FACULTY OF ENGINEERING OF THE UNIVERSITY OF PORTO



Self-adaptive optimization algorithms using machine learning for the Capacitated Vehicle Routing Problem

André João Pereira Alves de Sousa

MASTER THESIS PERFORMED IN THE SCOPE OF MASTER IN ELECTRICAL AND COMPUTERS ENGINEERING
MAJOR IN AUTOMATION

Supervisor: Luís Filipe Ribeiro dos Santos Guimarães

July 22, 2020

Self-adaptive optimization algorithms using machine learning for the Capacitated Vehicle Routing Problem

André João Pereira Alves de Sousa

MASTER THESIS PERFORMED IN THE SCOPE OF MASTER IN

ELECTRICAL AND COMPUTERS ENGINEERING
MAJOR IN AUTOMATION

July 22, 2020

Abstract

Machine Learning is believed to be of great assistance in the resolution of optimization problems. Even with the current breakthroughs in technology researchers keep identifying new complex problems that provide difficulties to commercial optimization solvers. This thesis project aims to develop a new algorithm that can compete with state-of-the-art solvers in the resolution of the Capacitated Vehicle Routing Problem.

Based on the literature review on machine learning models and the Vehicle Routing Problem a solver, a feature extractor and a regression technique were implemented and executed. Afterwards the analysis of the data obtained provided insights on what were the best available heuristics and where the developed algorithm could be framed. With this we conclude that not only it is possible to merge machine learning models with optimization methods but the developed algorithm is able to compete with the state-of-the-art heuristics.

Acknowledgments

I would like to thank to Professor Luís Filipe Ribeiro dos Santos Guimarães for the opporunity to develop this project and for all the support provided during the development of this dissertation.

To Carlos Carvalho my eternal colleague a great thank you for the entire journey of the university years and the help I needed in the dire hours.

I would also like to thank to my family for the help provided during the final days of this dissertation and for the support given over the years.

Finally, but not least a heartfelt thanks to Inês for always being there for me and always trying for me to improve my work.

Contents

1 Introduction **1**
- 1.1 Layout . . . 1
- 1.2 Context and Motivation . . . 1
- 1.3 Why Optimization and Machine Learning . . . 2
- 1.4 Objectives . . . 3

2 Literature Review **5**
- 2.1 Exact Optimization vs Approximate Optimization . . . 5
  - 2.1.1 Exact Algorithms . . . 5
  - 2.1.2 Constructive Heuristics . . . 7
  - 2.1.3 Meta-heuristics . . . 8
- 2.2 Machine Learning . . . 10
  - 2.2.1 Supervised Learning . . . 10
  - 2.2.2 Unsupervised Learning . . . 11
  - 2.2.3 Reinforcement Learning . . . 11
- 2.3 Optimization and Machine Learning . . . 12
- 2.4 Machine Learning & CVRP . . . 12

3 Problem Description **13**
- 3.1 Formal Description . . . 13
- 3.2 Mathematical Model . . . 14

4 Methodology **17**
- 4.1 Project Model . . . 17
  - 4.1.1 Data Model . . . 18
- 4.2 Implementation Instruments . . . 18
  - 4.2.1 Constructive Heuristics . . . 19
- 4.3 SOAM Development . . . 19
  - 4.3.1 Data selection and parsing . . . 19
  - 4.3.2 Instance Solver . . . 19
  - 4.3.3 Feature Extractor . . . 22
  - 4.3.4 Predicting Heuristic Performance . . . 25
  - 4.3.5 Verification and Validation . . . 26

5 Computational Experiments **29**
- 5.1 Data sets . . . 29
  - 5.1.1 Set A . . . 29
  - 5.1.2 Set B . . . 30
  - 5.1.3 Set E . . . 31
  - 5.1.4 Set F . . . 31
  - 5.1.5 Set M . . . 31
  - 5.1.6 Set P . . . 31
  - 5.1.7 Set Golden . . . 33
  - 5.1.8 Set Li . . . 33

5.1.9 Set X . . . 34
5.2 Instance Solver . . . 34
5.3 Result Analysis . . . 36
5.3.1 Distribution Analysis . . . 36
5.3.2 Performance Analysis . . . 37
5.3.3 Feature Importance . . . 43

6 Conclusions and future work **45**
6.1 Conclusions . . . 45
6.2 Future Work . . . 46

A Predictive Algorithm Results **47**

B Extracted Features **51**

References **55**

List of Figures

2.1 Solution Methodologies of Vehicle Routing Problem (Source: Anbuudayasankar et al. (2014)) 6
2.2 Branch & Bound Tree Diagram. 6
2.3 Evolution of VRP heuristics (Source: Gendreau et al. (2002)) 9
2.4 Behaviour of Random Forest Regression. 11
2.5 Interactions of a Markov Decision Process. 12

3.1 Subdivison of the VRP. 14
3.2 Visual Representation of a VRP. 15

4.1 Self-adapting Optimization Algorithm Machine Flow Model 18
4.2 Designed System Flow 18
4.3 Self-adapting Optimization Algorithm Machine Project Development 20
4.4 Visual Representation of A-n32-k5 21
4.5 Instance A-n37-k5 Quadrants 24
4.6 Ranks of heuristics in instance E-n33-k4 26

5.1 Instance A-n44-k6 30
5.2 Instance B-n44-k7 30
5.3 Instance E-n51-k5 31
5.4 Instance F-n135-k7 32
5.5 Instance M-n101-k10 32
5.6 Instance P-n20-k2 32
5.7 Instance Golden17 33
5.8 Instance Li23 33
5.9 Instance X-n129-k18 34
5.10 Rank vs Algorithm Fold 1 and 2 41
5.11 Rank vs Algorithm Fold 3 and 4 41
5.12 Rank vs Algorithm Fold 5, 6 and 7 42
5.13 Average Feature Importance 43

List of Tables

2.1 Computational comparison of four constructive heuristics (Source: Laporte and Semet (2002)) 8

4.1 Heuristics used present in OR-Tools Library 19
4.2 Estimators of features 22
4.3 Features List 27

5.1 Sample of the Instance Solver Results 35
5.2 Heuristic ranks for the Kruskal-Wallis test 36
5.3 SOAM × Path Most Constrained Arc Wilcoxon test 38
5.4 SOAM × Chirstofides Wilcoxon test 38
5.5 SOAM × Path Cheapest Arc Wilcoxon test 39
5.6 Analysis of the k-fold CV 40
5.7 Gap Differences to Best Solution Fold 1 through 4 41
5.8 Gap Differences to Best Solution Fold 5 through 7 42
5.9 Feature Importance by Heuristic 44

A.1 Results of fold 1 47
A.2 Results of fold 2 47
A.3 Results of fold 3 48
A.4 Results of fold 4 48
A.5 Results of fold 5 48
A.6 Results of fold 6 49
A.7 Results of fold 7 49
A.8 Instances present in each fold 50

B.1 Complete set of Extracted Features 51

Abbreviations and Symbols

AI	Artificial Intelligence
CV	Cross Validation
CVRP	Capacitated Vehicle Routing Problem
KPI	Key Performance Indicator
MAPE	Mean Absolute Percentage Error
MDP	Markov Decision Process
ML	Machine Learning
MSE	Mean Squared Error
MSP	Minimum Spanning Tree
OR	Operational Research
RL	Reinforcement Learning
RNN	Recurrent Neural Network
SOAM	Self-adapting Optimization Algorithm Machine
SMAPE	Symmetric Mean Absolute Percentage Error
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
VRPTW	Vehicle Routing Problem with Time Windows

Chapter 1

Introduction

This thesis report is developed in the scope of the dissertation on the Master Course in Electrical and Computers Engineering at Faculty of Engineering of the University of Porto.

On this chapter a contextualization of the problem is made as well as some regards on the current objectives of this project and the reason behind its development.

1.1 Layout

The layout of the report is as presented next. First we start by giving the current definitions and nomenclature of the problem, then we provide the current state of the art on the topic of machine learning and optimization.

On Chapter 3 we give the formal definition of the problem along with its mathematical model. It is then followed by the system model that was implemented as well as the methodology of the development of the system.

Finally in Chapter 5 we can find the description of the data being used to validate the project as well as the analysis of the obtained results. Afterwards there is a conclusion of the project and some remarks about future possible work in the same topic.

On Appendix A the results of the developed project can be seen and in Appendix B the features extracted from the instances are also shown.

1.2 Context and Motivation

The field of Operational Research (OR) is based primarily on developing models and algorithms for decision making. These algorithms have their main focus on real life scenarios with the typical goal of optimizing a certain performance. Within the problems found by OR there are several classes of problems, but we will focus on NP-hard problems which consists of problems like Capacitated Lot Sizing and Vehicle Routing and also some variants of the Traveling Salesman Problem. This thesis project will primarily focus on the sub-problem of the Vehicle Routing Problem (VRP) the Capacitated Vehicle Routing Problem (CVRP).

On the other side of this project we have Machine Learning (ML) which is designed so that with a finite set of data we can identify patterns and/or solutions for the problems at hand. ML is a sub-part of Artificial Intelligence (AI) but since we are not interested in making a full AI we will just focus on the ML

portion. There are several ways to train an AI as for example through supervised, unsupervised, deep or reinforcement learning. This way through a high number of test cases it is possible that after training the AI can replicate the expected results through the learning process it had.

As can be seen on Bengio et al. (2018) supervised learning is the easiest to implement from the set but shows a bad performance when it comes to making a generalization of the current learned policy. However when it comes to replicating the result from an "expert" it shows the best results.

As for Reinforcement Learning (RL) it is based on a Markov Decision Process (MDP) through a reward system. This model is harder to implement since the choosing of the reward function is not easy because of the "exploration VS exploitation" problem. This problem consists on finding the balance between finding new solutions (exploration) or being "stuck" on what seems to be the best solution but that can be only a local maximum (exploitation).

However, on the last years commercial optimization solver algorithms have been seeing an improvement in performance. Some problems that in previous times were not even thought to be solvable are now solved optimally thanks to breakthroughs in hardware and solver algorithms.

Because of these advancements in performance researchers were led to find even more complex problems leading once more to challenges to commercial solvers. These limitations made researchers develop tailor made algorithms to find a solution to the problems. Despite the great results obtained by these tailor made algorithms these are too specific and depend too much on the researcher knowledge and experience.

Also over the last years advances on the area of AI and therefore on ML have been showing great promise on being able to replicate or even surpass human knowledge and experience. Because of this researchers are led to believe that ML can be of great value to help on the resolution of ever more complex optimization problems.

This thesis project tries to tackle this problematic by joining optimization algorithms with ML methods in a way to create new decomposition algorithms that adapt to the instance of the problem. Specifically on the development of algorithms that can compete to state-of-the-art methods on the classical optimization problems, this project will be validated using instances of the CVRP.

1.3 Why Optimization and Machine Learning

While the sole use of optimization heuristics or meta-heuristics provides researchers with very good results those implemented algorithms are too focused on a single instance. Therefore they lack the ability to provide those same results on the whole problem family they are based on, which means they do not have good generalization properties.

In an attempt to change this a new solution is what is provided by this thesis project. The combination of those same optimization algorithms with a ML interface will allow that an algorithm designed for a single instance of a problem can successfully be generalized to similar instances through parameter tuning provided by the ML agent.

As can be seen on Ansótegui et al. (2009) successful implementation of similar algorithms for parameter configuration has already been achieved. Following this idea the ML agent will, from the instance properties, define what will be the best set of parameters to utilize. This entire process will benefit from a level of abstraction in such a way that the user will not require extensive knowledge on how the parameters work together to be able to obtain a good output from the system.

Applied to the CVRP this entire process is supposed to be able to create a system that can compete with the already state of the art solver algorithms. With this in mind the implemented ML algorithm will

determine which of the available optimization algorithms will provide the best result in the least possible time.

1.4 Objectives

This project has as a goal the creation of an algorithm based on ML that can identify the type of the given problem and with that information be able to determine which is the best method to apply to reach a good quality solution in the fastest time possible.

The reason behind this is to be able to develop a new algorithm that can on average perform better than always using the same method for every instance. This analysis is conducted by the average instead of on instance basis since the objective is to have something that is overall better than the alternatives. If this comparison was made on an instance basis then even though we could have excellent results on a single type of instance the whole project would not fit the objective since it would not be able to be applied to the generality of the instances.

On a first approach it will be necessary to determine which are the main characteristics of the problem that make a differentiation within that problem family. According to this information we will be able to determine which features to analyse such that we are able to create different clusters of different types of problems within the same problem family.

Then the type of ML to be used will have to be defined and implemented so that the AI can be trained to differentiate the several types of problem and choose the best available method.

Finally with the help of metrics from the website CVRPLIB (vrp.galgos.inf.puc-rio.br/index.php/en/) and NEO Research Group (neo.lcc.uma.es/vrp/vrp-instances/) there will be done extensive tests to the developed AI such that the results can be compared to the already best known results. The main goal of this battery of tests is to frame the applied solutions from the algorithm alongside the already known solutions within the lowest elapsed time.

In the end this project should be able to create a proof of concept of a system that is able to solve in a more efficient way new instances of the CVRP. It should also be able to estimate the expected result gap for the chosen method of resolution.

Chapter 2

Literature Review

On the current chapter we will focus on the review of the current state-of-the-art on the topics related to this project. This includes the research into combinations of optimization techniques with ML algorithms.

Section 2.1 consists on the description of the main differences between exact optimization algorithms and approximate optimization algorithms and also the description of some heuristics and exact methods. On section 2.2 an overview of Machine Learning types and algorithms is provided. It is then followed by a description of the combinations between Optimization and ML in section 2.3 and between ML and the CVRP in section 2.4

2.1 Exact Optimization vs Approximate Optimization

There are several methods of optimization like the exact methods, capable of getting the optimal solution and the approximated methods, which may not give the optimal solution but give a "good" solution to the problem.

As can be observed on Figure 2.1 which is found on Anbuudayasankar et al. (2014) the amount of ways one can tackle the VRP or the Travelling Salesman Problem (TSP) families are very high. This section dives into some of the usual exact methods, heuristics and meta-heuristics utilized to solve problems of the VRP family.

2.1.1 Exact Algorithms

Exact Optimization Algorithms are a kind of methods that can be used in the resolution process of most combinatorial optimization problems. This type of methods have a great strength in that they always provide the optimal solution for the problem, the downside for this is that given the complexity of the problems the time required to achieve said optimal solution can range from a couple of hours to several days. As exposed on Festa (2014) some of the most common of these approaches include the Branch & Bound and the Dynamic Programming that have both being under study since at least 1966.

Even though these approaches have the problem of exponential time consumption - the time required for these algorithms to achieve the optimal solution increases exponentially with the size of the problem - the Branch & Bound and the Branch & Price methods like those found in Fukasawa et al. (2004), Ralphs (2003) and Bard et al. (1998) in are still commonly used in the resolution of the VRP since they are able to provide the optimal solution to the problem.

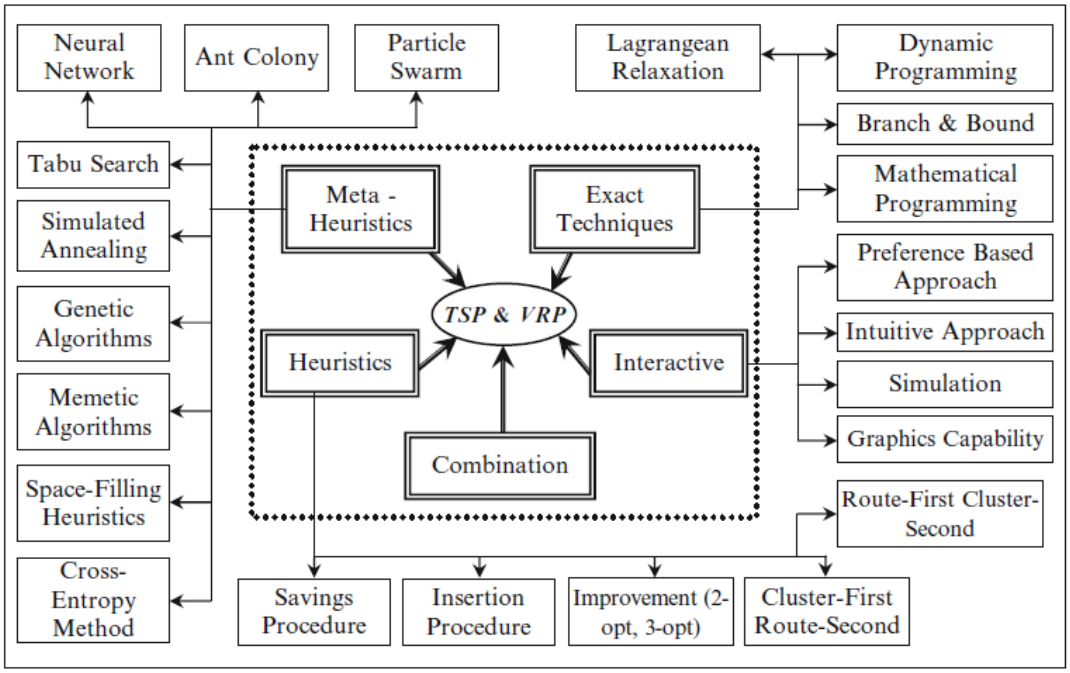


Figure 2.1: Solution Methodologies of Vehicle Routing Problem (Source: Anbuudayasankar et al. (2014))

2.1.1.1 Branch & Bound

The Branch & Bound algorithm is the most commonly known exact optimization technique. As proposed by Laporte et al. (1987) the Branch & Bound algorithm begins by using a *Branching* method by subdividing the main problem into several smaller problems which because of their size reduction are easier to solve creating a tree-like structure like the one represented in Figure 2.2.

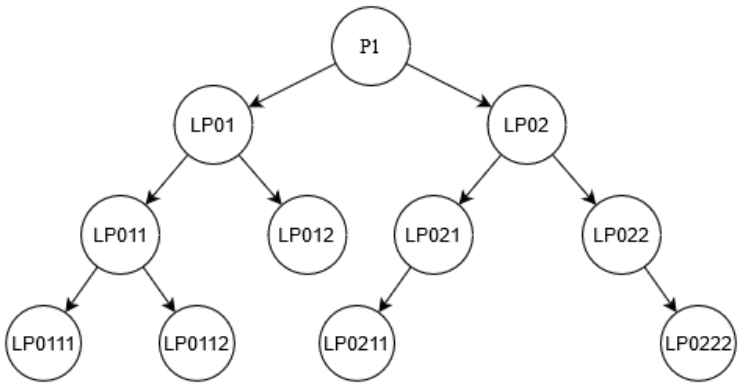


Figure 2.2: Branch & Bound Tree Diagram.

It is then followed by a *Bounding* method in that both upper and lower bounds are defined and as such if a node, or leaf, is beyond those bounds it is not explored further since it will not provide a better solution. With this bounding method - in theory - the entire tree will not need to be perused, however there are some exceptions that depending on the method of exploration will require the whole tree to be explored. The usage of the bounding method will reduce the time required to achieve the optimal solution.

2.1.2 Constructive Heuristics

Constructive heuristics are a type of algorithms that begin with an empty solution and successively constructs the solution until a complete solution is achieved. An example of some heuristics applied to the VRP can be seen on Laporte and Semet (2002) and also some heuristics will be briefly explained in this section.

- **Path Cheapest Arc**

The Path Cheapest Arc heuristic is also known as the Nearest Neighbour heuristic. It begins with a single node and extends the route to the nearest node, it then repeats this process always choosing the nearest node to the last chosen node until all nodes are included in the solution.

- **Savings**

The Savings algorithm was proposed by Clarke & Wright in 1964 and can be found on Clarke and Wright (1964) and is one of the most used heuristics for the VRP. It is based on the concept of savings, in that if two routes can be merged together into a single one a saving is generated.

Although the widespread use of this algorithm according to Vigo (1996) it has a worse performance when it comes to directed problems.

- **Sweep**

The Sweep algorithm by Wren & Holliday can be encountered on Wren and Holliday (1972) and was proposed in 1972 which is based on Elementary Clustering.

It works mainly by performing a sweep - as the name indicates - of the vertexes with the anchor point in the depot, when the maximum capacity is achieved a cluster is created and then the process repeats until all vertexes belong to a single cluster then each cluster will be solved as a simple TSP. If some after processing is implemented a certain degree of optimization can be achieved by exchanging the outlying vertexes with the adjacent clusters.

- **Christofides**

This algorithm was presented by Nicos Christofides in 1976 and can be seen on Christofides (1976).

Summed up the Christofides algorithm generates its routes by creating maximum matching of the nodes and then merging that matching with the Minimum Spanning Tree (MSP) created by those nodes an Euler tour is generated. Then to obtain the final solution repeated nodes are removed from the tour.

- **Parallel Cheapest Insertion**

This algorithm works by iteratively selecting the cheapest node - the one with the lowest demand - and inserts it at the cheapest position until all are connected. The cost for the insertion is calculated based on the arc cost of the nodes.

The sole difference between Parallel Cheapest Insertion and Local Cheapest Insertion is that while Parallel Insertion always chooses the cheapest node available Local Insertion chooses nodes by their order of creation and not cost.

- **Global Cheapest Arc**

Global Cheapest Arc algorithm functions by connecting from all the available nodes the two that generate the cheapest arc, this is done in an iterative way until the complete solution is achieved.

There is also a Local Cheapest Arc algorithm that has as a difference instead of choosing the two nodes it simply selects the first node without a successor and connects it to the node that creates the cheapest route.

Conclusion

Although these methods provide a very approximated result to the best known value, as can be seen on the Table 2.1 found in Laporte and Semet (2002) the execution of such algorithms in larger instances of the problems may be unfeasible because of the time taken to be able to compute the result, as can be seen by the blank spaces on the table. As such a new way to better solve these problems has arisen.

Table 2.1: Computational comparison of four constructive heuristics (Source: Laporte and Semet (2002))

	Sweep[1]		Generalized assignment[2]		Location-based heuristic[3]		Truncated branch-and-bound[4]		Best known solution
Problem	f^{*4}	Time[4]	f^{*2}	Time[2]	f^{*3}	Time[3]	f^{*4}	Time[4]	value
E051-05e	532	*12.2*	524	*9.3*	**524.6**	*68*	534	*7.1*	524.61[5]
E076-10e	874	*24.3*	857	*12.0*	848.2	*406*	871	*15.6*	835.26[5]
E101-08e	851	*65.1*	833	*17.7*	832.9	*890*	851	*38.2*	826.14[5]
E101-10c	937	*50.8*	824	*6.4*	826.1	*400*	816	*39.3*	819.56[5]
E121-07c	1266	*104.3*	—	—	1051.5	*1303*	1092	*51.1*	1042.11[5]
E151-12c	1079	*142.0*	1014	*33.6*	1088.6	*2552*	1064	*81.1*	1028.42[5]
E200-17c	1389	*252.2*	1420	*40.1*	1461.2	*4142*	1386	*138.4*	1291.45[5]
D051-06c	560	*11.4*	560	*15.2*	—	—	560	*5.3*	555.43[5]
D076-11c	933	*23.8*	916	*20.6*	—	—	924	*13.6*	909.63[5]
D101-09c	888	*58.5*	885	*52.2*	—	—	885	*33.4*	865.94[5]
D101-11c	949	*53.6*	876	*6.3*	—	—	878	*45.2*	866.37[5]
D121-11c	1776	*85.5*	—	—	—	—	1608	*61.8*	1541.14[6]
D151-14c	1230	*134.7*	1230	*121.3*	—	—	1217	*74.0*	1162.55[6]
D200-18c	1518	*238.5*	1518	*136.6*	—	—	1509	*135.6*	1395.85[5]

2.1.3 Meta-heuristics

Meta-heuristics are a kind of high-level algorithmic framework that is independent of the problem it is solving. Unlike exact methods meta-heuristics do not suffer from combinatorial explosion which is when the computing time needed to obtain the optimal solution of a NP-hard problem have an exponential increase as a function of the size of the problem. Compared to the usual heuristic methods used the meta-heuristics usually result in better local optima however since meta-heuristics can allow worse or infeasible solutions in the search process they tend to consume more time in giving the result.

As observed on Figure 2.3 which can be found on Gendreau et al. (2002) the trend for the solver algorithms for the VRP will tend to be on the reduction of computing time for meta-heuristic instead of further optimizing the already good classical heuristics.

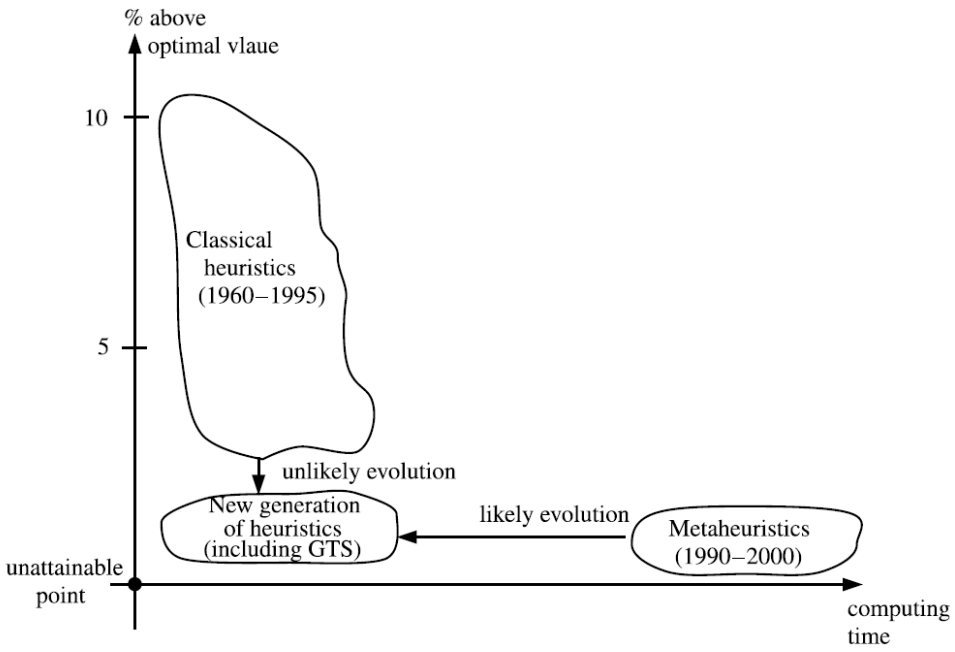


Figure 2.3: Evolution of VRP heuristics (Source: Gendreau et al. (2002))

As can be seen on Gendreau et al. (2002) the main types of meta heuristics found are:

- **Simulated Annealing**

Like proposed by Osman (1993) the simulated annealing algorithm is a type of meta-heuristic that is commonly used in the resolution of the VRP.

Its main concept is that after determining two possible routes the algorithm tries to swap customers between both routes until a improvement can be observed.

- **Tabu Search**

Tabu Search algorithms has seen some increased usage over the last years as can be seen on Taillard (1993), Toth and Vigo (2003) and Rego and Roucairol (1996).

In this type of algorithm just like in the simulated annealing the solutions are examined and the solutions are shifted to the best neighbouring solution. Since this process could lead to cycling of the solutions then the algorithms sets the current analysed solution as prohibited - or as the name indicates tabu.

- **Genetic Algorithms**

Genetic algorithms are population based meta-heuristics that have as a main principle the mimicry of the natural evolution processes. This kind of algorithms have 4 main stages to solve a problem.

First there will be chosen two chromosomes to be the parents of the next offspring. Then by performing crossover operators the two parents are merged creating two new offspring which afterwards will be randomly mutated with a mutation operator. Finally a new generation will be created by selecting the a part of the best solutions and generating the same amount of offspring so that all the generations have the same size.

Like can be seen on Potvin (1996) this type of algorithm can safely be applied to the VRP with the appropriate changes in mutation and crossover operators.

Conclusion

Besides the algorithms exposed above there are several others more like for example Ant Systems (Qi, 2008) or Neural Networks (Yoshiike and Takefuji, 1999). For an extensive text on meta-heuristics see Gendreau and Potvin (2010).

2.2 Machine Learning

The other aspect of this thesis is ML, which is an area of investigation that has as a basis the learning of new processes and behaviours by the AI in question. ML mainly works with finite sets of data, but it is well suited for signals where no mathematical formulation exists because the data distribution is not analytically resolved.

The latest breakthroughs in the area give as the main types of learning methods the Supervised Learning, Unsupervised Learning and RL.

RL focuses mainly on rewarding and penalizing the AI based on the response to the given situation, like if you are trying to teach a car to park with RL, you can reward him if he gets closer to the parking spot but you can penalize if the car hits an obstacle or climbs the sidewalk.

2.2.1 Supervised Learning

As far as supervised learning goes as seen on Kotsiantis (2007) it is usually implemented in cases where we have the correct expected output so that the AI can be trained to replicate such results. There are two kinds of behaviour for supervised learning it can be for classification, making a distinction inside the data for example correct or incorrect, or it can be of regression making predictions of the results in a linear and continuous way.

The main defect of Supervised Learning is that even though it can replicate with ease and accuracy the solutions it was presented it is not very good in the process of generalization, making it very inaccurate when the new data is very different from the data used to train the ML model like exposed in Smith and Martinez (2011).

Inside of the regression techniques there are several distinct types of techniques like Linear Regression, Support Vector Regression, Regression Trees and Random Forests. The regression techniques can be used with one or several features to predict one or more targets. The main goal of regression is to be able to predict a continuous value, it works by determining several types of behaviour that replicates the provided data and calculating the one with the lowest difference to the given data.

Even though there are several distinct types and implementations of ML models only the analysis of Random Forest Regression will be deepened since it will be the one used in this project.

Random Forest Regression is one type of regression technique that is comprised by an ensemble of individual decision trees regressions. The ensemble of models will act together and predict the value of the average of the individual predictions. The behaviour of an ensemble with 600 decision trees can be seen on Figure 2.4.

The main advantage of utilizing a random forest instead of a simple decision tree is that by having multiple trees output a value and then choosing the one that minimizes the Mean Squared Error (MSE) provides better results since that the trees will shield each other from their individual errors - provided that the errors are not all in the same direction. An effect that can boost this shielding is the fact of trying to

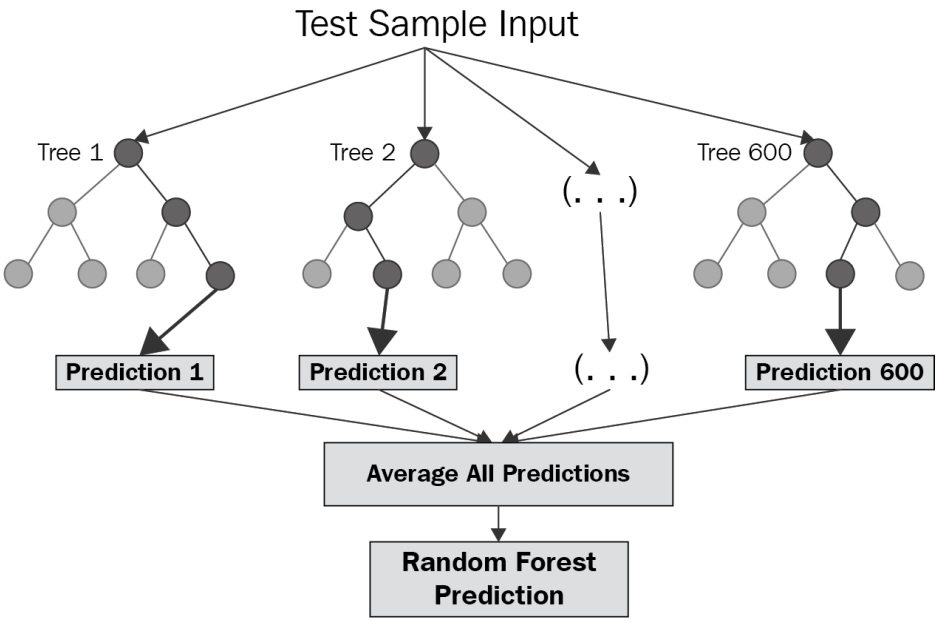


Figure 2.4: Behaviour of Random Forest Regression.

achieve the greatest number of uncorrelated trees, since they will most likely not have the same errors and therefore prevent error propagation.

2.2.2 Unsupervised Learning

Unsupervised learning has two main methods of being used. First it can be used as a clustering method since it can detect up to a certain range different properties and differences between the data and creating clusters of data points. Another use is the one of dimensionality reduction which is used to remove unimportant data or features from the data set provided to the ML algorithm.

As seen on Caron et al. (2018) sub-chapter 4.2 the method of clustering can provide excellent results on detecting some differences between data points. Because of this feature it may prove to be a useful learning type for the clustering of the different types of problems within the thesis project.

Although the purpose is not the same most ML models of Unsupervised Learning have very good capabilities to make generalizations of the rules learned, making them a good choice when the real data can be very different from the data set used on the training of the same model.

2.2.3 Reinforcement Learning

The RL process is based on a MDP which is demonstrated on Figure 2.5. This process works like follows at every time step.

First the AI, referred as agent, being on a given state of the environment takes an action based on its chosen policy. Then, as result of the action it is rewarded by the environment and switches into a new state.

The main goal of RL is to train the agent to maximize the total sum of the rewards given by the environment. The agent does not need to know the properties of the environment to work, these are learned by the agent in a dynamic way.

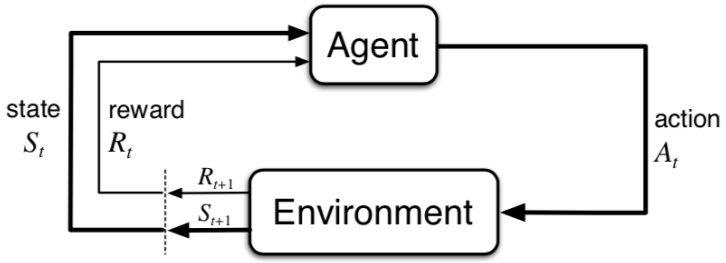


Figure 2.5: Interactions of a Markov Decision Process.

On the topic of RL the reward function is one of the hardest points to define since there is the possibility of the learning to converge on a local maxim or the possibility for it to be sparsely defined and therefore lacking on learning opportunities. To a more extensive read on RL refer to Sutton and Barto (2018).

2.3 Optimization and Machine Learning

Like mentioned on Chapter 1.3 the combination of ML techniques and optimization algorithms can be a great advantage to the development of new and better algorithms for solving the CVRPs family. Having both classes several methods and algorithms that can be implemented some specific combinations of both can be seen as better working than others. As proposed on Kadaba et al. (1991) one of the ways both ML and CVRP could be joined is through neural networks and genetic algorithms.

As can be seen on Nazari et al. (2018) one of the combinations that can be used is the RL method and the Neural Networks algorithm, more precisely the Recurrent Neural Network (RNN).

2.4 Machine Learning & CVRP

Although there are not many research papers available on the junction of ML and CVRP the ones that exist are mostly on the creation of ML models for solving the VRP some of which can be found on Nazari et al. (2018) and Peng et al. (2020).

These papers focus on the creation of new ML based algorithms to solve the VRP which can be compared to current solvers. Since the goal of this project is not to develop a new solution method but instead to create an algorithm that is able to determine the best course of action using current methods only the main ideas were used from these papers.

However as mentioned in Ren et al. (2006) one of the uses of ML in the CVRP family is the usage of different types of learning to implement the clustering methods required to create sub groups in such way to allow feasible initial solutions. With this in mind this thesis project is more similar to Cooray and Rupasinghe (2017) in which the model will be used to tune certain aspects of the solver.

Chapter 3

Problem Description

The problem at hand is to optimize the resolution of CVRP through the junction of the solver algorithms with ML process in a way to obtain the most effective solution possible on an autonomous way. With this in mind the problem can be separated in two parts. First the solver algorithm for the CVRP instances and then the decision on which type of learning process to implement as the way to train it.

3.1 Formal Description

The VRP family is defined as a NP-hard combinatorial optimization problem and is composed by problems that have as their main concern the distribution of goods between depots and clients. These problems have real-world applications like bus systems routing, waste collection and street cleaning among others. Inside this class of VRP there are several other types of problems which are created by adding restrictions to the base problem. The resolution of this kind of problem is of interest to industries and service providers since costs can be reduced by better planing of routes for goods delivery or costumer maintenance.

The CVRP is an extension of VRP on which a new constraint is added, on the case of CVRP it is the maximum capacity of the circuit vehicles. As shown on Figure 3.1 the CVRP is one of the simplest sub-problems of VRP.

The VRP have as main elements the origin (and in some cases the termination) of the circuit which are the depots, all the clients required to be served and the routes to all the clients. In a way to facilitate the visualization of this kind of problem one can represent them as a graph (either directed or undirected) on which the vertexes are the clients and depots and where the arcs represent the routes between points and have an associated weight. A visual representation of simple VRP and respective solution -each represented in a different color - can be seen on Figure 3.2.

As it can be seen on Toth and Vigo (2002) the main characteristics of clients are:

- the vertex of the client location;
- the size or amount of the goods to be delivered or picked up;

On special cases with more constraints like Vehicle Routing Problem with Time Windows (VRPTW) you can make use of other special characteristics like for example the period of the day when the client can be served or the time taken to load and unload the vehicle.

Besides the clients, the usual characteristics of the vehicles are:

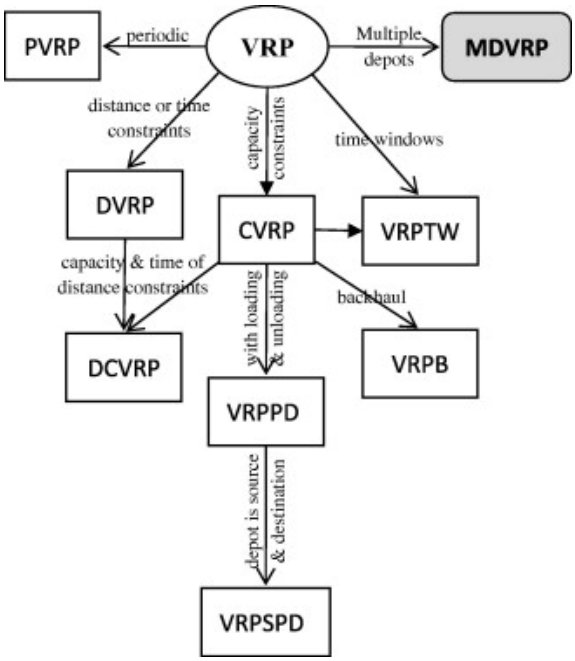


Figure 3.1: Subdivison of the VRP.

- the origin depot and if need be the termination depot;
- the vehicle load capacity (like in the case of CVRP);

Like the clients in more specific cases or in real life scenarios it can also be needed to know the possible routes for that kind of vehicle and may also be required to know the costs of using a vehicle (personnel, maintenance, etc.).

On most of these types of problems the main goal is to minimize the global cost of transportation either through uniforming the routes, reducing number of active vehicles or others.

3.2 Mathematical Model

The CVRP can be described mathematically as a mixed integer programming problem like follows as can be seen on Munari et al. (2016):

Minimize:

$$\sum_{i=0}^{N}\sum_{j=0}^{N} c_{ij}x_{ij} \tag{3.1}$$

Subject to:

$$\sum_{j=0}^{N} x_{ij} = 1, \qquad \text{for } i = 1,...,N \text{ and } i \neq j \tag{3.2}$$

$$\sum_{i=0}^{N} x_{ih} - \sum_{j=0}^{N} x_{hj} = 0, \qquad \text{for } h = 1,...,N \text{ and } i \neq h\ ,\ j \neq h \tag{3.3}$$

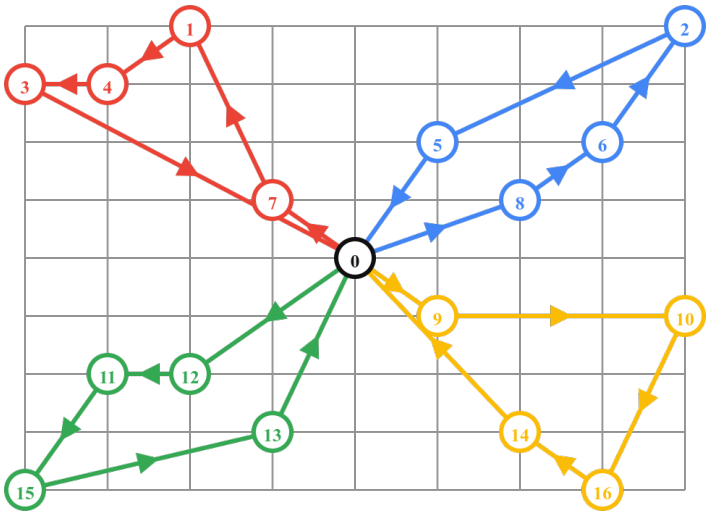


Figure 3.2: Visual Representation of a VRP.

$$\sum_{j=1}^{N} x_{0j} \leq K, \tag{3.4}$$

$$y_j \geq y_i + d_j x_{ij} - Q(1 - x_{ij}), \qquad \text{for } i, j = 0, ..., N \tag{3.5}$$

$$d_i \leq y_i \leq Q, \qquad \text{for } i = 0, ..., N \tag{3.6}$$

$$x_{ij} \in \{0, 1\}, \qquad \text{for } i, j = 0, ..., N \tag{3.7}$$

This problem mathematical formulation has as elements: The cost between nodes i and j is denoted as c_{ij}. K, N and Q are the number of vehicles available, the number of customers and depots available and the capacity of the vehicles, respectively. The demand from node i is given by d_i where the demand from the depot $d_0 = 0$. In addition there are two decision variables, y_i which corresponds to the cumulative demand from the route that serves node i up to that point and finally x_{ij} which corresponds to if the route is used or not.

Constraints (3.2) and (3.3) ensure the correct flow of vehicles between nodes such that if a vehicle enters a node it must leave the same node and also that each node is visited only once. Constraint (3.5) has the purpose of also avoiding subtours in the solution for instance cycling routes that never pass by the depot. Constraint (3.4) enforces that the number of existing routes does not surpass the number of available vehicles K.

The main distinction between the CVRP and the VRP is the set of equations (3.5) and (3.6) which enforces the limit of the vehicle capacity such that if a vehicle serves client i then all demands served by that vehicle can not be higher than the capacity Q.

Chapter 4

Methodology

On this chapter starting in Section 4.1 the modulated system will be presented. Then it is followed by the Tools used in the implementation of the model.

Finally in section 4.3 each step of the implementation process will be explained.

4.1 Project Model

As far as the whole thesis project goes, the generic model for the entire process is as follows on Figure 4.2. In order to infer the acuracy and effectiveness of the system some Key Performance Indicator (KPI) will have to be defined before hand.

Some KPIs that may be used are: The computation time took by the software to determine the solution; The Result Gap Percent based on best known results from the literature which is calculated according to

$$Gap(\%) = \frac{C_r - K_r}{K_r} \times 100, \tag{4.1}$$

where K_r is the best known result for the instance and C_r is the obtained from the implemented algorithm. In the eventuality of $C_r < K_r$ then the gap is still determined although the negative sign will indicate it is a new best result.

The designed project system, now referred to as Self-adapting Optimization Algorithm Machine (SOAM), can be visualized as a two part process.

First the SOAM will perform a feature extraction from the received instance in order to through forecasting methods and predefined KPIs be able to select the best optimization algorithm to use. Then it will execute the chosen algorithm and perform an analysis based on those same KPIs such that the data can be used in future executions.

The KPIs defined on the model as can be seen on Figure 4.1 have to be well defined as well as their evaluation method because as there are several KPI their evaluation will be made as a multi criterion evaluation. Therefore it must be careful studied what are the most important aspects to optimize since this choice will heavily impact those performances.

As can be seen on Figure 4.2 the SOAM interacts directly with the received instances. The connection between the KPIs and the SOAM is bidirectional because the results obtained from the execution of the algorithm may be useful for improving the encapsulated ML agents of the SOAM.

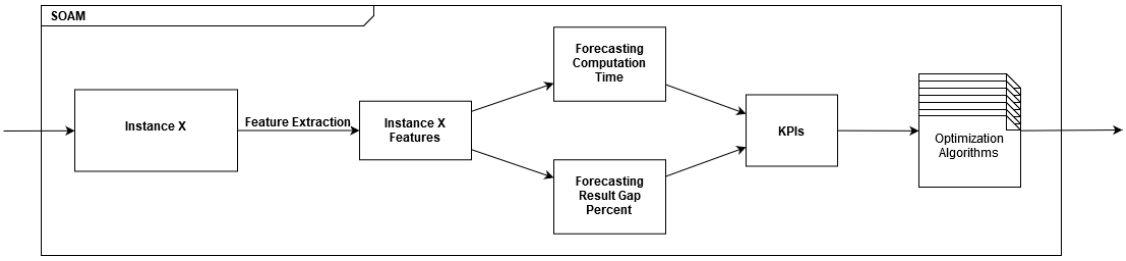


Figure 4.1: Self-adapting Optimization Algorithm Machine Flow Model

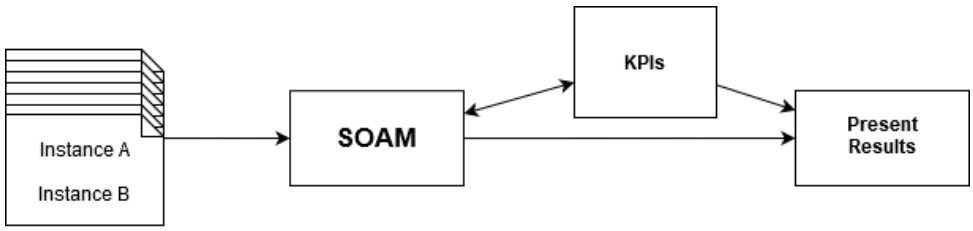


Figure 4.2: Designed System Flow

4.1.1 Data Model

To train and test the implemented model instances from CVRPLIB repository will be used. CVRPLIB (http://vrp.atd-lab.inf.puc-rio.br/) is a web site repository of instances of CVRP which contains 12 different data sets from the literature and correspondent solutions.

The data from CVRPLIB on CVRP comes with $n-1$ vertexes and one depot and with all the information relative to these points. On the other hand it also has the demand from every client and the maximum capacity of the vehicles in the fleet although it does not have a limit to the number of vehicles used.

The analysis that will be made to the results will be similar to the ones that can be seen on Table 8 of Augerat et al. (1995).

4.2 Implementation Instruments

On the development of this project several applications and programs were used. Python 3.7 was the coding language chosen for this project since the newer version 3.8 was not supported by all the required libraries of this project in the beginning of development and therefore to avoid switching distributions in the middle the same version was used until the end of the project.

Google OR-Tools is a software for combinatorial optimization that includes several kinds of solvers, from those the solver for Vehicle Routing was used. The Routing solver includes several implemented heuristics to provide a solution to the given problem and along with those it also provides some meta-heuristics for local search. The local search meta-heuristics try to escape a local minimum in order to improve the solution obtained by the heuristic. All the documentation related to OR-Tools Python Library can be found on Python Reference: Routing[1].

[1] developers.google.com/optimization/reference/python/constraint_solver/pywrapcp

4.2.1 Constructive Heuristics

The constructive heuristics found on OR-Tools can be seen on the documentation and the ones used in this project can be found in Table 4.1. Some of the heuristics available in the OR-Tools library were not used because the OR-Tools library in Python is an exported version of the one in C++ and for this reason not all methods behave properly in Python.

Table 4.1: Heuristics used present in OR-Tools Library

Heuristic		
Path Cheapest Arc	Path Most Constrained Arc	Savings
Christofides	Parallel Cheapest Insertion	Local Cheapest Insertion
Global Cheapest Arc	Local Cheapest Arc	First Unbound Min Value

4.3 SOAM Development

The development diagram of the SOAM can be observed in Figure 4.3. In this section each of the different steps will be explained in more detail. The two first steps will not be detailed here since they were already exposed in Section 3.1 and in Section 4.1.

4.3.1 Data selection and parsing

The data sets used in this project follow the same kind of data organization that is provided with TSPLIB 95 and are stored in a .vrp type file. To be able to store this data into the project an interpreter was developed, it was based on excerpts of code extracted from a website by Vipin Ajayakumar [2].It works by parsing the .vrp file using regular expression matching and dividing he information into its proper type.

After parsing the data a visual representation of the instance was created in an attempt to confirm the correct reading of the coordinates and demands found in the instance file. A visual representation of the instance A-n32-k5 can be observed in Figure 4.4. One of the images is the one available on the CVRPLIB repository and the other the one created for the confirmation of the correct interpretation of the data.

4.3.2 Instance Solver

So that the SOAM could have a baseline to be able to train, test and validate the results a solver for the CVRP instances was required to be implemented. The skeleton of the implemented solver is the one available in the OR-Tools library examples.

The heuristics in Table 4.1 are merged with the parser described in the previous section to create the Instance Solver. The pseudo-code of the solver algorithm can be seen on Algorithm 1, and the table produced by the results of the same solver can be found later on Chapter 5 on Table 5.1. The results are sorted first by *Result Gap* and if they are equal then they are sorted by *Run Time*. The result gap is calculated as described on Equation (4.1).

[2] vipinajayakumar.com/parsing-text-with-python/

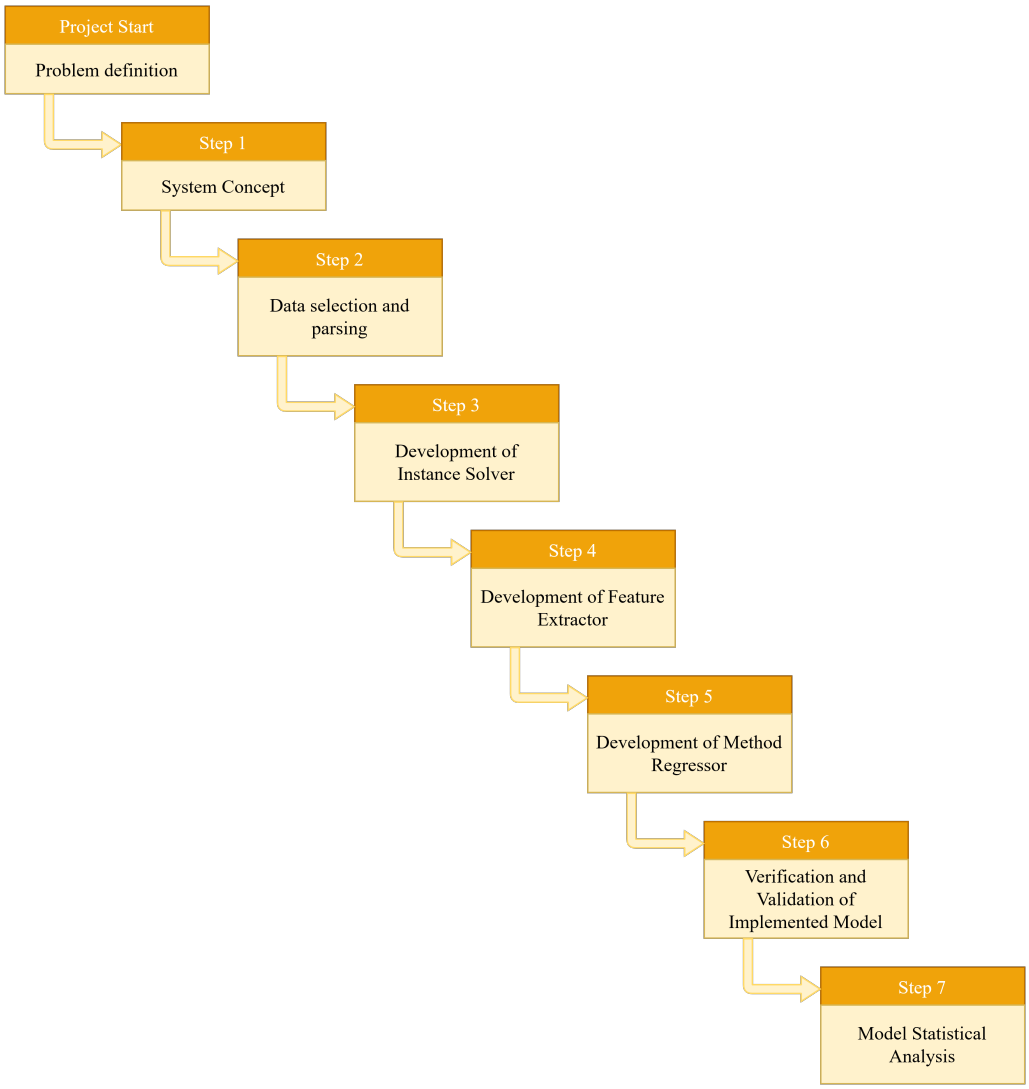
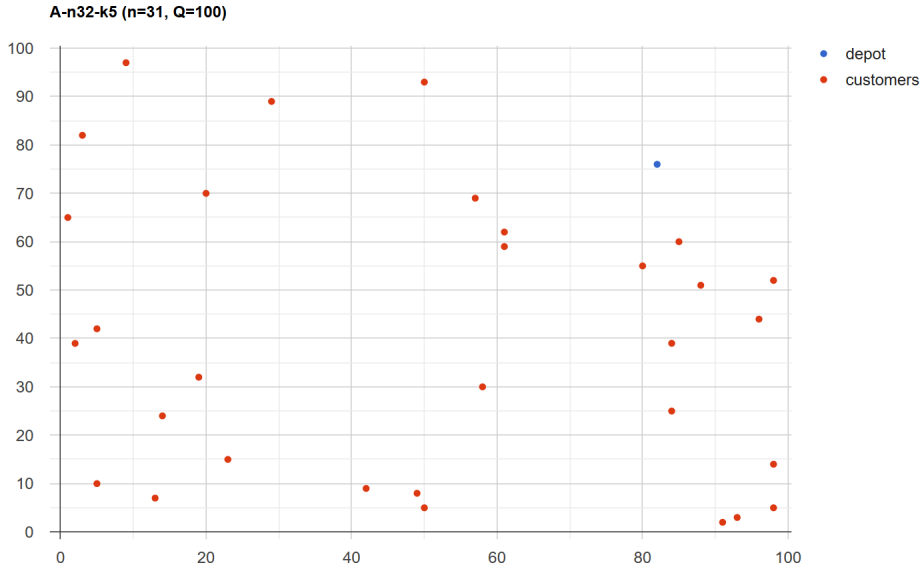


Figure 4.3: Self-adapting Optimization Algorithm Machine Project Development

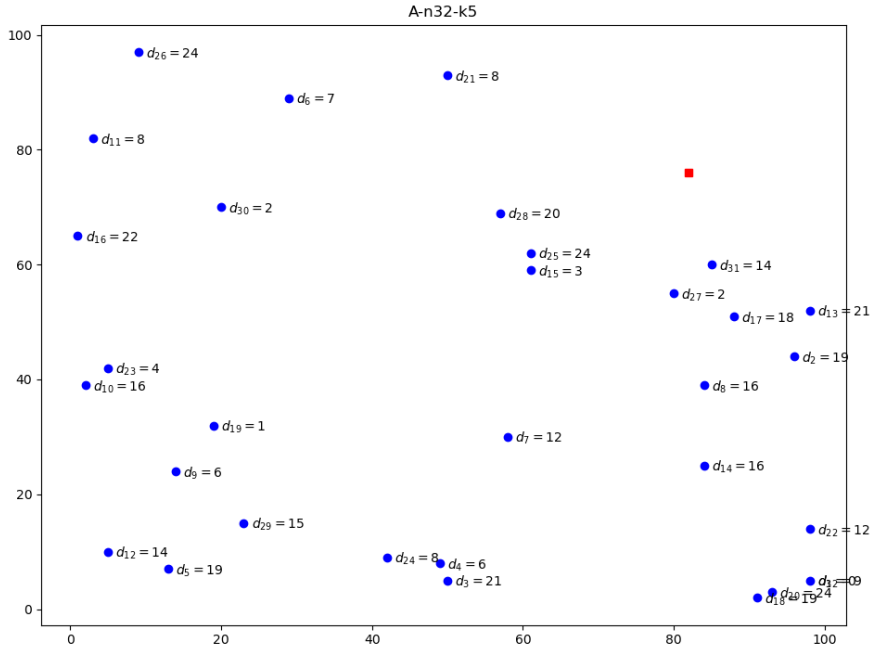
Some of the implemented heuristics are non-deterministic, like Path Cheapest Arc - for the same input values they do not provide the same result. For that reason the solver algorithm was executed 2 times in order to perform an average of the obtained results.

To be able to get a more reliable run time and since local search meta-heuristics are non deterministic the solver executes multiple times for the same configuration of parameters in order to be able to compute the average values of the result gap and run times. Besides in order to avoid the solver running during extremely long times in search of better local optima there were enforced three different kinds of stopping criteria to the solver. These were comprised of:

- Solution Limit
- Solver Time Limit
- Local Neighbour Search Time Limit



(a) Instance Representation on CVRPLIB repository



(b) Instance Representation for confirmation

Figure 4.4: Visual Representation of A-n32-k5

Algorithm 1: Instance Solver

```
foreach instance do
    Parse file;
    for 2 times do
        foreach Routing Option do
            Execute Solver;
            Save result;
            Save execution time;
        end
    end
end
Save results to .txt file;
Display results table;
```

For each of these a value was chosen, the solution limit was defined as 1000 times the number of nodes present in the instance, so if an instance had 32 nodes the solver would have a limit of 32000 viable solutions. The time limits were of 15 minutes in the total time to solve an instance and 5 minutes in the search for a local neighbour.

4.3.3 Feature Extractor

For the Supervised Learning Algorithm to have data to train on it was necessary to extract information from the instance file provided in the repository. Since the goal of the project is to be able to better determine the best resolution method just from the instance characteristics this was an essential cornerstone of the entire development.

The features that were found and extracted can be divided in 5 different groups: Instance Based, Enclosing Rectangle, Node Distribution, Geometric, Heuristic Features. In the following tables there is a number of estimators associated with each feature, if it is statistical estimators then associated are the following eight:

Table 4.2: Estimators of features

Estimator		
Total Value	Minimum Value	Maximum Value
Mean Value	Standard Deviation	Coefficient of Variation
Skewness	Kurtosis	

The total value estimator is determined by the sum of all the correspondent values. The mean value of is determined by the arithmetic mean of the values. Standard Deviation is calculated with resource to

$$S_d = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x})}{n-1}} \tag{4.2}$$

where $\bar{x}$ is the mean value of the data and n the number of points in the data. The Coefficient of Variation was determined by the division between the standard deviation and the mean value. Kurtosis and Skewness

are both determined with resource to python libraries which use Fisher's definition and the Fisher-Pearson coefficient of skewness respectively.

- **Instance Based Features**

Feature I1 and I3 are the number of clients of the instance and the value of the vehicles capacity respectively. As for feature I2 it is the statistical analysis with the estimators found in table 4.2 of the values of the demands from each client. I4 is determined by the result of $\left\lceil \frac{D_t}{Q} \right\rceil$, where D_t is the total demand of the instance and Q the maximum capacity of the vehicles.

The feature I5 is calculated with equation (4.3) where D is the total demand of the instance, K is the theoretical minimum number of vehicles and Q the vehicle capacity.

$$\frac{D}{(K+x)*Q}*100 \qquad \text{for } x \in \{0,1,2\} \tag{4.3}$$

Feature I6 is the division between the mean value of the demand and the vehicle capacities. As for I7 it is what was expeted as the average number of clients in each route and it is given by $\left\lfloor \frac{Q}{\overline{d}} \right\rfloor$ where $\overline{d}$ is the average client demand.

- **Enclosing Rectangle Features**

The enclosing rectangle is the smallest rectangle that encloses all of the instances nodes, it is calculated by determining the smallest and higher node in the x and y axis of the instance. Feature ER1 is composed by three elements, the width, the height, and the area of the enclosing rectangle. Feature ER2 is the geometric center of the enclosing rectangle and the two estimators are the x and y coordinates respectively.

Feature ER3 and ER4 are the density inside the minimum enclosing rectangle of clients and demand respectively. It is determined by the division of the number of clients or the total demand by the area of the enclosing rectangle.

- **Node Distribution Features**

For feature ND1 a statistical analysis with the estimators from Table 4.2 is performed on the set of distances from the depot to each client node. Feature ND2 is determined by adding up all the directional components of the client locations and dividing by the number of clients.

Feature ND3 which is the demand distribution center is similar to ND2 except that the directional components are weighted with the demand of the correspondent client. Both percentage differences ND4 and ND5 are calculated by the difference between the geometric or demand center, respectively, and the actual depot location divided by the width and height of the enclosing rectangle if determining differences in the X or Y axis.

For feature ND6 and ND7 from each client node it is determined the circular area that is equal to the enclosing rectangle area, and calculated the density of clients or demand within that same circular area. Finally for feature ND8 it is first determined what is the client neighborhood that is calculated with $\frac{ER_{area}}{N}$ where ER_{area} is the enclosing rectangle area and N the number of clients. With the client neighborhood calculated just like in ND6 and ND7 from each client the circular area with that size is determined and calculated the client density inside of it.

- **Geometric Features**

Geometric features are based on the analysis of each of the 4 quadrants of the instance, the quadrants are determined by the position of the depot. For the nodes that are vertically or horizontally aligned with the depot they belong to only one of the quadrants that they share. A visual representation of how the quadrants are defined can be seen on Figure 4.5 in green color.

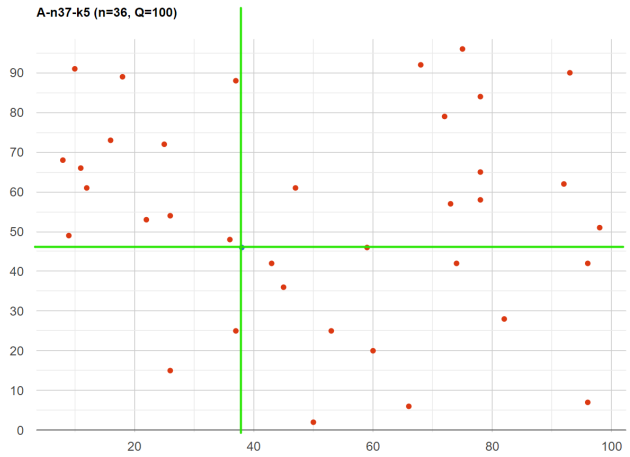


Figure 4.5: Instance A-n37-k5 Quadrants

Features G1 and G2 provide a complete analysis of the client distribution per quadrant and how much variation there is between quadrants. G2 is the analysis of G1 with the estimators exposed on Table 4.2. Features G3 to G8 provide the same type of analysis but of the demand distribution between quadrants.

- **Heuristic Features**

For feature H1 the Bin Packing problem of the instance demands and vehicle capacities was solved with resource to the offline heuristic of Best Fit Decreasing of which there is an overview on Algorithm 2 and its result is the value of the feature.

Algorithm 2: Best Fit Decreasing Heuristic

```
Sort demands;
foreach demand do
    best_fit := maximum;
    foreach open_bin do
        if (free_space < demand) ∧ (free_space - demand < best_fit) then
            Calculate new best_fit;
            Save best_bin;
        end
    end
    if best_fit = maximum then
        Open new bin;
        Insert into new bin;
    else
        Insert into best_bin;
    end
end
```

Output: Number of bins required to fit all demand

The implemented Best Fit Decreasing Algorithm works by first ordering all the demand values in a decreasing order so that the higher demands are first. Then with each demand it will repeat the same procedure of for each bin that is still available it will check that the free space on that bin is larger than the demand in question and that the space that would be left is smaller than what was already calculated. The point of both tests is that the algorithm will try to place the demand on the bin that will be the most full.

Since the Bin Packing problem resolution provides the list of space occupied by vehicle for feature H2 that value was converted into vehicle occupancy and then statistically analysed with the estimators from Table 4.2. In feature H4 the instance Minimum Spanning Tree - the smallest set of arcs that connect all nodes with the least cost possible - was determined using Prim's Algorithm and then a statistical analysis of the set of used arcs was performed with resource to the estimators on Table 4.2.

From each instance there were extracted a total of 134 features and then they are registered into a .csv file so that when it is required for them to be accessed the extractor will not have to be executed again. The complete list of features extracted is present on Appendix B.

4.3.4 Predicting Heuristic Performance

After extracting the features from the instances and executing the solver for all instances then a Machine Learning algorithm based on Supervised Learning is ready to be implemented. The main goal of this algorithm is to predict with basis on the instance features what is the result gap for each heuristic. This is the focal point of the SOAM since it is with this information that it will be able to decide what is the best heuristic to use in the resolution of new instances.

In Supervised Learning cases, the model to be trained requires the labels of the data to predict. The SOAM provides to the models the extracted features as the features to be trained upon and the results of the execution as the correspondent labels. Since the solver executed the same instance for different configurations, then for every type of configuration a regression technique was trained.

The regression algorithm implemented for each type of heuristic is based on a Random Forest ensemble. The implemented ensemble is comprised of 800 estimators - which means it is made out of 800 individual decision trees - and besides the ensemble has bootstrap, which means that when creating the individual decision trees the entire data set is not used on every tree. The ensemble has also set as the maximum number of features for each node to search while looking for the best split as $max_features = \sqrt{n_features}$.

The great advantage of the bootstrap technique and setting the maximum number of features different from the total is that it greatly decreases the correlation of the decision trees which vastly improves the prediction accuracy of the model.

After training the model it is tested to evaluate its predictive accuracy. Since the data can (and is expected to) provide a result of zero most of the times Mean Absolute Percentage Error (MAPE) can not be used to calculate the error since it would cause a division by zero as can be seen on Equation (4.4), where A_i is the actual value, F_i the predicted value and n is the number of instances.

$$MAPE = \sum_{i=1}^{n} \left| \frac{A_i - F_i}{A_i} \right| * \frac{100}{n} \tag{4.4}$$

As such a new way to determine the average error was attempted to be used, one of the variations of the Symmetric Mean Absolute Percentage Error (SMAPE). Unlike the MAPE it is able to calculate the error even when the actual value is zero even though when it happens the value of the error goes to 100%. The expression to calculate SMAPE can be seen on Equation (4.5) where A_i is the actual value and F_i the

predicted value.

$$SMAPE = \frac{\sum_{i=1}^{n} |A_i - F_i|}{\sum_{i=1}^{n} (A_i + F_i)} * 100 \tag{4.5}$$

In order to better train the regression model besides repeating the training process 20 times a cross validation technique is being used to split the data. The k-Fold Cross Validation technique implemented is described in algorithm 3 and was used with a k value of 5.

Algorithm 3: k-Fold Cross Validation

Input: k
Shuffle data set;
Split data set into k folds;
foreach *fold* **do**
    Set fold as test data;
    Set remaining data as training data set;
    Fit (train) the model on the training set and evaluate on test data;
    Retain model if better performance;
end

When the solver was executed it was verified that not all heuristics were able to provide a solution, some because they required more than 15 minutes to solve others because they could not find a feasible solution, in these cases it was assumed the value of the result gap to be 100% in order to have more instances to be able to train and test.

4.3.5 Verification and Validation

In order to be able to validate the data obtained from the regression models two comparisons are performed on a separate data set with 30 instances that are not used either in the training or evaluation of the models. The SOAM will be compared against the performance of always using the same heuristic - for all heuristics available - and also against the case when always the best heuristic is used.

Firstly its performance is evaluated based on the value of the mean result gap for this new validation data set. Then the rank mean is evaluated, this is determined by looking into each instance present in the validation set and determining the rank of each heuristic like can be seen on Figure 4.6.

E-n33-k4

First_Solution	Result_Gap	Rank
CHRISTOFIDES	0.240	1.0
PARALLEL_CHEAPEST_INSERTION	1.557	2.0
LOCAL_CHEAPEST_INSERTION	1.557	2.0
SAVINGS	2.036	4.0
PATH_MOST_CONSTRAINED_ARC	3.832	5.0
PATH_CHEAPEST_ARC	4.671	6.0
GLOBAL_CHEAPEST_ARC	5.868	7.0
LOCAL_CHEAPEST_ARC	5.868	7.0
FIRST_UNBOUND_MIN_VALUE	6.347	9.0

Figure 4.6: Ranks of heuristics in instance E-n33-k4

Table 4.3: Features List

Type	Feature	Description	Estimators
Instance Based	I1	Number of Clients	1
	I2	Total Demand	statistical
	I3	Vehicle Capacity	1
	I4	Theoretical Minimum Vehicle Quantity	1
	I5	Average Vehicle Occupancy	3
	I6	Ratio of Average Demand and Vehicle Capacity	1
	I7	Theoretical Average Number of nodes per route	1
Enclosing Rectangle	ER1	Minimal Enclosing Rectangle	3
	ER2	Enclosing Rectangle Center	2
	ER3	Client Density in Minimal Enclosing Rectangle	1
	ER4	Demand Density in Minimal Enclosing Rectangle	1
Node Distribution	ND1	Distance from Depot to nodes	statistical
	ND2	Client Geometric Distribution Center	2
	ND3	Client Demand Distribution Center	2
	ND4	Percentage Difference from Depot to ND2	2
	ND5	Percentage Difference from Depot to ND3	2
	ND6	Client Density with circular area equal to Enclosing Rectangle	statistical
	ND7	Demand Density with circular area equal to Enclosing Rectangle	statistical
	ND8	Client Density with circular area equal to neighborhood	statistical
Geometric	G1	Number of clients per quadrant	4
	G2	Statistics of G1	statistical
	G3	Total Demand per quadrant	4
	G4	Statistics of G3	statistical
	G5	Demand Statistics of Quadrant 1	statistical
	G6	Demand Statistics of Quadrant 2	statistical
	G7	Demand Statistics of Quadrant 3	statistical
	G8	Demand Statistics of Quadrant 4	statistical
Heuristic	H1	Bin Packing Problem	1
	H2	Bin Packing Problem Vehicle Occupancy	statistical
	H3	Bin Packing Occupancy vs Theoretical Occupancy	1
	H4	Prim's Minimum Spanning Tree	statistical

Chapter 5

Computational Experiments

On the following chapter the results obtained by the SOAM will be presented as well as the respective analysis in an attempt to frame the obtained results with the available solvers.

In a first stage the data sets used will be described. Then the obtained results will be presented, these results are comprised by those achieved by the solver which simply executed all instances with all heuristics and those results obtained by the SOAM.

5.1 Data sets

The data sets used were obtained from CVRPLIB repository as explained on Chapter 4 Section 4.1.1 and they come from 9 different sets from the literature and a total of 225 instances were used. These 225 instances are not all of the available on the repository since some of those, like the ones from Christofides et al. (1979) were left out since they were not new instances, but variations of other instances like exposed in Uchoa et al. (2017) "Instances CMT1, CMT2, CMT3, CMT4, CMT5, CMT11, and CMT12 correspond to instances E-n51-k5, E-n76-k10, E-n101-k8, M-n151-k12, M-n200-k16, M-n121-k7, and M-n100-k10, respectively. The only difference are the conventions...".

The most common sets that are used on testing new methods to solve the CVRP as seen on Uchoa et al. (2017) are from set A, B and P proposed by Augerat et al. (1995) and sets E, F and M that were proposed by Christofides and Eilon (1969), Fisher (1994) and Christofides et al. (1979), respectively. This group of sets are known as the ABEFMP standard, these instances besides the fact that mostly are provided with Euclidean distances, they follow a naming standard that reflects the number of points and routes of the problem like instance B-n44-k7 comes from set B, has 44 nodes and a requirement of 7 vehicles.

On the following sections the color code is blue for the depot node and red for the client nodes.

5.1.1 Set A

The instances provided in set A are found in Augerat et al. (1995) and this set is comprised of 27 instances. These instances were generated by choosing coordinates in random positions within a grid of $[0, 100] \times [0, 100]$ and the demands are determined by an uniform distribution, but that a tenth of those demands are multiplied by 3.

The instance files found in the library did not require any modification since they had all the information needed, including the optimal value of each instance. A visual representation of a single instance from this set can be seen on Figure 5.1.

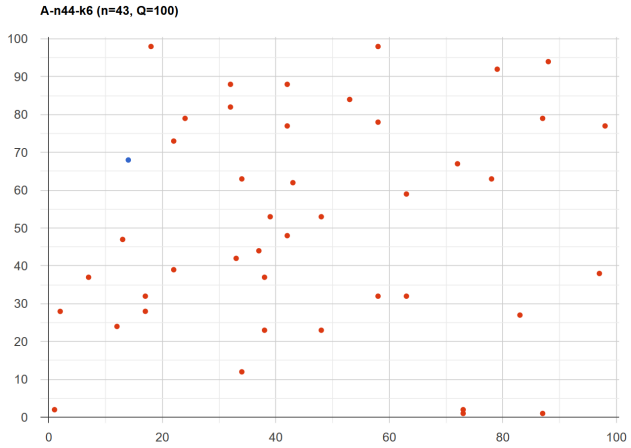


Figure 5.1: Instance A-n44-k6

5.1.2 Set B

The instances provided in set B are also from Augerat et al. (1995) and this set is comprised of 23 instances. These instances were generated by choosing coordinates within a grid of $[0, 100] \times [0, 100]$, those coordinates are chosen in such a way that they form clusters of nodes like can be seen on Figure 5.2.

The demands, like set A are determined by an uniform distribution that has a tenth of those demands multiplied by 3. Just like set A the instance files found in the library did not require any modification since they had all the information required.

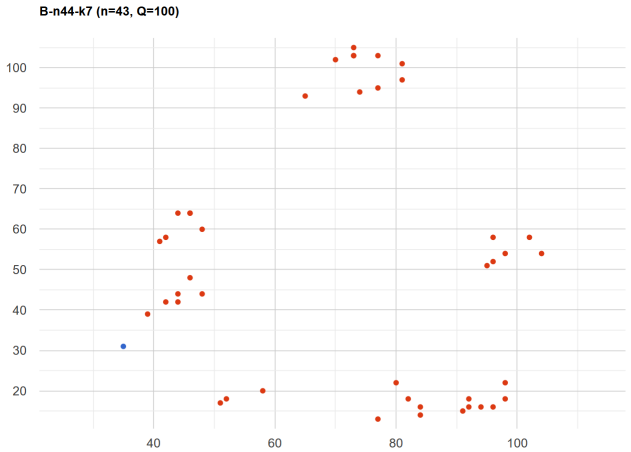


Figure 5.2: Instance B-n44-k7

5.1.3 Set E

The instances provided in set E are commonly associated with Christofides and Eilon (1969) are found in several pieces of literature like Dantzig and Ramser (1959) and Gaskell (1967). This set is comprised of 13 instances but from those two are not used because of the format they are provided in.

Of some of these instances there is no information on how they were generated, some are modifications of others of this same set and others are generated at random from a uniform distribution like the one observed in Figure 5.3. In either case the majority of the instances from this set are characterized by having the nodes equally spaced among each other.

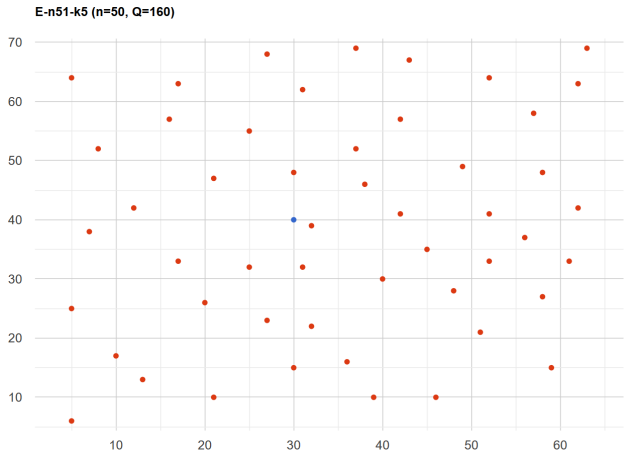


Figure 5.3: Instance E-n51-k5

5.1.4 Set F

The instances provided in set F are also found in Fisher (1994) and this set is only composed of 3 instances. These instances are generated from real life scenarios, two of them are from a day of grocery deliveries from the Peterboro (Ontario terminal) of National Grocers Limited and the other one from Exxon to the delivery of goods to gasoline service stations. A visual representation of a single instance from this set can be seen on Figure 5.4.

5.1.5 Set M

The instances provided in set M can be found in Christofides et al. (1979) and this set is comprised of 5 instances. The data of these instances was generated either by modifying other instances from set E or M or by grouping nodes into clusters as to represent practical cases found in the literature as the one that is represented in Figure 5.5.

5.1.6 Set P

The instances provided in set P are also found in Augerat et al. (1995) and this set is comprised of 24 capacity modified instances from sets A, B and E. The instance files found in the library did not require any modification since they had all the information needed, including the optimal value of each instance. A visual representation of a single instance from this set can be seen on Figure 5.6.

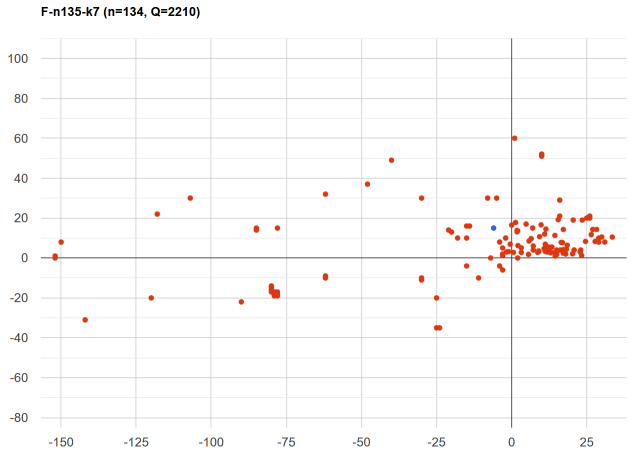


Figure 5.4: Instance F-n135-k7

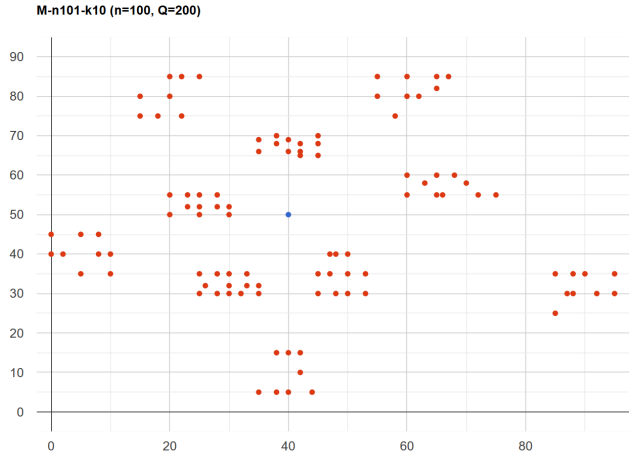


Figure 5.5: Instance M-n101-k10

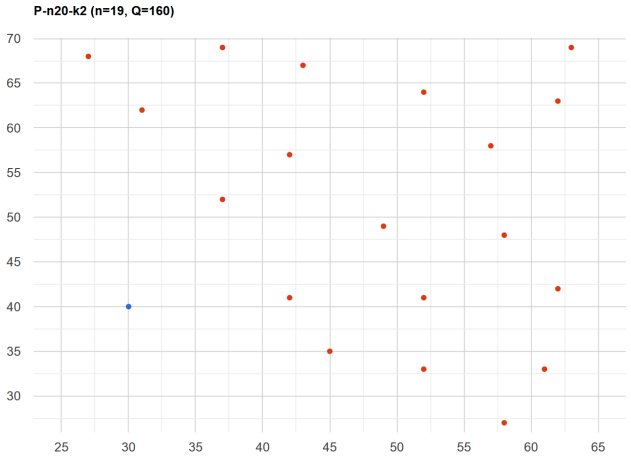


Figure 5.6: Instance P-n20-k2

5.1.7 Set Golden

The instances provided in this set are the ones proposed in Golden et al. (1998) and this set is comprised of 20 instances. The data of these instances was generated by creating several concentric geometric shapes like six-pointed stars, squares and circles. On most of these instances the depot node is in the center of the shapes but on some of concentric squares the depot node is placed on one of the square corners. A visual representation of one instance with six-pointed stars can be observed in Figure 5.7

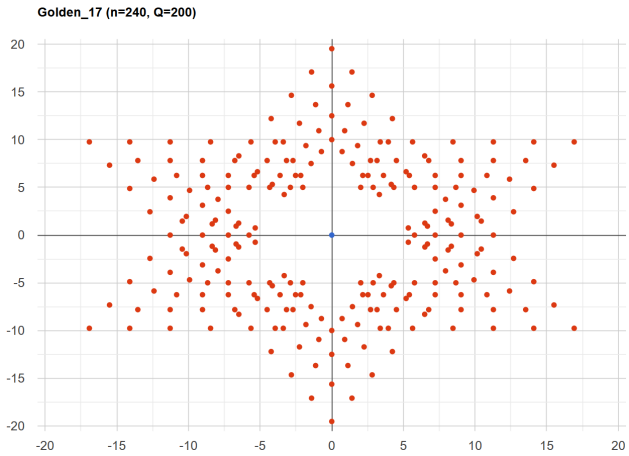


Figure 5.7: Instance Golden17

5.1.8 Set Li

The instances provided on this set can be found in Li et al. (2005) and this set is comprised of 12 large scale instances. In these instances the client nodes are placed concentrically to the depot like observed on Figure 5.8.

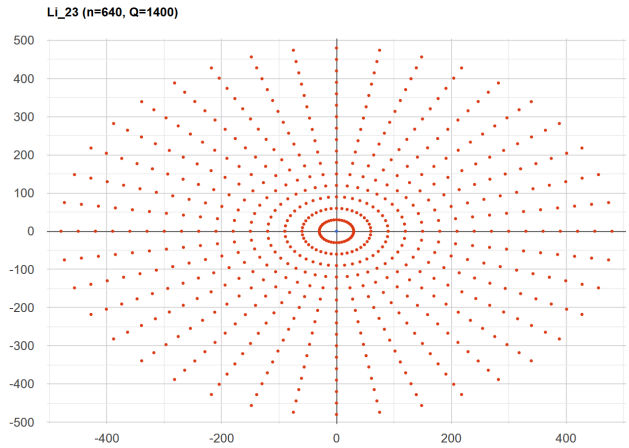


Figure 5.8: Instance Li23

5.1.9 Set X

The instances provided in set X are the benchmark proposed by Uchoa et al. (2017) and this set is comprised of 100 instances. These instances are generated according to different attributes those being: depot position; client position; demand distribution and average route size. A visual representation of an instance with eccentric depot and random-clustered clients can be seen on Figure 5.9

The files provided associated with these instances were modified to include the value of the current best known solution since they did not have it present. The value of the current best known solution of these instances was found in the CVRPLIB repository.

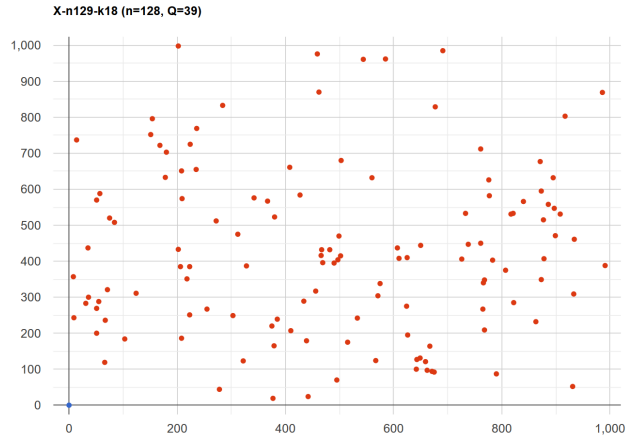


Figure 5.9: Instance X-n129-k18

5.2 Instance Solver

To be able to provide the baseline results from the selected instances the implemented solver was executed on a single machine with the following specifications:

Processor: Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz

Installed memory (RAM): 16,0 GB

System Type: 64-bit operating system, x64-based processor

The obtained results were saved to a .txt file so that the solver did not have to be executed every time those results were required. The results are also displayed as a table using the Python library *pandas* and a small sample of the resulting table can be seen on Table 5.1.

With a quick glance at the table one can observe that for instance A-n32-k5 the first solution heuristic of the savings algorithm failed to provide a solution and that in instance X-n143-k7 even though it was not the worse result achieved by an heuristic the Local Cheapest Arc heuristic took almost 7 times more to achieve a solution than any other heuristic.

Table 5.1: Sample of the Instance Solver Results

Instance	Best Solution	First Solution	Total Distance	Run Time	Result Gap
A-n32-k5	784* Optimal	PARALLEL_CHEAPEST_INSERTION	784	61,5	0
		PATH_MOST_CONSTRAINED_ARC	796	40	1,531
		PATH_CHEAPEST_ARC	796	62	1,531
		LOCAL_CHEAPEST_INSERTION	816	41	4,082
		LOCAL_CHEAPEST_ARC	824	36	5,102
		GLOBAL_CHEAPEST_ARC	831	40	5,995
		CHRISTOFIDES	839	57	7,015
		FIRST_UNBOUND_MIN_VALUE	1197	58	52,679
		SAVINGS		8,5	
A-n33-k5	661* Optimal	PATH_CHEAPEST_ARC	671	61	1,513
		GLOBAL_CHEAPEST_ARC	673	42	1,815
		PATH_MOST_CONSTRAINED_ARC	675	46	2,118
		LOCAL_CHEAPEST_ARC	684	86,5	3,48
		FIRST_UNBOUND_MIN_VALUE	685	63,5	3,631
		SAVINGS	692	35	4,69
		PARALLEL_CHEAPEST_INSERTION	717	38	8,472
		LOCAL_CHEAPEST_INSERTION	747	31	13,011
		CHRISTOFIDES	788	49,5	19,213
B-n31-k5	672* Optimal	PATH_CHEAPEST_ARC	672	30	0
		CHRISTOFIDES	677	38	0,744
		PARALLEL_CHEAPEST_INSERTION	677	38	0,744
		GLOBAL_CHEAPEST_ARC	678	40	0,893
		LOCAL_CHEAPEST_ARC	678	42	0,893
		LOCAL_CHEAPEST_INSERTION	686	33	2,083
		SAVINGS	686	39	2,083
		PATH_MOST_CONSTRAINED_ARC	688	44	2,381
		FIRST_UNBOUND_MIN_VALUE	688	59,5	2,381
X-n143-k7	15700* Optimal	GLOBAL_CHEAPEST_ARC	16557	1105,5	5,459
		LOCAL_CHEAPEST_ARC	17151	13249,5	9,242
		PARALLEL_CHEAPEST_INSERTION	17303	1210,5	10,21
		PATH_CHEAPEST_ARC	17376	1000,5	10,675
		CHRISTOFIDES	17533	1806	11,675
		PATH_MOST_CONSTRAINED_ARC	17543	842,5	11,739
		FIRST_UNBOUND_MIN_VALUE	17705	1957,5	12,771
		SAVINGS	18766	641	19,529
		LOCAL_CHEAPEST_INSERTION	19142	644,5	21,924

5.3 Result Analysis

In order to validate the results of the SOAM a k-Fold cross validation method was used with 7 folds, which in the 213 instances that were used creates several validation sets of about 30 instances each. The results obtained after executing the SOAM with the Cross Validation (CV) method can be observed on Tables A.1 to A.7. In an attempt to use the most data points available, the instances which did not provide a valid solution like A-n32-k5 with the Savings method as can be seen on Table 5.1 were assigned a result gap value of 100%.

On this section the analysis of the results obtained from the predictive model will be performed. Under analysis will be the ranks of the solutions provided by the heuristics and also the result gap difference obtained by the same heuristics.

5.3.1 Distribution Analysis

The first performance test performed was a Kruskal-Wallis H Test of the implemented algorithm against the three heuristics with the best Average Rank. The purpose of this test is to assess if the SOAM and the three best heuristics obtain on average the same rank of result gap or if one of them obtains a different average rank.

The data used in this test is the related to fold 6 which is presented in Table A.6. From the table we can extract that the Path Most Constrained Arc, Chirstofides and Path Cheapest Arc heuristics provide the best average rank, therefore they were chosen to perform the test.

Table 5.2: Heuristic ranks for the Kruskal-Wallis test

Algorithm	Heuristic Ranks																												
SOAM	1	3	1	1	1	1	2	1	3	1	1	1	2	1	1	2	2	2	1	3	2	1	1	2	2	3	2	2	3	2
PATH_MOST_CONSTRAINED_ARC	4	4	3	2	1	1	2	1	3	3	4	4	2	3	3	1	1	4	4	2	2	1	1	2	2	1	1	2	3	1
CHRISTOFIDES	3	1	4	4	4	4	1	3	1	1	3	3	1	4	1	4	4	1	3	1	1	3	3	1	4	2	4	4	1	4
PATH_CHEAPEST_ARC	2	2	2	3	3	1	4	4	2	4	1	2	4	2	4	2	2	2	1	3	4	4	4	4	1	3	2	1	2	2

The hypotheses for this test are:

H_0 - The four algorithms have the same rank distribution.

H_1 - At least two of the algorithms do not have the same rank distribution.

From the compiled ranks from the chosen heuristics found on Table 5.2 the test will be performed with a 5% level of significance, which means $\alpha = 0,05$. For this data we have a n value of 120 and a k value of 4, using Equation (5.1) the determined H_{STAT} value is of 10,064.

$$H = \frac{12}{n(n+1)} \sum_{i=1}^{k} \frac{R_i^2}{n_i} - 3(n+1) \tag{5.1}$$

Since the number of degrees of freedom equals $k-1$ then we have 3 degrees of freedom. With this information by looking in chi-square probabilities we can determine we have a χ^2_{CRIT} value of 7,815 and we can also determine the p-value which is of 0,0180.

As we have $\chi^2_{CRIT} < H_{STAT}$ and also $p-value < \alpha$ we can reject the null hypothesis and assume the rank distribution is not equal to the algorithms.

5.3.2 Performance Analysis

Because the null hypothesis H_0 was rejected then a left-tailed Wilcoxon Signed Rank Test with paired sample was performed for the combinations of the previous heuristics with the SOAM like follows:

- SOAM $\times$ Path Most Constrained Arc
- SOAM $\times$ Chirstofides
- SOAM $\times$ Path Cheapest Arc

The objective of this particular statistical test is to ascertain if the SOAM has the same performance of the heuristic it is being compared to, or if the SOAM performs better. In the case of the SOAM displaying a better performance than the compared heuristic then the null hypothesis will be rejected.

In all of the following tests the hypotheses will be:

H_0 - Rank mean difference = 0

H_1 - Rank mean difference > 0

$\gg$ **SOAM $\times$ Path Most Constrained Arc**

After performing the calculations of the differences between the SOAM ranks and the Path Most Constrained Arc ranks and determining the ranks of the absolute differences as can be seen on Table 5.3 the values of W^+ and W^- were determined by adding the positive and negative ranks respectively. So the value of W_{STAT} was determined by $W_{STAT} = min(W^+, W^-) = 30,5$. Since this is a one-tailed Wilcoxon test and $n < 17$ then the exact approach has to be used to determine W_{CRIT}. By looking into the Wilcoxon Signed-Ranks Table the value will be the one where $n = 17$ and $\alpha = 0,05 * 2$ since it is one-tailed which gives a W_{CRIT} value of 41.

As $W_{STAT} < W_{CRIT}$ then the null hypothesis is rejected, meaning the SOAM has a better performance than Path Most Constrained Arc if analysed by rank. In any case,the p-value of the test was determined and is 0,0147 and since it is less than α then the null hypothesis can be safely rejected.

$\gg$ **SOAM $\times$ Chirstofides**

After performing the calculations of the differences between the SOAM ranks and the Chirstofides ranks and determining the ranks of the absolute differences like can be seen on Table 5.4 the values of W^+ and W^- were determined by adding the positive and negative ranks respectively. So the value of W_{STAT} was determined by $W_{STAT} = min(W^+, W^-) = 81$. Since this is a one-tailed Wilcoxon test the value on the Wilcoxon Signed-Ranks Table will be the one where $n = 28$ and $\alpha = 0,05 * 2$ which gives a W_{CRIT} value of 130.

As $W_{STAT} < W_{CRIT}$ then the null hypothesis is rejected, meaning the SOAM has a better performance than Chirstofides if analysed by rank. The p-value was also calculated giving a value of 0,0027 which as being lesser than the level of significance of 5% confirms that the null hypothesis can be rejected.

$\gg$ **SOAM $\times$ Path Cheapest Arc**

Repeating the same procedure exposed on the previous section which is recorded on Table 5.5 the values of W^+ and W^- were determined by adding the positive and negative ranks respectively. So the value of W_{STAT} was determined to be $W_{STAT} = min(W^+, W^-) = 25$. Since this is a one-tailed Wilcoxon test the value

Table 5.3: SOAM × Path Most Constrained Arc Wilcoxon test

SOAM Ranks	Path Most Constrained Arc	Difference	Positive Ranks	Negative Ranks
1	4	-3		15,5
3	4	-1		4,0
1	3	-2		10,5
1	2	-1		4,0
1	1	0		
1	1	0		
2	2	0		
1	1	0		
3	3	0		
1	3	-2		10,5
1	4	-3		15,5
1	4	-3		15,5
2	2	0		
1	3	-2		10,5
1	3	-2		10,5
2	1	+1	4,0	
2	1	+1	4,0	
2	4	-2		10,5
1	4	-3		15,5
3	2	+1	4,0	
2	2	0		
1	1	0		
1	1	0		
2	2	0		
2	2	0		
3	1	+2	10,5	
2	1	+1	4,0	
2	2	0		
3	3	0		
2	1	+1	4,0	
n = 17		Sum:	30,5	122,5

Table 5.4: SOAM × Chirstofides Wilcoxon test

SOAM Ranks	Chirstofides	Difference	Positive Ranks	Negative Ranks
1	3	-2		15,0
3	1	+2	15,0	
1	4	-3		26,0
1	4	-3		26,0
1	4	-3		26,0
1	4	-3		26,0
2	1	+1	3,5	
1	3	-2		15,0
3	1	+2	15,0	
1	1	0		
1	3	-2		15,0
1	3	-2		15,0
2	1	+1	3,5	
1	4	-3		26,0
1	1	0		
2	4	-2		15,0
2	4	-2		15,0
2	1	+1	3,5	
1	3	-2		15,0
3	1	+2	15,0	
2	1	+1	3,5	
1	3	-2		15,0
1	3	-2		15,0
2	1	+1	3,5	
2	4	-2		15,0
3	2	+1	3,5	
2	4	-2		15,0
2	4	-2		15,0
3	1	+2	15,0	
2	4	-2		15,0
n = 28		Sum:	81	325

Table 5.5: SOAM × Path Cheapest Arc Wilcoxon test

SOAM Ranks	Path Cheapest Arc	Difference	Positive Ranks	Negative Ranks
1	2	-1		5,0
3	2	+1	5,0	
1	2	-1		5,0
1	3	-2		12,5
1	3	-2		12,5
1	1	0		
2	4	-2		12,5
1	4	-3		18,0
3	2	+1	5,0	
1	4	-3		18,0
1	1	0		
1	2	-1		5,0
2	4	-2		12,5
1	2	-1		5,0
1	4	-3		18,0
2	2	0		
2	2	0		
2	2	0		
1	1	0		
3	3	0		
2	4	-2		12,5
1	4	-3		18,0
1	4	-3		18,0
2	4	-2		12,5
2	1	+1	5,0	
3	3	0		
2	2	0		
2	1	+1	5,0	
3	2	+1	5,0	
2	2	0		
n =	20	Sum:	25	185

on the Wilcoxon Signed-Ranks Table will be the one where $n = 20$ and $\alpha = 0,05 * 2$ which gives a W_{CRIT} value of 60.

As $W_{STAT} < W_{CRIT}$ then the null hypothesis is rejected, meaning the SOAM has a better performance than Path Cheapest Arc if analysed by rank. The p-value of this statistic was determined and found to be 0,0014 which proves that the null hypothesis can be safely rejected since it is less than α.

5.3.2.1 Folds Analysis

The same process of performing a Kruskal-Wallis H Test followed by the Wilcoxon Signed Rank Test for the top three heuritics was repeated with all of the k-folds from the CV method. The results of this analysis of all folds can be seen on Table 5.6.

The reason for repeating the process to all the folds is due to being able to test the entire SOAM. Since the performance of all algorithms - SOAM and the heuristics - varies in accordance with the used instances.

On the table we can observe that only on 1 of 7 folds we could safely reject the null hypothesis with a confidence level of 95% - or level of significance of 5%.

In Figures 5.10 to 5.12 it can be seen a box-plot with whiskers of the ranks obtained by each algorithm for each fold.

There is still some noticeable differences in between heuristics like for example the First Unbound Min Value heuristic even though somewhat consistent with the ranks obtained is one of those with the poorest performance. While for instance Global Cheapest Arc is everything but consistent on the results since in all folds the box-plot goes from the minimum value possible 1 to the maximum value of 9.

Table 5.6: Analysis of the k-fold CV

Fold	Heuristics to Compare	Kruskal-Wallis H Test			Wilcoxon Signed Rank Test			
	$\alpha = 5\%$	H Value	p-value	Reject H_0	n	W Value	p-value	Reject H_0
1	Path Most Constrained Arc	5,961	0,1135	No	20	38,5	0,0065	Yes
	Local Cheapest Arc				23	94,5	0,0929	No
	Global Cheapest Arc				27	100,5	0,0167	Yes
2	Path Cheapest Arc	4,221	0,2386	No	24	112,0	0,1388	No
	Path Most Constrained Arc				13	30,0	0,1394	No
	Parallel Cheapest Insertion				29	124,5	0,0222	Yes
3	Path Most Constrained Arc	7,562	0,0560	No	14	20,0	0,0207	Yes
	Christofides				30	118,0	0,0093	Yes
	Local Cheapest Arc				23	51,5	0,0043	Yes
4	Path Most Constrained Arc	4,795	0,1875	No	17	54,0	0,1434	No
	Path Cheapest Arc				17	36,0	0,0276	Yes
	Christofides				30	125,0	0,0135	Yes
5	Global Cheapest Arc	2,015	0,5693	No	28	190,5	0,3880	No
	Path Cheapest Arc				22	124,5	0,4741	No
	Path Most Constrained Arc				18	47,5	0,0490	Yes
6	Path Most Constrained Arc	10,064	0,0180	Yes	17	30,5	0,0147	Yes
	Christofides				28	81,0	0,0027	Yes
	Path Cheapest Arc				20	25,0	0,0014	Yes
7	Path Cheapest Arc	4,012	0,2602	No	21	78,5	0,0992	No
	Path Most Constrained Arc				17	36,0	0,0276	Yes
	Global Cheapest Arc				28	119,0	0,0279	Yes

However like expected from the data on Table 5.6 in some folds, like fold 5 for example, there are not many differences between the SOAM and the Path Cheapest Arc Heuristic. This small difference could already be an indicator that in that case it would be difficult to prove a better performance by the SOAM .

Even though the Kruskal-Wallis H Test was not able to reject the null hypothesis this does not however confirm it. For example on fold 3 even though H_0 was not rejected it was by a small difference. This may be justified by the lack of power of the Kruskal-Wallis H Test.

5.3.2.2 Result Gap Analysis

A second stage of analysis was performed with basis on the result gap. The objective of this analysis was to be able to determine which heuristic had a better result gap difference on average for each fold. Then by determining the average of those values obtain the global average of result gap difference. With this information it is possible to discern what algorithm provides the closest solutions to the optimal value.

The information from tables A.1 to A.7 was combined and provided the results seen on Tables 5.7 and 5.8. All the values presented on the tables are in percentage.

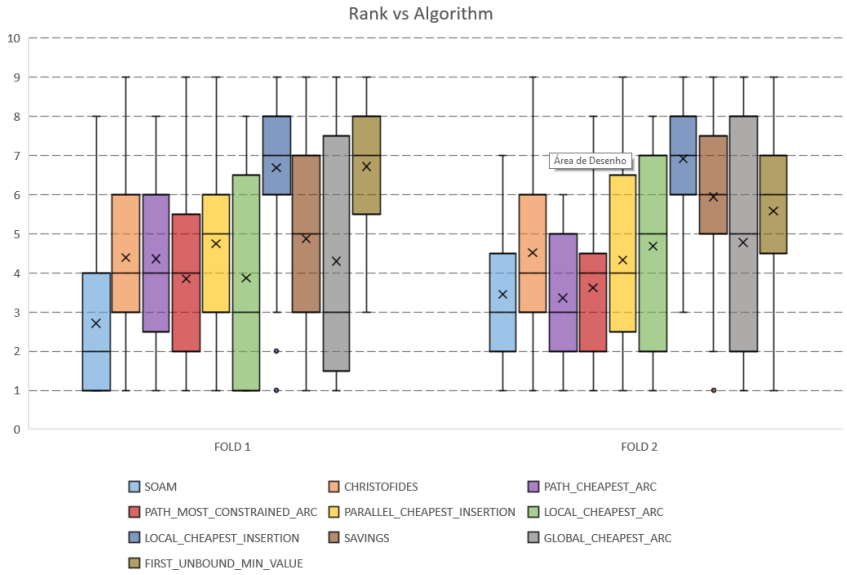


Figure 5.10: Rank vs Algorithm Fold 1 and 2

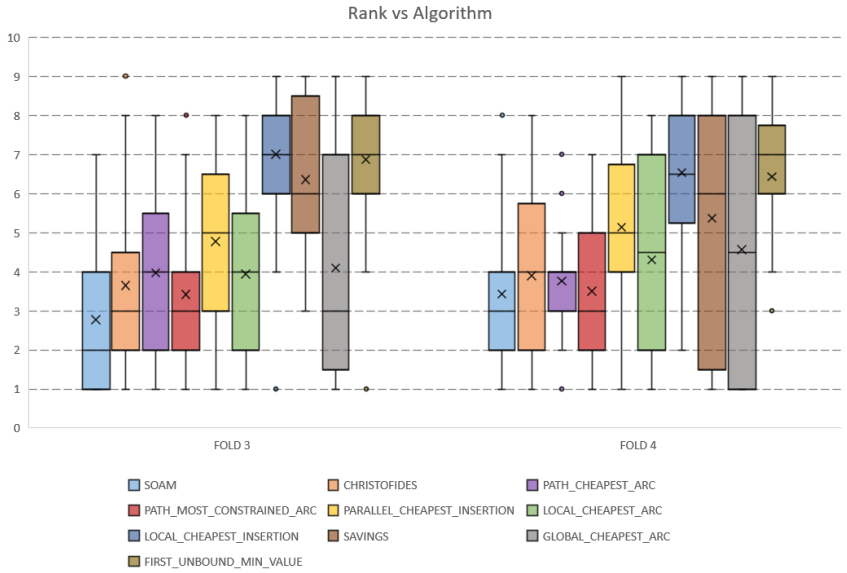


Figure 5.11: Rank vs Algorithm Fold 3 and 4

Table 5.7: Gap Differences to Best Solution Fold 1 through 4

Heuristic	Fold 1			Fold 2			Fold 3			Fold 4		
	Min	Max	Difference to Best	Min	Max	Difference to Best	Min	Max	Difference to Best	Min	Max	Difference to Best
SOAM	-2,335	24,880	***2,588***	-7,107	21,923	3,471	-1,643	12,216	***2,011***	-0,910	16,194	***2,595***
CHRISTOFIDES	0,000	24,880	5,469	-5,338	100,000	7,872	-0,829	28,293	4,484	0,000	19,149	3,579
FIRST_UNBOUND_MIN_VALUE	1,373	100,000	33,644	-5,219	100,000	18,229	0,000	100,000	25,200	-1,719	100,000	21,083
GLOBAL_CHEAPEST_ARC	0,612	100,000	31,516	-0,345	100,000	41,240	-2,156	100,000	25,574	0,000	100,000	41,270
LOCAL_CHEAPEST_ARC	-1,127	100,000	23,311	-6,793	100,000	31,993	-1,327	100,000	14,470	0,000	100,000	26,929
LOCAL_CHEAPEST_INSERTION	2,237	53,178	11,324	-5,923	100,000	17,501	-0,829	57,853	13,713	0,000	100,000	14,015
PARALLEL_CHEAPEST_INSERTION	0,000	24,212	5,815	-2,503	30,278	4,483	-1,327	22,659	5,187	0,966	26,015	6,062
PATH_CHEAPEST_ARC	0,000	19,001	4,692	-7,107	18,644	3,396	-1,327	17,947	4,414	-0,910	17,274	3,292
PATH_MOST_CONSTRAINED_ARC	-2,335	22,472	4,534	-5,420	17,313	***2,967***	-1,643	17,486	3,002	-1,820	15,764	3,152
SAVINGS	1,789	100,000	23,349	-4,858	100,000	22,285	2,083	100,000	35,270	-2,022	100,000	25,374

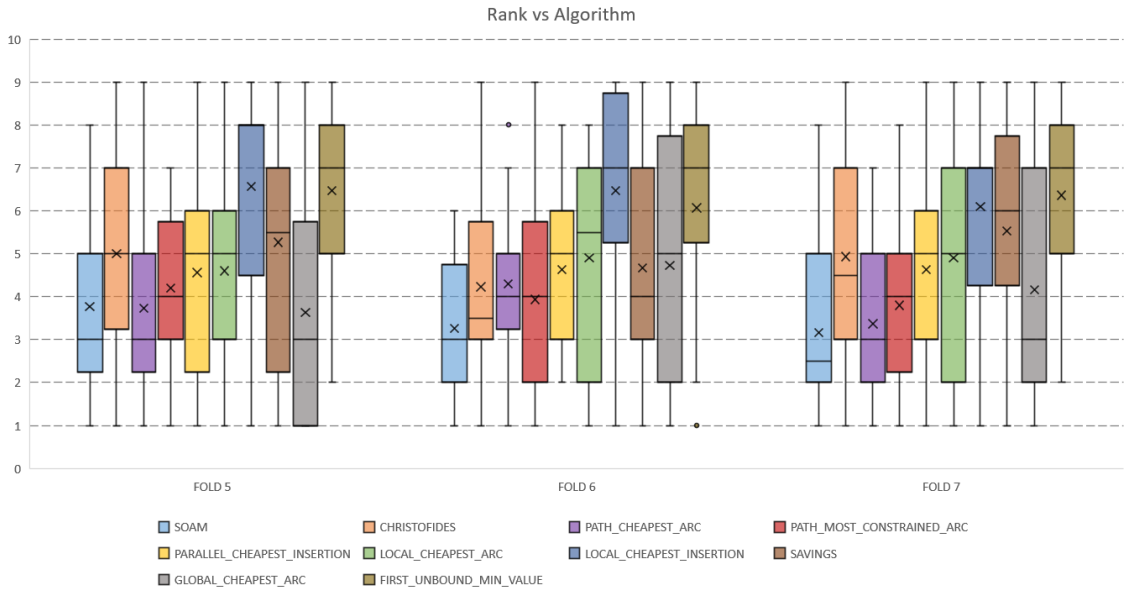


Figure 5.12: Rank vs Algorithm Fold 5, 6 and 7

Table 5.8: Gap Differences to Best Solution Fold 5 through 7

Heuristic	Fold 5			Fold 6			Fold 7			Average
	Min	Max	Difference to Best	Min	Max	Difference to Best	Min	Max	Difference to Best	Difference to Best
SOAM	-5,225	13,901	3,155	0,176	17,835	***2,274***	0,000	16,878	***3,195***	***2,755***
CHRISTOFIDES	-3,572	100,000	8,368	-0,586	15,719	3,519	1,109	31,397	6,480	5,682
FIRST_UNBOUND_MIN_VALUE	-3,595	100,000	20,191	0,626	100,000	21,692	0,135	100,000	32,158	24,600
GLOBAL_CHEAPEST_ARC	-7,615	100,000	19,882	-0,928	100,000	34,878	-1,830	100,000	38,057	33,202
LOCAL_CHEAPEST_ARC	-5,431	100,000	15,624	0,604	100,000	29,325	0,000	100,000	33,289	24,991
LOCAL_CHEAPEST_INSERTION	-5,837	55,548	9,330	0,176	37,194	9,623	0,567	62,013	12,682	12,598
PARALLEL_CHEAPEST_INSERTION	-2,148	26,747	4,624	0,439	22,445	3,595	0,451	100,000	8,433	5,457
PATH_CHEAPEST_ARC	-5,225	19,962	***3,026***	0,176	24,561	3,323	-0,340	25,738	3,718	3,694
PATH_MOST_CONSTRAINED_ARC	-5,225	14,217	3,712	0,176	18,818	3,665	0,540	21,487	4,148	3,597
SAVINGS	-4,590	100,000	23,349	-6,052	100,000	10,297	0,650	100,000	21,822	23,107

The values highlighted in green in each column are those that have the lowest difference to the best case. As it can be seen the SOAM has the best difference to the best solution in 5 out of 7 folds. As such it also has the best Average Difference to the best solution with a value of 2,76%.

5.3.3 Feature Importance

While not explored further the list of feature importance's for each regression model was extrapolated with resource to the Python library *sklearn*. The complete list of importance can be seen on Table 5.9. After examining the importance of each feature for each regression model it was possible to conclude that there was not a dominant feature in the decision process being the highest importance found of 7,184% in the Global Cheapest Arc regression model for Feature I5_+1.

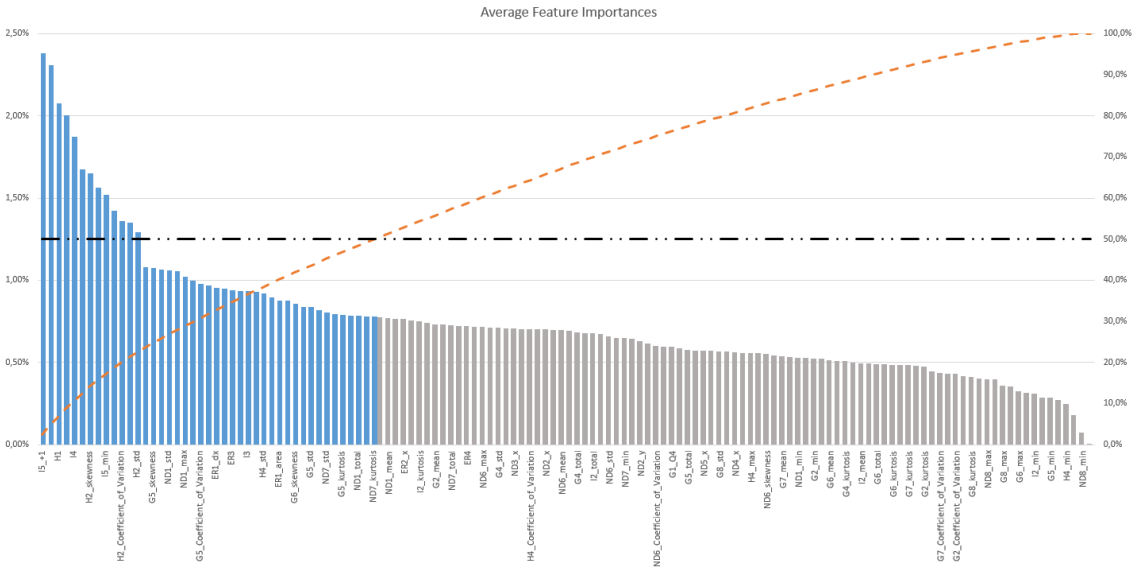


Figure 5.13: Average Feature Importance

The Pareto chart present in Figure 5.13 is created by determining the average feature importance within all regression models. In order to achieve a global importance of 50% it is required to use 43 of the extracted features.

Also there is not a feature that was across the board dominant, from which we can conclude that the determining factors for each kind of regression model are not the same to all heuristics. This is due to the fact that the highest average importance is roughly 2,4% which is not significant.

Table 5.9: Feature Importance by Heuristic

SMAPE	18,08%	49,12%	25,21%	18,17%	20,75%	21,07%	38,63%	18,82%	22,25%
Feature	PATH CHEAPEST ARC	SAVINGS	CHRISTOFIDES	PATH MOST CONSTRAINED ARC	PARALLEL CHEAPEST INSERTION	LOCAL CHEAPEST INSERTION	FIRST UNBOUND MIN VALUE	GLOBAL CHEAPEST ARC	LOCAL CHEAPEST ARC
I1	0,495%	0,498%	0,551%	0,615%	0,894%	0,774%	0,453%	1,249%	0,809%
I2_total	0,682%	0,388%	0,654%	0,937%	0,872%	0,936%	0,681%	0,353%	0,588%
I2_min	0,387%	0,144%	0,727%	0,294%	0,375%	0,157%	0,489%	0,079%	0,139%
I2_mean	0,486%	0,687%	0,597%	0,504%	0,302%	0,453%	0,887%	0,169%	0,368%
I2_max	0,372%	0,202%	0,507%	0,479%	0,222%	0,145%	0,868%	0,132%	0,277%
I2_std	0,718%	0,941%	0,846%	0,889%	1,214%	0,361%	1,596%	0,208%	0,573%
I2_Coefficient_of_Variation	1,344%	1,163%	1,298%	1,193%	2,142%	0,434%	1,495%	0,181%	0,458%
I2_skewness	0,816%	1,090%	0,975%	0,822%	0,972%	2,069%	2,120%	0,299%	0,344%
I2_kurtosis	0,796%	0,710%	1,079%	0,693%	0,822%	0,434%	1,408%	0,265%	0,533%
I3	0,972%	0,912%	1,137%	1,007%	0,765%	1,068%	1,240%	0,577%	0,722%
I4	0,571%	0,947%	1,181%	0,616%	0,753%	0,822%	0,857%	5,991%	5,121%
I5_min	0,883%	1,846%	0,758%	1,215%	0,747%	0,715%	0,592%	3,786%	3,122%
I5_+1	0,951%	1,101%	1,169%	0,985%	0,924%	1,659%	0,950%	7,184%	6,490%
I5_+2	0,863%	1,037%	1,637%	0,904%	1,017%	1,139%	1,040%	7,121%	6,002%
I6	1,163%	4,296%	2,753%	0,960%	1,598%	3,061%	1,122%	1,622%	1,452%
I7	0,915%	3,044%	1,921%	0,766%	1,218%	2,687%	0,982%	1,243%	1,311%
ER1_dx	0,713%	0,529%	0,392%	0,636%	0,849%	0,897%	0,617%	2,413%	1,524%
ER1_dy	0,663%	0,447%	0,415%	0,639%	0,449%	0,758%	0,489%	1,298%	0,680%
ER1_area	0,750%	0,540%	0,428%	0,641%	0,918%	0,975%	0,534%	2,125%	0,995%
ER2_x	0,533%	0,438%	1,087%	0,587%	0,613%	0,333%	0,618%	0,829%	1,849%
ER2_y	0,456%	0,847%	2,176%	0,378%	0,425%	0,641%	0,591%	0,413%	0,977%
ER3	0,934%	0,892%	1,105%	0,641%	1,209%	1,163%	1,178%	0,602%	0,717%
ER4	1,041%	0,587%	0,731%	0,769%	0,668%	0,721%	0,814%	0,356%	0,800%
ND1_total	0,665%	0,716%	0,435%	0,616%	0,410%	0,745%	0,453%	2,340%	0,671%
ND1_min	0,655%	0,562%	0,603%	0,550%	0,520%	0,445%	0,603%	0,211%	0,615%
ND1_mean	0,892%	0,596%	0,399%	0,579%	0,602%	0,688%	0,446%	1,575%	1,159%
ND1_max	0,953%	0,499%	0,804%	0,579%	0,845%	2,808%	0,566%	1,170%	0,949%
ND1_std	0,904%	0,508%	0,843%	0,754%	1,643%	3,097%	0,408%	0,810%	0,556%
ND1_Coefficient_of_Variation	0,951%	0,690%	1,466%	1,489%	1,604%	0,869%	0,704%	0,248%	0,389%
ND1_skewness	0,555%	1,422%	1,020%	0,984%	0,501%	0,417%	0,853%	0,370%	0,387%
ND1_kurtosis	0,953%	0,495%	0,537%	0,688%	0,572%	0,532%	1,066%	0,569%	0,403%
ND2_x	0,633%	0,516%	0,508%	0,508%	0,476%	0,532%	1,122%	0,656%	1,367%
ND2_y	0,467%	0,592%	0,500%	0,382%	1,154%	0,744%	0,459%	0,376%	1,006%
ND3_x	0,705%	0,498%	0,698%	0,682%	0,465%	0,689%	0,869%	0,474%	1,263%
ND3_y	0,677%	0,635%	0,499%	0,618%	1,516%	0,506%	0,532%	0,294%	1,007%
ND4_x	0,834%	0,477%	0,538%	0,602%	0,637%	0,804%	0,586%	0,223%	0,352%
ND4_y	0,584%	0,451%	0,503%	0,832%	1,873%	0,849%	0,591%	0,205%	0,481%
ND5_x	0,711%	0,634%	0,816%	0,635%	0,574%	0,562%	0,504%	0,248%	0,462%
ND5_y	0,508%	0,438%	0,367%	0,675%	1,963%	1,089%	0,598%	0,257%	0,429%
ND6_total	0,614%	0,629%	0,581%	0,608%	0,820%	0,965%	0,475%	1,016%	0,718%
ND6_min	0,771%	0,491%	0,850%	0,729%	0,689%	1,004%	0,513%	0,829%	0,943%
ND6_mean	0,637%	0,635%	0,475%	0,648%	0,622%	1,092%	0,491%	0,939%	0,730%
ND6_max	0,577%	0,704%	0,732%	0,574%	0,733%	0,812%	0,493%	1,194%	0,640%
ND6_std	0,555%	0,457%	0,601%	0,581%	0,594%	0,638%	0,476%	1,399%	0,633%
ND6_Coefficient_of_Variation	0,757%	0,689%	0,657%	0,836%	0,471%	0,585%	0,601%	0,483%	0,344%
ND6_skewness	0,508%	0,476%	0,625%	0,647%	0,762%	0,534%	0,571%	0,426%	0,401%
ND6_kurtosis	0,910%	0,962%	0,840%	1,009%	0,707%	0,634%	0,575%	0,453%	0,381%
ND7_total	0,653%	0,663%	0,655%	0,629%	0,687%	1,062%	0,449%	1,044%	0,687%
ND7_min	0,655%	0,711%	0,689%	0,867%	0,754%	0,871%	0,550%	0,335%	0,404%
ND7_mean	0,832%	0,590%	0,913%	0,561%	0,638%	1,078%	0,574%	0,556%	0,370%
ND7_max	0,501%	0,603%	0,620%	0,590%	0,574%	0,350%	0,552%	0,313%	0,259%
ND7_std	0,634%	1,149%	1,233%	1,032%	0,706%	1,019%	0,792%	0,333%	0,328%
ND7_Coefficient_of_Variation	1,027%	0,729%	1,077%	0,929%	1,344%	1,414%	0,515%	0,660%	0,368%
ND7_skewness	0,568%	0,558%	0,868%	0,674%	0,917%	0,451%	0,624%	0,395%	0,327%
ND7_kurtosis	0,942%	1,277%	0,796%	0,887%	0,589%	1,020%	0,608%	0,456%	0,430%
ND8_total	1,446%	0,384%	0,741%	1,322%	0,766%	0,496%	0,530%	0,572%	0,814%
ND8_min	0,033%	0,133%	0,031%	0,102%	0,043%	0,007%	0,209%	0,001%	0,112%
ND8_mean	1,549%	0,628%	0,931%	0,900%	0,857%	0,643%	0,702%	0,415%	0,386%
ND8_max	0,573%	0,338%	0,641%	0,389%	0,449%	0,238%	0,419%	0,233%	0,317%
ND8_std	1,487%	0,602%	0,867%	1,174%	1,061%	0,387%	0,568%	0,354%	0,486%
ND8_Coefficient_of_Variation	1,223%	1,095%	0,773%	1,007%	0,719%	0,938%	1,305%	0,332%	0,481%
ND8_skewness	1,632%	0,834%	0,609%	0,941%	0,697%	0,879%	1,656%	0,549%	0,576%
ND8_kurtosis	1,630%	0,602%	0,562%	0,567%	0,623%	1,303%	1,960%	0,694%	0,609%
G1_Q1	0,524%	1,157%	0,484%	0,462%	0,503%	1,089%	0,634%	0,326%	0,337%
G1_Q2	0,392%	0,371%	0,492%	0,529%	0,417%	0,314%	0,396%	0,352%	0,306%
G1_Q3	0,727%	0,539%	0,365%	0,729%	0,564%	0,442%	0,474%	0,299%	0,166%
G1_Q4	0,926%	0,311%	0,479%	0,810%	1,344%	0,481%	0,511%	0,360%	0,137%
G2_total	0,573%	0,558%	0,638%	0,645%	0,866%	0,792%	0,388%	1,388%	0,722%
G2_min	0,676%	0,287%	0,405%	0,590%	1,249%	0,619%	0,300%	0,403%	0,177%
G2_mean	0,620%	0,565%	0,673%	0,612%	0,931%	0,690%	0,364%	1,440%	0,691%
G2_max	0,948%	0,620%	0,614%	1,171%	0,399%	0,773%	0,547%	0,648%	0,529%
G2_std	0,463%	0,483%	0,419%	0,328%	0,330%	0,584%	0,458%	0,271%	0,671%
G2_Coefficient_of_Variation	0,334%	0,343%	0,658%	0,435%	0,772%	0,459%	0,393%	0,177%	0,298%
G2_skewness	0,343%	1,516%	2,733%	0,384%	0,486%	0,510%	0,948%	0,172%	0,468%
G2_kurtosis	0,440%	0,591%	0,698%	0,524%	0,518%	0,615%	0,439%	0,144%	0,319%
G3_Q1	0,486%	0,592%	0,531%	0,674%	0,398%	1,262%	0,677%	0,195%	0,314%
G3_Q2	0,462%	0,755%	0,352%	0,585%	0,438%	0,334%	0,981%	0,216%	0,439%
G3_Q3	1,115%	0,438%	0,450%	0,932%	0,568%	0,318%	0,660%	0,124%	0,137%
G3_Q4	1,028%	0,303%	0,441%	1,042%	1,288%	0,294%	0,438%	0,120%	0,127%
G4_total	0,769%	0,370%	0,845%	0,751%	0,713%	1,015%	0,644%	0,344%	0,682%
G4_mean	0,758%	0,257%	0,621%	0,918%	0,851%	0,885%	0,754%	0,365%	0,638%
G4_std	0,888%	0,463%	0,342%	0,612%	0,856%	1,376%	0,934%	0,249%	0,694%
G4_Coefficient_of_Variation	0,514%	0,441%	0,602%	0,497%	0,864%	1,101%	0,410%	0,217%	0,384%
G4_skewness	0,499%	0,931%	0,551%	0,450%	0,498%	0,353%	0,700%	0,182%	0,276%
G4_kurtosis	0,553%	0,452%	0,531%	0,587%	0,599%	0,919%	0,523%	0,099%	0,297%
G6_total	0,453%	0,627%	0,386%	0,568%	0,481%	0,391%	0,932%	0,178%	0,409%
G6_min	0,369%	0,364%	0,607%	0,307%	0,295%	0,234%	0,382%	0,139%	0,139%
G6_mean	0,409%	0,522%	0,379%	0,738%	0,429%	0,836%	0,743%	0,188%	0,371%
G6_max	0,316%	0,378%	0,411%	0,350%	0,255%	0,142%	0,803%	0,108%	0,151%
G6_std	0,627%	0,618%	0,488%	0,642%	0,336%	0,265%	1,221%	0,134%	0,151%
G6_Coefficient_of_Variation	0,734%	0,744%	0,759%	0,810%	0,491%	0,294%	0,620%	0,110%	0,125%
G6_skewness	0,898%	1,161%	1,360%	0,889%	0,450%	0,562%	1,841%	0,272%	0,272%
G6_kurtosis	0,518%	0,674%	0,612%	0,593%	0,387%	0,258%	0,712%	0,157%	0,471%
G7_total	1,121%	0,470%	0,518%	0,807%	0,544%	0,351%	0,630%	0,146%	0,211%
G7_min	0,222%	0,360%	0,428%	0,293%	0,173%	0,470%	0,261%	0,096%	0,125%
G7_mean	0,946%	0,502%	0,761%	1,084%	0,521%	0,245%	0,460%	0,158%	0,173%
G7_max	0,776%	0,251%	0,553%	0,472%	0,382%	0,246%	0,714%	0,121%	0,097%
G7_std	1,524%	0,471%	0,507%	0,852%	0,490%	0,288%	0,796%	0,151%	0,181%
G7_Coefficient_of_Variation	0,592%	0,445%	0,627%	0,659%	0,420%	0,293%	0,505%	0,260%	0,140%
G7_skewness	0,814%	0,730%	0,787%	0,689%	0,307%	0,377%	0,974%	0,295%	0,183%
G7_kurtosis	0,893%	0,585%	0,441%	0,410%	0,358%	0,404%	0,868%	0,240%	0,149%
G8_total	0,984%	0,293%	0,415%	0,845%	1,151%	0,358%	0,509%	0,183%	0,125%
G8_min	0,273%	0,289%	0,252%	0,268%	0,375%	0,363%	0,492%	0,116%	0,145%
G8_mean	0,587%	0,360%	0,457%	0,721%	0,430%	0,287%	0,687%	0,112%	0,125%
G8_max	0,583%	0,222%	0,430%	0,527%	0,483%	0,170%	0,617%	0,101%	0,092%
G8_std	0,839%	0,417%	0,394%	1,079%	0,876%	0,228%	0,975%	0,136%	0,139%
G8_Coefficient_of_Variation	1,103%	0,941%	0,798%	1,442%	1,695%	0,291%	0,569%	0,158%	0,167%
G8_skewness	0,508%	0,875%	0,871%	0,411%	0,352%	0,217%	0,655%	0,290%	0,206%
G8_kurtosis	0,362%	0,552%	0,428%	0,460%	0,454%	0,432%	0,577%	0,158%	0,262%
H1	0,558%	0,992%	1,341%	0,639%	0,689%	1,001%	0,822%	6,548%	6,093%
H2_min	1,057%	3,516%	1,855%	2,113%	0,907%	1,545%	0,983%	0,369%	0,468%
H2_mean	1,321%	1,828%	1,011%	1,606%	0,879%	1,431%	0,571%	1,956%	1,549%
H2_max	0,010%	0,007%	0,010%	0,002%	0,002%	0,000%	0,005%	0,000%	0,001%
H2_std	1,276%	2,922%	1,267%	1,775%	0,648%	1,647%	0,756%	0,848%	0,482%
H2_Coefficient_of_Variation	1,360%	2,778%	1,399%	1,901%	0,976%	1,634%	0,785%	0,935%	0,473%
H2_skewness	0,622%	1,020%	0,647%	0,715%	0,636%	0,850%	0,932%	4,068%	5,362%
H2_kurtosis	0,625%	1,160%	0,852%	0,734%	0,620%	0,729%	0,854%	4,216%	5,262%
H3	0,045%	0,158%	0,021%	0,204%	0,041%	0,030%	0,040%	0,347%	0,745%
H4_total	1,560%	0,680%	0,560%	1,096%	0,680%	1,044%	0,502%	2,209%	0,627%
H4_min	0,201%	0,208%	0,325%	0,228%	0,753%	0,162%	0,137%	0,078%	0,140%
H4_mean	0,684%	0,851%	1,458%	0,793%	1,409%	1,305%	0,914%	0,533%	0,762%
H4_max	0,490%	0,824%	0,730%	0,436%	0,307%	0,375%	0,889%	0,350%	0,629%
H4_std	0,944%	1,483%	1,541%	0,606%	0,709%	0,704%	1,080%	0,531%	0,703%
H4_Coefficient_of_Variation	0,986%	0,943%	0,695%	1,285%	0,575%	0,380%	0,813%	0,246%	0,408%
H4_skewness	1,025%	0,870%	0,711%	0,771%	0,939%	0,881%	0,612%	0,412%	0,430%
H4_kurtosis	1,945%	0,813%	0,854%	2,141%	1,425%	0,660%	0,844%	0,341%	0,583%
G5_total	0,484%	0,635%	0,387%	0,637%	0,447%	1,188%	0,825%	0,183%	0,380%
G5_min	0,236%	0,483%	0,573%	0,208%	0,217%	0,218%	0,407%	0,103%	0,110%
G5_mean	0,665%	0,516%	0,543%	0,713%	0,586%	0,450%	1,119%	0,158%	0,244%
G5_max	0,400%	0,302%	0,354%	0,612%	0,333%	0,233%	1,212%	0,121%	0,318%
G5_std	0,678%	0,526%	0,738%	1,138%	1,604%	0,474%	1,739%	0,165%	0,458%
G5_Coefficient_of_Variation	1,200%	0,753%	0,895%	0,932%	2,395%	0,350%	1,807%	0,165%	0,322%
G5_skewness	0,637%	0,842%	0,733%	1,193%	1,231%	2,872%	1,569%	0,231%	0,350%
G5_kurtosis	0,642%	0,740%	0,879%	0,796%	1,262%	0,408%	1,473%	0,287%	0,595%

Chapter 6

Conclusions and future work

On this chapter the work conclusions will be presented as well as some remarks that may be considered during future work on the subject.

6.1 Conclusions

Like mentioned on the beginning of this report the main goal of this project was to be able to develop an algorithm that through machine learning models would be able to compete with state-of-the-art solvers in the resolution of the benchmarks found in the literature. With this in mind the project was planned and developed accordingly in an attempt to achieve a valid algorithm that was able to achieve the determined goal.

First the solver was implemented in an attempt to make the most of the available time since it was predictable that the execution of all the benchmark instances would take more than a day to fully complete. As the project begun several "dry runs" with just 5 different instances were used in an attempt to verify most of the behaviour of the implemented solver.

Then even though the feature extractor was able to extract 137 different features there would be most likely other features that could be used and extracted and maybe even provide a better importance to the models.

Finally the work on the regression models begun and while at start the fact that only about 30% of the full data-set was being used to train, test and validate the models the results obtained were not the expected. This setback was dealt with when the solver was able to execute for the entire data-set and all the instances were used on the training, testing and validation process.

Even though during the development of the project there were good indications that the algorithm was somewhat providing the expected results, it was not until the obtained results were analysed that a complete realization of the algorithm performance was achieved. Like observed in the last chapter it can be affirmed with good confidence that the developed project is able to compete with the current state-of-the-art algorithms.

On a final approach to the problem and goal that was developed I feel confident that the project was able to meet the expected results even though it suffered from some setbacks and it was not developed in its entirety.

6.2 Future Work

As related to future work that can be developed on top of this project there are a several things that come to mind:

First some deep analysis of the current features and their contributions - positive or negative - to each model and then an attempt to find more features to use.

Another point for future work is the implementation of the regression techniques for the local search meta-heuristics, which was not implemented even though originally planned since the time required to obtain the results from the solver on the current machine would take roughly 2 months.

Besides those two points another thing that could later be developed is the ability to make the decision of which is the best heuristics not only based on the value of the result gap but also with the value of the run time for the solver and the number of vehicles required.

Also for future work instead of using the regression techniques to predict based on the instance it is solving, it could be experimented with clustering techniques to identify the "family" of instances and then choosing the best heuristic for that type of "family".

Appendix A

Predictive Algorithm Results

Table A.1: Results of fold 1

Type	Total Result Gap	Average Result Gap	Minimum Result Gap	Maximum Result Gap	Difference to Implemented Algorithm	Heuristic Average Rank
ALWAYS_BEST	134,702	4,345	-2,335	10,464	-2,588	1
SOAM	214,916	6,933	-2,335	24,880		2,710
PATH_MOST_CONSTRAINED_ARC	275,264	8,879	-2,335	22,472	1,947	3,839
PATH_CHEAPEST_ARC	280,151	9,037	0,000	19,001	2,104	4,355
CHRISTOFIDES	304,254	9,815	0,000	24,880	2,882	4,387
PARALLEL_CHEAPEST_INSERTION	314,981	10,161	0,000	24,212	3,228	4,742
LOCAL_CHEAPEST_INSERTION	485,753	15,669	2,237	53,178	8,737	6,677
LOCAL_CHEAPEST_ARC	857,355	27,657	-1,127	100,000	20,724	3,871
SAVINGS	858,508	27,694	1,789	100,000	20,761	4,871
GLOBAL_CHEAPEST_ARC	1111,706	35,861	0,612	100,000	28,929	4,290
FIRST_UNBOUND_MIN_VALUE	1177,661	37,989	1,373	100,000	31,056	6,710

Table A.2: Results of fold 2

Type	Total Result Gap	Average Result Gap	Minimum Result Gap	Maximum Result Gap	Difference to Implemented Algorithm	Heuristic Average Rank
ALWAYS_BEST	142,483	4,596	-7,107	11,605	-3,471	1
PATH_MOST_CONSTRAINED_ARC	234,466	7,563	-5,420	17,313	-0,504	3,613
PATH_CHEAPEST_ARC	247,753	7,992	-7,107	18,644	-0,075	3,355
SOAM	250,088	8,067	-7,107	21,923		3,452
PARALLEL_CHEAPEST_INSERTION	281,461	9,079	-2,503	30,278	1,012	4,323
CHRISTOFIDES	386,514	12,468	-5,338	100,000	4,401	4,516
LOCAL_CHEAPEST_INSERTION	685,004	22,097	-5,923	100,000	14,030	6,903
FIRST_UNBOUND_MIN_VALUE	707,589	22,825	-5,219	100,000	14,758	5,581
SAVINGS	833,316	26,881	-4,858	100,000	18,814	5,935
LOCAL_CHEAPEST_ARC	1134,254	36,589	-6,793	100,000	28,521	4,677
GLOBAL_CHEAPEST_ARC	1420,914	45,836	-0,345	100,000	37,769	4,774

Table A.3: Results of fold 3

Type	Total Result Gap	Average Result Gap	Minimum Result Gap	Maximum Result Gap	Difference to Implemented Algorithm	Heuristic Average Rank
ALWAYS_BEST	118,912	3,836	-2,156	10,369	-2,011	1
SOAM	181,239	5,846	-1,643	12,216		2,774
PATH_MOST_CONSTRAINED_ARC	211,959	6,837	-1,643	17,486	0,991	3,419
PATH_CHEAPEST_ARC	255,756	8,250	-1,327	17,947	2,404	3,968
CHRISTOFIDES	257,903	8,319	-0,829	28,293	2,473	3,645
PARALLEL_CHEAPEST_INSERTION	279,705	9,023	-1,327	22,659	3,176	4,774
LOCAL_CHEAPEST_INSERTION	544,001	17,548	-0,829	57,853	11,702	7,000
LOCAL_CHEAPEST_ARC	567,476	18,306	-1,327	100,000	12,459	3,935
FIRST_UNBOUND_MIN_VALUE	900,114	29,036	0,000	100,000	23,190	6,871
GLOBAL_CHEAPEST_ARC	911,717	29,410	-2,156	100,000	23,564	4,097
SAVINGS	1212,293	39,106	2,083	100,000	33,260	6,355

Table A.4: Results of fold 4

Type	Total Result Gap	Average Result Gap	Minimum Result Gap	Maximum Result Gap	Difference to Implemented Algorithm	Heuristic Average Rank
ALWAYS_BEST	131,058	4,369	-2,022	13,740	-2,595	1
SOAM	208,894	6,963	-0,910	16,194		3,433
PATH_MOST_CONSTRAINED_ARC	225,623	7,521	-1,820	15,764	0,558	3,500
PATH_CHEAPEST_ARC	229,822	7,661	-0,910	17,274	0,698	3,767
CHRISTOFIDES	238,415	7,947	0,000	19,149	0,984	3,900
PARALLEL_CHEAPEST_INSERTION	312,924	10,431	0,966	26,015	3,468	5,133
LOCAL_CHEAPEST_INSERTION	551,499	18,383	0,000	100,000	11,420	6,533
FIRST_UNBOUND_MIN_VALUE	763,558	25,452	-1,719	100,000	18,489	6,433
SAVINGS	892,283	29,743	-2,022	100,000	22,780	5,367
LOCAL_CHEAPEST_ARC	938,914	31,297	0,000	100,000	24,334	4,300
GLOBAL_CHEAPEST_ARC	1369,160	45,639	0,000	100,000	38,676	4,567

Table A.5: Results of fold 5

Type	Total Result Gap	Average Result Gap	Minimum Result Gap	Maximum Result Gap	Difference to Implemented Algorithm	Heuristic Average Rank
ALWAYS_BEST	94,432	3,148	-7,615	10,940	-3,155	1
PATH_CHEAPEST_ARC	185,215	6,174	-5,225	19,962	-0,129	3,733
SOAM	189,072	6,302	-5,225	13,901		3,767
PATH_MOST_CONSTRAINED_ARC	205,777	6,859	-5,225	14,217	0,557	4,200
PARALLEL_CHEAPEST_INSERTION	233,144	7,771	-2,148	26,747	1,469	4,567
CHRISTOFIDES	345,458	11,515	-3,572	100,000	5,213	5,000
LOCAL_CHEAPEST_INSERTION	374,339	12,478	-5,837	55,548	6,176	6,567
LOCAL_CHEAPEST_ARC	563,147	18,772	-5,431	100,000	12,469	4,600
GLOBAL_CHEAPEST_ARC	690,893	23,030	-7,615	100,000	16,727	3,633
FIRST_UNBOUND_MIN_VALUE	700,160	23,339	-3,595	100,000	17,036	6,467
SAVINGS	794,893	26,496	-4,590	100,000	20,194	5,267

Table A.6: Results of fold 6

Type	Total Result Gap	Average Result Gap	Minimum Result Gap	Maximum Result Gap	Difference to Implemented Algorithm	Heuristic Average Rank
ALWAYS_BEST	156,105	5,204	-6,052	13,215	-2,274	1
SOAM	224,310	7,477	0,176	17,835		3,267
PATH_CHEAPEST_ARC	255,798	8,527	0,176	24,561	1,050	4,300
CHRISTOFIDES	261,681	8,723	-0,586	15,719	1,246	4,233
PARALLEL_CHEAPEST_INSERTION	263,969	8,799	0,439	22,445	1,322	4,633
PATH_MOST_CONSTRAINED_ARC	266,061	8,869	0,176	18,818	1,392	3,933
LOCAL_CHEAPEST_INSERTION	444,809	14,827	0,176	37,194	7,350	6,467
SAVINGS	465,028	15,501	-6,052	100,000	8,024	4,667
FIRST_UNBOUND_MIN_VALUE	806,861	26,895	0,626	100,000	19,418	6,067
LOCAL_CHEAPEST_ARC	1035,844	34,528	0,604	100,000	27,051	4,900
GLOBAL_CHEAPEST_ARC	1202,440	40,081	-0,928	100,000	32,604	4,733

Table A.7: Results of fold 7

Type	Total Result Gap	Average Result Gap	Minimum Result Gap	Maximum Result Gap	Difference to Implemented Algorithm	Heuristic Average Rank
ALWAYS_BEST	135,090	4,503	-1,830	11,460	-3,195	1
SOAM	230,944	7,698	0,000	16,878		3,167
PATH_CHEAPEST_ARC	246,628	8,221	-0,340	25,738	0,523	3,367
PATH_MOST_CONSTRAINED_ARC	259,542	8,651	0,540	21,487	0,953	3,800
CHRISTOFIDES	329,501	10,983	1,109	31,397	3,285	4,933
PARALLEL_CHEAPEST_INSERTION	388,074	12,936	0,451	100,000	5,238	4,633
LOCAL_CHEAPEST_INSERTION	515,564	17,185	0,567	62,013	9,487	6,100
SAVINGS	789,752	26,325	0,650	100,000	18,627	5,533
FIRST_UNBOUND_MIN_VALUE	1099,830	36,661	0,135	100,000	28,963	6,367
LOCAL_CHEAPEST_ARC	1133,753	37,792	0,000	100,000	30,094	4,900
GLOBAL_CHEAPEST_ARC	1276,806	42,560	-1,830	100,000	34,862	4,167

Table A.8: Instances present in each fold

Fold 1	**Fold 2**	**Fold 3**	**Fold 4**	**Fold 5**	**Fold 6**	**Fold 7**
A-n36-k5	A-n34-k5	A-n44-k6	A-n69-k9	A-n32-k5	A-n80-k10	A-n33-k5
A-n38-k5	A-n60-k9	A-n53-k7	B-n34-k5	A-n33-k6	B-n41-k6	A-n45-k6
A-n48-k7	A-n62-k8	A-n63-k10	B-n45-k5	A-n37-k5	B-n43-k6	A-n46-k7
A-n54-k7	A-n64-k9	B-n31-k5	B-n45-k6	A-n37-k6	B-n50-k8	B-n44-k7
A-n55-k9	B-n63-k10	B-n39-k5	B-n57-k9	A-n39-k5	E-n101-k8	B-n50-k7
A-n61-k9	B-n66-k9	B-n56-k7	E-n76-k8	A-n39-k6	E-n23-k3	B-n52-k7
A-n63-k9	E-n22-k4	B-n67-k10	F-n45-k4	A-n45-k7	Golden_14	B-n64-k9
A-n65-k9	Golden_10	F-n135-k7	Golden_17	B-n35-k5	Golden_20	B-n68-k9
B-n38-k6	Golden_16	Golden_12	Golden_3	B-n51-k7	Golden_9	E-n101-k14
B-n57-k7	Golden_19	Golden_18	Golden_5	B-n78-k10	Li_23	F-n72-k4
E-n76-k14	Li_25	Golden_2	Golden_6	E-n30-k3	P-n50-k10	Golden_4
Golden_1	M-n200-k17	Golden_7	Li_24	E-n33-k4	P-n50-k7	Li_22
Golden_11	P-n19-k2	Li_21	Li_28	E-n51-k5	P-n51-k10	Li_26
Golden_13	P-n22-k2	Li_30	M-n121-k7	E-n76-k10	P-n65-k10	M-n151-k12
Golden_15	P-n60-k15	Li_32	P-n16-k8	E-n76-k7	P-n70-k10	M-n200-k16
Golden_8	X-n106-k14	M-n101-k10	P-n50-k8	Li_27	X-n101-k25	P-n55-k8
Li_29	X-n120-k6	P-n101-k4	P-n55-k15	Li_31	X-n110-k13	X-n125-k30
P-n21-k2	X-n200-k36	P-n20-k2	X-n228-k23	P-n55-k7	X-n139-k10	X-n176-k26
P-n45-k5	X-n214-k11	P-n22-k8	X-n242-k48	X-n129-k18	X-n143-k7	X-n190-k8
P-n60-k10	X-n266-k58	P-n23-k8	X-n275-k28	X-n167-k10	X-n148-k46	X-n195-k51
P-n76-k4	X-n280-k17	P-n40-k5	X-n289-k60	X-n172-k51	X-n157-k13	X-n256-k16
P-n76-k5	X-n284-k15	P-n55-k10	X-n331-k15	X-n186-k15	X-n181-k23	X-n261-k13
X-n1001-k43	X-n313-k71	X-n115-k10	X-n351-k40	X-n237-k14	X-n209-k16	X-n303-k21
X-n134-k13	X-n411-k19	X-n233-k16	X-n359-k29	X-n317-k53	X-n219-k73	X-n393-k38
X-n162-k11	X-n513-k21	X-n251-k28	X-n367-k17	X-n376-k94	X-n270-k35	X-n491-k59
X-n204-k19	X-n548-k50	X-n308-k13	X-n429-k61	X-n420-k130	X-n298-k31	X-n573-k30
X-n223-k34	X-n627-k43	X-n322-k28	X-n480-k70	X-n439-k37	X-n449-k29	X-n716-k35
X-n294-k50	X-n749-k98	X-n327-k20	X-n613-k62	X-n459-k26	X-n561-k42	X-n783-k48
X-n344-k43	X-n766-k71	X-n401-k29	X-n641-k35	X-n502-k39	X-n685-k75	X-n876-k59
X-n384-k52	X-n801-k40	X-n856-k95	X-n655-k131	X-n536-k96	X-n701-k44	X-n979-k58
X-n733-k159	X-n895-k37	X-n957-k87				

Appendix B

Extracted Features

Table B.1: Complete set of Extracted Features

Type	Feature	Description	Estimator
Instance Based	I1	Number of Clients	Total
	I2	Client Demands	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	I3	Vehicle Capacity	Total
	I4	Theoretical Minimum Vehicle Quantity	Total
	I5	Average Vehicle Occupancy for K, K+1, K+2	K
			K+1
			K+2
	I6	Ratio of Average Demand and Vehicle Capacity	Total
	I7	Theoretical Average Number of nodes per route	Total
Enclosing Rectangle	ER1	Minimal Enclosing Rectangle	X Coordinate
			Y Coordinate
			ER1_area
	ER2	Enclosing Rectangle Center	ER2_x
			ER2_y
	ER3	Client Density in Minimal Enclosing Rectangle	Density
	ER4	Demand Density in Minimal Enclosing Rectangle	Density
Node Distribution	ND1	Distance from Depot to each node	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis

Table B.1 continued from previous page

Type	Feature	Description	Estimator
Node Distribution	ND2	Client Geometric Distribution Center	X Coordinate
			Y Coordinate
	ND3	Client Demand Distribution Center	X Coordinate
			Y Coordinate
	ND4	Percentage Difference from Depot to ND2	X Difference
			Y Difference
	ND5	Percentage Difference from Depot to ND3	X Difference
			Y Difference
	ND6	Client Density with circular area equal to Enclosing Rectangle	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	ND7	Demand Density with circular area equal to Enclosing Rectangle	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	ND8	Client Density with circular area equal to neighborhood	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
Geometric	G1	Q1 Clients	Total
		Q2 Clients	Total
		Q3 Clients	Total
		Q4 Clients	Total
	G2	Number of Clients per Quadrant	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	G3	Q1 Demand	Total
		Q2 Demand	Total
		Q3 Demand	Total
		Q4 Demand	Total
	G4	Statistics of G3	Total
			Mean
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis

Table B.1 continued from previous page

Type	Feature	Description	Estimator
Geometric	G5	Client Demands of Quadrant 1	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	G6	Client Demands of Quadrant 2	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	G7	Client Demands of Quadrant 3	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	G8	Client Demands of Quadrant 4	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
Heuristic	H1	Bin Packing Problem	Total
	H2	Bin Packing Problem Vehicle Occupancy	Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis
	H3	Bin Packing Occupancy vs Theoretical Occupancy	Total
	H4	Prim's Minimum Spanning Tree	Total
			Minimum
			Mean
			Maximum
			Standard Deviation
			Coefficient of Variation
			Skewness
			Kurtosis

References

Python Reference: Routing | OR-Tools | Google Developers. URL https://developers.google.com/optimization/reference/python/constraint_solver/pywrapcp.

Vipin Ajayakumar. Parsing text with Python. URL https://www.vipinajayakumar.com/parsing-text-with-python/.

S. P. Anbuudayasankar, K. Ganesh, and Sanjay Mohapatra. Survey of Methodologies for TSP and VRP. In *Models for Practical Routing Problems in Logistics*, volume 9783319050, pages 11–42. Springer International Publishing, Cham, 2014.

Carlos Ansótegui, Meinolf Sellmann, and Kevin Tierney. A gender-based genetic algorithm for the automatic configuration of algorithms. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5732 LNCS(0644113):142–157, 2009.

P Augerat, J M Belenguer, E Benavent, D Corberán Naddef, and G Rinaldi. Computational results with a branch and cut code for the capacitated vehicle routing problem. *Technical Report RR 949-M*, (January), 1995.

Jonathan F. Bard, Liu Huang, Moshe Dror, and Patrick Jaillet. A branch and cut algorithm for the VRP with satellite facilities. *IIE Transactions*, 30(9):821–834, 9 1998.

Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine Learning for Combinatorial Optimization: a Methodological Tour d'Horizon. pages 1–34, 2018.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep Clustering for Unsupervised Learning of Visual Features. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 11218 LNCS:139–156, 7 2018.

N Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. (February), 1976.

N. Christofides and S. Eilon. An Algorithm for the Vehicle-dispatching Problem. *Journal of the Operational Research Society*, 20(3):309–318, 9 1969.

N. Christofides, A. Mingozzi, and P. Toth. The vehicle routing problem. In *Combinatorial optimization, vol 1*, chapter 11, pages 315–338. Wiley Interscience, 1979.

G. Clarke and J. W. Wright. Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operations Research*, 12(4):568–581, 8 1964.

P. L. N. U. Cooray and Thashika D. Rupasinghe. Machine Learning-Based Parameter Tuned Genetic Algorithm for Energy Minimizing Vehicle Routing Problem. *Journal of Industrial Engineering*, 2017: 1–13, 2017.

G. B. Dantzig and J. H. Ramser. The Truck Dispatching Problem. *Management Science*, 6(1):80–91, 10 1959.

P. Festa. A brief introduction to exact, approximation, and heuristic algorithms for solving hard combinatorial optimization problems. In *2014 16th International Conference on Transparent Optical Networks (ICTON)*, pages 1–20. IEEE, 7 2014.

Marshall L. Fisher. Optimal solution of vehicle routing problems using minimum K-trees. *Operations Research*, 42(4):626–642, 1994.

Ricardo Fukasawa, Jens Lysgaard, Marcus Poggi de Aragão, Marcelo Reis, Eduardo Uchoa, and Renato F. Werneck. Robust Branch-and-Cut-and-Price for the Capacitated Vehicle Routing Problem. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3064, pages 1–15. 2004.

T. J. Gaskell. Bases for Vehicle Fleet Scheduling. *Journal of the Operational Research Society*, 18(3): 281–295, 9 1967.

Michel Gendreau and Jean-Yves Potvin, editors. *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*. Springer US, Boston, MA, 2010.

Michel Gendreau, Gilbert Laporte, and Jean-Yves Potvin. 6. Metaheuristics for the Capacitated VRP. In *The Vehicle Routing Problem*, pages 129–154. Society for Industrial and Applied Mathematics, 1 2002.

Bruce L. Golden, Edward A. Wasil, James P. Kelly, and I-Ming Chao. The Impact of Metaheuristics on Solving the Vehicle Routing Problem: Algorithms, Problem Sets, and Computational Results. In *Fleet Management and Logistics*, volume 1, pages 33–56. Springer US, Boston, MA, 1998.

Nagesh Kadaba, Kendall E Nygard, and Paul L Juell. Integration of adaptive machine learning and knowledge-based systems for routing and scheduling applications. *Expert Systems with Applications*, 2(1):15–27, 1 1991.

Sotiris Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. *Informatica (Ljubljana)*, 31, 2007.

G. Laporte, Y. Nobert, and S. Taillefer. A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. *Mathematical Modelling*, 9(12):857–868, 1987.

Gilbert Laporte and Frédéric Semet. 5. Classical Heuristics for the Capacitated VRP. In *The Vehicle Routing Problem*, pages 109–128. Society for Industrial and Applied Mathematics, 1 2002.

Feiyue Li, Bruce Golden, and Edward Wasil. Very large-scale vehicle routing: new test problems, algorithms, and results. *Computers & Operations Research*, 32(5):1165–1179, 5 2005.

Pedro Munari, Twan Dollevoet, and Remy Spliet. A generalized formulation for vehicle routing problems. (1):1–19, 6 2016.

Mohammadreza Nazari, Afshin Oroojlooy, Lawrence V. Snyder, and Martin Takáč. Reinforcement Learning for Solving the Vehicle Routing Problem. *Advances in Neural Information Processing Systems*, 2018-Decem:9839–9849, 5 2018.

Ibrahim Hassan Osman. Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41(4):421–451, 12 1993.

Bo Peng, Jiahai Wang, and Zizhen Zhang. A Deep Reinforcement Learning Algorithm Using Dynamic Attention Model for Vehicle Routing Problems. *Communications in Computer and Information Science*, 1205 CCIS:636–650, 2 2020.

Jean-Yves Potvin. Genetic algorithms for the traveling salesman problem. *Annals of Operations Research*, 63(3):337–370, 6 1996.

Chengming Qi. An Ant Colony Algorithm with Stochastic Local Search for the VRP. In *2008 3rd International Conference on Innovative Computing Information and Control*, pages 464–464. IEEE, 2008.

T.K. Ralphs. Parallel branch and cut for capacitated vehicle routing. *Parallel Computing*, 29(5):607–629, 5 2003.

César Rego and Catherine Roucairol. A Parallel Tabu Search Algorithm Using Ejection Chains for the Vehicle Routing Problem. In *Meta-Heuristics*, pages 661–675. Springer US, Boston, MA, 1996.

Chunyu Ren, Xiao-bo Wang, and Jin-ying Sun. Research on VRP of Optimizing Based on Fuzzy C-Means Clustering and Iga Under Electronic Commerce. In *2006 International Conference on Machine Learning and Cybernetics*, number August, pages 2461–2467. IEEE, 8 2006.

Michael R. Smith and Tony Martinez. Improving classification accuracy by identifying and removing instances that should be misclassified. In *The 2011 International Joint Conference on Neural Networks*, pages 2690–2697. IEEE, 7 2011.

Richard S Sutton and Andrew G Barto. *Reinforcement Learning: An Introduction*. Second edition, 2018.

É. Taillard. Parallel iterative search methods for vehicle routing problems. *Networks*, 23(8):661–673, 12 1993.

Paolo Toth and Daniele Vigo. 1. An Overview of Vehicle Routing Problems. In *The Vehicle Routing Problem*, pages 1–26. Society for Industrial and Applied Mathematics, 1 2002.

Paolo Toth and Daniele Vigo. The Granular Tabu Search and Its Application to the Vehicle-Routing Problem. *INFORMS Journal on Computing*, 15(4):333–346, 11 2003.

Eduardo Uchoa, Diego Pecin, Artur Pessoa, Marcus Poggi, Thibaut Vidal, and Anand Subramanian. New benchmark instances for the Capacitated Vehicle Routing Problem. *European Journal of Operational Research*, 257(3):845–858, 2017.

Daniele Vigo. A heuristic algorithm for the asymmetric capacitated vehicle routing problem. *European Journal of Operational Research*, 89(1):108–126, 2 1996.

Anthony Wren and Alan Holliday. Computer Scheduling of Vehicles from One or More Depots to a Number of Delivery Points. *Journal of the Operational Research Society*, 23(3):333–344, 9 1972.

N. Yoshiike and Y. Takefuji. Vehicle routing problem using clustering algorithm by maximum neural networks. In *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials. IPMM'99 (Cat. No.99EX296)*, volume 2, pages 1109–1113. IEEE, 1999.