50 to 100 nodes).

3. The **Divided graph** modifications will faster find better solutions than the basic version and the Reduced graph modification for **biggest** graphs (above 100 nodes).

Those hypotheses will be examined in this Report.

3 Methodology

In this section, all information about the methodology of this work is stated.

Datasets

We used two datasets to test the proposed algorithms. Those can be found in the widely-used online repository [1]. Those datasets were used to determine the correctness of the algorithm and for testing the appropriate hypotheses.
From the set **Augerat 1995 - Set A**, four graphs were used:

- A-n32-k05 (32 nodes) - small graph 1
- A-n44-k06 (44 nodes) - small graph 2
- A-n60-k09 (60 nodes) - medium graph 1
- A-n69-k09 (69 nodes) - medium graph 2

From the set **Uchoa et al. 2014**, two graphs were used:

- X-n101-k25.xml (101 nodes) - big graph 1
- X-n120-k6.xml (120 nodes) - big graph 2

Big graph 2 is a unique set because every vertex has the same demand.

Additional parameters for the datasets

The datasets mentioned above were designed for the CVRP without restricting the maximum route for the vehicle or the number of available vehicles. Those parameters of the set were chosen so that the Greedy algorithm had a problem finding the right solution. One can follow the process of selecting those parameters in [2] in
'MSI2/project1/src/JupyterNotebooks/Additional parameters for datasets.ipynb'.

Examination of the distributions

For every problem and algorithm, the calculations were performed **11 times** to examine the differences in distributions of results. We have also compared the median solution for every algorithm. These comparisons can be viewed in Section 5.

Fixed time of optimization

For every case, the algorithm was running approximately **5 minutes**. In that, the results are comparable.

Algorithms

Three versions of the Ant Colony algorithm were compared with themselves and the greedy algorithm. The algorithms are in-depth characterized in the Conspectus.

Parameters

Although it was not the aim of this work, we performed basic parameter tuning. In those small tests, we found the Basic version of the Ant Colony algorithm works best with parameters $\boldsymbol{\alpha = 2, \beta = 3.5}$ and the modifications Reduced and Divided graph works well with $\boldsymbol{\alpha = 1, \beta = 1}$. Nevertheless, more comprehensive tests are needed to determine the best hyperparameters for that task.

The other parameters were used the same for every version of the algorithm, as follow: **starting_pheromone=1**, **Q=1**, **ro=0.9**, **cars_penalty=0.1**. As mentioned before, the parameter **max_time=300** was used, and **max_iter=10000**, but the max_iter parameter had no effect, because the time had passed before the iteration limit.

Basic Ant Colony is different from description in Conspectus

The Basic version of Ant Colony described in the Conspectus turns out to be relatively ineffective for a given CVRP task. We think it was caused by choosing the wrong hyperparameters for the given task. We excluded the possibility that it was caused by the small number of iterations performed because, after about one minute of learning, all the ants were following the same non-optimal path. We renamed this version "Primary" and changed it with two modifications.

The first modification addressed the problem of the tendency of the population to follow the same path. We were inspired by the RIDGE linear estimator, where the matrix $X^T X$ will have an improved condition number by adding the $\lambda \cdot I$ term so that the inverse $(X^T X + \lambda \cdot I)^{-1}$ is more stable. So we did make the transition matrix more stable by adding the $0.01 \cdot E$ term to it, where E is a matrix of ones. Therefore, for every step, the ant has a 1% chance to completely ignore the pheromone and make a move according to the uniform distribution on the possible vertices. This normalization did the trick, and the population of ants was no longer getting stuck in a single solution.

The second modification is the additional heuristic for finding the solution with fewer vehicles. We have noticed that the algorithm has bigger problems serving people close to the warehouse. We think this is happening because the penalty for going back to the warehouse and then serving the next person is too small for people close to the warehouse. We de-

cided to add the new parameter: **cars penalty**. This works as follows: if the ant finds the solution with the correct number of cars, meaning smaller or equal to the goal, the pheromone is distributed normally. If ant found the solution with a bigger number of cars, then the pheromone it distributes is smaller. The penalty is proportional to the % of redundant cars, meaning that if the ant_1 uses 2 times more vehicles than the goal and the ant_2 uses 6 times more vehicles than the goal, then the penalty for ant_1 will be 3 times smaller than for ant_2. The bigger the **cars penalty** parameter, the bigger the penalty. If the ant finds the solution with more than $\frac{\text{number of cars}}{\text{cars penalty}}$ cars, then no pheromone is distributed. As mentioned in the previous paragraph, we used the value **cars penalty = 0.1** for this parameter.

The Primary Ant Colony algorithm with those two additional modifications is in this Report named Basic. The other two modifications, namely Reduced and Divided, were implemented and analyzed as in the Conspectus.

4 How to reproduce the results

As mentioned in the Section 1, the code developed for this project is available in GitHub repository [2]. There, one can follow the instructions in the ‘README.md‘ file.

One has to download the repository. Next, open the folder ‘MSI2/project1/src/AntColony/‘ in the terminal and type the ‘pip install .‘ commend. This command will install the package ‘AntColony‘. Then, one can run the scripts in the ‘MSI2/project1/src/scripts‘ folder and the Jupyter Notebooks in the ‘MSI2/project1/src/JupyterNotebooks‘ folder.

We performed all the computations with the seed for the random number generator. We used the seed ‘1234‘ in all of the places. This seed was chosen arbitrarily and did not change during development.

Note that the script was developed to run for a fixed time. To be precise, for 5 minutes for every optimization on every problem on every iteration. There were six problems to be solved, 3 Ant Colony algorithms and every one of them was solved 11 times. This sums up to 16.5 hours of computing time. We performed the calculations on MacBook Air 2017. If one will perform the same steps as we did, one will also have the fixed 16.5 hours of computing time on his machine, but the number of performed iterations on the problems will be different. In every log file generated by the script, there is written the number of iterations performed.

5 Analysis of the algorithms

In this section, we present an in-depth analysis of the results of algorithms. We performed 11 optimizations on every one of 6 sets from Section 3 and every algorithm of interest.

For every dataset, we present two plots. One is the distribution of all the results. The other is the graph visualizing the median solution for every algorithm.

We believe that this study will enlarge the reader a more in-depth understanding of the features of the algorithms.

The greedy algorithm is, for obvious reasons, the fastest algorithm. This is also the only one considered that is non-random. The Ant Colony algorithm is known to find better solutions the longer they evaluate. We restricted the search up to five minutes for every optimization. In that, the results are comparable between different algorithms.

Distribution of solutions

On every graph, the point represents one process of learning the algorithm. The points are grouped on the x-axis by the algorithm they represent. On the y-axis, they are placed at the value of the path the algorithm found at the end of the learning.

Median solution

The median solution from the previous graph is selected and plotted. On every graph, the points represent the customers, and the lines represent the path the algorithm chose for the supplier to take. When the algorithm decides the same supplier will serve the customers, those points are coloured by the same colour. The numbers in the points represent the load for a customer. A point with a zero load is a warehouse.

Recall from the Conspectus that we suspected that the Basic Ant Colony algorithm would work best for small sets because it is unrestricted to a part of nodes. Therefore, better possible solutions are possible for it to reach. This is also true for a bigger set. However, it is less likely that the algorithm will find those for bigger sets. Therefore we suspected that the Restricted and the Divided would work better for those.

5.1 Small set

Small sets were defined as those up to 50 nodes.

Small set 1; 32 nodes

The results of optimization on this set can be seen in Figure 1 and 2.

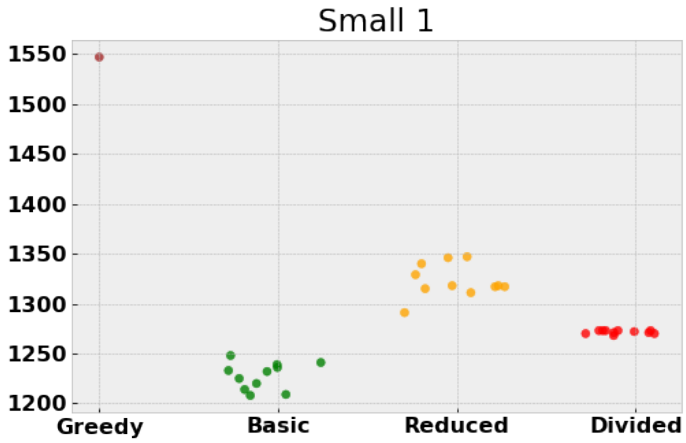

Figure 1: Small set 1; comparison of costs of all results of algorithms

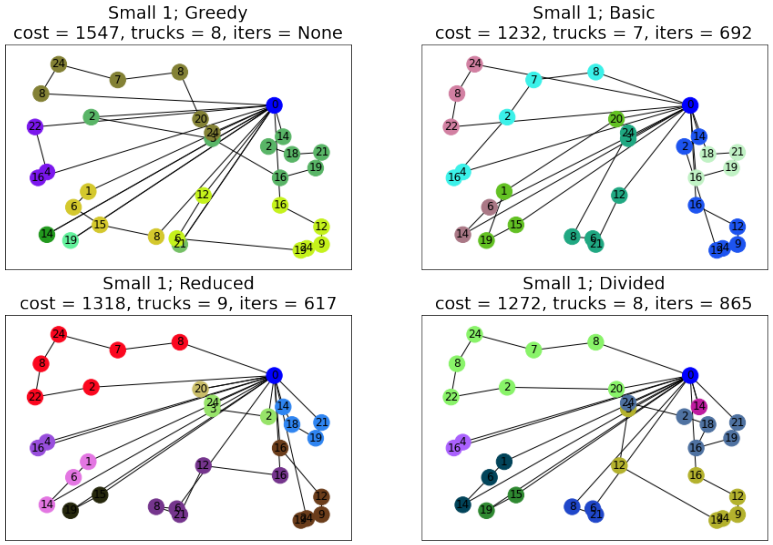

Figure 2: Small set 1; comparison of median results of algorithms

For this set, the results are as expected. The algorithm's Basic version always found a better solution than other algorithms.

the distribution of the Divided is also interesting - they have minimal variance. Every time the algorithm runs, every cluster finds an optimal or near-optimal solution.

Every found solution was better than Greedy's one. This dataset is the only instance of this happening. Closer analysis shows that this

behaviour is caused by the nodes in the bottom-left corner, which every other algorithm found a better way of solving.

The Basic algorithm found the solution with only seven vehicles, as required by the set. Take note that the plotted median solution for Divided cannot be easily improved to have seven vehicles (looks like the red node with "14" can be attached to the purple nodes with "4" and "16". However, this would result with violation of s_max parameter).

However, one can see that the plotted median result of Reduced can be easily improved. The lime nodes of "2", "3", and "24" can be combined with the pale yellow "20". This is the recurring pattern in this analysis that we will not mention more. We encourage the reader to try to find the improvement in the plotted median patterns. It is hard for all the Greedies but relatively easy for other algorithm's solutions, especially for larger graphs.

Small set 2; 44 nodes

The results of optimization on this set can be seen in Figure 3 and 4.

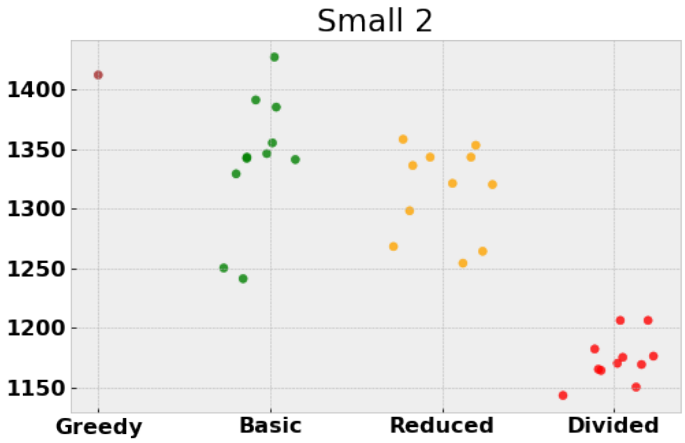

Figure 3: Small set 2; comparison of costs of all results of algorithms

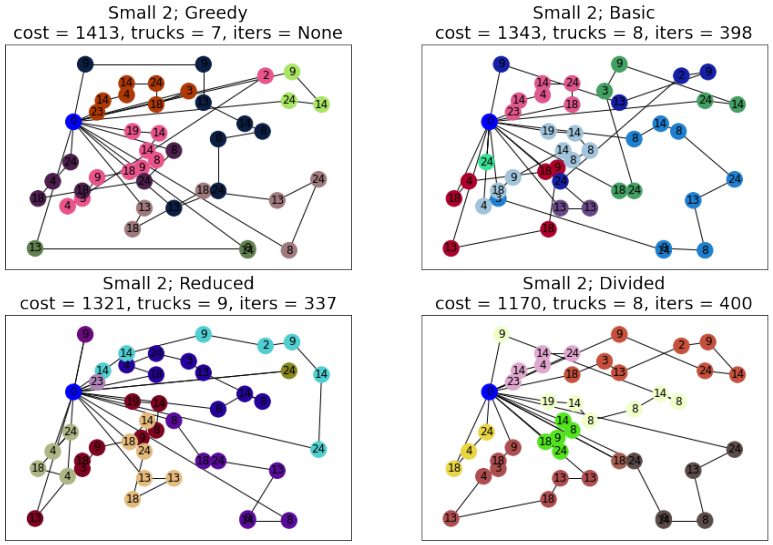

Figure 4: Small set 2; comparison of results of algorithms

For this set, the results are not as expected. The dataset was so complicated that the Divided modification did the best for it. The worst solution found by Divided was better than the best found by any other algorithm.

What is more, there was a single instance of Basic who did worse than Greedy. We were surprised this happened for such a small graph. However, one can see the results of Basic have significant variance. Much

bigger than the Reduced and Divided, and much bigger than the previous graph. We think this is a sign that the considered graph has a lot of "local minimas".

5.2 Medium set

Medium sets were defined as those from 50 to 100 nodes.

Medium set 1; 60 nodes
The results of optimization on this set can be seen in Figure 5 and 6.

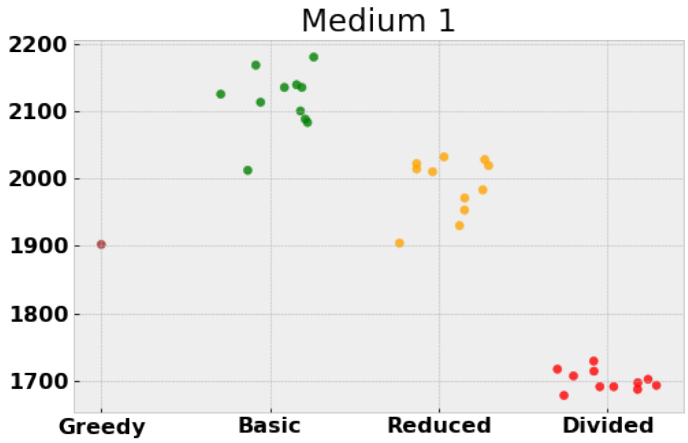

Figure 5: Medium set 1; comparison of costs of all results of algorithms

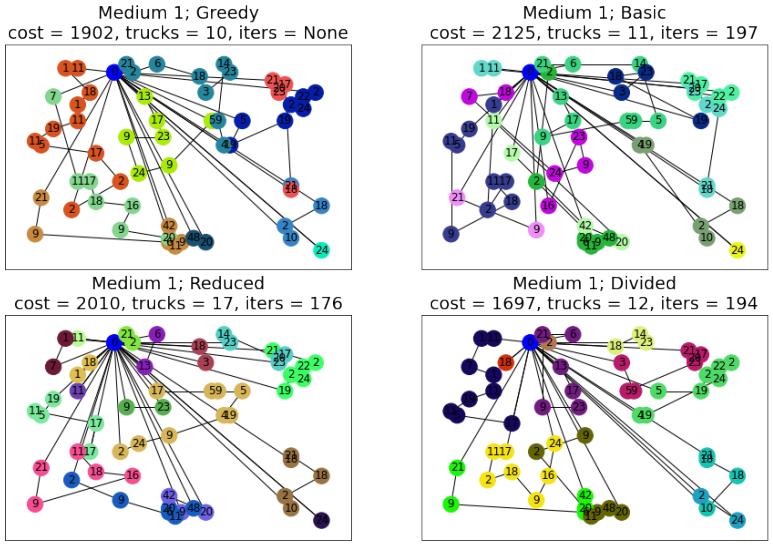

Figure 6: Medium set 1; comparison of median results of algorithms

We can see that this set was big and complicated enough that the Divided modification was the only one better than the Greedy. This is the behaviour we were expecting for big graphs.

The inspection of the plot of the medium result can give us the intuition that the Divided considers more basic patterns than the other algorithms.

After analyzing the plots, one can suspect the Basic and Reduced

methods did not reach their plateau. This is the observation that needs more data to make conclusions about. Similar thoughts show up after the larger graphs' analysis, so we will not mention them.

Medium set 2; 69 nodes
The results of optimization on this set can be seen in Figure 7 and 8.

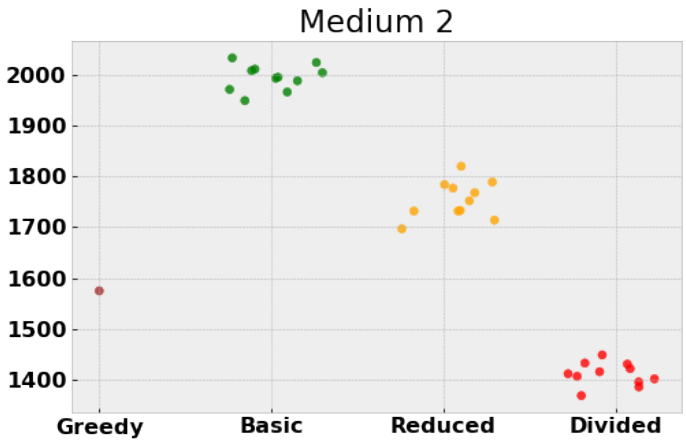

Figure 7: Medium set 2; comparison of costs of all results of algorithms

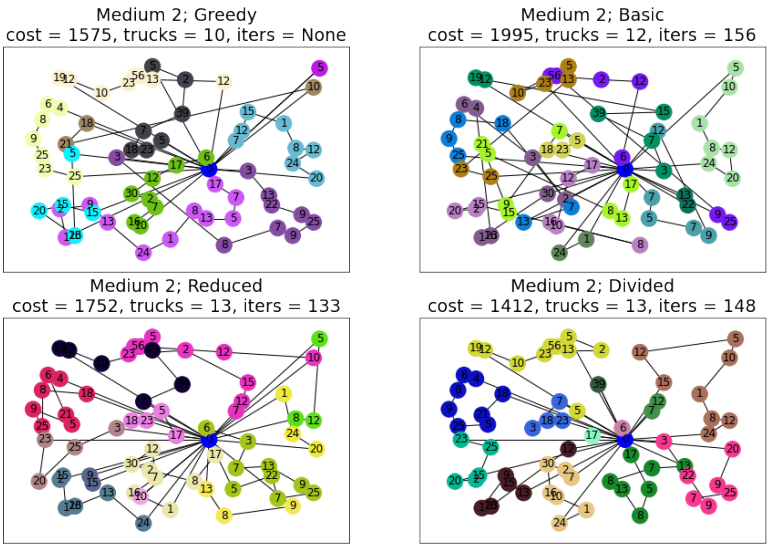

Figure 8: Medium set 2; comparison of median results of algorithms

The results for this set are similar to the previous one. However, after analyzing the median plot, the difference between the Basic and the Reduced seams is way more significant. The ants in the Basic's median solution go back and forth through the whole map, while in the Reduced, they are more local-oriented. This property is not surprising - in fact, this was the desired behaviour and the inspiration for the Reduced modification.

5.3 Big sets

Big sets were defined as those with more than 100 nodes.

Big set 1; 101 nodes
The results of optimization on this set can be seen in Figure 9 and 10.

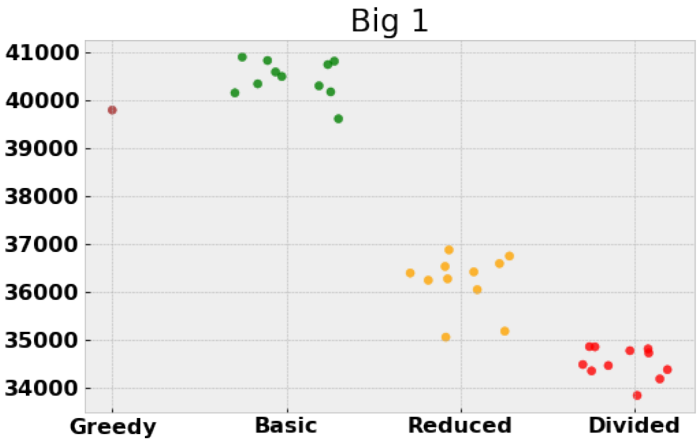

Figure 9: Big set 1; comparison of costs of all results of algorithms

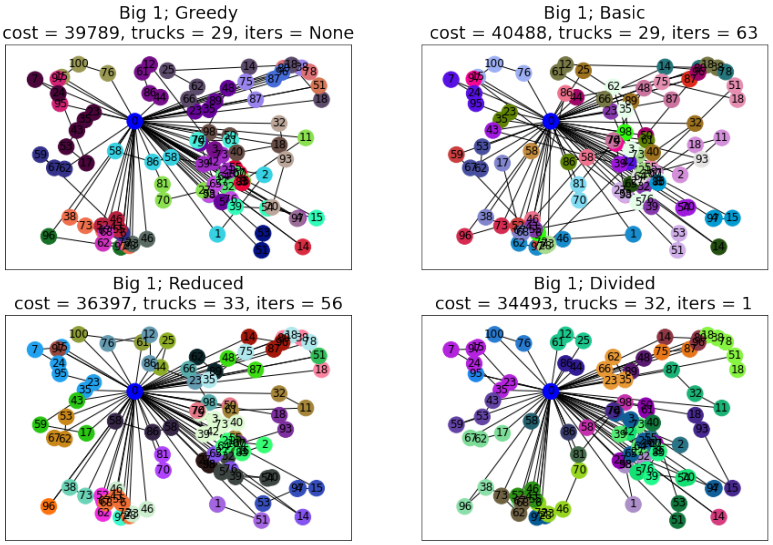

Figure 10: Big set 1; comparison of median results of algorithms

The results of this dataset correspond to the third hypothesis. The Divided modification is way better than any other method. What is more, one can see that the note in the title of Divided's median graph says "iters=1". This is because there was only one iteration for the biggest cluster of nodes (the one in the bottom-right corner). One can see in the figure that other parts of this image are solved better than this big cluster. This suggests that if this part got more time to find a better

solution, it would find a way better. Therefore the overall leadership of the Divided modification will be even more significant.

Big set 2; 120 nodes
The results of optimization on this set can be seen in Figure 11 and 12.

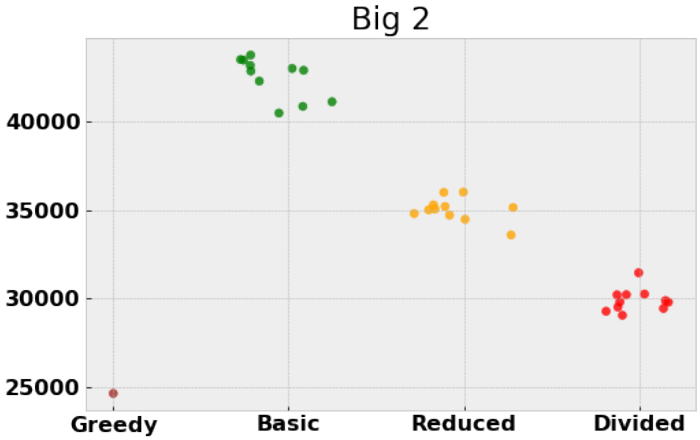

Figure 11: Big set 2; comparison of costs of all results of algorithms

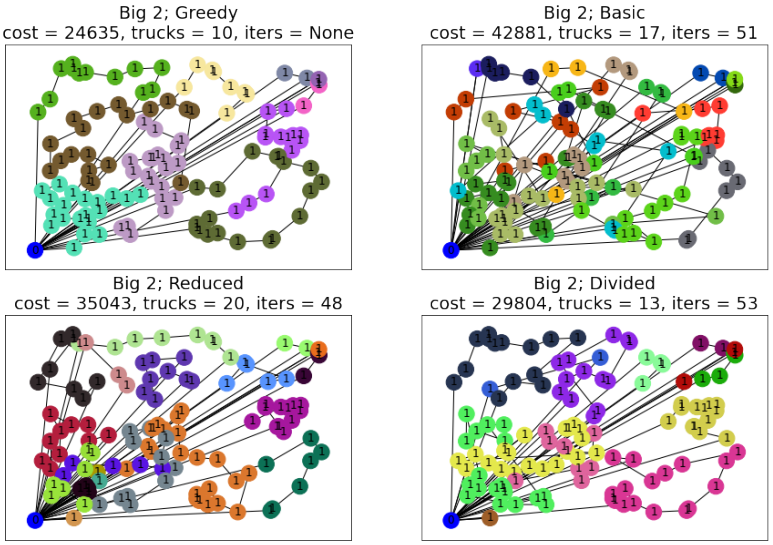

Figure 12: Big set 2; comparison of median results of algorithms

As mentioned before, this set is interesting because all the nodes have

the same order. This simplification makes this problem way more manageable for the Greedy algorithm. It does not surprise that Greedy easily outperforms other algorithms. On the graph of median solutions, one can see that the Greedy is not doing any long jumps. This is opposed to the Greedy behaviour in previous datasets.

One can see that the median solution of Divided makes way fewer long jumps than the other Ant Colony algorithms. However, still, the Greedy does not make any of those.

6 Conclusions and discussion

We see that the behaviour of those algorithms is significantly different depending on the dataset. Sometimes the algorithms performed according to the hypothesis and our intuition, sometimes counter-intuitive.

One sure thing, algorithms did not hit their plateau in 5 minutes. More computational power will provide significantly better solutions. There even was a single cluster for the Big 1 dataset, where only a single iteration was performed with Divided modification.

One exciting behaviour that can be seen is the dataset Big 2, where the CVRP was oversimplified by setting all the orders the same. With such a simplification, the Greedy algorithm did an excellent job of finding the solution. The simplification may be the key to the problem. This slogan inspired the Divided modification that was the best performing for most problems.

6.1 Reflections on the approach and its efficiency

We cannot hide that most of the plotted median solutions made by the algorithms are visually worse than the solution made by a human in the given 5 minutes. We wanted those algorithms to find the solutions to that complicated problem. However, in light of the data provided here, the developed software cannot do so. However, do not forget that this was just our first try at this approach. In Section 6.2 we explore possible improvements to the software that we believe will significantly close us up to this goal.

We think the algorithms can be successfully used in the human-in-the-loop setting. With such a setting, the human could get the results of those algorithms and point out the noticeable improvements. Alternatively, the appropriate software can be used on the output of the algorithms, such as 2-OPT or 3-OPT correction.

Nevertheless, we think we did a pretty good job developing the software for solving the CVRP. We did so especially considering the behaviour of the Primary algorithm. The two modifications described in **Basic Ant Colony is different from description in Conspectus** paragraph of Section 3 were outstanding improvements for the algorithm. What is more, our intuition for the Reduced and Divided modification agreed with the obtained results.

6.2 Possible improvements

As mentioned in the previous paragraph, the software developed during this project can be improved for better results. Also, the developed

software can be used to make better and more in-depth analyses. We propose some such improvements:

1. Usage of bigger computing power
2. Search for better hyperparameters
3. Sample with more than 11 examples
4. Other datasets and other s_max and max_cars parameters of the used sets - including the search for the rule, when the Reduced modification is the best of the four algorithms
5. More in-depth analysis of the differences between Primary and Basic algorithm
6. The software can save not only the lowest-cost solution overall but also the lowest-cost solution for every number of used vehicles

6.3 Personal lessons learned

During the development of the software and analyzing the data, we made a significant improvement as the developers and the computer analyst. We list things that this project helped us improve:

1. Greedy algorithm is sometimes a useful solution. It is always an excellent reference point.
2. Optimal parameters are highly dependent on the dataset
3. Sometimes, it is better to make a less ambitious plan, especially when there are tight deadlines.
4. Setting the constant time of the calculations increases the variance of the results - the computer sometimes takes less and sometimes more time to perform the calculations, and it is hard to control
5. Python is slow
6. Development of private Python package is surprisingly easy

References

[1] J.E. Mendoza. VRP-REP: the vehicle routing community repository. http://www.vrp-rep.org. [Online; accessed 14-03-2022].

[2] Chojecki Przemysław. GitHub page with the code for this project. https://github.com/PrzeChoj/MSI2/. [Online; accessed 14-03-2022].