2015

The Application of the Self Organizing Map to the Vehicle Routing Problem

Meghan Steinhaus
University of Rhode Island, mksteinhaus@gmail.com

THE APPLICATION OF THE SELF ORGANIZING MAP TO THE VEHICLE ROUTING PROBLEM

BY

MEGHAN STEINHAUS

A DISSERTATION SUBMITTED IN PARTIAL FULFILLMENT OF THE

REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

IN

MECHANICAL, INDUSTRIAL AND SYSTEMS ENGINEERING

UNIVERSITY OF RHODE ISLAND

2015

DOCTOR OF PHILOSOPHY DISSERTATION

OF

MEGHAN STEINHAUS

APPROVED:

Dissertation Committee:

Major Professor Manbir Sodhi

Edmund Lamagna

Gregory Jones

Nasser H. Zawia

DEAN OF THE GRADUATE SCHOOL

UNIVERSITY OF RHODE ISLAND

2015

ABSTRACT

The Vehicle Routing Problem (VRP) is an $\mathcal{NP}$-hard, combinatorial optimization problem. For $\mathcal{NP}$-hard problems, there is no known polynomial time algorithm to solve these problems. Therefore, for many moderately sized problems, these problems cannot be reliably solved to optimality. This is the case with the VRP, where problem instances with over 100 cities are not easily solved using exact methods. Therefore, the majority of VRP research focuses on heuristics.

Artificial Neural Networks (ANNs) are inspired by the functions in the human brain. Researchers have applied ANNs across a wide range of problems with great success. Since solving the VRP relies heavily on heuristics, and ANNs have shown to be effective heuristics for numerous applications, this research seeks to determine the effectiveness of applying ANNs to the VRP

The first part of this work investigates the effectiveness of the existing application of ANNs for solving the VRP. An updated Self Organizing Map (SOM) algorithm is proposed for solving the VRP. The proposed SOM incorporates fuzzy logic in order to overcome the need for parameter tuning for each new problem. Experiments are conducted, and the results indicate that the performances of the proposed algorithm exceeds previous results. Further, a comparison is made to other constructive heuristics which makes it clear that the proposed algorithm is a competitive constructive heuristic for solving the VRP.

The second part of this research investigates the VRP in the context of the algorithm selection problem. This work utilizes the SOM as a tool for both exploratory data analysis of the diversity of the existing VRP benchmark problem sets, as well as a prediction tool for algorithm selection. 23 VRP problem characteristics are examined across 102 VRP benchmark problems, and a method for automatic extraction of these problem characteristics is proposed.

Finally, both an unsupervised and supervised SOM are trained and tested for prediction of algorithm performance. The results indicate that the SOM is capable of distinguishing between algorithm performance based on the 23 problem characteristics extracted from each VRP instance.

ACKNOWLEDGMENTS

First, I would like to thank my family. My husband Gary has provided unending support throughout this journey, and I am so, deeply grateful for everything he does for our family. My children inspire me daily to worry a little less and laugh a little bit more, both of which have been vitally important throughout this journey.

I would also like to thank Professor Manbir Sodhi, my major professor for his guidance and mentoring throughout this process. He has helped make this experience both challenging and fun, and his constant encouragement has been invaluable over the years. I would also like to thank my committee members Professor Edmund Lamagna and Dr. Gregory Jones for their feedback, time, and encouragement.

Finally, I would like to thank my colleagues, mentors, and friends at the United States Coast Guard Academy. I am grateful to work in such a supportive environment, with such wonderful people. I would also like to acknowledge and thank the U.S. Coast Guard for supporting my education.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGMENTS iii

TABLE OF CONTENTS iv

LIST OF TABLES vii

LIST OF FIGURES viii

CHAPTER

1 Introduction 1

1.1 Optimization 1

1.2 Artificial Neural Networks 3

1.3 Combinatorial Optimization and Artificial Neural Networks 3

List of References 4

2 The Vehicle Routing Problem 5

2.1 Origin 5

2.2 Problem Formulation 6

2.2.1 Mathematical Programming Formulations 6

2.3 Approaches to Solving CVRP 11

2.3.1 Exact Algorithms 11

2.3.2 Heuristic Methods 12

2.3.3 Metaheuristics 16

2.4 Benchmark Problem Sets 21

List of References 21

3 Applications of ANNs to the CVRP 27

3.1 Artificial Neural Networks 27

3.2 The Application of ANNs to Combinatorial Optimization Problems 27

3.2.1 Hopfield Neural Network 27

Page

3.2.2 Self Organizing Map 28

List of References 34

4 An Updated SOM Approach to the CVRP 38

4.1 Introduction 38

4.2 Background 39

4.2.1 Vehicle Routing Problem 39

4.2.2 Self Organizing Map 40

4.3 Updated SOM for CVRP 44

4.3.1 Proposed Algorithm 44

4.3.2 Parameter Sensitivity 48

4.4 Fuzzy Logic Control 49

4.4.1 SOM with Fuzzy Logic Control 52

4.5 Experimental Results 52

4.5.1 Discussion 55

4.5.2 Conclusion and Future Work 58

List of References 58

5 Using a SOM for Algorithm Prediction 61

5.1 Introduction 61

5.2 Algorithm Selection Problem 63

5.3 Feature Selection 65

5.3.1 Clustering 68

5.3.2 Problem Features for the CVRP 79

5.4 SOM for Exploratory Data Analysis and Prediction 81

5.5 Experimental Setup 84

5.5.1 Problem Characteristics and Diversity in Benchmark Problems 84

5.5.2 Using a SOM for Algorithm Performance Prediction 86

5.6 Experimental Results 88

Page

5.6.1 Problem Characteristics and Diversity in Benchmark Problems 88

5.6.2 Using a SOM for Algorithm Performance Prediction 97

5.7 Discussion 102

5.8 Conclusions and Future Work 108

List of References 109

6 Conclusion and Future Work 113

6.0.1 Conclusion 113

6.0.2 Future Work 114

APPENDIX

BIBLIOGRAPHY 171

LIST OF TABLES

Table		Page
1	Parameter settings tested	48
2	Best solutions found in 100 replications of the proposed algorithm	48
3	Parameter Descriptions	53
4	Fuzzy Rules	54
5	Number of feasible solutions found out of 100 replications	55
6	Comparison of the best solutions found by each algorithm	56
7	Coefficient of variation for problem characteristics across the 102 benchmark problem sets	91
8	Number of Problems an Algorithm Performed Best	98
9	Unsupervised SOM Predicted Best Algorithm Versus the Known Best Algorithm for the *Testing Set*	100
10	Supervised SOM Predicted Best Algorithm Versus the Known Best Algorithm for the *Testing Set*	100
11	Unsupervised SOM Prediction Performance	101
12	Unsupervised SOM Prediction Performance	101

LIST OF FIGURES

Figure		Page
1	Comparison of various polynomial and exponential time complexity functions [4].	2
2	Example to illustrate the variables y_{ij} and y_{ji} on a route [5].	10
3	string cross	16
4	string exchange	16
5	string relocation	16
6	Different Distributions of Customers	22
7	HNN representation of a TSP [17]	28
8	Two Layer Neural Network Used in EN and SOFM	29
9	Updating Positions of Winning Node and Neighbors	30
10	Evolution of Network for TSP [26]	31
11	A Two Layer SOM Network for TSP	41
12	Evolution of SOM Network for CVRP with N=50 and K=5	42
13	Example of node competition with bias term; assume that vehicle 2 is overloaded and vehicle 1 is not.	46
14	General structure of fuzzy logic controller [32]	50
15	Triangular membership functions used in FLC. The fuzzification process uses 15a and 15b, and the defuzzification process uses 15c and 15d.	54
16	Rice's [3] original diagram of the algorithm selection model.	64
17	Rice's enhanced algorithm selection model to include the feature space [6].	64
18	Example of how to use the $k-distance$ plot to identify Eps.	70
19	$n = 9$ cities placed uniformly on a 400 x 400 grid. The distance from c_i to its 4 nearest neighbors is $d = \frac{400}{2} = 200$.	71
20	Example of using Equation 40 to define Eps parameter in DBSCAN.	71
21	5th degree polynomial fit to a $k-distance$ plot using *numpy.polyfit* in Python.	72
22	Comparing the relationship between the 'valley' in the curve and the inflection point. The approximate inflection point is found in the yellow rectangle, whereas the first 'valley' is located in one of the red circles.	73

Figure		Page
23	Example of the difference between the approximate locations of the inflection point and 'valley' in the fitted $k - distance$ curve.	73
24	Choosing *Eps* from values within 2 standard deviations above the mean is ineffective in the presence of large $k - distance$ outliers.	75
25	Choosing *Eps* where all three methods yield similar results.	76
26	Choosing *Eps* where choosing *Eps* using Equation 40 is not sufficient.	77
27	Choosing *Eps* where the method of choosing *Eps* between $[mean, mean + 2\sigma$] is not sufficient.	78
28	number of clusters	89
29	proportion of distinct distances	89
30	Histograms of each algorithms performance, measured by PDB, across all benchmark problems	90
31	Scatter plots of algorithm performance (percent deviation above best known solution) versus the number of cities in the problem	92
32	The codebook vectors of the trained SOM based on the 102 benchmark problem sets	94
33	Plot of the average distance between a node and its neighbors	94
34	Plot of the number of input vectors that are mapped to each node	94
35	K means plot to identify the 'elbow' in the curve	95
36	Trained SOM with six clusters	95
37	SOM component plane: number of cities	97
38	SOM component plane: ratio of the number of clusters to the number of cities	97
39	SOM component plane: number of clusters	97
40	SOM component plane: y-coordinate of the depot	97
41	102 benchmark problems mapped to the trained SOM	98
42	Best performing algorithm for each problem instance mapped to the trained SOM where circle=C& W, triangle=Sweep, and plus sign=Fuzzy SOM	98
43	Codebook vectors of the trained SOM	99
44	Average distance between a node and its neighboring nodes	99
45	The number of input vectors that are mapped to each node	99

Figure **Page**

46 Mapping of the best algorithm for each of the problem instances in the *training set* 99

.47 Number of Cities 119

.48 Standard Deviation of the Distance Matrix 119

.49 X-coordinate of City Centroid 119

.50 Y-coordinate of City Centroid 119

.51 Radius of Problem Instance 120

.52 Proportion of Distinct Distances 120

.53 Standard Deviation of the Nearest Neighbor Distances 120

.54 Coefficient of Variation for Nearest Neighbor Distances 120

.55 Ratio of the Number of Clusters to the Number of Cities 121

.56 Ratio of the Number of Outliers to the Total Number of Cities 121

.57 Ratio of the Number of Edge Cities to Total Number of Cities 121

.58 Number of Clusters 121

.59 Mean Radius of the Clusters 122

.60 X-coordinate of the Depot 122

.61 Y-coordinate of the Depot 122

.62 Standard Deviation of Demand 122

.63 Ratio of Total Demand to Total Vehicle Capacity 123

.64 Ratio of the Max Cluster Demand to Vehicle Capacity 123

.65 Ratio of Outlier Demand to Overall Demand 123

.66 Ratio of the Maximum City Demand to Vehicle Capacity 123

.67 Average Number of Customers Per Route 124

.68 Area of the Rectangle 124

.69 Minimum Number of Trucks 124

.70 Scatter plots of algorithm performance (percent deviation above best known solution) versus the number of cities in the problem 125

.71 Scatter plots of algorithm performance (percent deviation above best known solution) versus the standard deviation of the distance matrix in the problem 126

Figure **Page**

.72 Scatter plots of algorithm performance (percent deviation above best known solution) versus the x coordinate of the instance centroid 127

.73 Scatter plots of algorithm performance (percent deviation above best known solution) versus the y coordinate of the instance centroid 128

.74 Scatter plots of algorithm performance (percent deviation above best known solution) versus the radius of the problem instance 129

.75 Scatter plots of algorithm performance (percent deviation above best known solution) versus the proportion of distinct distances in the distance matrix (rounded to two decimal values) 130

.76 Scatter plots of algorithm performance (percent deviation above best known solution) versus the standard deviation of the nearest neighbors 131

.77 Scatter plots of algorithm performance (percent deviation above best known solution) versus the coefficient of variation of the nearest neighbors 132

.78 Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of clusters to the total number of cities for each problem instance 133

.79 Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of outliers to the total number of cities for each problem instance 134

.80 Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of cities at the edge of a cluster (identified by DBSCAN) to the total number of cities for each problem instance 135

.81 Scatter plots of algorithm performance (percent deviation above best known solution) versus the number of clusters found by DBSCAN for each problem instance 136

.82 Scatter plots of algorithm performance (percent deviation above best known solution) versus the mean radius of the clusters for each problem instance . . 137

.83 Scatter plots of algorithm performance (percent deviation above best known solution) versus the x coordinate of the depot for each problem instance . . . 138

.84 Scatter plots of algorithm performance (percent deviation above best known solution) versus the y coordinate of the depot for each problem instance . . . 139

.85 Scatter plots of algorithm performance (percent deviation above best known solution) versus the standard deviation of the demand for each problem instance 140

.86 Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of total demand to the available capacity (based on the minimum number of trucks required for the problem) for each problem instance 141

Figure **Page**

.87 Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of the largest cluster demand to a single vehicle's capacity for each problem instance 142

.88 Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of sum of all outlier demands to the overall demand for each problem instance 143

.89 Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of the single largest city demand to the vehicle capacity for each problem instance 144

.90 Scatter plots of algorithm performance (percent deviation above best known solution) versus the average number of customers per route for each problem instance 145

.91 Scatter plots of algorithm performance (percent deviation above best known solution) versus the area of the rectangle that all problems lie within for each problem instance 146

.92 Scatter plots of algorithm performance (percent deviation above best known solution) versus the minimum number of trucks required for each problem instance 147

.93 SOM component plane: number of cities 148

.94 SOM component plane: standard deviation of the distance matrix 148

.95 SOM component plane: x-coordinate of city centroid 148

.96 SOM component plane: y-coordinate of city centroid 148

.97 SOM component plane: radius of problem instance 148

.98 SOM component plane: proportion of distinct distances 148

.99 SOM component plane: standard deviation of the nearest neighbor distances 149

.100 SOM component plane: coefficient of variation for nearest neighbor distances 149

.101 SOM component plane: ratio of the number of clusters to the number of cities 149

.102 SOM component plane: ratio of the number of outliers to the total number of cities 149

.103 SOM component plane: ratio of the number of edge cities to total number of cities 149

.104 SOM component plane: number of clusters 149

.105 SOM component plane: mean radius of the clusters 149

.106 SOM component plane: x-coordinate of the depot 149

Figure **Page**

.107 SOM component plane: y-coordinate of the depot 150

.108 SOM component plane: standard deviation of demand 150

.109 SOM component plane: ratio of total demand to total vehicle capacity 150

.110 SOM component plane: ratio of the max cluster demand to vehicle capacity . 150

.111 SOM component plane: ratio of outlier demand to overall demand 150

.112 SOM component plane: ratio of the maximum city demand to vehicle capacity 150

.113 SOM component plane: average number of customers per route 150

.114 SOM component plane: area of the rectangle 150

.115 SOM component plane: minimum number of trucks 151

.116 Histograms of the number of cities in both the existing benchmark problem set and the newly generated problem set 152

.117 Histograms of the standard deviation of the distance matrix in both the existing benchmark problem set and the newly generated problem set 153

.118 Histograms of the x-coordinate of the centroid in both the existing benchmark problem set and the newly generated problem set 153

.119 Histograms of the y-coordinate of the centroid in both the existing benchmark problem set and the newly generated problem set 154

.120 Histograms of the problem radius in both the existing benchmark problem set and the newly generated problem set 154

.121 Histograms of the proportion of distinct distances in the distance matrix in both the existing benchmark problem set and the newly generated problem set 155

.122 Histograms of the standard deviation of the nearest neighbors in both the existing benchmark problem set and the newly generated problem set 155

.123 Histograms of the problem radius in both the existing benchmark problem set and the newly generated problem set 156

.124 Histograms of the ratio of the number of clusters to the number of cities in both the existing benchmark problem set and the newly generated problem set 156

.125 Histograms of the ratio of the number of outliers to the number of cities in both the existing benchmark problem set and the newly generated problem set 157

.126 Histograms of the ratio of the number of cities on the edge of a cluster to the total number of cities in both the existing benchmark problem set and the newly generated problem set 157

.127 Histograms of the number of clusters in both the existing benchmark problem set and the newly generated problem set 158

Figure **Page**

.128 Histograms of the mean radius of the clusters in both the existing benchmark problem set and the newly generated problem set 158

.129 histograms of the x-coordinate of the depot in both the existing benchmark problem set and the newly generated problem set 159

.130 Histograms of the y-coordinate of the depot in both the existing benchmark problem set and the newly generated problem set 159

.131 Histograms of the standard deviation of demand in both the existing benchmark problem set and the newly generated problem set 160

.132 Histograms of the ratio of the total demand to the total vehicle capacity in both the existing benchmark problem set and the newly generated problem set 160

.133 Histograms of the ratio of the maximum cluster demand to vehicle capacity in both the existing benchmark problem set and the newly generated problem set 161

.134 Histograms of the ratio of the outlier demand to the total demand in both the existing benchmark problem set and the newly generated problem set 161

.135 Histograms of the ratio of the maximum demand to the vehicle capacity in both the existing benchmark problem set and the newly generated problem set 162

.136 Histograms of the average number of customers per route in both the existing benchmark problem set and the newly generated problem set 162

.137 Histograms of the area of the rectangle the cities lie within for both the existing benchmark problem set and the newly generated problem set 163

.138 Histograms of the minimum number of trucks required in both the existing benchmark problem set and the newly generated problem set 163

.139 K-means plot for the trained SOM for prediction 167

.140 8 clusters in the trained SOM for prediction 167

.141 SOM component plane: number of cities 168

.142 SOM component plane: standard deviation of the distance matrix 168

.143 SOM component plane: x-coordinate of city centroid 168

.144 SOM component plane: y-coordinate of city centroid 168

.145 SOM component plane: radius of problem instance 168

.146 SOM component plane: proportion of distinct distances 168

.147 SOM component plane: standard deviation of the nearest neighbor distances 168

.148 SOM component plane: coefficient of variation for nearest neighbor distances 168

.149 SOM component plane: ratio of the number of clusters to the number of cities 169

Figure **Page**

.150 SOM component plane: ratio of the number of outliers to the total number of cities 169

.151 SOM component plane: ratio of the number of edge cities to total number of cities 169

.152 SOM component plane: number of clusters 169

.153 SOM component plane: mean radius of the clusters 169

.154 SOM component plane: x-coordinate of the depot 169

.155 SOM component plane: y-coordinate of the depot 169

.156 SOM component plane: standard deviation of demand 169

.157 SOM component plane: ratio of total demand to total vehicle capacity 170

.158 SOM component plane: ratio of the max cluster demand to vehicle capacity . 170

.159 SOM component plane: ratio of outlier demand to overall demand 170

.160 SOM component plane: ratio of the maximum city demand to vehicle capacity 170

.161 SOM component plane: average number of customers per route 170

.162 SOM component plane: area of the rectangle 170

.163 SOM component plane: minimum number of trucks 170

CHAPTER 1

Introduction

This research is motivated by the intersection of combinatorial optimization and artificial neural networks. The most basic underlying question is how the strengths of artificial neural networks can be used to solve inherently difficult combinatorial optimization problems. This introductory chapter provides a brief overview of both combinatorial optimization and artificial neural networks. Chapters 2 and 3 provide a detailed background of the Vehicle Routing Problem (which is the specific combinatorial optimization problem this research is focused on), as well as the existing artificial neural network approaches to solving the vehicle routing problem. Chapter 4 introduces an updated artificial neural network approach to solving the vehicle routing problem, along with the experimental results and discussion. Chapter 5 uses a SOM as a tool for analyzing the diversity of CVRP benchmark problems, as well as a prediction tool for choosing the best performing algorithm for a new CVRP problem instance. Chapter 6 provides the conclusion as well as directions for future work.

1.1 Optimization

Optimization problems deal with maximizing or minimizing a function, normally of many variables and subject to constraints. In general, optimization problems can be divided into two categories: problems with continuous variables and problems with discrete variables. An optimization problem whose solution can be found within a finite set of possible solutions is called a combinatorial optimization problem [1].

The practicality of combinatorial optimization problems (COPs) to everyday life cannot be overstated. Combinatorial optimization problems exist across a wide range of disciplines and are solved on a daily basis. In his overview of the early history of combinatorial optimization, Schrijver [2] suggests primitive examples of COPs such as searching for food or short paths, as well as more modern applications such as creating a mailman's route, assigning jobs amongst people, and the transporting of goods. Although combinatorial optimization emerged as its own field of study in the middle of the twentieth century, the roots of many problems can be traced back decades or even centuries [2].

The study of solving combinatorial optimization problems is deeply intertwined with the study of computational complexity. Computational complexity refers to the categorization of

Time complexity function	Size n					
	10	20	30	40	50	60
n	.00001 second	.00002 second	.00003 second	.00004 second	0.00005 second	.00006 second
n^2	.0001 second	.0004 second	.0009 second	.0016 second	.0025 second	.0036 second
n^3	.001 second	.008 second	.027 second	.064 second	.125 second	.216 second
n^5	.1 second	3.2 seconds	24.3 seconds	1.7 minutes	5.2 minutes	13.0 minutes
2^n	.001 second	1.0 second	17.9 minutes	12.7 days	35.7 years	366 centuries
3^n	.059 second	58 minutes	6.5 years	3855 centuries	$2x10^8$ centuries	$1.3x10^{13}$ centuries

Figure 1: Comparison of various polynomial and exponential time complexity functions [4].

problem difficulty based on the efficiency of computer methods for solving these problems [3]. Although an in depth discussion of $\mathcal{P}$ versus $\mathcal{NP}$ is beyond the scope of this thesis, it is important to note that many combinatorial optimization problems are $\mathcal{NP}$-Hard. This indicates that there is not a known, polynomial time algorithm that can find the optimal solution.

In the study of computational complexity, algorithms are generally described as either a *polynomial time algorithm* or an *exponential time algorithm*. A *polynomial time algorithm* is an algorithm in which the time complexity function can be expressed in terms of a polynomial, whereas an *exponential time algorithm* cannot. Additionally, an *exponential time algorithm* does not necessarily have to be an exponential function, it is just not a polynomial function. For example, an algorithm with a time complexity function of n^{logn} is considered an *exponential time algorithm* [4].

The importance of time complexity function is illustrated in 1. As seen in this table provided by Garey and Johnson [4], it is clear that as the problem size, n, increases, it is not feasible to use *exponential time algorithms* for solving these problems. For the numerous $\mathcal{NP}$-hard combinatorial optimization problems, this is the case. There are currently no known, *polynomial time algorithms* for solving these $\mathcal{NP}$-hard problems. Therefore a great deal research into combinatorial optimization focuses on heuristic algorithms.

1.2 Artificial Neural Networks

The study of Artificial Neural Networks (ANNs) is inspired by the organization and functions of the biological neural networks in the human brain [5]. At the core of ANN research is the recognition that although the performance of modern computers is superior to that of humans for computational tasks, the relative ease at which the human brain can process and solve complex tasks dominates even the best computers. Jain and Mao [6] describe many of the superior, inherent capabilities of the human brain as compared to the modern computer: massive parallelism, distributed representation and computation, learning ability, generalization ability, adaptivity, inherent contextual information processing, fault tolerance, and low energy consumption. In the study of ANNs, the goal is to simulate functions of the brain with a computer in order to exploit these capabilities.

Scientists across many disciplines have developed various ANNs in order to solve problems such as: pattern classification, speech synthesis and recognition, adaptive interfaces between humans and complex physical systems, function approximation, image data compression, associative memory, clustering, forecasting and prediction, combinatorial optimization, nonlinear system modeling, and control [7]. While researchers have found success in the application of ANNs to many of these problems, their application to optimization has been less successful [8].

1.3 Combinatorial Optimization and Artificial Neural Networks

Simply put, an ANN is a heuristic. It is a method, modeled after the human brain, used to find a good solution to a problem. Since the solvability of real-world sized combinatorial optimization problems relies almost exclusively on heuristics, it is reasonable to question whether ANNs are, or can be developed, as useful tools for solving these combinatorial optimization problems. To date, the most widely studied combinatorial optimization problem that ANN research has focused on is the Traveling Salesman Problem (TSP), with the Vehicle Routing Problem (VRP) being the second most studied [9]. The most current literature indicates that ANNs are inferior to the best heuristics for the TSP as well as the VRP [10, 11]. Although this has created pessimism in some researchers, other researchers encourage continued research in these areas [10, 11] with the belief that small changes, over time, can lead to large changes; as well as the belief that advances in this work might be beneficial to other research areas; and that ANNs might become competitive with the availability of massively parallel computers.

List of References

[1] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. Courier Dover Publications, 1998.

[2] S. Alexander, "On the history of combinatorial optimization (till 1960)," *Handbooks in Operations Research and Management Science: Discrete Optimization*, vol. 12, p. 1, 2005.

[3] E. A. Lamagna, "Infeasible computation: Np-complete problems," *ABACUS*, vol. 4, no. 3, pp. 18–33, 1987.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability*. wh freeman, 2002, vol. 29.

[5] J. Brownlee, *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, 2011.

[6] A. K. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[7] M. H. Hassoun, *Fundamentals of artificial neural networks*. MIT press, 1995.

[8] I. H. Osman and G. Laporte, "Metaheuristics: A bibliography," *Annals of Operations Research*, vol. 63, no. 5, pp. 511–623, 1996.

[9] M. Schwardt and J. Dethloff, "Solving a continuous location-routing problem by use of a self-organizing map," *International Journal of Physical Distribution & Logistics Management*, vol. 35, no. 6, pp. 390–408, 2005.

[10] E. Cochrane and J. Beasley, "The co-adaptive neural network approach to the euclidean travelling salesman problem," *Neural Networks*, vol. 16, no. 10, pp. 1499–1525, 2003.

[11] J.-C. Créput and A. Koukam, "The memetic self-organizing map approach to the vehicle routing problem," *Soft Computing*, vol. 12, no. 11, pp. 1125–1141, 2008.

CHAPTER 2

The Vehicle Routing Problem

2.1 Origin

The Vehicle Routing Problem (VRP) consists of finding the optimal routing of a fleet of vehicles amongst a set of cities, where each city has a demand for goods that the vehicles must deliver, and each vehicle has a constraint on the capacity of goods it can transport. Examples of the VRP include the distribution of goods such as in the beverage or food industries, the collection of waste, or the delivery of mail or newspapers [1]. The applications of the VRP span many industries, each with its own unique attributes, which have resulted in many variations of Vehicle Routing Problems.

The VRP was first introduced as the Truck Dispatching Problem in 1959 [2]. In their original work on this problem, Dantzig and Ramser introduced the Truck Dispatching Problem as a generalization of the Traveling Salesman Problem (TSP). Although the term 'traveling salesman problem' became relevant in the mathematics community in the early 1930's [3], William Cook provides an extensive history of the TSP prior to its formal entrance into the mathematics field, which goes back nearly two centuries, in his book *In Pursuit of the Traveling Salesman* [4] .

Given a set of cities, the objective of the TSP is to find the shortest route of visiting all the cities and returning to the starting point. Dantzig and Ramser generalize the TSP to introduce the Truck Dispatching Problem by specifying that deliveries be made at each one of the cities. Furthermore, the carrier of these deliveries is constrained by the capacity it can transport in one load. If there is only one truck (carrier) with an infinite capacity, this is simply the TSP.

In the initial introduction of the Truck Dispatching Problem, the authors spent the majority of the paper discussing one variant of the problem. In their problem formulation, Dantzig and Ramser assumed that for the fleet of vehicles and set cities, 'only one product is to be delivered and that all trucks have the same capacity C' [2]. This formulation of the problem has subsequently become the most widely studied variant of Vehicle Routing Problems [5], and this formulation is referred to as the Capacitated Vehicle Routing Problem (CVRP). From the original introduction as the Truck Dispatching Problem, however, the Vehicle Routing Problem literature consists of numerous variations of the general problem. In addition to the CVRP, additional problems which have received a great deal of attention in the literature [5] are as follows:

- The Distance Constrained VRP
- The VRP with Time Windows
- The VRP with Backhauls
- The VRP with Pickups and Delivery

This research is focused on the CVRP.

2.2 Problem Formulation

As previously stated, this research focuses on the CVRP. Within the CVRP, there are two general variations. One in which the distances or costs associated with each arc (connection between nodes) are symmetric, and in the other case, these distances or costs are asymmetric. This research is specifically concerned with the symmetric CVRP, which is described as the following graph theoretic problem. Let $G = (V, A)$ be an undirected, complete graph, where $V = \{0, \ldots, n\}$ is the set of nodes, and A is the set of arcs which connect the nodes. A cost, c_{ij}, is associated with each arc, $(i, j) \in A$, where $i \neq j$. In this research the cost associated with each arc is simply the Euclidean distance between the nodes i and j, and because the problem is symmetric, $c_{ij} = c_{ji}$. Node 0 corresponds to the depot, whereas the remaining nodes, $V^{'} = V \setminus \{0\}$, correspond to the customers. Each customer $i \in V^{'}$ has a deterministic, non-negative demand, d_i, for goods. The goods are transported between the depot and the customers by one of vehicles in the known, homogeneous set of K vehicles. Each vehicle has a maximum capacity, Q, that is identical across the set of vehicles, and $d_i \leq Q$ for each $i = 1, \ldots, n$. The objective of the CVRP is to find K vehicle routes that start and end at the depot, with the minimum cost, where the demand of each customer is met, and the capacity of each vehicle is not violated.

2.2.1 Mathematical Programming Formulations

As described by Toth and Vigo [5], there are three general ways that the CVRP has been modeled in the literature. These three approaches are: *the vehicle flow formulation, the commodity flow formulation, and the set partitioning formulation.* Within each of these formulations, there are numerous variations and extensions. The three basic models are described below, and in each formulation, the graph $G(V, A)$ is assumed to be a complete graph. In line with the graph theoretic description, d_i is the demand associated with each city i; and given a set $S \subseteq V$, the

total demand for the set is $d(S) = \sum_{i \in S} d_i$. The formulations found in [5] follow.

Vehicle Flow Formulation

The basic *vehicle flow formulation* uses n^2 binary variables, x_{ij}, which correspond to each arc $(i, j) \in A$. If the arc (i, j) is in the optimal solution, $x_{ij} = 1$, otherwise, $x_{ij} = 0$.

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij} \tag{1}$$

subject to:

$$\sum_{i \in V} x_{ij} = 1 \qquad \forall j \in V \setminus \{0\}, \tag{2}$$

$$\sum_{j \in V} x_{ij} = 1 \qquad \forall i \in V \setminus \{0\}, \tag{3}$$

$$\sum_{i \in V} x_{i0} = K, \tag{4}$$

$$\sum_{j \in V} x_{0j} = K, \tag{5}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq r(S) \qquad \forall S \subseteq V \setminus \{0\}, S \neq \emptyset, \tag{6}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V. \tag{7}$$

The first two constraints, equations 2 and 3, ensure that each customer vertex has precisely one arc entering and leaving. The next two constraints, equations 4 and 5, ensure that exactly K (which corresponds to the number of vehicles) arcs enter and leave the depot vertex. The next constraint, equation 6, ensures that the solution routes are connected and that the capacity constraints are met. For a set $S \subseteq V \setminus \{0\}$, the term $r(S)$ refers to the minimum number of vehicles required to serve all of the customers in S, based upon the total demand of the set, $d(S)$. The term $r(S)$ is the optimal solution to the Bin Packing Problem, which is also an $\mathcal{NP}$-hard optimization problem. A valid [6] and common [5] substitute for $r(S)$ in the above formulation is the lower bound of the Bin Packing Problem $\lceil \frac{d(S)}{Q} \rceil$.

The above *vehicle flow formulation* is a basic model for an asymmetric CVRP. The *vehicle flow formulation* is easily refined for the symmetric CVRP (which is the focus of the current research) as follows:

$$\min \sum_{i \in V \setminus \{n\}} \sum_{j \geq i} c_{ij} x_{ij} \tag{8}$$

subject to:

$$\sum_{h \leq i} x_{hi} + \sum_{j \geq i} x_{ij} = 2 \qquad \forall i \in V \setminus \{0\}, \tag{9}$$

$$\sum_{j \in V \setminus \{0\}} x_{0j} = 2K, \tag{10}$$

$$\sum_{i \in S} \sum_{\substack{h \leq i \\ h \notin S}} x_{hi} + \sum_{i \in S} \sum_{\substack{j \geq i \\ j \notin S}} x_{ij} \geq 2r(S) \qquad \forall S \in V \setminus \{0\}, S \neq \emptyset, \tag{11}$$

$$x_{ij} \in \{0, 1\} \qquad \forall i, j \in V \setminus \{0\}, i \leq j, \tag{12}$$

$$x_{0j} \in \{0, 1, 2\} \qquad \forall j \in V \setminus \{0\}. \tag{13}$$

The symmetric CVRP *vehicle flow formulation* stems from the TSP formulation from Dantzig, et al [7], which was then extended to the symmetric CVRP formulation in [8, 9]. In the symmetric formulation of the problem, $c_{ij} = c_{ji}$, therefore the decision variable x_{ij} can take on a value of $\{0, 1, 2\}$, which represents the number of times that the arc is traversed in the final solution. In the case of a route with only one customer, $c_{ij} = 2$ since the (i, j) arc must be traversed twice in order to travel to and from the customer. The first constraint in the symmetric CVRP formulation, equation 9, is a combination of the first two constraints, equation 2 and equation 3, in the asymmetric formulation. This new constraint ensures that there is exactly one arc entering and one arc leaving each vertex (not including the depot vertex). The second constraint, equation 10, ensures that there is one arc entering and one arc exiting the depot vertex for each of the K vehicles in the problem. The third constraint, equation 10, ensures the continuity of the solution, as well as enforcing that the vehicle capacity constraints are not violated by requiring an adequate number of edges to enter each subset of vertices.

In addition to different versions of the *vehicle flow formulation* for the asymmetric and symmetric CVRP, additional versions and variations exist. Toth and Vigo [5] provide a detailed overview of the extensions and modifications of this model. One of the most notable extensions is the three index model [10, 11], which uses a binary variable x_{ijk} which is equal to 1 if vehicle k travels on arc (i, j).

Commodity Flow Formulation

The first application of the *commodity flow model* was to an oil delivery problem [12]. Subsequent work [13, 14, 5] extended the model to both the TSP and VRP. These early formulations of the *commodity flow formulation* are detailed in [15]. And a more recent example by Baldacci, et al [16] extends the TSP model introduced by Finke, et al [17].

The *commodity flow formulation* deals with an extended graph $G^{'} = \left(V^{'}, A^{'}\right)$. In this extended graph, $V^{'} = V \cup \{n+1\}$ where $\{n+1\}$ is a copy of the depot, and $A^{'} = A \cup \{(i, n+1) : i \in V\}$. As with the previously described model, the binary variables, x_{ij}, correspond to arc (i, j). If the arc (i, j) is in the optimal solution, $x_{ij} = 1$, otherwise, $x_{ij} = 0$. In addition to the variables introduced in the *vehicle flow formulation*, the *commodity flow formulation* introduces two new variables, y_{ij} and y_{ji}. If a vehicle travels along arc (i, j), the variable y_{ij} corresponds to the vehicle's load as it travels this arc, and the variable y_{ji} corrpesonds to the available capacity on the vehicle as it travels arc (i, j). For each arc $(i, j) \in A^{'}$, $y_{ij} + y_{ji} = Q$, where Q is the maximum capacity of the vehicle.

Toth and Vigo [5] describe the flow variables y_{ij} and y_{ji} as representing two directed paths through the solution. One path starts at the depot vertex, 0, and travels to the last vertex $n+1$. The variables on this path, y_{ij}, represent the vehicle load. The vehicle leaves the depot with a load equal to the sum of all customers' demands for the route. At each customer, the appropriate demand is delivered, and the vehicle ends at node $n+1$ empty. The second path starts at vertex $n+1$ and travels to the depot vertex, 0. This second path can be described as the vehicle leaving vertex $n+1$ empty, and at each customer, the demand, d_i is picked up by the vehicle. Therefore, when the vehicle arrives at the depot vertex, 0, the vehicle's load is equal to the total demand of all the customers along the path. Figure 2 illustrates the variables y_{ij} and y_{ji} for an example route with four cities, where $Q = 25$ [5]. Each city's demand is listed by the vertex.

The following is the *commodity flow formulation* for the symmetric CVRP.

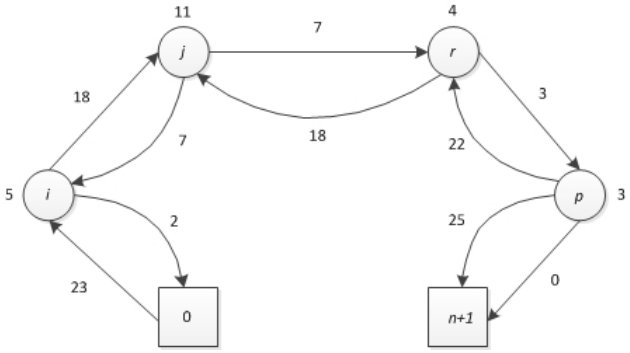


Figure 2: Example to illustrate the variables y_{ij} and y_{ji} on a route [5].

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij} \tag{14}$$

subject to:

$$\sum_{j\in V'} (y_{ji} - y_{ij}) = 2d_i \qquad \forall i \in V^{'} \setminus \{0, n+1\}, \tag{15}$$

$$\sum_{j\in V'\setminus\{0,n+1\}} y_{0j} = d(V \setminus \{0, n+1\}), \tag{16}$$

$$\sum_{j\in V'\setminus\{0,n+1\}} y_{j0} = KQ - d(V \setminus \{0, n+1\}), \tag{17}$$

$$\sum_{j\in V'\setminus\{0,n+1\}} y_{n+1j} = KQ, \tag{18}$$

$$y_{ij} + y_{ji} = Qx_{ij} \qquad \forall (i,j) \in A^{'}, \tag{19}$$

$$\sum_{j\in V'} (x_{ij} + x_{ji}) = 2 \qquad \forall i \in V^{'} \setminus \{0, n+1\}, \tag{20}$$

$$y_{ij} \geq 0 \qquad (i,j) \in A^{'}, \tag{21}$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A^{'}. \tag{22}$$

The constraints in Equations 15 - 18 ensure the appropriate flow between the depot vertex 0 and the vertex $n+1$. The next constraint, in Equation 19, ensures feasible values for y_{ij} and y_{ji}. The vehicle load along arc (i,j) plus the available capacity $y_{j,i}$ along arc (i,j) must always equal the vehicle's total capacity Q. And the constraint in Equation 20 ensures the continuity of the solution.

Set Partitioning Formulation

The *set partitioning formulation* of the VRP was introduced in 1964 by Balinski and Quandt [18]. It is possible for this model to require an exponential number of binary variables. Each binary variable, $x_j, j = 1, \ldots, r$, is associated with one of the r feasible circuits of G . A feasible circuit is a subset of vertices of V that does not violate the capacity constraint. The variable $x_j = 1$ if the circuit is included in the final solution, and $x_j = 0$ otherwise. Each of the r circuits has a minimum cost associated with it c_j^*; and a binary variable $a_{ij} = 1$ if vertex i is included in circuit j.

$$\min \sum_{j=1}^{r} c_j^* x_j \tag{23}$$

subject to:

$$\sum_{j=1}^{r} a_{ij} x_j = 1 \qquad \forall i \in V \setminus \{0\}, \tag{24}$$

$$\sum_{j=1}^{r} x_j = K, \tag{25}$$

$$x_j \in \{0, 1\} \qquad \forall j = 1, \ldots, r. \tag{26}$$

$$\tag{27}$$

The first constraint, Equation 24, ensures that each vertex is included in the solution, and the second constraint, Equation 25, requires exactly K circuits (vehicle routes) in the solution. It is important to note that finding c_j^* is, itself, an $\mathcal{NP}$-hard optimization problem [19].

2.3 Approaches to Solving CVRP

As previously described, the CVRP is an $\mathcal{NP}$-hard, which makes it difficult to solve to optimality for a non-trivial sized problem. The two general categories of algorithms used to solve the CVRP are exact algorithms and heuristic algorithms. Exact algorithms are concerned with finding the optimal solution to the CVRP, whereas heuristics are concerned with finding a good solution, but not necessarily the optimal.

2.3.1 Exact Algorithms

To date, the largest CVRP that has been solved to optimality consists of 134 cities [20, 21, 22, 23, 16]. Even though there have been significant advances in exact methods for solving the CVRP in the past 50 years, the existing exact methods are not yet capable of reliably solving

real-world-sized vehicle routing problems [19]. The earliest survey of exact methods for solving the VRP was the work by Laporte and Norbert [15]; which has been subsequently updated in 2002 [5], 2007 [24, 25]; 2008 [26], and 2010 [20]. Due to the limitations of these exact methods, the majority of VRP research focuses on heuristics [27].

2.3.2 Heuristic Methods

The first heuristic proposed for solving the VRP was introduced by Dantzig and Ramser with their initial introduction of the 'Vehicle Dispatching Problem' [2]. Since the introduction of the VRP, a significant amount of research has focused on heuristics methods for solving the $\mathcal{NP} - hard$ problem. The VRP heuristic literature is immense, and an extensive review of VRP heuristics can be found in [5, 28, 29].

VRP heuristics can be broadly divided into two categories: classical heuristics, and meta-heuristics [27]. A general overview of these two categories follows.

Classical Heuristics

Classical heuristics are described as methods that do not make a deep exploration of the search space, have fairly short computing times, and are capable of producing fairly good quality solutions [30]. Laporte [27] further describes classical heuristics with the fact that every step of the heuristic is a decent, which means that the heuristic never allows inferior solutions (even temporarily). These classical heuristics only move from one solution if the subsequent solution is superior. Classical heuristics are often divided into two categories: constructive and improvement heuristics.

Constructive Heuristics Constructive heuristics solve a problem by progressively building the routes. As the heuristic builds the solution, it monitors the cost, and ensures that the solution remains feasible. Three of the most widely known constructive heuristics are the Clarke and Wright algorithm, the sweep algorithm, and the Fisher and Jaikumar algorithm.

Clarke and Wright Savings Algorithm

One of the best known constructive heuristics is the Clarke and Wright Savings Algorithm [31]. This algorithm was introduced in 1964 and is still popular today [19]. The algorithm begins with n feasible routes, one back and forth route between the depot and each customer. On a given iteration, two routes $(0, \ldots, i, \ldots, 0)$ and $(0, \ldots, j, \ldots, 0)$ can be merged together if the resulting route $(0, \ldots, i, j, \ldots, 0)$ is feasible with regard to the capacity constraint. If these routes

are merged, there is a savings of $s_{ij} = c_{i0} + c_{0j} - c_{ij}$. The algorithm has two versions: a parallel implementation and a sequential implementation.

Both the parallel and sequential versions of the algorithm are initialized identically. The first step is to create a vehicle route, $(0, i, 0)$ for each city, $i \in V \setminus \{0\}$. Next, a list of savings $s_{ij} = c_{i0} + c_{0j} - c_{ij}$ for each $(i, j) \in A$ where $i \neq j$ is created. After the creation of the savings list, the steps of the parallel and sequential versions of the algorithm diverge.

In the parallel version of the algorithm, the next step consists of identifying the largest savings, s_{ij} in the savings list; followed by the determination of whether the two routes can be feasibly merged. If the two routes from the savings list, one containing the edge $(i, 0)$ and the other containing the edge $(0, j)$, can be feasibly merged, the two routes are combined resulting in a new route consisting of the edge (i, j). In this parallel version, multiple routes are being developed at the same time. This process continues until all s_{ij} have been inspected for potential route merging.

In the sequential version of the algorithm, after the creation of the n routes and the savings list, each route is fully expanded before the next route is considered. To accomplish this, one route $(0, i, \ldots, j, 0)$, is considered at a time. The current route is merged with another route that contains the largest savings, s_{ki} or s_{jl}, that can be feasibly merged with the current route. This process continues for the current route, until there are no more feasible merges. Then the next route is considered.

Several modifications have been introduced for the Clarke and Wright algorithm since its debut in 1964. One of the limitations of the Clarke and Wright algorithm is its tendency for the quality of the route construction to deteriorate near the end of the algorithm [30]. To overcome this, a shape parameter was introduced to the savings calculations by Gaskell [32] and Yellow [33], and subsequent work into the appropriate settings for this shape parameter was presented by Golden, et al [10]. The incorporation of a matching algorithm was introduced by Desrochers and Verhoog [34] as well as Altinkemer and Gavish [35] with the intent of optimizing the route merges at each iteration of the algorithm. Finally, enhancements to improve the computational time were presented by Paessens [36]; and Nelson et al [37], introduced efficient data structure usage. An in depth survey of the various modifications to the Clarke and Wright algorithm can be found in [30].

Sweep Algorithm

Another well known classical heuristic for the VRP is the sweep algorithm. The sweep

algorithm was introduced by Gillet and Miller [38] as a way to divide a large VRP into smaller subproblems. To implement the sweep algorithm, each city (node) must be represented by its polar coordinates (θ_i, r_i). The plane is centered at the depot, and one city, i^* is chosen such that $\theta_i^* = 0$. The polar coordinates for the remaining cities are found accordingly.

The cities are then ordered, by their increasing polar angle values. Starting with the first vehicle, $k = 1$, and the unrouted city with the smallest polar angle (the top of the city list), cities are assigned to vehicle $k = 1$ until the capacity of the vehicle has been met. Once the vehicle capacity is met, the next vehicle is initialized, $k = 2$, and the unrouted city with the smallest polar angle is assigned to this vehicle. Moving down the list of cities, a city is added to the current vehicle, k, until the vehicle capacity has been met, at which point a new vehicle is initialized and the process continues. The algorithm halts once all cities are assigned to a vehicle.

Once the cities have been partitioned amongst the k vehicles, each vehicle's route is found by solving the corresponding TSP. The TSP can be solved exactly or approximately. This sweep algorithm is sometimes referred to as a two phase method, which refers to the fact that one phase of the algorithm is devoted to clustering or partitioning the cities, and the other phase is concerned with the vehicle's route amongst these cities.

Fisher and Jaikumar Algorithm

The last well known classical heuristic that is discussed is the Fisher and Jaikumar algorithm [11]. Instead of relying on the geometric nature of the problem to form the clusters, the Fisher and Jaikumar algorithm first solves a Generalized Assignment Problem in order to divide the cities amongst the k vehicles. Similar to the sweep algorithm, the Jaikumar and Fisher algorithm is also commonly referred to as a two phase method, due to the fact that the tasks of clustering the cities and finding the routes are completed in two separate steps.

The first step of the algorithm is to identify k seed vertices $j_k \in V$. Each of these seeds corresponds to one of the K vehicle routes. Once these seeds are identified, a cost for allocating a customer, i, to each route, k, is computed: $d_{ik} = \min\{c_{0i}+c_{ij_k}+c_{j_k0}, c_{0j_k}+c_{j_ki}+c_{i0}\}-(c_{0j_k}+c_{j_k0})$. Next, the generalized assignment problem is solved with d_{ik} as the costs, the customer weights q_i, and the vehicle capacity Q. Once all customers are assigned to a route resulting from the generalized assignment problem, a TSP is then solved to find the optimal routing for each vehicle.

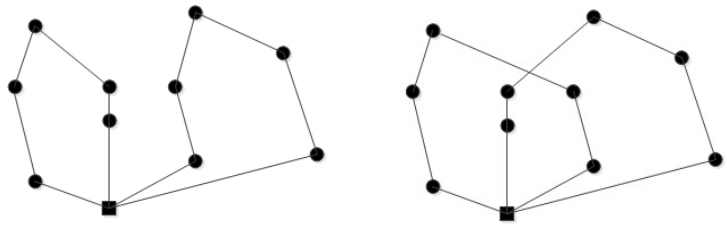
Improvement Heuristics The second category of classical heuristics is improvement heuristics. Improvement heuristics for a VRP are applied to an existing solution. Often these improve-

ment heuristics are applied to the results of previously described classical heuristics [28]. With this existing solution, improvements can be made within individual routes, or across many routes at the same time.

Improvements made to a single route at a time, intra-route improvements, stem from the $\lambda - opt$ TSP heuristic introduced by Lin and Kernighan [39]. Starting with a solution, the $\lambda - opt$ procedure removes λ edges from the solution and then reconnects the remaining segments of the solution in every possible combination. When a superior solution is found, the procedure is repeated, and this process continues until there are no further improvements. At this point, the solution is said to be $\lambda - opt$. Several improvements have been introduced to the $\lambda - opt$ method. Instead of setting a λ value at the beginning of the algorithm, Lin and Kernighan suggest changing λ dynamically while the algorithm is running [40]. Another modification presented by Or [41] entails removing and reinserting a chain of 3 vertices until no more improvements can be found, and this procedure is then repeated with chains of 2 vertices, followed by a single vertex. Additional modifications to intra-route improvement heuristics and modifications are described in [30, 42].

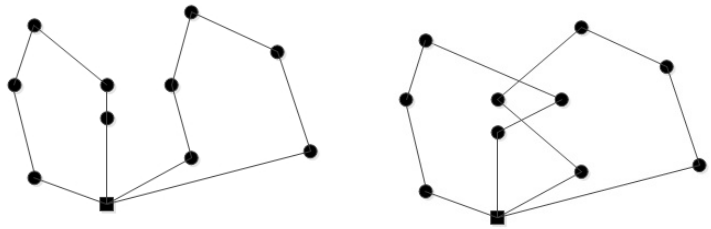
Improvements made across multiple routes are called inter-route improvements. Just as with an intra-route improvement, an inter-route improvement heuristic begins with a solution to the VRP. With this initial solution, an attempt is made to move cities/nodes amongst routes with the goal of finding a superior solution. Van Breedam [43] classifies inter-route improvement heuristics into four categories: string cross, string exchange, string relocation, and string mix. Van Breedam traces the main ideas of these heuristics to the previous work of Dror and Levy [44] and Savelsbergh [45]. Van Breedam's inter-route improvement heuristics are briefly described as follows:

- The string cross move consists of crossing two arcs of two distinct routes. This is equivalent to the exchange of route segments between the routes (see Figure 3).
- The string exchange swaps two sets of K vertices between routes. Normally $K = 1, 2$ (see Figure 4).
- The string relocation move shifts a string of K vertices from one route to another. Just as with the string cross, normally $K = 1, 2$ (see Figure 5).
- The string mix move is the better move: string exchange or string relocation.



(a) String Cross before (b) string cross after

Figure 3: string cross



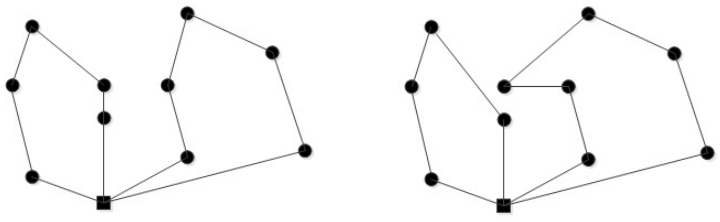
(a) String Exchange before (b) string exchange after

Figure 4: string exchange

One additional inter-route improvement heuristic to note is the work by Thompson and Psaraftis [46]. The authors develop a general b-cyclic, k-transfer system where k vertices are shifted to the next route in circular permutation of b routes. Additional inter-route improvements heuristics as well as modifications to the ones described can be found in [47, 48, 49, 50, 51, 30, 28].

2.3.3 Metaheuristics

The second general category for solving a CVRP is metaheuristics. Unlike classical heuristics, metaheuristics employ a deep search of the solution space [30], and they utilize concepts and procedures from the classical VRP heuristics [27]. In 1986 Fred Glover introduced the term



(a) String Relocation before (b) string relocation after

Figure 5: string relocation

'metaheuristic' to describe a heuristic that can be described generally, but which employs more specific heuristics necessary for the specific problem at hand [52]. Glover went on to predict that this type of metaheuristic will become very useful for solving combinatorial optimization problems. Since the initial introduction, numerous metaheuristics have been developed for various combinatorial optimization problems.

Now, instead of developing algorithms for a specific problem from scratch, these metaheuristics can be tailored to a specific problem, or type of problems [29]. Many metaheuristics have been shown to find nearly optimal solutions in an acceptable computation time for various combinatorial optimization problems [53]. The implementation of metaheuristics specifically to the VRP have been extremely successful [29].

Due to the volume of metaheuristics introduced in the literature, this section does not provide a complete survey of all applications of metaheuristics to the VRP. The most recent detailed survey of metaheuristics for the VRP can be found in [29]. This section will focus on three broad categories of metaheuristics [24, 27, 19]: local search, population search, and learning mechanisms.

Local Search

A local search heuristics is initialized with a solution. A neighborhood is defined around the current solution, and at each iteration, the heuristic moves to a new solution in this neighborhood. The procedure terminates based on predefined stopping criteria and returns the best solution found throughout the search. Given this general framework, various algorithms can be developed [27]. Two examples of local search algorithms are the simulated annealing and tabu search.

Simulated Annealing

Simulated annealing (SA) was first introduced as a technique for optimization in 1983 [54]. The general form of simulated annealing is described by Gendreau, et al [55], as follows. At each iteration of the simulated annealing algorithm, a solution x is randomly taken from the neighborhood of the current solution x_t. If the cost of this neighboring solution is less than the cost of the current solution, $f(x) \leq f(x_t)$, then the current solution is updated: $x_{t+1} = x$. If the cost of the neighboring solution is not an improvement, then the current solution is updated, $x_{t+1} = x$ with a probability of p_t; and $x_{t+1} = x_t$, otherwise. The probability, p_t is normally a decreasing function of both the iteration, t, and $f(x) - f(x_t)$, and can be defined as:

$$p_t = \exp\left(-[f(x) - f(x_t)]/\theta_t\right) \tag{28}$$

In Equation 28, θ_t represents the 'temperature' at iteration t, and the rule that defines and updates θ_t is called the 'cooling schedule'. Normally θ_t is a decreasing function. As a result, the probability of updating the current solution to a worse solution decreases over time. The algorithm's stopping criteria usually depends on the number of iterations, or the frequency of updates toward a better solution.

This meta-strategy generally guides the algorithm, however, the specifics of how the neighborhood is searched, as well as how the cooling schedule is defined significantly impact the algorithm's performance. One of the earliest and most successful implementations for the CVRP was by Osman in 1993 [48]. In his implementation, the algorithm was initialized with a solution found by the Clarke and Wright algorithm, and he employed similar inter and intra-route improvement techniques described in the previous section. Osman's simulated annealing algorithm resulted in generally good solutions, however, his computation times were rather long, and his results were not competitive with other location search heuristics, such as the tabu search.

Tabu Search

Another well known metaheuristic for the VRP is the tabu search, which can be traced back to Glover [52]. Similar to simulated annealing, the tabu search begins with an initial solution. At each iteration, the current solution is updated with the best solution found in the surrounding neighborhood, even if there is no improvement from the current solution. To avoid cycling amongst solutions, solutions that have been recently examined are marked as 'tabu' for a certain number of iterations. Again, whereas the general strategy across different tabu search algorithms is the same, the details of how the neighborhood is searched, the rules for updating solutions, and how the tabu routes are stored in memory are all factors that significantly influence a tabu search algorithm's performance.

Gendreau, et al [55] describe the earliest applications of a tabu search to the VRP. Although the earliest work of Willard [56] in 1989 was improved upon by Pureza and Franca [57] just two years later, Gendreau et al describes neither implementation of having very good results. However, more refined search strategies led to significant improvements in the application of the tabu search to VRP.

In his 1993 paper, Osman not only outlined an SA approach to solving the CVRP, he also introduced a tabu search metaheuristic [48]. In his tabu search, Osman used the same inter and intra-route improvements as he did with his SA algorithm for the neighborhood search. For the tabu search, he tested two different schemes for updating the current solution. In one scheme, he

searched the entire neighborhood to find the best solution before updating the current solution. In the second scheme, he updated the current solution as soon as an admissible improvement was found. Only feasible and non-tabu solutions were considered for an update.

Gendreau, et al, soon improved upon many of the results of Osman with a more intricate tabu search algorithm for the CVRP, which they called TABUROUTE. The authors defined a neighborhood as all of the solutions that can be found by removing one vertex from the current solution and reinserting into a different route that contains one of the vertex's nearest neighbors. This *generalized insertion procedure*, GENI, was previously created by the authors as a post-optimization procedure for the TSP [58]. Another notable difference with the TABUROUTE is that the algorithm's update procedure is such that updates can be made with infeasible solutions.

These are two of the earliest and most successful implementations of a tabu search for the CVRP. A thorough listing of further developments in the application of the tabu search to the CVRP can be found in [29]. In addition to simulated annealing and tabu search, additional local searches noted in [19] include variable neighborhood search [59], very large neighborhood search [60], and adaptive large neighborhood search [61].

Population Search

Population searches include algorithms that employ the use of a population of solutions throughout the searching procedure. The most famous example of this type of algorithm is the Genetic Algorithm (GA) pioneered by John Holland in 1975 [62]. Unlike the local searches, previously described, which are initialized with one starting solution, a GA is initialized with a population of solutions. Many combinatorial optimization problems have been solved using GA's [53], including the VRP. A basic outline of a GA follows.

1 An initial population of solutions is created.

2 The fitness of each solution is evaluated.

3 For each iteration, $t = 1, \ldots, T$, the following is repeated m times:

  [a] Select two parents (from the population of solutions).

  [b] Create two offspring from the parent solutions using crossover.

  [c] With some small probability, apply mutation to each offspring.

  [d] The worst elements in the current population are replace with the $2m$ offspring.

4 Return the best solution

Potvin [63] details many of the successes as well as the inherent difficulties of developing a GA suitable for solving the VRP, and he provides a comprehensive literature review of the GA's developed for solving the VRP. Many of the algorithms do precisely follow the general scheme of the GA, however, they utilize many of the evolutionary principles from the GA. Some notable GA and evolutionary approaches to solving the CVRP are found in [64, 65, 66, 67].

Additional population based searches for the VRP include the Adaptive Memory Procedure introduced by Rochat and Taillard [68]. In this algorithm, crossover and local search procedures are executed on a population of solutions produced by the tabu algorithm. Starting with an initial population over very good solutions (results of tabu search), the idea is to generate even better solutions with the crossover and local search. Lastly, the Particle Swarm Optimization technique also utilizes a population of solutions and has beens successfully applied to the CVRP [69, 70].

Learning Mechanisms

Learning mechanisms include algorithms that learn from one iteration of the algorithm to the next. The two algorithms in this category, described in [27, 19, 28], are Artificial Neural Networks and Ant Colony Optimization. ANNs have been used to solve various combinatorial optimization problems, including the VRP. The application of ANNs to the VRP will be covered extensively in Chapter 3.

Ant algorithms were first introduced in 1991 by Colormi et al, and are based on the real life behavior of ant colonies who are searching for food [71]. Initially, ants wander randomly from their colony in search of food. As they move about, the ants continually release a pheromone trail along the path they travel. These pheromones are a means of communication between ants, and they provide an indication about the length of the path, as well as the quality of food encountered. If additional ants follow the same path, the strength of the pheromone along the path is increased. Over time, however, a pheromone evaporates. As a result, the most frequented paths will maintain a high level of pheromone, whereas the less traveled paths will lose their pheromone. When the path to a high quality food source becomes frequented by an increasing number of ants, this only encourages additional ants to follow.

The original work by Colormi et al was applied to the TSP, which is described as follows in [30]. Each pair of vertices, v_i, v_j has two corresponding values. The first value is a *visibility*

variable, $\nu_{ij} = \frac{1}{d_{ij}}$, where d_{ij} is the Euclidean distance between the two vertices. The second corresponding value for each pair of vertices is the pheromone trail, Γ_{ij}, which is updated as the algorithm iterates. At each iteration of the algorithm, artificial ants construct n new TSP tours based on a nearest neighbor heuristic where the distance between vertices v_i, v_j, consists of both Euclidean distance, ν, as well as the pheromone value, Γ_{ij}. At the end of each iteration, a fraction, ρ, of the pheromone for each edge evaporates, and pheromones for the edges included in the resulting tours are increased. The pheromone between two vertices is updated as $\Gamma_{ij} = \rho\Gamma_{ij} + \sum_{k=1}^{N} \delta_{ij}^k$, where $\delta_{ij}^k = \frac{1}{L_k}$ if ant k uses edge (v_i, v_j), and the length of the resulting tour is L_k. Tours are repeatedly constructed and the pheromone trails are updated for a specified number of iterations.

The original ant approach to the TSP was improved in [72, 73], and subsequently extended to the VRP in [74, 75, 76, 77].

2.4 Benchmark Problem Sets

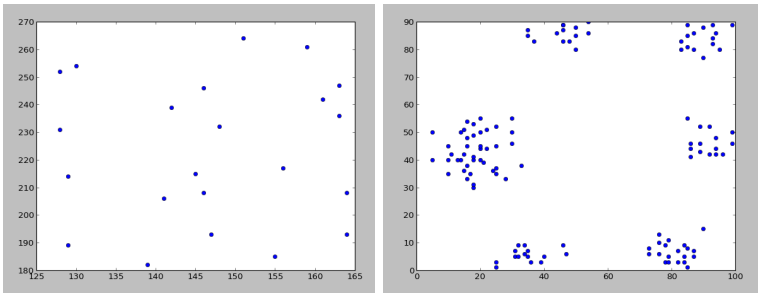
There are a significant number of benchmark problems sets that exist to test VRP algorithms. One of the most comprehensive collections of these problem sets can be found in [78]. These benchmark problem sets provide a straightforward way to compare the solution quality of different algorithms across many different problem instances. However, even within one type of problem, the CVRP for example, there are numerous problem characteristics that can vary, and these variations can have a significant impact on the performance of a given algorithm. The No Free Lunch Theorems [79] state that there is not one general purpose algorithm that will outperform all other algorithms on all optimization problems, and this idea can be extended to performance within a specific type of problem (ie, the CVRP). Certain CVRP algorithms might perform well when the customers are evenly distributed within an area, however these algorithms might not perform as well on problems where the customers are clustered.

List of References

[1] B. L. Golden, A. A. Assad, and E. A. Wasil, "Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries," *The vehicle routing problem*, vol. 9, pp. 245–286, 2002.

[2] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[3] E. Lawler, J. Lenstra, A. Rinnooy Kan, and D. Shmoys, *The Traveling Salesman Problam*, 1st ed. John Wiley and Sons Ltd., 1985.



(a) uniformly distributed cities (b) clustered cities

Figure 6: Different Distributions of Customers

[4] W. Cook, *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, 2012.

[5] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[6] G. Cornuejols and F. Harche, "Polyhedral study of the capacitated vehicle routing problem," *Mathematical Programming*, vol. 60, no. 1-3, pp. 21–52, 1993.

[7] G. Dantzig, R. Fulkerson, and S. Johnson, "Solution of a large-scale traveling-salesman problem," *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[8] G. Laporte and Y. Nobert, "A branch and bound algorithm for the capacitated vehicle routing problem," *Operations-Research-Spektrum*, vol. 5, no. 2, pp. 77–85, 1983.

[9] G. Laporte, Y. Nobert, and M. Desrochers, "Optimal routing under capacity and distance restrictions," *Operations research*, vol. 33, no. 5, pp. 1050–1073, 1985.

[10] B. L. Golden, T. L. Magnanti, and H. Q. Nguyen, "Implementing vehicle routing algorithms," *Networks*, vol. 7, no. 2, pp. 113–148, 1977.

[11] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.

[12] W. Garvin, H. Crandall, J. John, and R. Spellman, "Applications of linear programming in the oil industry," *Management Science*, vol. 3, no. 4, pp. 407–430, 1957.

[13] B. Gavish and S. Graves, "The traveling salesman problem and related problems," 1979.

[14] B. Gavish and S. Graves, "Scheduling and routing in transportation and distributions systems: Formulations and new relaxations." 1982.

[15] G. Laporte and Y. Nobert, "Exact algorithms for the vehicle routing problem," *North-Holland Mathematics Studies*, vol. 132, pp. 147–184, 1987.

[16] R. Baldacci, E. Hadjiconstantinou, and A. Mingozzi, "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, 2004.

[17] G. Finke, A. Claus, and E. Gunn, "A two-commodity network flow approach to the traveling salesman problem," *Congressus Numerantium*, vol. 41, no. 1, pp. 167–178, 1984.

[18] M. L. Balinski and R. E. Quandt, "On an integer program for a delivery problem," *Operations Research*, vol. 12, no. 2, pp. 300–304, 1964.

[19] G. Laporte, "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, 2009.

[20] R. Baldacci, P. Toth, and D. Vigo, "Exact algorithms for routing problems under vehicle capacity constraints," *Annals of Operations Research*, vol. 175, no. 1, pp. 213–245, 2010.

[21] J. Lysgaard, A. N. Letchford, and R. W. Eglese, "A new branch-and-cut algorithm for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 100, no. 2, pp. 423–445, 2004.

[22] R. Fukasawa, H. Longo, J. Lysgaard, M. P. d. Aragão, M. Reis, E. Uchoa, and R. F. Werneck, "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical programming*, vol. 106, no. 3, pp. 491–511, 2006.

[23] P. Augerat, J. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi, *Computational results with a branch and cut code for the capacitated vehicle routing problem.* IMAG, 1995.

[24] J.-F. Cordeau, G. Laporte, M. W. Savelsbergh, and D. Vigo, "Vehicle routing," *Transportation, handbooks in operations research and management science*, vol. 14, pp. 367–428, 2007.

[25] R. Baldacci, P. Toth, and D. Vigo, "Recent advances in vehicle routing exact algorithms," *4OR*, vol. 5, no. 4, pp. 269–298, 12 2007, copyright - Springer-Verlag 2007; Last updated - 2012-06-29.

[26] B. L. Golden, S. Raghavan, and E. A. Wasil, *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges.* Springer Science & Business Media, 2008, vol. 43.

[27] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics (NRL)*, vol. 54, no. 8, pp. 811–819, 2007.

[28] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany, "New heuristics for the vehicle routing problem," in *Logistics Systems: Design and Optimization*, A. Langevin and D. Riopel, Eds. Springer US, 2005, pp. 279–297. [Online]. Available: http://dx.doi.org/10.1007/0-387-24977-X_9

[29] M. Gendreau, J.-Y. Potvin, O. Bräumlaysy, G. Hasle, and A. Løkketangen, "Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography," in *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, B. L. Golden, S. Raghavan, and E. A. Wasil, Eds. Springer, 2008, pp. 143–169.

[30] G. Laporte and F. Semet, "Classical heuristics for the capacitated vrp," in *The Vehicle Routing Problem*, P. Toth and D. Vigo, Eds. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications, 2002, pp. 109–128.

[31] G. u. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations research*, vol. 12, no. 4, pp. 568–581, 1964.

[32] T. Gaskell, "Bases for vehicle fleet scheduling," *OR*, pp. 281–295, 1967.

[33] P. Yellow, "A computational modification to the savings method of vehicle scheduling," *Operational Research Quarterly*, pp. 281–283, 1970.

[34] M. Desrochers and T. Verhoog, "A matching based savings algorithm for the vehicle routing problem," *Cahiers du GERAD*, 1989.

[35] K. Altinkemer and B. Gavish, "Parallel savings based heuristics for the delivery problem," *Operations Research*, vol. 39, no. 3, pp. 456–469, 1991.

[36] H. Paessens, "The savings algorithm for the vehicle routing problem," *European Journal of Operational Research*, vol. 34, no. 3, pp. 336–344, 1988.

[37] M. D. Nelson, K. E. Nygard, J. H. Griffin, and W. E. Shreve, "Implementation techniques for the vehicle routing problem," *Computers & Operations Research*, vol. 12, no. 3, pp. 273–283, 1985.

[38] B. E. Gillett and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Operations research*, vol. 22, no. 2, pp. 340–349, 1974.

[39] S. Lin, "Computer solutions of the traveling salesman problem," *Bell System Technical Journal, The*, vol. 44, no. 10, pp. 2245–2269, 1965.

[40] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[41] I. Or, *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Xerox University Microfilms, 1976.

[42] G. Babin, S. Deneault, and G. Laporte, "Improvements to the or-opt heuristic for the symmetric travelling salesman problem," *Journal of the Operational Research Society*, vol. 58, no. 3, pp. 402–407, 2007.

[43] A. Van Breedam, *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints*. RUCA, 1994.

[44] M. Dror and L. Levy, "A vehicle routing improvement algorithm comparison of a greedy and a matching implementation for inventory routing," *Computers & Operations Research*, vol. 13, no. 1, pp. 33–45, 1986.

[45] M. Savelsbergh, "Computer aided routing," Ph.D. dissertation, Erasmus University, The Netherlands, 1988.

[46] P. M. Thompson and H. N. Psaraftis, "Cyclic transfer algorithm for multivehicle routing and scheduling problems," *Operations research*, vol. 41, no. 5, pp. 935–946, 1993.

[47] R. Fahrion and M. Wrede, "On a principle of chain-exchange for vehicle-routeing problems (1-vrp)," *Journal of the Operational Research Society*, pp. 821–827, 1990.

[48] I. H. Osman, "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of operations research*, vol. 41, no. 4, pp. 421–451, 1993.

[49] J.-Y. Potvin, T. Kervahut, B.-L. Garcia, and J.-M. Rousseau, "The vehicle routing problem with time windows part i: tabu search," *INFORMS Journal on Computing*, vol. 8, no. 2, pp. 158–164, 1996.

[50] J. Xu and J. P. Kelly, "A network flow-based tabu search heuristic for the vehicle routing problem," *Transportation Science*, vol. 30, no. 4, pp. 379–393, 1996.

[51] C. Rego and C. Roucairol, "A parallel tabu search algorithm using ejection chains for the vehicle routing problem," in *Meta-Heuristics*. Springer, 1996, pp. 661–675.

[52] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.

[53] M. Gendreau and J.-Y. Potvin, "Metaheuristics in combinatorial optimization," *Annals of Operations Research*, vol. 140, no. 1, pp. 189–213, 2005.

[54] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, *et al.*, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[55] M. Gendreau, G. Laporte, and J.-Y. Potvin, "Metaheuristics for the capacitated vrp," in *The Vehicle Routing Problem*, P. Toth and D. Vigo, Eds. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications, 2002, pp. 129–154.

[56] J. Willard, "Vehicle routing using r-optimal tabu search," 1989.

[57] V. Pureza and P. Franca, "Vehicle routing problems via tabu search metaheuristic," 1991.

[58] M. Gendreau, A. Hertz, and G. Laporte, "New insertion and postoptimization procedures for the traveling salesman problem," *Operations Research*, vol. 40, no. 6, pp. 1086–1094, 1992.

[59] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[60] Ö. Ergun, J. B. Orlin, and A. Steele-Feldman, "Creating very large scale neighborhoods out of smaller ones by compounding moves," *Journal of Heuristics*, vol. 12, no. 1-2, pp. 115–140, 2006.

[61] S. Ropke and D. Pisinger, "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation science*, vol. 40, no. 4, pp. 455–472, 2006.

[62] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* University of Michigan Press, Ann Arbor, 1975.

[63] J.-Y. Potvin, "State-of-the art review- evolutionary algorithms for vehicle routing," *INFORMS Journal on Computing*, vol. 21, no. 4, pp. 518–548, 2009.

[64] B. M. Baker and M. Ayechew, "A genetic algorithm for the vehicle routing problem," *Computers& Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.

[65] C. Prins, "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.

[66] E. Alba and B. Dorronsoro, "Computing nine new best-so-far solutions for capacitated vrp with a cellular genetic algorithm," *Information Processing Letters*, vol. 98, no. 6, pp. 225–230, 2006.

[67] D. Mester and O. Bräysy, "Active-guided evolution strategies for large-scale capacitated vehicle routing problems," *Computers & Operations Research*, vol. 34, no. 10, pp. 2964–2975, 2007.

[68] y. Rochat and É. D. Taillar, "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of heuristics*, vol. 1, no. 1, pp. 147–167, 1995.

[69] A.-l. Chen, G.-k. Yang, and Z.-m. Wu, "Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem," *Journal of Zhejiang University Science A*, vol. 7, no. 4, pp. 607–614, 2006.

[70] C. Pornsing, "A particle swarm optmiization for the vehicle routing problem," Ph.D. dissertation, University of Rhode Island, 2014.

[71] A. Colorni, M. Dorigo, and V. Maniezzo, "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142. Paris, France, 1991, pp. 134–142.

[72] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *Systems, Man and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.

[73] M. Dorigo and L. M. Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

[74] B. Bullnheimer, R. F. Hartl, and C. Strauss, "Applying the ant system to the vehicle routing problem," in *Meta-Heuristics*. Springer, 1999, pp. 285–296.

[75] B. Bullnheimer, R. F. Hartl, and C. Strauss, "An improved ant system algorithm for the vehicle routing problem," *Annals of operations research*, vol. 89, pp. 319–328, 1999.

[76] M. Reimann, M. Stummer, and K. Doerner, "A savings based ant system for the vehicle routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Morgan Kaufmann Publishers, Inc., 2002, pp. 1317–1326.

[77] M. Reimann, K. DOerner, and R. F. Hartl, "D-ants: Savings based ants divide and conquer the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 4, pp. 563–591, 2004.

[78] University of Malaga. "Neo: Networking and emerging optimization group." Jan. 2013. [Online]. Available: http://neo.lcc.uma.es/vrp/

[79] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

CHAPTER 3

Applications of ANNs to the CVRP

3.1 Artificial Neural Networks

Interest in ANNs began with the original work on networks of neurons by McCulloch and Pitts in 1943 [1]. Jain and Mao [2] describe this as the first of three distinct peaks in the research interest of ANNs. The second peak followed Rosenblatt?s perceptron convergence theorem in 1962 [3], which was a continuation of his original introduction of the perceptron in 1958 [4]. The book by Minsky and Papert in 1969 [5], however, purported to show the limitations of the perceptron and caused many researchers to lose interest in pursuing further work with ANNs. The third resurgence of interest in ANNs began in the 1980's with the introduction of the Hopfield Network [6]. In addition to Hopfield's work, McClelland and Rumelhart [7] revived the multilayer feed forward networks which was the original work of Werbos [8].

ANNs can be generally broken down into two learning categories: supervised and unsupervised. Supervised learning can be thought of as learning with a teacher [9]. With supervised learning, the ANN is trained with a set of data where the desired output is known. During this training phase, the network output is compared to the desired output, and any discrepancy is used to update the weights of the network in a manner such that the error is minimized. Unsupervised learning is used when the desired output for a given input is unknown. The goal of unsupervised learning is to detect similarities in the input data in order to cluster or categorize the data. The three prominent applications of ANNs to combinatorial optimization problems are examples of unsupervised learning.

3.2 The Application of ANNs to Combinatorial Optimization Problems

3.2.1 Hopfield Neural Network

The first application of an ANN to a combinatorial optimization problem was in 1985 when Hopfield and Tank applied the Hopfield Neural Network (HNN) [6, 10] to the Traveling Salesman Problem (TSP) [11]. The HNN is an example of an unsupervised learning method. To solve an n city TSP problem with an HNN, it requires a network of n^2 fully connected neurons where the rows can be thought of as the cities, and the columns represent the position in which the city is visited (figure 1 illustrates the HNN representation of a TSP). An energy function is used to represent the state of the network (which is similar to the objective function of the optimization

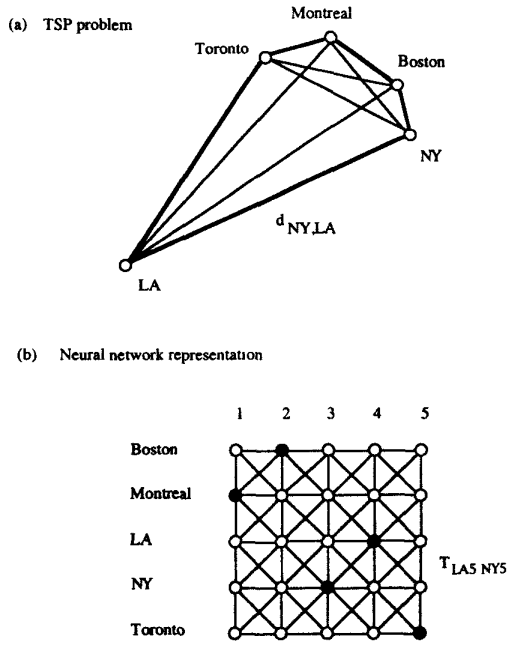


Figure 7: HNN representation of a TSP [17]

problem), and the neuron connection weights are derived from this energy function. Once the neurons are initialized, the network evolves in such a way that the energy function continues to decrease until a minimum is reached. Ideally this minimum corresponds to a good, feasible solution, but subsequent research suggested otherwise. Although the HNN sparked enthusiasm in both the ANN and Operations Research communities [12], within a few years, it also generated skepticism as subsequent researchers were unable to replicate the results of Hopfield and Tank [13]. To date, researchers have been able to address the issues of parameter setting and convergence [14, 15]. However, the vast majority of research into the use of ANNs for combinatorial optimization ha focused on the HNN [16] since it is easily generalized to a wide range of Combinatorial Optimization Problems HNN has been researched extensively and applied to many other combinatorial optimization problems [15].

3.2.2 Self Organizing Map

The Elastic Net (EN) and the Self-Organizing Map (SOM) are two additional ANN approaches to solving combinatorial optimization problems. Both of these methods use a geometric approach to solving a TSP, yet they differ in how their respective networks evolve. Due to the similarities between the EN and SOM approaches to solving combinatorial optimization problems, these two methods are often grouped together in the literature as a SOM approach [12].

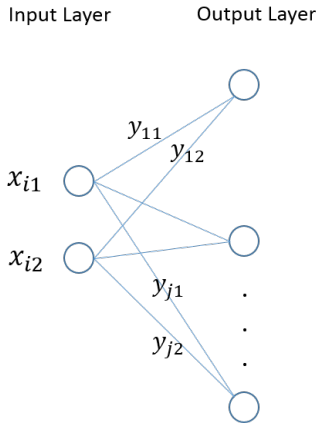
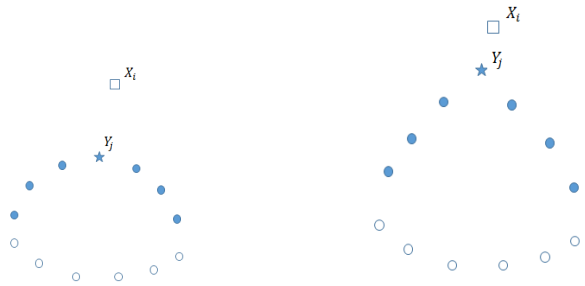


Figure 8: Two Layer Neural Network Used in EN and SOFM

The Elastic Net was proposed in 1987 [18] for solving a TSP and is described as a procedure where a small, circular band of neurons begin near the center of the cities. This band is then stretched and elongated toward each city in such a way that an energy function is decreased. The application of the SOM to the TSP was an extension of Kohonen's original work on Self-Organizing Feature Maps (SOM) [19, 20, 21] which was not specific to combinatorial optimization problems or the TSP. Kohonen introduced the SOM as an ANN that identifies patterns and similarities in input data and then organizes itself in a manner that represents this relationship amongst the data [12]. The extension of the SOM to the TSP [22, 23, 24] resulted in a similar approach as the EN. The SOM approach to the TSP also began with a small circular ring of neurons centered amongst the cities, which was then moved and stretched toward the cities. The SOM, unlike the EN and HNN, did not use an energy function in its formulation. The capabilities of both the EN and SOM seem to surpass those of the Hopfield Network for solving the TSP [17, 25].

Both the EN and SOM employ a similar, geometric strategy for solving the TSP. The network consists of two layers, where the input layer allows for the input of one city, X_i, at a time by its x and y components, (X_{i1}, X_{i2}). The output layer consists of all of the nodes in the problem, where the number of nodes, M, is greater than or equal to the number of cities, N. This is a fully connected, two layer network, where the connection weights correspond to the (x, y) coordinates of the corresponding node, Y_j.

The weights are initialized randomly, normally in such a way that the nodes form a small ring near the center of the cities. At each step, one city is input to the network, and all nodes compete to determine which is the closest. In the simplest case, the distance between a node



(a) winning node and neighbors identified

(b) positions updated

Figure 9: Updating Positions of Winning Node and Neighbors

and city is measured only by Euclidean distance. Once a winner is identified, the weights of the winning node and its neighbors are updated by being moved closer to the city. The number of neighbors that are affected by the winning node is a parameter in the problem, and decreases over time. As the distance between the winning node and its neighbors (in terms of how many 'nodes' they are away from each other) increases, the movement toward the city decreases. Figure 9 illustrates a network where Y_j was found to be the winning node for city X_i. The shaded nodes around Y_j represent the neighbors whose positions will be updated along with Y_j. The algorithm halts once each city has a node within an acceptable distance. The evolution of a network for a TSP is shown in Figure 10.

Both the SOM and EN have been applied to combinatorial optimization problems beyond the TSP, including the Multiple Traveling Salesman Problem (mTSP) [27, 28, 29, 26, 30], and the Vehicle Routing Problem [31, 32, 26, 30]. To extend the ideas from the TSP to the CVRP, in general, two changes must be made. The first is that instead of the problem consisting of one ring of nodes, as in the TSP described previously, in the CVRP there are K rings, where K = number of vehicles; therefore each ring corresponds to one of the vehicle routes. The second aspect that must be considered is the capacity constraint in the CVRP. There are multiple variations of how researchers have embedded this constraint into the ANN formulation. In addition to the different methods of embedding the capacity constraint, the applications of the SOM and EN to the CVRP also vary in the node update rules. A brief description of the general ideas behind the applications of EN and SOM to the CVRP follows.

In 1991 Ghaziri [33] presented an application of the SOM to the VRP which he called the Hierarchical Deformable Net (HDN). In this algorithm, the nodes compete by Euclidean distance

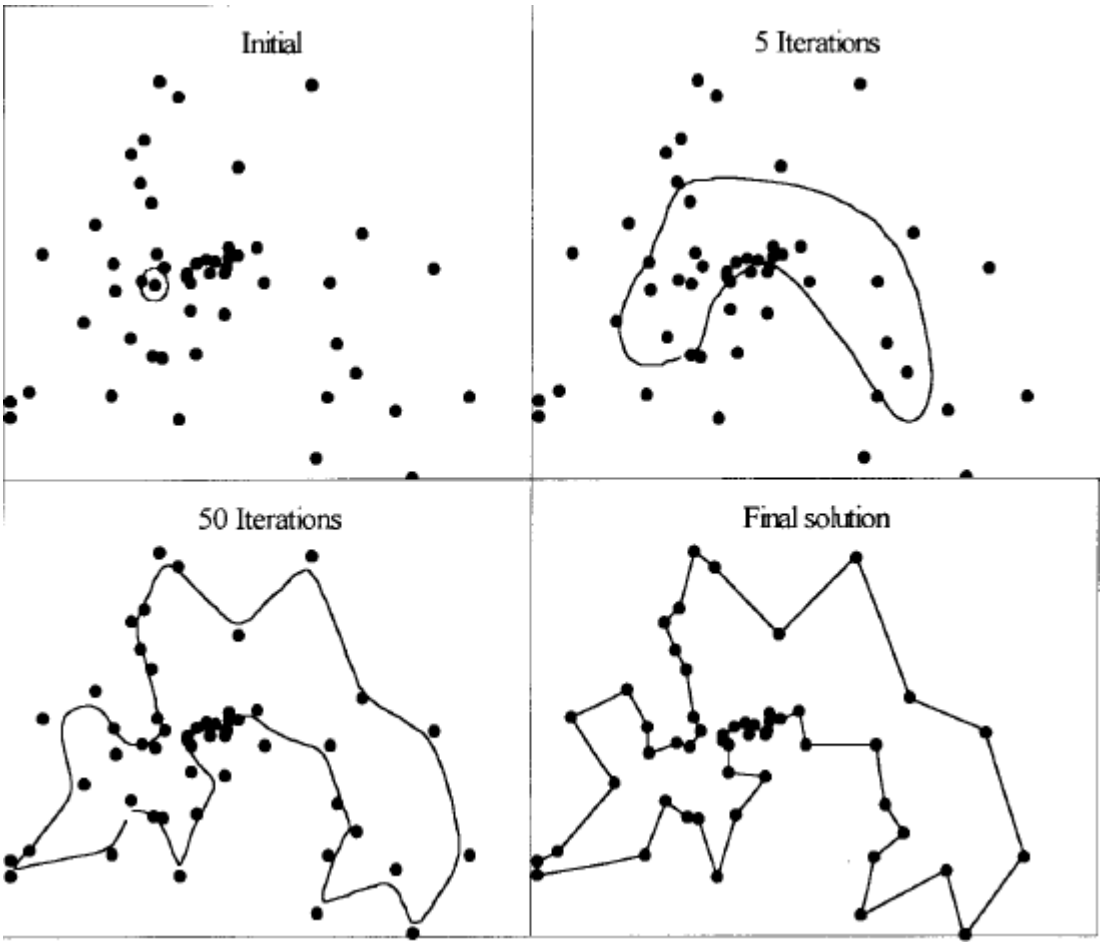


Figure 10: Evolution of Network for TSP [26]

as previously described, however, the closest node on each of the K rings is found. After one node on each ring is identified, he uses a probabilistic approach to determine the overall winning node. He calculates the probability in such a way that as the network evolves, the probability that a winning node is found on an overloaded route is nearly zero. This ensures the feasibility of the solution. Once a winning node is selected, the node position of the winner and its neighbors are updated as described above, where the winner and neighbors are moved in a straight line distance toward the city. Ghaziri compared his algorithm to three well-known heuristics: Clarke and Wright; Mole and Johnson; and the Sweep Algorithm. He tested the algorithms on three standard test problems [34] and determined that his algorithm was competitive with the existing heuristics. Ghaziri later extended his HDN to the CVRP with time windows [35] as well as the Vehicle Routing Problem with Backhauls [36].

Also in 1991, Matsuyama [37] presented an extension of the SOM to a vehicle routing problem with different types of demands, and each truck was not necessarily capable of picking up every type of demand. He tested his algorithm on the 532 city symmetric TSP data presented by [38], to which he added the demands. In his algorithm, nodes compete to determine which was the winner based on Euclidean distance, but to account for the capacity constraints of the vehicles, the Euclidean distance was multiplied by a 'handicap' factor. A bigger handicap indicates that the current ring is near or over its capacity constraint. In order to update the node positions in his algorithm, Matsuyama uses two factors. A node's movement is influenced by both the position of the city, as well as the position of the node's immediate neighbor on both sides. This position update rule is similar to the update rule presented in the application of the EN to the TSP [18]. Matsuyama's results for the VRP are ambiguous because he did not make a comparison with any other algorithms, however, his work does provide insight into how the handicap factor can be used to account for the capacity constraint in a CVRP.

Vakhutinsky and Golden [31] extended the EN to the CVRP in 1994. To account for the capacity constraint in the VRP, they incorporated an inhibitor coefficient, and they use the same node update rule as the original EN application to the TSP [18]. To test their algorithm, they use five well known test problems from [39]. However, instead of comparing their results to the best known algorithms at the time, Vakhutansky and Golden compare their 1994 results to the 1971 results from [39] and determine that their results are comparable. This isn't a very telling comparison however, since the best known results in 1994 were significantly improved from 1971.

In 1997 Torki, et al [32] developed a SOM application to the CVRP, which incorporates a

bias term to account for the capacity constraint. When the nodes compete to determine which node was the closest, both the Euclidean distance and this bias term are considered. Similar to Matsuyama and the EN, Torki also incorporates neighboring node position when updating a node's position. Torki et al applies their algorithm to ten standard test problems and compare the results to those of Vakhutansky and Golden's EN application to the CVRP [31]. The results of Torki et al's SOM algorithm clearly surpasses the results of the EN.

Modares et al [26] presented a SOM approach to the CVRP in 1999 that is very similar to Torki's algorithm. Similar to Torki, Modares' SOM approach to the CVRP uses a bias term to identify the winning node, as well as incorporating neighbor positions when updating node positions. The only difference in these two approaches is a slightly different bias term. The Modares paper uses the exact same test problems as Torki, and reports the exact same results.

In 2005, Schwardt and Dethloff presented an application of the SOM to the continuous location-routing problem (LRP) [40]. They use 11 benchmark CVRP data sets to test their algorithm. They test their algorithm on both the LRP, which is essentially the same as a CVRP except that the depot location is variable, and they also test their algorithm on the standard CVRP. In their SOM algorithm, they use a bias term in addition to Euclidean distance when determining the winning node for a city. However, unlike the other algorithms discussed so far, Schwardt and Dethloff also include a 'tabu counter' which is utilized when a vehicle route violates the capacity constraint during the evolution of the network. When a route is over capacity, the city (assigned to the over-capcity route) that is the closest to a node on a different route becomes 'tabu' for this over-capacity route. For a specified number of iterations, all nodes on this over-capacity route are excluded from the competition for the 'tabu' city. Once the specified number of iterations pass, the nodes on this once 'tabu' route can compete again. When updating their node positions, the algorithm only uses the distance between a node and the city in order to update the positions of the winning node and its neighbors. The results of their algorithm on the 11 benchmark CVRP data sets are superior to previously reported SOM results [32, 26]. In their conclusions, they discuss the parameter sensitivity between different problem sets, as well as the fact that despite the inclusion of a bias term and 'tabu counter', their algorithm can result in infeasible solutions.

In 2008 Creput and Koukam presented a memetic VRP algorithm which embedded a SOM in an evolutionary algorithm [41]. In their algorithm, the authors begin with a population of identical problems, and each member in the population undergoes a specified number of iterations

of the SOM algorithm; the SOM algorithm is the inner loop of the memetic algorithm. After the specified number of iterations of the SOM algorithm, each problem is forced to a solution. The best solution is found within the population, and the population is then updated to replace the worst solutions with the best solutions. At this point, operations are conducted to introduce perturbations in order to make the solutions feasible. Once these solutions are found, the process starts again with a new population in the SOM phase. The algorithm continues for a predetermined number of iterations, and the best solution is reported. The authors first test their algorithm on 14 Christofides benchmark problems and compare their results to all other SOM results that have been reported on these same problems [33, 26, 40]. Overall, the results of the memetic SOM outperform other SOM heuristics. The authors then test their algorithm on the 20 benchmark problems presented by Golden et al [42]. The authors compare their algorithm's performance to the results of the best performing CVRP algorithms presented in [43]. Although the memetic SOM does not outperform all other algorithms, the algorithm is competitive with a few heuristics on solution quality and computation time.

In addition to the methods already described, ANNs have also been applied to the CVRP in two additional ways that combine ANNs with other well known heuristics. The first method trains an ANN to select the best VRP heuristic for a given problem [44] based on characteristics of the problem set. This approach recognizes that a heuristic's performance is influenced by characteristics of a problem. The study identifies important problem characteristics and then uses them to train an ANN to identify which heuristic, from a predetermined finite set, will perform best on the given problem. The second method uses an ANN to enhance the initialization phase of a route construction heuristic for a VRP with time windows [45]. The ANN is specifically used to help cluster the cities prior to the start of the construction heuristic.

List of References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[2] A. K. Jain, J. Mao, and K. Mohiuddin, "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

[3] F. Rosenblatt, *Principles of Neurodynamics*. Spartan Book, 1962.

[4] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

[5] M. Minsky and S. Papert, "Perceptron: An introduction to computational geometry," *The MIT Press, Cambridge, expanded edition*, vol. 19, p. 88, 1969.

[6] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[7] J. L. McClelland, D. E. Rumelhart, P. R. Group, *et al.*, "Parallel distributed processing," *Explorations in the microstructure of cognition*, vol. 2, 1986.

[8] P. WERBOS, "Beyond regression: New tools for prediction and analysis in the behavioral sciences," *PhD thesis, Harvard University*, 1974.

[9] R. Rojas, *Neural networks: a systematic introduction*. Springer, 1996.

[10] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[11] J. J. Hopfield and D. W. Tank, "neural computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[12] K. A. Smith, "Neural networks for combinatorial optimization: A review of more than a decade of research," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 15–34, 1999.

[13] G. Wilson and G. Pawley, "On the stability of the travelling salesman problem algorithm of hopfield and tank," *Biological Cybernetics*, vol. 58, no. 1, pp. 63–70, 1988.

[14] K. Tan, H. Tang, and S. Ge, "On parameter settings of hopfield networks applied to traveling salesman problems," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 5, pp. 994–1002, 2005.

[15] M. N. Syed and P. M. Pardalos, "Neural network models in combinatorial optimization," in *Handbook of Combinatorial Optimization*. Springer, 2013, pp. 2028–2087.

[16] U.-P. Wen, K.-M. Lan, and H.-S. Shih, "A review of hopfield neural networks for solving mathematical programming problems," *European Journal of Operational Research*, vol. 198, no. 3, pp. 675–687, 2009.

[17] J.-Y. Potvin, "State-of-the-art surveythe traveling salesman problem: A neural network perspective," *ORSA Journal on Computing*, vol. 5, no. 4, pp. 328–348, 1993.

[18] R. Durbin and D. Willshaw, "An analogue approach to the travelling salesman problem using an elastic net method," *Nature*, vol. 326, no. 6114, pp. 689–691, 1987.

[19] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[20] T. Kohonen, "Clustering, taxonomy, and topological maps of patterns," in *Proceedings of the sixth international conference on pattern recognition*. Washington, DC: IEEE Computer Soc. Press, 1982, pp. 114–128.

[21] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Springer-Verlag, 2001.

[22] J. Fort, "Solving a combinatorial problem via self-organizing process: an application of the kohonen algorithm to the traveling salesman problem," *Biological Cybernetics*, vol. 59, no. 1, pp. 33–40, 1988.

[23] B. Angeniol, G. de La Croix Vaubois, and J.-Y. Le Texier, "Self-organizing feature maps and the travelling salesman problem," *Neural Networks*, vol. 1, no. 4, pp. 289–293, 1988.

[24] H. Ritter and K. Schulten, "Kohonen's self-organizing maps: Exploring their computational capabilities," in *Neural Networks, 1988., IEEE International Conference on*. IEEE, 1988, pp. 109–116.

[25] E. Cochrane and J. Beasley, "The co-adaptive neural network approach to the euclidean travelling salesman problem," *Neural Networks*, vol. 16, no. 10, pp. 1499–1525, 2003.

[26] A. Modares, S. Somhom, and T. Enkawa, "A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems," *International Transactions in Operational Research*, vol. 6, no. 6, pp. 591–606, 1999.

[27] E. Wacholder, J. Han, and R. Mann, "A neural network algorithm for the multiple traveling salesmen problem," *Biological Cybernetics*, vol. 61, no. 1, pp. 11–19, 1989.

[28] C.-Y. Hsu, M.-H. Tsai, and W.-M. Chen, "A study of feature-mapped approach to the multiple travelling salesmen problem," in *Circuits and Systems, 1991., IEEE International Sympoisum on*. IEEE, 1991, pp. 1589–1592.

[29] M. Goldstein, "Self-organizing feature maps for the multiple traveling salesmen problem," in *Proceedings of the IEEE International Conference on Neural Networks*. IEEE, 1990, pp. 258–261.

[30] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[31] A. I. Vakhutinsky and B. Golden, "Solving vehicle routing problems using elastic nets," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 7. IEEE, 1994, pp. 4535–4540.

[32] A. Torki, S. Somhon, and T. Enkawa, "A competitive neural network algorithm for solving vehicle routing problem," *Computers & industrial engineering*, vol. 33, no. 3, pp. 473–476, 1997.

[33] H. E. Ghaziri, "Solving routing problems by a self-organizing map," in *Artificial Neural Networks, 1991, International Conference on Artificial Neural Networks*. ICANN, 1991, pp. 829–834.

[34] N. Christofides and S. Eilon, "An algorithm for the vehicle-dispatching problem," *OR*, pp. 309–318, 1969.

[35] H. Ghaziri, "Supervision in the self-organizing feature map: Application to the vehicle routing problem," in *Meta-Heuristics*. Springer, 1996, pp. 651–660.

[36] H. Ghaziri and I. H. Osman, "Self-organizing feature maps for the vehicle routing problem with backhauls," *Journal of Scheduling*, vol. 9, no. 2, pp. 97–114, 2006.

[37] Y. Matsuyama, "Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems," in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. 1. IEEE, 1991, pp. 385–390.

[38] M. Padberg and G. Rinaldi, "Optimization of a 532-city symmetric traveling salesman problem by branch and cut," *Operations Research Letters*, vol. 6, no. 1, pp. 1–7, 1987.

[39] S. Eilon, C. Watson-Gandy, and N. Christofides, *Distribution management*. Griffin London, 1971.

[40] M. Schwardt and J. Dethloff, "Solving a continuous location-routing problem by use of a self-organizing map," *International Journal of Physical Distribution & Logistics Management*, vol. 35, no. 6, pp. 390–408, 2005.

[41] J.-C. Créput and A. Koukam, "The memetic self-organizing map approach to the vehicle routing problem," *Soft Computing*, vol. 12, no. 11, pp. 1125–1141, 2008.

[42] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results," in *Fleet management and logistics*. Springer, 1998, pp. 33–56.

[43] J.-F. Cordeau, M. Gendreau, A. Hertz, G. Laporte, and J.-S. Sormany, "New heuristics for the vehicle routing problem," in *Logistics Systems: Design and Optimization*, A. Langevin and D. Riopel, Eds. Springer US, 2005, pp. 279–297. [Online]. Available: http://dx.doi.org/10.1007/0-387-24977-X_9

[44] D. Tuzun, M. A. Magent, and L. I. Burke, "Selection of vehicle routing heuristic using neural networks," *International Transactions in Operational Research*, vol. 4, no. 3, pp. 211–221, 1997.

[45] J.-Y. Potvin and C. Robillard, "Clustering for vehicle routing with a competitive neural network," *Neurocomputing*, vol. 8, no. 2, pp. 125–139, 1995.

CHAPTER 4

An Updated SOM Approach to the CVRP

4.1 Introduction

Since the introduction of the Hopfield Neural Network (HNN) for solving the Traveling Salesman Problem (TSP) three decades ago [1, 2, 3], Artificial Neural Networks (ANN) have been applied to nearly every type of combinatorial optimization problem (COP) [4]. Following the introduction of the HNN, two additional ANN approaches for solving the TSP were proposed: the Elastic Net (EN) method [5], and the Self Organizing Map (SOM) [6, 7, 8]. The EN and SOM both offer a geometric approach to solving the TSP. Subsequent research on the use of neural networks for solving COPs can be related to either the HNN approach or the SOM approach [4], where the EN is grouped with the SOM as a result of their similarities. The vast majority of research into the use of ANNs for combinatorial optimization has focused on the HNN [9] since it is easily generalized to a wide range of COPs. The geometric nature of the SOM limits its application across all COPs [10]. For COPs that are geometric in nature, however, such as the Euclidean TSP and Vehicle Routing Problem (VRP); the SOM approach appears to surpass the performance of the HNN [11, 12].

This research is concerned with the use of a SOM to solve the Capacitated Vehicle Routing Problem (CVRP). The CVRP is the most basic and widely studied VRP [13]. It is an NP-hard combinatorial optimization problem which is closely related to the TSP, yet it is significantly more difficult to solve [14]. The original SOM approaches to solving a TSP [5, 6, 7, 8] have been extended to the VRP [15, 16, 17, 18, 19, 20, 21, 22].

Questions regarding the effectiveness of ANNs for combinatorial optimization problems date back to the Wilson and Pawley [23] critique of the HNN for the TSP. Despite continued research with competitive results, the application of ANNs to COPs was never able to overcome the early criticism [4]. The limited amount of ANN research into the VRP was described in 2007 by Laporte [14], as 'rather unsuccessful and this line of research seems to have been abandoned.' This criticism is likely rooted in two issues with the existing literature. First, not all research into the use of ANNs for the TSP and VRP have made use of publicly available benchmark data sets [12], therefore it is difficult to draw solid conclusions about the effectiveness of these ANN approaches. Second, the results of the application of ANNs for the TSP and VRP do not appear to be competitive with the best known Operations Research (OR) heuristics [24, 12, 25]. Burke

points out however [24], that the SOM approach is a constructive heuristic, and should only be compared with similar approaches to draw conclusions about its performance. Contradictory to Laporte's description, other researchers encourage a continued focus into the use of ANNs for combinatorial optimization problems, specifically the use of the SOM for the TSP and VRP [12, 25] for the following reasons:

- Over time, the incremental improvements to the SOM might lead to more competitive approaches to the VRP.
- The SOM algorithm is easily parallelized, and might become competitive as massively parallel computers become more widely available.
- Insights from the application of the SOM to the VRP might result in improvements in heuristics for other combinatorial optimization problems.
- Unlike the HNN, the SOM can be applied to very large problem instances.

The purpose of this chapter is two-fold. First, this chapter will present an updated SOM algorithm for solving the CVRP. The proposed algorithm will improve upon the best known SOM results for solving the CVRP, and the proposed algorithm will incorporate fuzzy logic for automatic parameter control. The second goal is to provide a robust, side by side comparison of the proposed SOM algorithm to two other constructive heuristics for the CVRP: the Sweep algorithm [26] and the Clarke and Wright algorithm [27]. This comparison will help provide clear insight into the effectiveness of the SOM approach to solving the CVRP.

4.2 Background

4.2.1 Vehicle Routing Problem

The vehicle routing problem (VRP) is a well-known combinatorial optimization problem concerned with finding the optimal routing of goods between a central depot and a set of customers [13]. The CVRP introduces a restriction on the vehicle capacities. This paper is concerned with the symmetric CVRP, which is described as follows. Let $G = (V, A)$ be an undirected, complete graph, where $V = \{0, \ldots, n\}$ is the set of nodes, and A is the set of arcs which connect the nodes. A cost, c_{ij}, is associated with each arc, $(i, j) \in A$, where $i \neq j$. In this research the cost associated with each arc is the Euclidean distance between the nodes i and j, and because the problem is symmetric, $c_{ij} = c_{ji}$. Node 0 corresponds to the depot, whereas the remaining nodes, $V' = V \setminus \{0\}$, correspond to the customers. Each customer $i \in V'$ has a deterministic, non-

negative demand, d_i, for goods. The goods are transported between the depot and the customers by one of the vehicles in the known, homogeneous set of K vehicles. Each vehicle has a maximum capacity, Q, that is identical across the set of vehicles, and $d_i \leq Q$ for each $i = 1, \ldots, n$. The objective of the CVRP is to find K vehicle routes that start and end at the depot, with the minimum cost, where the demand of each customer is met, and the capacity of each vehicle is not violated.

The TSP is a special case of the CVRP, where the capacity of a vehicle is unbounded, $Q = \infty$, and there is only one vehicle in the fleet, $K = 1$. First, the general application of the SOM to the TSP will be presented, followed by the application of the SOM to the CVRP.

4.2.2 Self Organizing Map

A general description of the SOM approach to the TSP is based upon the 'elastic band' idea that originates from the EN [5]. The neurons in the SOM are ordered along a circular band, and as the network evolves, the band of neurons is stretched toward the cities until each city is eventually associated with one neuron on the band. This association between the neurons and cities corresponds to a solution. The extension of the SOM algorithm from the TSP to the VRP requires the modification of the learning process. For the VRP, the rules which govern how the nodes are updated must now incorporate some type of penalties which will account for the capacity constraints in the VRP.

SOM for the TSP The SOM is applied to an N city TSP using a two layer network with M output nodes where $M \geq N$. The input node, X_i, represents the coordinates of a city x_{i1}, x_{i2}, whereas the coordinates of the output nodes, Y_j, are represented by the connection weights between the layers y_{j1}, y_{j2}. The input and output layers are fully connected, as in Figure 11, and the weights between the layers are initialized to small values such that the neurons form a small circular ring near the center of the cities.

Each city i, where $i = 1, \ldots, N$, is iteratively input to the network, and the output nodes, Y_j, compete based on Euclidean distance to be declared the winner. If X_i represents the coordinates of the i^{th} input city, and Y_j represents the weights of the j^{th} node, the winning node, J, is found to be the node with the smallest Euclidean distance to the input city, according to the following equation.

$$J = \text{ArgMin}_j\{\|X_i - Y_j\|\} \tag{29}$$

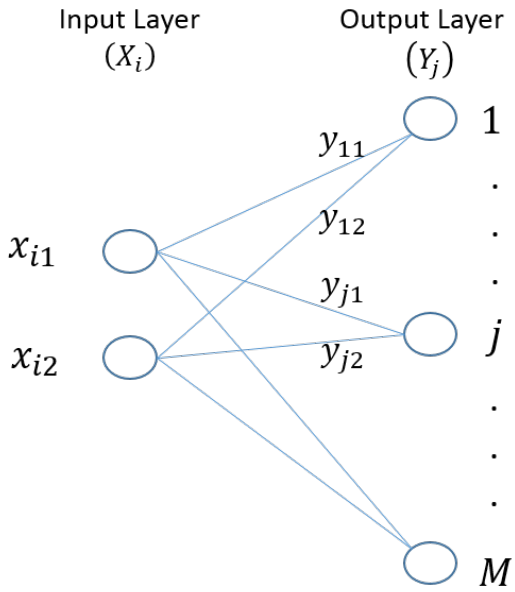


Figure 11: A Two Layer SOM Network for TSP

After the winning node J is found for a city, the position (weights) of J and its neighbors are updated as follows:

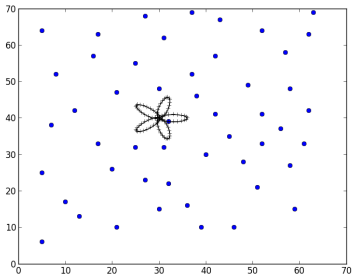
$$Y_j^{new} = Y_j^{old} + \mu F(d_j)(X_i - Y_j^{old}), j = 1, \ldots, M \tag{30}$$

In ((30)), μ is the learning rate and d_j is the number of nodes between node j and the winning node J. This distance is found as follows:

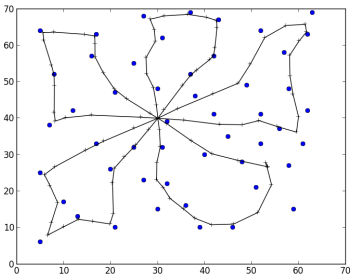
$$d_j = min[|j - J|, M - |j - J|] \tag{31}$$

The function $F(\cdot)$ is defined in such a way that as the network evolves, the influence of the winning node on its neighbors decreases. The algorithm continues to loop through the set of cities, presenting one city at a time to the network, causing the weights of the winning node and its neighbors to be updated. The algorithm terminates when there is one node sufficiently close to each city, or when a max number of iterations has been reached.

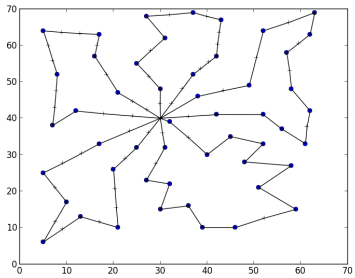
SOM for the CVRP To extend the SOM application from the TSP to the CVRP, two factors must be considered: the number of vehicles in the problem as well as the vehicle capacity constraint. Applying the SOM to the CVRP, there are K rings of neurons (one ring for each of the K vehicles/routes), where each ring consists of M neurons. The output neurons must now be indexed by both ring (route) and position within the route, Y_j^k.



(a) Initial SOM



(b) after 60 iterations



(c) Final solution, 178 iterations

Figure 12: Evolution of SOM Network for CVRP with N=50 and K=5

To account for the capacity constraint of the CVRP, two approaches have been followed in the literature. Both of these approaches incorporate a mechanism to penalize routes that are over capacity when choosing the winning node for an input city X_i. One of the earliest applications of the SOM to the CVRP uses a probabilistic approach to choosing the winning node J [16]. In this algorithm, each time an input city, X_i, is presented to the network, the closest node (measured by Euclidean distance) on each route is identified. From these K nodes, the winning node is chosen in such a way that the probability of the winning node being chosen from an overloaded route tends toward zero as the network evolves.

The second approach incorporates a bias term to penalize overloaded routes during the competition phase of the algorithm. The earliest variation of this approach multiplies the Euclidean distance by a 'handicap' factor as the nodes compete using (29) [15]. With this method, a larger 'handicap' indicates that the current route is near or over the capacity constraint. In subsequent SOM approaches to the VRP, a bias term is added to the Euclidean distance in (29) as the nodes compete [19, 20, 21]. With this approach, a winning node has the smallest combination of Euclidean distance and bias term. Amongst the published results of the application of the SOM to the CVRP that utilize benchmark problem sets, it is clear that results from the bias term

approach in [19, 20, 21] surpass the results from previous SOM approaches [16, 17, 18].

The SOM approach to the CVRP is usually initialized with K small petaloids that start and end at the depot [19, 20, 21]. The network evolves as a winning node, J, is identified for each input city, X_i, and the positions of the winning node and its neighbors are updated. Figure 12 graphically illustrates how a SOM network evolves. A solution is found when each city has a node within a specified distance, or when a maximum number of iterations has occurred.

There are numerous variations of the SOM approach to the TSP and VRP, and a complete literature survey of these SOM algorithms can be found in [12, 25]. Cochrane and Beasley summarize four general areas where the SOM approaches differ from one another:

- number of output neurons, Y_j, used in the network
- order in which the cities are presented to the network
- equation for finding the 'winning' neuron J
- the number of neighbors updated, as well as the equation for updating neurons

Shortcomings of SOM for the CVRP The results of the bias term approaches of applying the SOM to the CVRP of Torki, et al [19] and Modares, et al [20], appear to be promising. Although these approaches differ slightly in their bias term calculations, their results are identical, and all solutions are with 7% of the best known solution at the time of publication. In an attempt to replicate these results, however, it became clear that details regarding parameter settings and the bias term updates were critical to the quality of the published results. Despite using various parameter settings, and multiple methods for updating the bias term, the results could not be replicated. Many of the results were either infeasible, or feasible but of poor quality. This suggested the criticality of the parameter settings in order to strike a balance between the competing demands of finding a short route while not violating the capacity constraint.

The results of Schwardt et al [21] also seemed promising, but the authors indicate that different parameter settings were used for the various test problems, and details of these parameter settings were not provided. Schwardt does, however, address the fact that the SOM can result in infeasible solutions, which is why the parameter settings are adjusted for each problem. Schwardt also introduced additional mechanisms into the SOM in order to achieve feasible solutions more often.

Reviewing these previous results, it is clear that the parameter settings are critical in the solution quality for the bias term approach of using the SOM for the CVRP. Success of this approach can only be found if there is an appropriate balance between the competing demands of cost and feasibility for a solution. The shortcomings of the current literature for the SOM application to the CVRP are that the few published results using benchmark data sets do not offer enough details and insight into finding the appropriate parameter settings for the algorithm.

4.3 Updated SOM for CVRP

This section proposes an update to the bias term approach to the SOM for application to the CVRP. The proposed algorithm improves upon these previous results by strengthening the bias term as well as introducing a new *restricted* mechanism to help increase the feasibility of solutions. When developing the proposed algorithm, careful consideration was given to the four general areas where SOM approaches differ as described at the end of the previous section. The parameter settings for this updated approach are explored.

4.3.1 Proposed Algorithm

For a CVRP problem with N cities and K vehicles, where each city has a demand q, and each vehicle has a capacity Q, the proposed algorithm is initialized with M neurons divided amongst K rings. These K rings of neurons are initialized into small petaloids, where each is centered at a random point in the plane that is bound by the maximum and minimum X and Y coordinates amongst all of the N cities. Randomly initializing the location of the rings, instead of centering them at the depot, injects more stochasticity into the network and can lead to a more robust search of the solution space.

After the network is initialized, it moves into the competition phase. During this phase the nodes repeatedly compete to be declared 'winner' for each city including the depot. One epoch consists of all cities and depot being presented to the network once. At the beginning of each epoch, the cities are randomly ordered and presented to the network one at a time. Randomly ordering the cities is another means to increase the robustness of the algorithm [20]. When a city is presented, one winner is declared amongst all of the nodes, whereas when the depot is presented, a winner is declared on each route. An *inhibit* term is used to ensure that a node may only be declared a winner once per epoch, as described in [20]. This helps force the separation of nodes on each ring. The competition rules are discussed next.

Accounting for the capacity constraint is the most critical factor to ensure the feasibility

of solutions when extending the SOM from the TSP to the CVRP. The proposed algorithm incorporates a bias term in the competition rule in order to account for the capacity constraint. When the nodes compete to be declared the 'winner' for a city, X_i, the competition is based on a combination of a Euclidean distance and the bias term as follows:

$$J = \text{ArgMin}_j\{\|X_i - Y_j\|\} + \nu B_k \tag{32}$$

where

$$\nu = \frac{\nu}{1 + \delta\nu} \tag{33}$$

and

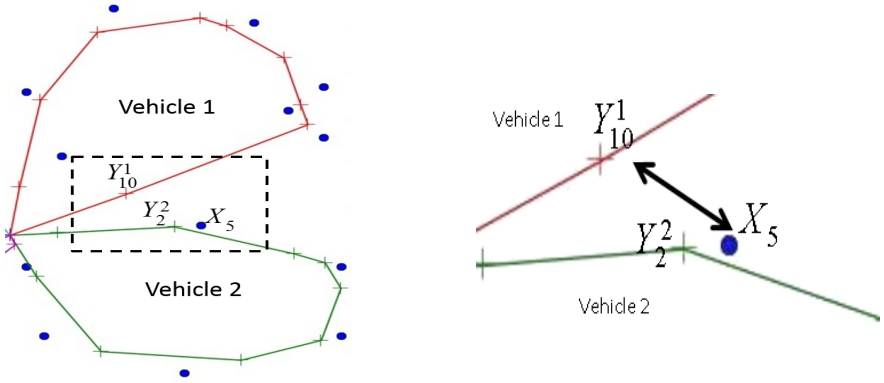
$$B_k = \left(1 + \frac{\sum_l q_l^k}{Q}\right)^2 \tag{34}$$

Where ν is initialized to a value that is proportional to the average distance between each node and the depot; and δ controls how quickly ν decreases and is initialized to a small value.

The Euclidean distance in Equation (32) is used to help ensure a short route, whereas the second term, the bias term, is used to help find a feasible route. The bias term is calculated with (34), and its purpose is to penalize overloaded routes. The proposed bias term differs from the existing literature as it provides a larger difference in the penalties for routes that are near or over capacity. The term $\frac{\sum_l q_l^k}{Q}$ in Equation (34) provides a ratio of the sum of all demand l assigned to route k, to the available capacity on the route. The proposed bias term provides a larger penalty for routes that are near or over capacity, compared with previous research. The bias term is updated at the end of each epoch, and when updating the bias term, the demand q_l for each city, l, is assigned to the route that corresponds to the closest node at the end of the epoch. Figure 13 shows the competition between nodes Y_{10}^1 and Y_2^2 for city X_5. Assuming that vehicle 2 is over capacity and has a high bias term, node Y_{10}^1 is declared the winner despite having a larger Euclidean distance from city X_5.

As the network evolves, the Euclidean distance between nodes and cities decreases, therefore it is critical that the weight of the bias term also decreases. If the weight is not decreased accordingly, the bias term will dominate Equation (32) resulting in solutions that are feasible, but of poor quality. The parameter ν serves to decrease the bias term weight, and ν is updated according to Equation (33) after each epoch.

After the winning node J is found for a city, X_i, the position of the winning node and its



(a) Nodes Y_{10}^1 and Y_2^2 compete for city X_5.

(b) Node Y_{10}^1 is declared winner.

Figure 13: Example of node competition with bias term; assume that vehicle 2 is overloaded and vehicle 1 is not.

neighbors are updated as follows:

$$Y_j{}^{k,new} = Y_j{}^{k,old} + \Delta Y_j{}^k \tag{35}$$

$$\Delta Y_j{}^k = \mu F(d_j)(X_i - Y_j{}^k) + \lambda(Y_{j+1}^k - 2Y_j{}^k + Y_{j-1}{}^k),$$
$$\{\forall j | F(d_j) \geq 0\} \tag{36}$$

The magnitude and direction that a node is updated is determined by Equation (36). The first term in Equation (36) moves the node toward the current city, X_i. The magnitude of this movement is determined by the learning rate, μ; the neighborhood function, $F(\cdot)$; and the current distance between the node and city, $(X_i - Y_j{}^k)$. The second term in Equation (36) originated from the Elastic Net algorithm. This term helps strengthen the force between neighboring nodes, and helps reduce the possibility of a route crossing over itself. The weight of this second term, λ, decreases at the end of each epoch in order to allow the network to converge.

As previously stated, the function $F(\cdot)$ is defined as a decreasing function, so that as the network evolves the winning node has less influence on its neighboring nodes.

$$F(G, d_j) = \begin{cases} \exp^{((-d_j{}^2)/(G^2))}, & d \leq H \\ 0, & otherwise \end{cases} \tag{37}$$

Where $H = .2 * M$, and G is the gain parameter that decreases as the network evolves.

At the end of each epoch, an intermediate solution is found by determining the closest node to each city. Each node cannot be assigned to more than one city. This intermediate solution is

used to update the bias term B_k according to Equation (34). The parameters ν, λ, and G are also decreased at this time.

Although the proposed algorithm provides a strong bias term for penalizing routes that are near or over capacity; it is possible for the solution to be infeasible. In order to overcome this, a *restricted* term is introduced to encourage the network to evolve toward a feasible solution. When the intermediate solution is found at the end of each epoch, any route that is over capacity Q becomes *restricted*. During each epoch, a *restricted* route cannot be declared a 'winner' for cities beyond its capacity Q. This *restricted* mechanism forces the network to explore more feasible solutions, and evolve the network accordingly. Once *restricted*, a route can remain *restricted* for a certain number of iterations, or until the route is no longer overloaded in the intermediate solution. An outline of the algorithm follows.

Step 1 Initialization:

Initialize parameters μ, δ, λ, G, and α; set $\nu = c *$ (avg distance of all cities to the depot). Let N = the number of cities plus the depot, and M = the number of nodes; and let *closeEnough* be the maximum distance required between a city and node for termination. Construct the nodes into K small petaloids placed randomly in the plane.

Step 2 Randomization:

Randomize the order of cities. Let i be the index of each city. Set $i = 1$. Set $inhibit = False$ for each node; *restricted* = *False* and *capacityCount* = 0 for each route.

Step 3 Competition:

Present city X_i to the network. Using (32), all qualified[1]nodes compete to determine the winning node J. If the depot is presented, one winning node is found on each of the K routes. Set *Inhibit* = *True* for the winning node(s), and update the $capacityCount_k$ for the route that corresponds to node J.

Step 4 Adaptation:

Move node J and its neighbors using (35).

Step 5 Increment:

Set $i = i + 1$. If $i \leq N$, go to Step 2; otherwise, go to Step 6.

Step 6 Test Convergence:

If each city has a node within a *closeEnough* distance, STOP. Otherwise, set $\lambda = \lambda(1-\alpha)$;

Table 1: Parameter settings tested

Settings		
run	δ	c
1	.01	1
2	.01	10
3	.01	100
4	.005	1
5	.005	10
6	.005	100

Table 2: Best solutions found in 100 replications of the proposed algorithm

Name	Size	Best Known (# vehicles)	Torki, et al [a]	Schwardt, et al	Proposed	
					Result	PDB [b]
eil22	21	375.3 (4)	390 (4)	375.3 (4)	375.3 (4)	0
eil23	22	569 (3)	585 (3)	570.2 (3)	569.7 (3)	0.13
eil30	29	534 (3)	557 (3)	537.1 (3)	539.9 (3)	1.10
eil33	32	835 (4)	889 (4)	915.4 (4)	846 (4)	1.32
C1	50	524.6 (5)	537 (5)	526.3 (5)	524.6 (5)	0
C2	76	835.3 (10)	876.3 (10)	902.7 (10)	860.3 (10)	2.97
C3	100	826.1 (8)	863 (8)	838.9 (8)	829.6 (8)	0.42
C11	120	1042.1 (7)	1066 (7)	1111.4 (7)	1049.7 (7)	0.73
C4	150	1028.4 (12)	1082 (12)	1074.4 (12)	1071.1 (12)	4.15
C5	199	1291.3 (16)	1386 (17)	1367.5 (17)	1365.3 (17)	5.73

[a] The results from Torki, et al, and Modares, et al are identical, therefore, they are combined into one column.
[b] percent deviation from the best known solution

$G = G(1-\alpha)$; update ν according to (33); update B_k according to (34); update *restricted* routes; and go to Step 2.

4.3.2 Parameter Sensitivity

Computational experiments were conducted with the updated SOM algorithm on ten standard CVRP instances [28, 29] from the literature. The same problem instances were used as Torki et al, and Modares et al, in order to make a clear comparison between the algorithms. Additionally, three parameters were varied in this experiment: the number of epochs that a route

[1] A node is qualified to compete if it is not inhibited, and if it is *restricted*, it cannot have exceeded the capacity Q for 'winning' nodes in one epoch.

is *restricted*, c, and δ. Two settings were used for the number of epochs that a route remains *restricted*. The first setting keeps the route *restricted* until the route is no longer overloaded in the intermediate solution; the second setting keeps a route *restricted* for ten epochs from when it is first categorized as *restricted*. Within each of these two settings, the parameters c and δ varied according to Table 1.

Each of the parameters that were chosen to vary have a significant influence on the ability of the algorithm to find a feasible solution. The parameter c is used to initialize the weight of the bias term at the beginning of the algorithm. The algorithm updates and uses δ at the end of each epoch to decrease the weight of the bias term. The remaining parameters were initialized as follows: $\mu = .6$; $\lambda = .3$; $G = .4 * M$; $\alpha = .03$; and $M = 4 * N$. These initial parameter settings are largely based upon the works of Torki, et al, and Modares, et al [19, 20].

For each parameter setting, 100 replications of the SOM were conducted, and the best solution amongst all parameter settings was recorded in Table 2, along with the percent deviation from the best known solution (PDB). The results of the proposed algorithm are compared to the best known results from the CVRP SOM literature [19, 20, 21].

One parameter setting did not lead to all of the best solutions. Each problem appeared to be uniquely sensitive to the parameter settings, and a few parameter and problem pairings led to no feasible solutions. This observation reinforces the importance of the parameters in maintaining a balance between the competing demands of finding a short solution and the feasibility of the solution. This finding is in line with the results of Scwardt and Dethloff [21] who reported having to adjust the parameter settings for each new problem.

Given the sensitivity of the algorithm's success to the parameters c and δ, it is desirable to control the algorithm's parameters automatically, instead of having to tune the parameters for each new problem. The intention is to identify those parameters that are most influential in keeping the balance between a short, yet feasible solution. Once these parameters are identified, all problems will start out with the same initial settings. As the network evolves, the automatic parameter control will tune these parameters in a manner that results in good quality, feasible solutions across all problems.

4.4 Fuzzy Logic Control

Background A fuzzy logic controller (FLC) can be described as a model which can produce a mapping between an input and output where the mapping is defined using linguistic variables [30].

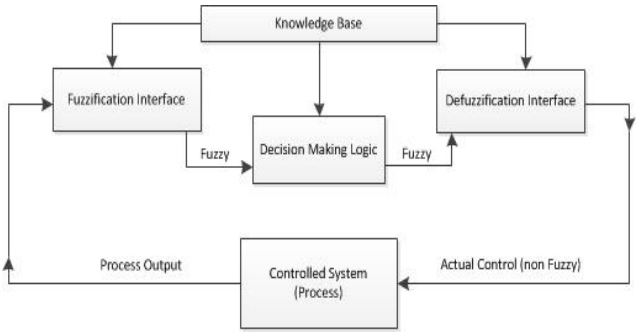


Figure 14: General structure of fuzzy logic controller [32]

The linguistic variables are a means to incorporate expert, human knowledge into an automatic control strategy. Fuzzy logic has been used extensively for controlling parameters in the Genetic Algorithm [30, 31] and a similar approach can be applied to the current SOM algorithm for the CVRP.

The generic form of an FLC is shown in Figure 14. In the current application, the SOM algorithm is the controlled system. Performance measures (Process Output) will be extracted from the SOM as it is evolving, these measures will be fuzzified and then passed through the fuzzy decision making logic. The results of the fuzzy decision making logic will then be defuzzified and the SOM parameters will be updated accordingly.

The purpose of incorporating an FLC into the SOM is to provide parameter control based on how the algorithm is performing. If the algorithm is converging toward a feasible solution, the appropriate parameters should be adjusted to relax the weight of the bias term. On the other hand, if the SOM is converging toward an infeasible solution, these parameters must be adjusted to increase the weight of the bias term with intent of encouraging the algorithm to find a feasible solution. The first step in designing the FLC is to determine what metrics will be used to assess the algorithm's performance.

Input In order to measure how well the SOM is converging toward a feasible solution, two metrics are chosen. Each metric is evaluated at the end of an epoch, and are based on the current, intermediate solution of the SOM. The first metric in Equation (38) is the ratio of the maximum demand assigned to a route, to the truck capacity Q. If $x_1 \leq 1$, this indicates that all routes are feasible. If $x_1 \geq 1$, however, this indicates that at least one route is infeasible. The second metric in Equation (39) is a ratio of the current and previous x_1 values. This ratio provides an indication as to whether the feasibility of the solution is getting better, worse or

staying the same.

$$x_1 = \frac{\text{Max}_k\{capacityCount_k\}}{Q} \tag{38}$$

$$x_2 = \frac{x_1^{curr}}{x_1^{last}} \tag{39}$$

Output The next step is to identify which parameters will be controlled by the FLC. In the experiment described in Section 4.3.2, only the parameters c, δ, and the length of *restricted* were adjusted, but the algorithm consists of many other parameters that may be controlled. Table 3 provides a list of all the parameters in the SOM algorithm with a brief description, and if/how they are currently updated.

Two parameters were chosen to be updated by the FLC: and δ and α. In the current version of the algorithm, the value of δ is initialized at the beginning of the algorithm, and held constant. δ is used to decrement the weighting factor ν at the end of the each epoch. The weight of the bias term must decrease as the algorithm evolves in order to ensure that the algorithm converges, however, how quickly the weight decreases has a significant impact on the quality of the solution. If the weight decreases too quickly, then there is a greater chance that the capacity constraint will be violated and the solution will be infeasible. However, if the weight decreases too slowly, this is likely to result in a solution that is feasible, but of poor quality with regard to route length. By allowing the FLC to control the value of δ, δ can be updated based on how well the algorithm is converging toward a feasible solution. If the intermediate solution is infeasible, the value of δ should be reduced, thus making the weighting factor ν decrease more slowly. If the algorithm is moving toward a feasible solution, however, δ can be increased to allow the algorithm to converge to a solution more quickly. Increasing and decreasing the value of δ will not change the fact that ν is a monotonically decreasing function, it will only change how quickly ν decreases.

The second parameter chosen to be controlled by the FLC is α. The role of α is to reduce the values of both G and λ, which are parameters related to the influence of neighboring nodes on each other. When a winning node is chosen, this winning node and its neighbors' positions are updated. The number of neighboring nodes which are updated decreases as the SOM evolves and is determined by Equation (37). G is an input to Equation (37), and as G decreases, so does the value of the output of Equation (37). The parameter λ is used in Equation (36). When a node's position is updated, the movement is a combination of the node's distance to the current

city, as well as the position of it's two immediate neighboring nodes. λ serves as the weight of the neighboring nodes positions for this update. When the neighboring nodes have a stronger influence in a node's positional update, this not only helps to ensure a shorter route, it also provides a more robust search of the solution space.

Fuzzifier/Inference/Defuzzifier This experiment used triangular membership functions, the *AND* operator for fuzzification; the *OR* operator and the *clipping* method to determine the consequence; and the weighted average defuzzification strategy. The triangular membership functions can be found in Figure (15). With two inputs, x_1 and x_2, as well as two outputs, y_1 (α) and y_2 (β), a total of 18 fuzzy rules were used in the FLC; 9 for each output. These fuzzy rules can be found in Table (4). [p]

4.4.1 SOM with Fuzzy Logic Control

The FLC is incorporated into Step 4.3.1 of the updated algorithm outlined in Section 4.3.1. The FLC is used to update the parameters α and δ every 10 epochs. In order to incorporate this update into the previously described, updated algorithm, Step 4.3.1 is be revised as follows:

Step 6 Test Convergence: Test Convergence: If each city has a node within a *closeEnough* distance, STOP. Otherwise, if the epoch number mod 10 is equal to 0, update α and δ with the output of the FLC: $\alpha = y_1 * \alpha$; $\delta = y_2 * \delta$. Set $\lambda = \lambda(1 - \alpha)$; $G = G(1 - \alpha)$; update ν according to (33); update B_k according to (34); update *restricted* routes; and go to Step 2

The parameters λ, G and ν are updated at the end of each epoch regardless of whether the FLC is used. Using the FLC to update α and λ more frequently than 10 epochs resulted in the algorithm taking longer for convergence, and it did improve the solution quality. By allowing 10 epochs to pass, the input into the FLC, x_1 and x_2, seem to provide a more stable indicator as to how the SOM is evolving as compared to a higher frequency.

4.5 Experimental Results

In order to evaluate the effectiveness of fuzzy logic for parameter control in the updated SOM, the same ten standard CVRP instances [28, 29] were used as in the previous section. The Fuzzy SOM is initialized with the same parameter values as outlined in the previous section; however, instead of varying the parameters c, δ, and the length that a route is *restricted*, these values are initialized as follow: $c = 1, \delta = .01$ and a route remains *restricted* until it is not overloaded in the intermediate solution. For each problem, the best solution out of 100 replications of the Fuzzy

Table 3: Parameter Descriptions

Parameter	Description	Update Frequency	Update Rule
μ	learning rate	N/A	N/A
δ	used to decrement ν	N/A	N/A
λ	weight of neighbor positions in (36)	end of each epoch	$\lambda = \lambda * (1 - \alpha)$
G	gain parameter; input to (37)	end of each epoch	$G = G * (1 - \alpha)$
α	used to decrement λ and G	N/A	N/A
c	coefficient for initial value of ν	N/A	N/A
ν	weight of bias term	end of each epoch	$\nu = \frac{\nu}{1 + \delta\nu}$
closeEnough	minimum distance a node must be to a city in final solution	N/A	N/A
restrictedLength	how long a route is restricted if it is overloaded	N/A	N/A

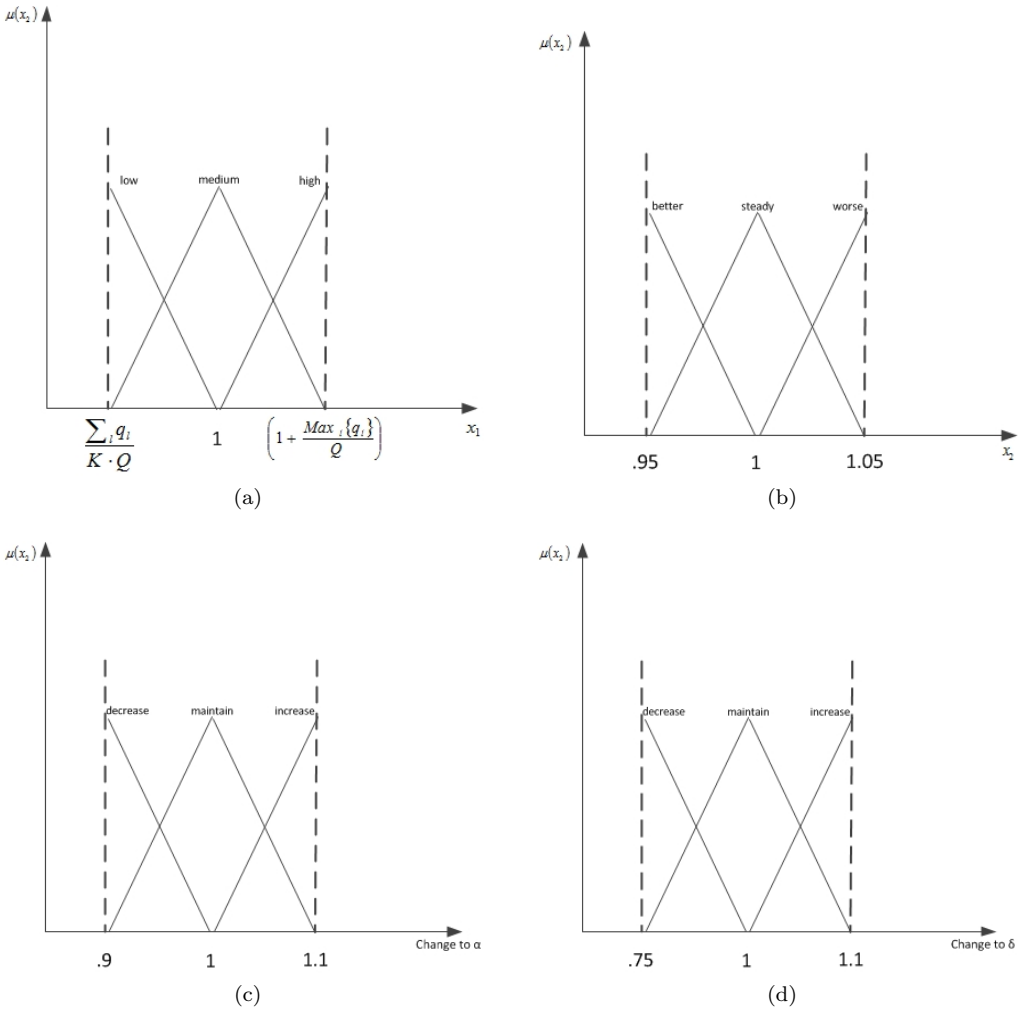


Figure 15: Triangular membership functions used in FLC. The fuzzification process uses 15a and 15b, and the defuzzification process uses 15c and 15d.

Table 4: Fuzzy Rules

Rule	IF		THEN	
	x_1	x_2	change α	change δ
1	low	better	increase	increase
2	low	same	increase	increase
3	low	worse	increase	steady
4	medium	better	same	same
5	medium	same	same	same
6	medium	worse	same	same
7	high	better	same	same
8	high	same	decrease	decrease
9	high	worse	decrease	decrease

Table 5: Number of feasible solutions found out of 100 replications

Problem Name	Updated SOM (no FLC) *restricted* until not overloaded						Fuzzy SOM
	$\delta = .01$	$\delta = .01$	$\delta = .01$	$\delta = .005$	$\delta = .005$	$\delta = .005$	$\delta = .01$
	$c = 1$	$c = 10$	$c = 100$	$c = 1$	$c = 10$	$c = 100$	$c = 1$
eil22	34	37	30	42	43	54	100
eil23	67	66	62	87	78	82	100
eil30	13	23	15	20	23	37	73
eil33	42	43	48	54	53	59	95
C1	11	12	15	15	15	17	54
C2	0	0	1	0	2	1	8
C3	46	47	57	86	91	89	92
C11	0	1	0	8	9	7	32
C4	15	12	14	49	49	51	71
C5	2	9	5	23	19	32	58

SOM are reported. The results of the Fuzzy SOM are compared to the results of the updated SOM with no fuzzy logic. A comparison is made in both the number of feasible solutions as well as the solution quality. Table 5 provides a side by side comparison of the number of feasible solutions found by the updated SOM with fuzzy logic for parameter control, with a subset of the parameters tested for the updated SOM without fuzzy logic.

In order to understand the performance of the Fuzzy SOM as a constructive heuristic, the proposed algorithm is compared to two well-known constructive heuristics: the parallel Clarke and Wright algorithm [27, 13] and the Sweep algorithm [26]. The Sweep Algorithm implementation used in this research follows the algorithm outlined in [33]. Once the customers were separated into routes according to the Sweep algorithm, the routes were then improved using a 2-opt heuristic. This Sweep algorithm was repeated n times, with each city used as the starting vertex once. The best result of these n replications is reported. The results are found in Table 6.

4.5.1 Discussion

The results of the updated SOM in Table 2 surpass previous applications of the SOM to the CVRP. These preliminary results also provide insight into the sensitivity of the SOM algorithm to the parameter settings. Although one set of parameter values did not lead to all of the best solutions, examining the algorithm's performance under different parameter settings provided insight into the delicate balance between the competing demands of finding a short route and

Table 6: Comparison of the best solutions found by each algorithm

Name	Size	Best Known (no. vehicles)	Clarke & Wright	Sweep w/ 2 opt	updated SOM	Fuzzy SOM	
						Result	PDB
eil22	21	375.3 (4)	388.8 (4)	398.3 (4)	375.3 (4)	375.3 (4)	0
eil23	22	569 (3)	621.1 (3)	581.0 (3)	569.7 (3)	569.7 (3)	0.1
eil30	29	534 (3)	534.5 (4)	508.1 (4)	539.9 (3)	539.9 (3)	1.1
eil33	32	835 (4)	843.1 (4)	879.6 (4)	846.0 (4)	845.0 (4)	1.2
C1	50	524.6 (5)	584.6 (6)	531.9 (5)	524.6 (5)	524.6 (5)	0
C2	76	835.3 (10)	900.3 (10)	902.8 (11)	860.3 (10)	862.2 (10)	3.2
C3	100	826.13 (8)	886.8 (8)	865.2 (8)	829.6 (8)	834.1 (8)	1.0
C11	120	1042.1 (7)	1068.1 (7)	1272.5 (7)	1049.7 (7)	1051.8 (7)	0.9
C4	150	1028.4 (12)	1133.4 (12)	1098.6 (12)	1071.1 (12)	1069.1 (12)	4.0
C5	199	1291.3 (16)	1395.7 (17)	1419.64 (17)	1365.3 (17)	1372.3 (17)	5.73

finding a feasible solution.

The purpose of introducing fuzzy logic to control the parameter settings is to alleviate the need to tune the parameters for each new problem. The need for parameter control is a result of the sensitive balance between finding a feasible solution and finding a short route. Comparing the results of the Fuzzy SOM to the updated SOM without fuzzy, it is clear that the FLC for parameter control finds feasible solutions more often, and the quality of these solutions is commensurate with the results of manually tuning the SOM parameters.

Table 5 provides the number of feasible solutions found in 100 replications for a portion of the parameter settings evaluated in Section 4.3.1. In general, as the value of c increased and the value of δ decreased, the algorithm found feasible solutions more often, however, these solutions had longer route lengths. As c increases and δ decreases, the weight of the bias term ν is initialized to a larger value (c is the weighting factor), and as the algorithm evolves, ν decreases more slowly (δ controls how quickly ν decreases). Under these circumstances, there is a greater emphasis on the bias term in Equation (32), which places more emphasis on the algorithm finding a feasible solution versus a short route. With a smaller value of c and a larger value of δ, however, the emphasis is on the first term in Equation (32), which will likely result in more infeasible solutions. This trend is evident in Table 5. When $\delta = .005$, across all values of c, there is an overall increase in the number of feasible solutions found.

The Fuzzy SOM requires no manual parameter tuning, and it is initialized with only one setting of c and δ. The FLC in the Fuzzy SOM is designed to output two values, y_1 and y_2, which are update factors for the terms α and δ. By updating α and δ, the balance between finding a short route and a feasible solution is shifted, if necessary, based on how the algorithm is evolving. It is clear from Table 5 that the Fuzzy SOM is successful in finding feasible solution more often compared to the manually tuned SOM.

Finding more feasible solutions is only beneficial if the solution quality remains competitive. It is clear from Table 6 that the solution quality is nearly unchanged by incorporating fuzzy logic for parameter control. The Fuzzy SOM not only finds feasible solutions more often, but the solution quality is nearly identical to the results of multiple attempts of manual parameter control.

Finally, the proposed Fuzzy SOM algorithm outperforms the parallel Clarke and Wright Algorithm as well as the Sweep Algorithm on nearly all of the 10 benchmark problems. The best known solution for 'eil30' is improved when the number of trucks is increased to 4, and both

the Clarke and Wright as well as the Sweep Algorithm outperform the Fuzzy SOM by finding a better solution with an increased number of vehicles. Other than this problem, the results of the Fuzzy SOM are superior to the Sweep Algorithm on all remaining benchmark problems; and the Fuzzy SOM outperforms the Clarke and Wright Algorithm on 8 out of the 9 remaining problems.

4.5.2 Conclusion and Future Work

Although the use of a SOM for solving a CVRP has not always been highly regarded in the literature, when directly compared to other construction heuristics, the SOMs performance cannot be overlooked. The performance of the proposed Fuzzy SOM is superior to the Sweep and Clarke and Wright algorithm results, and this algorithm improves existing SOM results on nine out of ten problems. This provides strong evidence that continued SOM research for the CVRP is promising and should persist.

This research demonstrates the credibility of the SOM as a constructive heuristic for the CVRP as well as the effectiveness of fuzzy logic for dynamic parameter control. As massively parallel computers continue to become more readily available, one of the most attractive aspects of the SOM algorithm is its inherent potential for parallelization. Further, the simple bias term concept of the proposed algorithm allows for a straightforward extension to other VRP variants. Future research might consider combining the existing SOM algorithm with local improvement heuristics or a hybridization with existing evolutionary techniques.

List of References

[1] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[2] J. J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

[3] J. J. Hopfield and D. W. Tank, "neural computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

[4] K. A. Smith, "Neural networks for combinatorial optimization: A review of more than a decade of research," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 15–34, 1999.

[5] R. Durbin and D. Willshaw, "An analogue approach to the travelling salesman problem using an elastic net method," *Nature*, vol. 326, no. 6114, pp. 689–691, 1987.

[6] J. Fort, "Solving a combinatorial problem via self-organizing process: an application of the kohonen algorithm to the traveling salesman problem," *Biological Cybernetics*, vol. 59, no. 1, pp. 33–40, 1988.

[7] B. Angeniol, G. de La Croix Vaubois, and J.-Y. Le Texier, "Self-organizing feature maps and the travelling salesman problem," *Neural Networks*, vol. 1, no. 4, pp. 289–293, 1988.

[8] H. Ritter and K. Schulten, "Kohonen's self-organizing maps: Exploring their computational capabilities," in *Neural Networks, 1988., IEEE International Conference on*. IEEE, 1988, pp. 109–116.

[9] U.-P. Wen, K.-M. Lan, and H.-S. Shih, "A review of hopfield neural networks for solving mathematical programming problems," *European Journal of Operational Research*, vol. 198, no. 3, pp. 675–687, 2009.

[10] M. N. Syed and P. M. Pardalos, "Neural network models in combinatorial optimization," in *Handbook of Combinatorial Optimization*. Springer, 2013, pp. 2028–2087.

[11] J.-Y. Potvin, "State-of-the-art surveythe traveling salesman problem: A neural network perspective," *ORSA Journal on Computing*, vol. 5, no. 4, pp. 328–348, 1993.

[12] E. Cochrane and J. Beasley, "The co-adaptive neural network approach to the euclidean travelling salesman problem," *Neural Networks*, vol. 16, no. 10, pp. 1499–1525, 2003.

[13] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[14] G. Laporte, "What you should know about the vehicle routing problem," *Naval Research Logistics (NRL)*, vol. 54, no. 8, pp. 811–819, 2007.

[15] Y. Matsuyama, "Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems," in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. 1. IEEE, 1991, pp. 385–390.

[16] H. E. Ghaziri, "Solving routing problems by a self-organizing map," in *Artificial Neural Networks, 1991, International Conference on Artificial Neural Networks*. ICANN, 1991, pp. 829–834.

[17] H. Ghaziri, "Supervision in the self-organizing feature map: Application to the vehicle routing problem," in *Meta-Heuristics*. Springer, 1996, pp. 651–660.

[18] A. I. Vakhutinsky and B. Golden, "Solving vehicle routing problems using elastic nets," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 7. IEEE, 1994, pp. 4535–4540.

[19] A. Torki, S. Somhon, and T. Enkawa, "A competitive neural network algorithm for solving vehicle routing problem," *Computers & industrial engineering*, vol. 33, no. 3, pp. 473–476, 1997.

[20] A. Modares, S. Somhom, and T. Enkawa, "A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems," *International Transactions in Operational Research*, vol. 6, no. 6, pp. 591–606, 1999.

[21] M. Schwardt and J. Dethloff, "Solving a continuous location-routing problem by use of a self-organizing map," *International Journal of Physical Distribution & Logistics Management*, vol. 35, no. 6, pp. 390–408, 2005.

[22] H. Ghaziri and I. H. Osman, "Self-organizing feature maps for the vehicle routing problem with backhauls," *Journal of Scheduling*, vol. 9, no. 2, pp. 97–114, 2006.

[23] G. Wilson and G. Pawley, "On the stability of the travelling salesman problem algorithm of hopfield and tank," *Biological Cybernetics*, vol. 58, no. 1, pp. 63–70, 1988.

[24] L. Burke, "conscientious neural nets for tour construction in the traveling salesman problem: the vigilant net," *Computers & operations research*, vol. 23, no. 2, pp. 121–129, 1996.

[25] J.-C. Créput and A. Koukam, "The memetic self-organizing map approach to the vehicle routing problem," *Soft Computing*, vol. 12, no. 11, pp. 1125–1141, 2008.

[26] B. E. Gillett and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Operations research*, vol. 22, no. 2, pp. 340–349, 1974.

[27] G. u. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations research*, vol. 12, no. 4, pp. 568–581, 1964.

[28] N. Christofides and S. Eilon, "An algorithm for the vehicle-dispatching problem," *OR*, pp. 309–318, 1969.

[29] N. Christofides, A. Mingozzi, and P. Toth, "The vehicle routing problem," in *Combinatorial Optimization*, N. Mingozzi, P. Toth, and C. Sandi, Eds. London: John Wiley and Sons, 1979.

[30] A. Michael and H. Takagi, "Dynamic control of genetic algorithms using fuzzy logic techniques," in *Proceedings of the Fifth International Conference on Genetic Algorithms*. Citeseer, 1993, pp. 76–83.

[31] F. Herrera and M. Lozano, "Adaptation of genetic algorithm parameters based on fuzzy logic controllers," *Genetic Algorithms and Soft Computing*, vol. 8, pp. 95–125, 1996.

[32] C. C. Lee, "Fuzzy logic in control system: Fuzzy logic in controller part i," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 20, no. 2, pp. 404–418, 1990.

[33] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International transactions in operational research*, vol. 7, no. 4-5, pp. 285–300, 2000.

CHAPTER 5

Using a SOM for Algorithm Prediction

5.1 Introduction

As researchers have continued to develop more sophisticated algorithms over the years, it is not always clear how these algorithms perform across a broad range of problems [1]. Combing through the CVRP research, it is clear that there are many good algorithms that are capable of finding good solutions to benchmark problems. Sometimes researchers test their algorithm on a broad range of problem sets, and other times they don't. For an important problem such as the CVRP, for which real-world problems exist across numerous facets of industry, an important question remains: given a real world problem, what is the best performing algorithm? Or, as Smith-Miles and Lopes ask, "Which algorithm in a broad portfolio is likely to be best for a relevant set of problem instances," [1]. This is a difficult question. First, an algorithm's performance can be measured by different factors. The two most common are: solution quality and time. The second reason that this is a difficult question is because every problem instance is unique, and there are numerous characteristics that can vary.

With their introduction of the No Free Lunch Theorem [2], Wolpert and Macready contend that 'the average performance of any pair of algorithms across all possible problems is identical.' This notion that one single algorithm will not outperform all other algorithms, all of the time, is widely recognized by researchers and is directly related to the 1976 work of John Rice, who formalized the algorithm selection problem [3]. Rice's abstract model seeks to identify the best performing algorithm given a collection of problem instances and a set of features describing the problems.

Various research communities have focused on the algorithm selection problem over the years, albeit under various names. Within the machine learning community, an early work related to the algorithm selection problem was presented by Aha [4] in 1992. In this paper, Aha was critical of the practice within the machine learning community of evaluating algorithm performance on a limited number of data sets, and then generalizing these results to other data sets. Aha describes these generalizations as 'suspect', and he proposed a method which "...derive[s] rules of the form 'this algorithm outperforms these other algorithms on these dependent measures for databases with these characteristics' "[4]. It was in this paper [4] that Aha introduced the term 'meta-learning' which is used extensively in the machine learning literature [5]. Despite the

clear relationship between meta-learning in the machine learning field and the algorithm selection problem, the majority of research regarding meta-learning does not mention Rice's work [6].

In addition to the machine learning community, the algorithm selection problem has also been a focus within the artificial intelligence field [7]. Despite the over-arching goal of developing new, better performing algorithms for solving problems, the artificial intelligence community is also keenly aware of the fact that there is not one algorithm that will outperform all other algorithms across all problems [2]. The work by Tsang et al [8], provides an early example of approaching the algorithm selection problem within the artificial intelligence community. The authors study the mapping between problems and algorithm performance for the constraint satisfaction problem, and conclude that because there is not one dominant algorithm across all problems, future research should investigate how an algorithm's performance relates to the problem characteristics [8]. More recent investigations from the artificial intelligence community into the algorithm selection problem include [9, 10, 11, 12].

Machine learning and artificial intelligence are just two broad categories of how the literature related to the algorithm selection problem can be divided. There are actually numerous disciplines involved in this research, either focused on some form of the entire algorithms selection problem, or just one aspect of it. Specifically within the Operations Research field, related to the algorithm selection problem is the investigation of how the landscape of the fitness function of a combinatorial optimization problem can affect algorithm performance [6, 13, 14, 15]. Cheeseman [16] introduced the notion of phase transitions, and he asserts that many $\mathcal{NP}$-hard problems have one parameter around which a transition from easy to hard problems can be defined. Despite the seemingly disjoint research going on across these research communities, Smith-Miles provides insight into how the work amongst these communities can be bridged to make stronger future gains [6]. The major components of the algorithm selection problem that bridge the communities working in this field are as follows [6]:

(i) the availability of a large, diverse set of problem instances;

(ii) a significant number of algorithms that can applied to the problem instances;

(iii) metrics to measure the algorithm performance; and

(iv) suitable features to characterize the problem instances.

The aim of this research is to investigate the VRP in the context of the algorithm selection

problem. The current method for introducing new algorithms relies heavily on the algorithm's performance on benchmark problem sets [17], and this is true for the CVRP. There are a significant number of benchmark problem sets that have been introduced for the CVRP, and as new algorithms are introduced, algorithm performance is often measured by how well they solve these problem sets. However, as algorithm performance has improved dramatically on many of these benchmark problems over the past few decades, a valid question remains: can the algorithm's performance be generalized beyond these benchmark problems, or is the algorithm simply well suited for these particular instances? This question is deeply rooted in the algorithm selection problem. To date, there does not appear to be any investigation into the diversity of the existing benchmark problem sets for the CVRP.

The first part of this work will focus on examining the diversity of the existing benchmark problems for the VRP, which relates back to part (i) of the components of the algorithm selection problem described by Smith-Miles [6]. If the existing benchmark problems are diverse, the NFL theorem indicates that one algorithm's performance should vary across these problems. The diversity of the existing benchmark problems will be examined through exploratory data analysis using an unsupervised SOM. Key problem features taken from the literature will be examined for their influence on algorithm performance, which relates to part (iv) of the algorithm selection problem. The key question motivating this section of the work is whether the existing benchmark problem sets offer a diverse range of problem characteristics.

The second part of this research will train an SOM for algorithm prediction. The problem instances used for training and testing the prediction model will include the benchmark problem instances explored during the first part of this work, as well as additional problems derived from these benchmark problems. VRP researchers have previously demonstrated the effectiveness of a feed-forward neural network for choosing the best VRP algorithm amongst a portfolio [18, 19], however, to date, the use of an SOM has not been explored for predicting the best performing algorithm amongst a portfolio for the VRP. Both an unsupervised and supervised SOM will be trained and tested as prediction models. The over-arching goal of both parts of this current research is to gain insight into problem characteristics and algorithm performance for the VRP.

5.2 Algorithm Selection Problem

The framework for the algorithm selection problem was formalized by Rice in 1976 [3]. In his relatively simple model, given a problem space $\mathcal{P}$ and an algorithm space $\mathcal{A}$, the goal is to

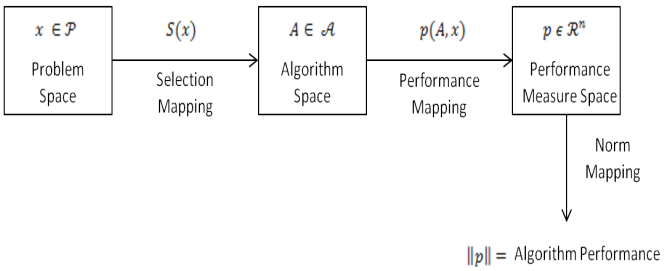


Figure 16: Rice's [3] original diagram of the algorithm selection model.

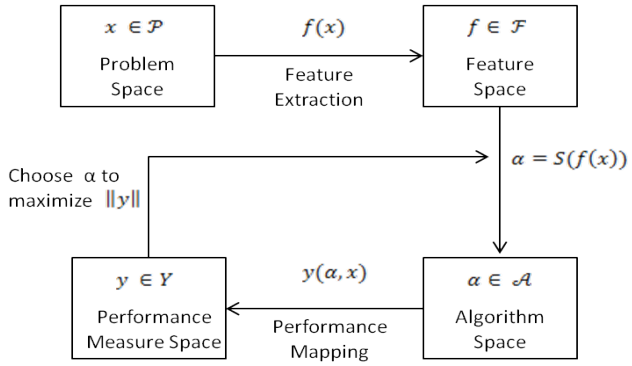


Figure 17: Rice's enhanced algorithm selection model to include the feature space [6].

find the performance mapping, $S(x)$, for each problem-algorithm pair [7]. Rice's original model can be seen in Figure 16. Rice describes the objective of the model as 'to determine $S(x)$ so as to have high algorithm performance' [3].

In the same paper, Rice [3] refines his basic model to include the problem features that can be used in order to determine the selection mapping. Figure 17 was presented by Smith-Miles [6] and slightly enhances Rice's model to explicitly indicate that the goal is choose α to maximize algorithm performance. The four main characteristics of the model are [3, 6]:

- **Problem Space** The problem space $\mathcal{P}$ represents the set of instances in a problem class.
- **Feature Space** The feature space $\mathcal{F}$ contains the problem characteristics extracted from $\mathcal{P}$.
- **Algorithm Space** The algorithm space $\mathcal{A}$ is the set of algorithms considered for solving the problems in $\mathcal{P}$.
- **Performance Space** The performance space $\mathcal{Y}$ represents the mapping of each algorithm to the set of performance metrics.

Smith-Miles formally states the algorithm selection problem as follows: For a given problem instance $x \in \mathcal{P}$, with feature vector $f(x) \in \mathcal{F}$, find the selection mapping $S(f(x))$ into algorithm space $\mathcal{A}$, such that the selected algorithm $\alpha \in \mathcal{A}$ maximizes the performance metric $\|y\|$ for $y(\alpha, x) \in \mathcal{Y}$.

Ideally, the intent is determine the optimal selection mapping $S(x)$ that results in the best performing algorithm being chosen for *any* problem x. However, due to the limitations of problems, mappings and algorithms that one can include in practice, a good solution to the algorithm selection problem is one that improves the best known results [7]. Focusing on the problems, features, and how the selection mapping is done is of utmost importance.

As Rice points out [3], and many other researchers have either investigated or called for [16, 17, 1], determining what features affect algorithm performance is one of the most critical components of the algorithm selection problem. The identification of these problem features for the CVRP are discussed next.

5.3 Feature Selection

To adequately model the mapping between problem instances and algorithm performance, it is critical to have a diverse population of problem instances. To measure the diversity amongst a set of problems, a list of features for categorizing the problems must be identified, and these features are then used to help understand the strengths and weaknesses of an algorithm's performance. This relationship between problem features and algorithm performance is paramount to answering the question 'for which types of problem instances can we expect a given algorithm in a portfolio to perform well and why?' [1].

In order to determine the most promising features for characterizing CVRP problem instances, the existing literature regarding the algorithm selection problem for the VRP is analyzed. Nygard et al [18] built a rather sophisticated, multi-layered neural network used to select the optimal algorithm for a given VRP instance. In this study, the problem features extracted from the VRP instances were strictly related to the clusters within the problem. In the 1997 study by Tuzun et al [19], the authors provide a more straightforward ANN for selecting the optimal VRP algorithm. In this work, the authors note the importance of four general aspects of a VRP problem:

- *Spatial Distribution of Customers* This describes how the customers are distributed within the service area of the problem. This will describe if customers are clustered, or evenly

distributed and is in line with the problem feature included in [18].

- *Depot Position* The authors note that 'the quality of a routing depends on the position of the depot where the routes originate from,' therefore it is reasonable to expect differences in algorithm performance based on this problem characteristic.

- *Average Number of Customers on a Tour* This relates the average customer demand to the vehicle capacity, and offers insight into how long each route might be.

- *Demand Structure* This refers to how the demand is distributed amongst customers. The authors describe how it is not always reasonable to assume the demand is evenly distributed, as many real world instances might include a few customers with a very high demand relative to the truck capacity.

In order to assess the statistical significance of these problem characteristics and their effect on an algorithm's performance, Tuzun et al devised an experimental design in which they generated 1280 VRP problem instances by varying six characteristics related to the previously described VRP characteristics:

- number of clusters,
- clustering ratio,
- spread ratio,
- position of the depot,
- average number of customers per tour,
- demand structure.

The first three of these characteristics relate to the spatial distribution of the customers, and the remaining three relate to each remaining previously described category. Tuzun et al found all of these problem characteristics to be statistically significant in affecting the tour length produced by a VRP algorithm.

Due to the fact that there is very little research into the algorithm selection problem for the VRP, attention is now turned into a special case of the vehicle routing problem, the Traveling Salesman Problem. Due to the similarities between the problems, it is a reasonable assumption

that the research into the characteristics of what makes a TSP instance difficult to solve [16, 20, 21, 22, 1, 23] can be related to the VRP.

When determining what features to include in a performance prediction model, it is important to keep in mind the length of time required to calculate these features [22]. Although fitness landscape analysis is useful in determining the difficulty of a problem [1], these metrics are not desirable for inclusion into a performance prediction model due to the extensive time and analysis required.

Twelve features are introduced in [22] to categorize a TSP problem instance, and these features are subsequently used in [23]. The authors [22] conclude that these features allow for the grouping of problem instances into distinct classes, and these features are useful for predicting algorithm performance. These twelves features are as follows:

- 1) the standard deviation of the distances in the distance matrix,
- 2) the (x, y) coordinates of the city centroid,
- 3) the radius of the TSP instance (defined as the mean distance from each city to the centroid),
- 4) the fraction of distinct distances in the distance matrix (with 2 decimal point precision),
- 5) the rectangular area within which the cities lie
- 6) the variance of the normalized nearest neighbor distances,
- 7) the coefficient of variation of the normalized nearest neighbor distances,
- 8) the cluster ratio (the ratio of the number of clusters to the number of cities),
- 9) the outlier ratio (ratio of number of outliers to cities),
- 10) the ratio of cities near the edge of a cluster,
- 11) the mean radius of the clusters (mean distance, from any city in a cluster to the cluster centroid, averaged over all clusters),
- 12) number of cities in the problem.

In the original work identifying these features [22], the authors note that number of clusters and the number of outliers, required for features 8 and 9, were found using GDBSCAN [24]

which is an extension of the DBSCAN [25]. This study by Smith-Miles et al [22] extracts the features directly from a problem instance, in real-time. The previously described work [18, 19] regarding the features for the CVRP, generated the problem instances based on the specific features. Therefore, the features were not extracted from a problem instance, the features for each problem instance were already known through the problem generation phase. In order to analyze features from existing problem sets, which is the aim of the current research, the ability to extract features from any given problem set, in real-time is key.

5.3.1 Clustering

Given the importance of clustering in analyzing algorithm performance on the CVRP [18, 19], as well as the TSP [22], it is included as a feature in this current research. Choosing an appropriate clustering algorithm for this task is paramount. In general, the users of clustering algorithms bear a large amount of responsibility in the algorithm's success, due to the fact that every algorithm has at least one user-defined parameter that helps to define what a cluster is [20]. Two common variants of these parameters are either knowing the number of clusters, k, in the dataset ahead of time, or defining a minimum distance between points in order to be considered part of the same cluster. Requiring the pre-existing knowledge of how many clusters exist in a VRP problem is simple for problems generated with pre-defined specifications, however, it is not feasible in the current setting where the purpose is to analyze existing VRP problem sets. Defining the second common parameter, the minimum distance between points, seems more reasonable in the current setting, specifically if all problems are standardized within the same bounded region of the $x - y$ plane.

To cluster the TSP problems in [20, 22], the authors used the hierarchical clustering algorithm, GDBSCAN. As previously mentioned, the GDBSCAN is an extension of the DBSCAN. The GDBSCAN extends the DBSCAN by extending the notion of what constitutes a neighborhood of objects, as well as how to count the number of objects within a neighborhood [24]. However, in the current context of the Euclidean VRP, neighborhoods will be determined by Euclidean distance, and the size of neighborhoods is measured by the cardinality of the set. In this case, we do not require any of the extensions beyond the original DBSCAN.

The DBSCAN requires two input parameters, Eps and $MinPts$. Eps is the maximum Euclidean distance allowed between two points for them to be considered part of the same neighborhood. $MinPts$ is the minimum number of points that are required to constitute a

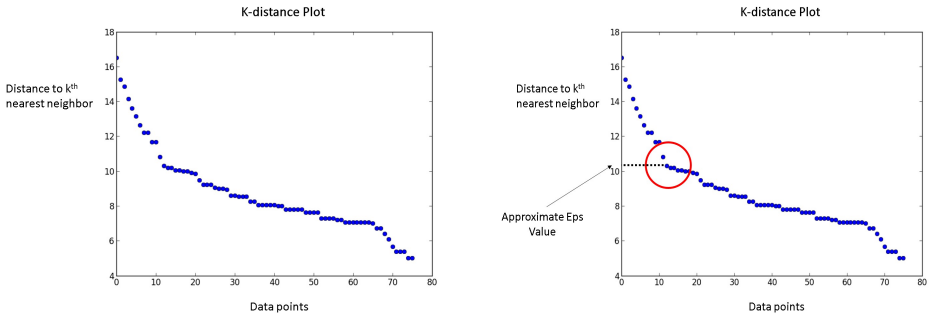
cluster. In order to cluster a dataset, DBSCAN starts at any arbitrary points, and identifies all points that are within a distance of Eps from this point. If the $MinPts$ requirement for a cluster is met, these points are considered to be part of the same cluster. The algorithm follows this logic for every point in the dataset. At the end of the algorithm, each point either has a cluster assignment, or it is considered an outlier. Within each cluster, each point is considered either a core point or a border point. A core point has at least $MinPts$ data points within a distance of Eps from it; whereas a border point is within the required Eps distance of another point in the cluster, but does not have the required $MinPts$ within an Eps distance to be considered a core point.

For the parameter $MinPts$, the DBSCAN authors [25] report that in their experiments, using a $MinPts \geq 4$ does not yield a significant difference from using $MinPts = 4$. However, increasing $MinPts$ does require a additional computation time. Therefore, the authors recommend using $MinPts = 4$ for all 2-dimensional data. Using $MinPts \leq 4$ is possible, and should be decided by the user based on domain knowledge. A $MinPts = 1$ does not make sense, as every data point could possibly become a cluster by itself.

The value of $MinPts$ is used to determine the second parameter Eps. Referring to the value of $MinPts$ as k, a list of the distances to the k^{th} nearest neighbor for each point is defined. This list is sorted descendingly and plotted as in Figure 18a. With regard to the $k - distance$ plot, when an Eps value is chosen, all points with an equal or smaller $k - distance$ will be core points, and all of the points with a larger $k - distance$ than the chosen Eps are not core points. To find this threshold value to distinguish between core points and non-core points the plot must be inspected for the first 'valley' [25]. Figure 18b illustrates the location of the first 'valley' in the $k - distance$ plot.

The DBSCAN authors [25] suggest an interactive approach to choosing Eps. When the user visually inspects the $k - distance$ graph for the first valley, the corresponding Eps value is used with the DBSCAN algorithm. If the results are satisfactory to the user, then the work is complete. However, if the DBSCAN results are not satisfactory, a different surrounding values of Eps are examined until a satisfactory result is reached. From their description, there is not a straightforward, automatic implementation of this process, due to the high reliance on user input. In the current research, the aim is to have an automatic clustering algorithm without any requirement or inference from the user.

The GDBSCAN algorithm was used with the clustering of TSP problems in [22], and the



(a) $k - distance$ plot (b) identifying the first valley

Figure 18: Example of how to use the $k - distance$ plot to identify Eps.

author set the $Eps = 40$ for all problems. In this work, the problems were generated by the authors, and each problem consisted of 100 cities generated within a $[0, 400]$ by $[0, 400]$ grid. Using an $Eps = 40$ roughly corresponds to the expected value of the 4th nearest neighbor distance, if the cities were uniformly placed on the 400 x 400 grid. If 100 cities are placed equidistant from each other on the grid, then the distance between any city that is not along the edge, and its 4th nearest neighbor is $dist = \frac{400}{(\sqrt{100}-1)} = 44.44$. A pictoral example of this can be seen in Figure 19. This distance to the 4th nearest neighbor can be generalized as follows, given a $b \times b$ grid where n items are placed uniformly:

$$d_4 = \frac{b}{\sqrt{n} - 1}$$

For the previous work on the TSP problem [22], it appears as though the authors used the following equation:

$$Eps = \frac{b}{\sqrt{n}} \tag{40}$$

Although it is known that the authors used $Eps = 40$ when running their clustering algorithm, their exact motivation is unclear. It does however, make sense to use an approximate value of the expected distance of the 4th nearest neighbor. This would insinuate that if the cities are not uniformly placed in the grid, then any point with a 4th nearest neighbor less than $Eps = 40$ should be considered a core point.

For the current VRP benchmark problem sets, DBSCAN was run using Equation 40 to determine Eps. Although this method of determining Eps provided good results on many problems, the results were not always favorable. An example of a problem where Equation 40 did not result in a desirable clustering of the problem can be found in Figure 20b. In Figure 20b, the colored circles represent cities that belong to a cluster. If the circle is large, this is a core point,

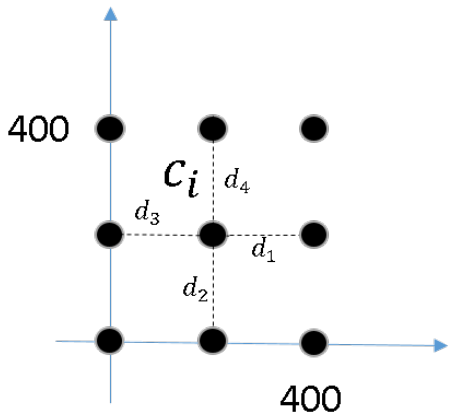
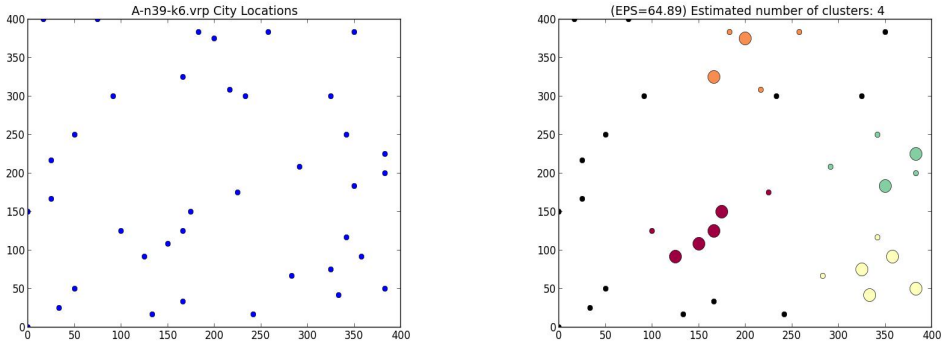


Figure 19: $n = 9$ cities placed uniformly on a 400 x 400 grid. The distance from c_i to its 4 nearest neighbors is $d = \frac{400}{2} = 200$.



(a) locations of the cities for a CVRP instance

(b) identifying the first valley

Figure 20: Example of using Equation 40 to define *Eps* parameter in DBSCAN.

smaller colored circles represent non-core points of the cluster. Any point that is not colored is considered an outlier by DBSCAN.

In Figure 20, the cities are fairly spread out across the plane. Although there are some cities that are closer to one another than others, it is difficult to clearly see any clusters in this graph. By setting *Eps* according to Equation 40, however, DBSCAN finds 4 clusters amongst the cities. Although clustering does consist of an element of subjectivity regarding what does or does not constitute a cluster, this current research does not consider the existence of any clusters in the data shown in Figure 20.

One of the advantages of the DBSCAN algorithm has to do with its inherent ability to identify clusters with varying densities. The DBSCAN authors describe that 'the main reason

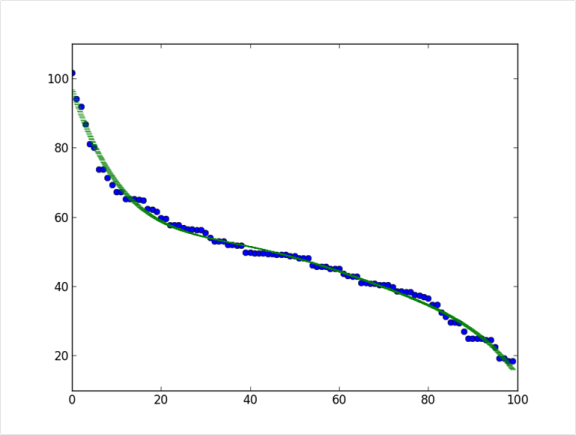


Figure 21: 5th degree polynomial fit to a $k - distance$ plot using *numpy.polyfit* in Python.

why we recognized the clusters is that within each cluster we have a typical density of points which is considerably higher than outside of the cluster' [25]. By setting the *Eps* value according to Equation 40, this does not allow an instance-specific definition of what it means for a cluster to have a 'considerably higher' [25] density as compared to the surrounding points. Due to this disadvantage of setting *Eps* according to Equation 40, three additional approaches for automatically determining the *Eps* value were examined.

The second method examined for determining the *Eps* value investigates how to automatically identify the 'valley' in the $k - distance$ graph. In this case, a polynomial is first fitted to the $k - distance$ graph. In order to fit the polynomial, the *numpy.polyfit* library of Python was used. An example of a curve fitted to the $k - distance$ plot can be seen in Figure 21. Once the polynomial is fit, the idea is that the 'valley' in the graph will be near the first inflection point of the polynomial. Once examined, however, this was determined to not be a reliable method. Figure 22 shows the approximate locations of the inflection point and the 'valley.' The inflection point is approximately located within the yellow rectangle, whereas the 'valley' is located in one of the two red circles. Figure 23 provides two additional examples of the difference between the 'valley' and the inflection points.

The inflection point of the fitted polynomial and the 'valley' in the plot do not precisely correspond. Although the inflection point serves as an easily computable point in the polynomial, it is not necessarily the exact point of interest. Perhaps a better description of the 'valley' is the location where the change in the slope is *sufficiently* small, *relative* to the surrounding points, and not necessarily the actual inflection point of the fitted polynomial. The challenge with

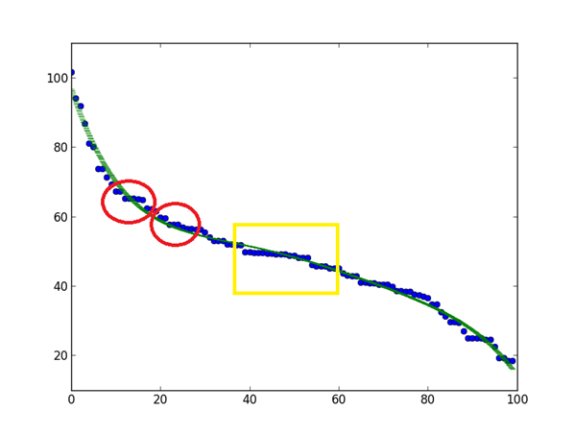
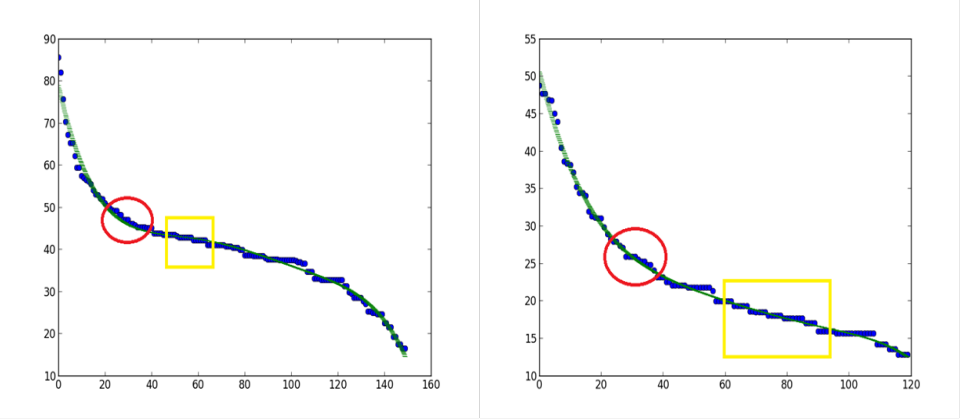


Figure 22: Comparing the relationship between the 'valley' in the curve and the inflection point. The approximate inflection point is found in the yellow rectangle, whereas the first 'valley' is located in one of the red circles.



(a) (b)

Figure 23: Example of the difference between the approximate locations of the inflection point and 'valley' in the fitted $k - distance$ curve.

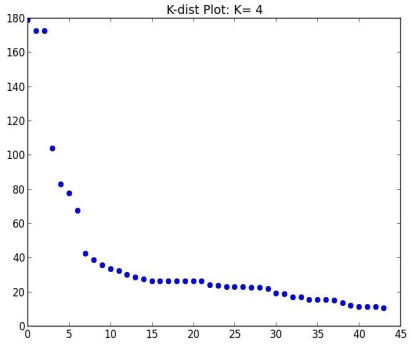
pursuing this method of identifying *Eps*, is that with descriptive words such as *sufficiently* and *relative*, this approach will require some sort of user-defined parameter to define what it means to be *sufficiently* small, or how the *relative* points are defined. Due to the fact that parameters must be user-defined in order to automatically define *Eps* with this approach, the problem of automatically defining parameters still exists.

As the DBSCAN authors describe choosing the value of *Eps*, they refer to all of the points to the left of the *Eps* value as 'noise' [25]. When referring to data as 'noise', this insinuates that the data is an anomaly or an outlier. Therefore, it is reasonable to assume that the majority of points in the $k-distance$ graphs will be considered core points, as 'noise' in data is often a small percentage of the overall data. Due to the fact that automatically detecting the 'valley' in the $k-distance$ plot is not easily attainable, attention is now focused on approximating the *Eps* value based on the expected amount of noise in the data.

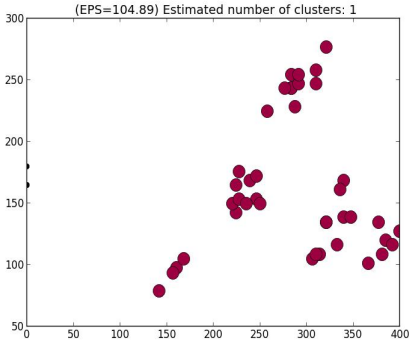
Chebyshev's Theorem is useful in determining a threshold point in the $k-distance$ plot, where the majority of $k-distance$ values fall to the right of this point. Chebyshev's Theorem states that for any distribution, at least $\left(1-\frac{1}{k^2}\right)\%$ of the data points lie within k standard deviations of the mean. Therefore, it is expected that 75% of points lie within 2 standard deviations of the mean. Due to the fact that the majority of points are less than this desired threshold point, attention is only focused on possible *Eps* values which are 2 standard deviations above the mean.

For this approach to identifying an appropriate *Eps* value, a range of values is explored for *Eps*, similar to the work in [20]. First, the mean and standard deviation of the $k-distance$ values are identified for a given problem instance. The range $[mean, mean+2\sigma]$ is divided into s equally spaced values ($s=10$ to begin with). Next, the DBSCAN algorithm is run using each of these values for *Eps*. For each of the s runs of DBSCAN, the number of clusters found is counted. After all s runs of DBSCAN are complete, the *Eps* value(s) which correspond to the most frequent number of clusters found are recorded in a list, and the median of this list is used for the final *Eps* value. In the event that there are an equal number of occurrences for different numbers of clusters, the routine is run again, but this time, the range of *Eps* values is divided into $2s$ equally spaced values. This continues until the greatest number of clusters found is unique.

This method of identifying *Eps* works well quite often. However, there are certain conditions under which the clustering is unfavorable. In problem instances where the $k-distance$ values are skewed, with a few very large values, this method of finding *Eps* is not sufficient. In this case,



(a) $k - distance$ plot with outliers



(b) Choosing *Eps* within 2 standard deviations above the mean.

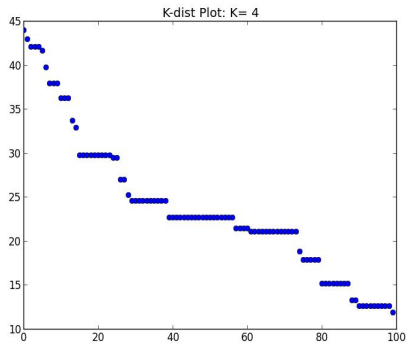
Figure 24: Choosing *Eps* from values within 2 standard deviations above the mean is ineffective in the presence of large $k - distance$ outliers.

the range of *Eps* values examined is too high and actually consists of a considerable amount of outlier $k - distance$ values. As a result of using an *Eps* value that is too high, the data points generally turn into one large cluster, as in Figure 24. The *k-distance* graph in Figure 24a shows the existence of a few outliers. Due to these outliers, searching for *Eps* within 2 standard deviations above the mean results in one large cluster of data, as seen in Figure 24b.

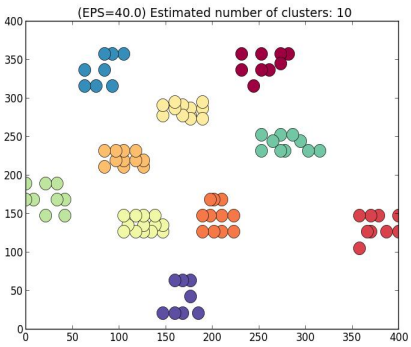
Because the use of the mean in the previous approach is adversely affected by extreme outliers, the final method explored for finding *Eps* eliminates the bias introduced by the outliers. In this final method, the range of *Eps* values tested is between the median and 85th percentile of the $k - distance$ values. By restricting the range to $[median, 85thpercentile]$, there is no longer an adverse influence of extreme outliers, and this range inherently indicates that the *Eps* value used will be a threshold that the majority of the points are below. A side-by-side comparison of the three main approaches for identifying *Eps* can be found in Figures 25,26, and 27. Figure 25 shows a problem instance where all three approaches result in a similar clustering, whereas Figures 26 and 27 show the side by side comparison of all three methods on instances where one approach was previously shown to be inadequate. The second approach, using curve fitting, is not presented, as it was never implemented in a way that did not require user defined parameters.

The four methods examined for how to automatically define *Eps* for use in the DBSCAN algorithm are as follows:

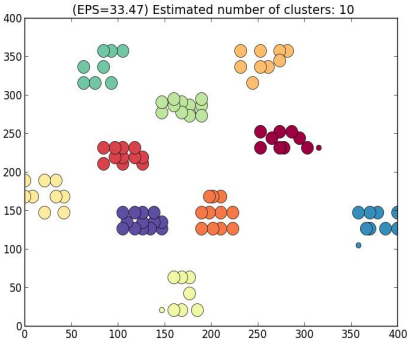
1) Define $Eps = \frac{400}{\sqrt{n}}$ where 400 is the length of the unit square (400×400) in which all problems are standardized to. This is a similar approach as used in [20, 22].



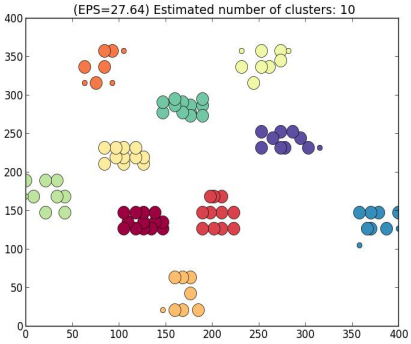
(a) $k - distance$ plot with outliers



(b) Using $Eps = \frac{400}{\sqrt{n}}$

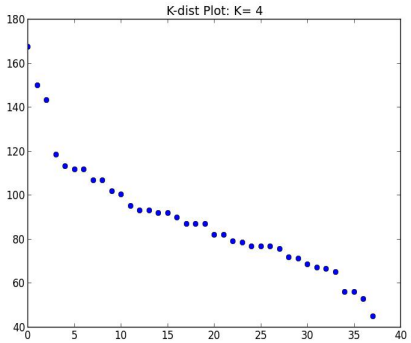


(c) Choosing Eps between $[mean, mean + 2\sigma\ ]$

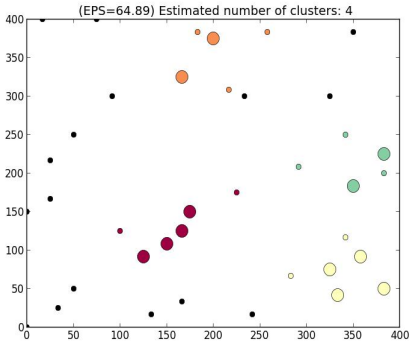


(d) Choosing Eps between $[median, 85^{th}\%\ ]$

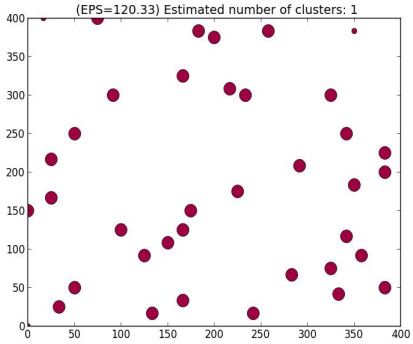
Figure 25: Choosing Eps where all three methods yield similar results.



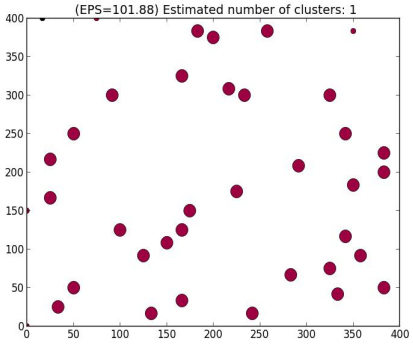
(a) $k - distance$ plot with outliers



(b) Using $Eps = \frac{400}{\sqrt{n}}$

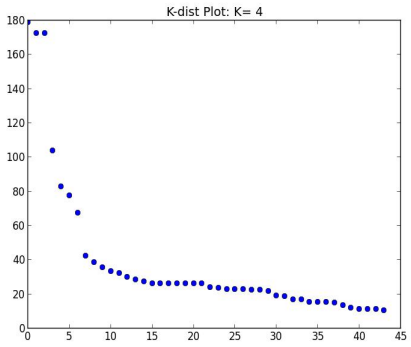


(c) Choosing Eps between $[mean, mean + 2\sigma\ ]$

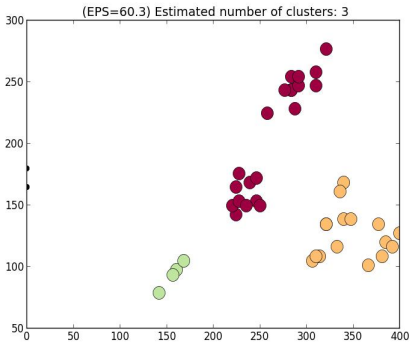


(d) Choosing Eps between $[median, 85^{th}\%\ ]$

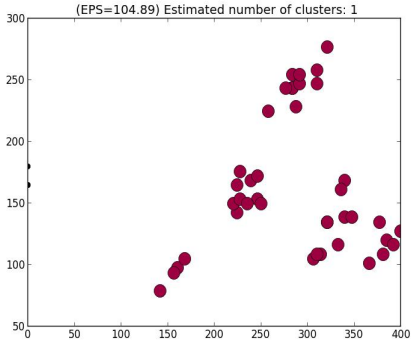
Figure 26: Choosing Eps where choosing Eps using Equation 40 is not sufficient.



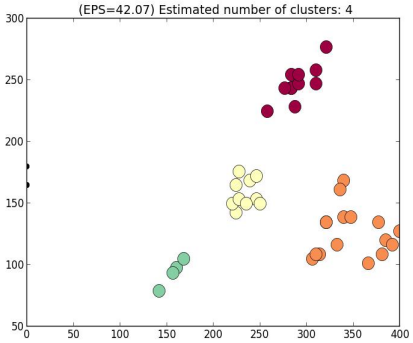
(a) $k - distance$ plot with outliers



(b) 3 clusters found using $Eps = \frac{400}{\sqrt{n}}$



(c) 1 cluster found choosing Eps between $[mean, mean + 2\sigma\ ]$



(d) 4 clusters found choosing Eps between $[median, 85^{th}\%\ ]$

Figure 27: Choosing Eps where the method of choosing Eps between $[mean, mean + 2\sigma\ ]$ is not sufficient.

2) Fit a polynomial to the $k - distance$ graph, and look for Eps near the point of inflection.

3) Run DBSCAN multiple times using a range of values between $[mean, mean + 2\sigma\ ]$ for Eps, and choose an Eps value that corresponds to the most frequent number of clusters found across all runs of DBSCAN.

4) Run DBSCAN multiple times using a range of values between $[median, 85^{th}\%\ ]$ for Eps, and choose an Eps value that corresponds to the most frequent number of clusters found across all runs of DBSCAN.

The fourth method was deemed most effective by visual inspection of the clusters produced across the benchmark problem sets. Although this method cannot recreate the human reasoning and preference involved by a person inspecting the $k - distance$ graph for the 'valley' as the DBSCAN authors advise, this fourth method for automatic Eps selection performs well in this VRP problem setting. The premise of using an Eps value between the median and $85^{th}\%$ of the $k - distance$ graph aligns with the DBSCAN authors' description of where to find the 'valley' in the $k - distance$ graph. Additionally, the fact that multiple Eps are tested in the DBSCAN algorithm is in line with visual inspection of the $k - distance$ algorithm, where it is often easy to identify an *area* where the 'valley' is, but it is more difficult to pinpoint an exact value, such as in Figure 18.

5.3.2 Problem Features for the CVRP

For both parts of this research, problem features for each CVRP instance are required, and the same 21 CVRP problem features will be used in both parts. These features are compiled from the previously described work on both the TSP [22], as well as the previous work on algorithm selection for the CVRP [18, 19]. One key aspect that makes the current work differ from the previously discussed studies, is that this current research seeks to analyze existing CVRP benchmark problem sets, whereas the previous studies generated their own problem sets. By analyzing the existing problems, careful consideration must be given to how the problems will be standardized to allow for a side-by-side comparison. It is important that the problems are transformed to the same area in the $x - y$ plane in order to make a fair comparison between many of the desired problem features. In order to make this transformation for each problem, the following formula is used to tranform each problem to the same (400×400) square. These

transformed coordinates are used for all required calculations during the feature extraction.

$$x_{new} = \frac{400(x_{curr} - min^*)}{(max^* - min^*)} y_{new} = \frac{400(y_{curr} - min^*)}{(max^* - min^*)} \tag{41}$$

In Equation 41, min^* and max^* are the smallest and largest coordinates from the original problem (amongst both the x and y coordinates). The following list provides a description of the 23 problem features, as well as a brief description of how the values are calculated.

1) **Number of cities** This excludes the depot.

2) **Standard deviation of the distance matrix** The euclidean distance between the city coordinates are used to create the distance matrix.

3) **X coordinate of the centroid** The centroid is found using the city coordinates and does not include the depot in the calculation.

4) **Y coordinate of the centroid** The centroid is found using the city coordinates and does not include the depot in the calculation.

5) **Radius of the VRP instance** This is the average distance between each city and the centroid.

6) **Fraction of distinct distances in the distance matrix** This is calculated with 2 decimal points.

7) **Standard deviation of nearest neighbor distances** Only the first nearest neighbor is used in this calculation.

8) **Coefficient of variation of the nearest neighbor distances** This is found by dividing the standard deviation of the nearest neighbor distances by the mean of the nearest neighbor distances.

9) **Cluster ratio** The number of clusters is found by using DBSCAN as described in the previous section. The number of clusters is divided by the number of cities.

10) **Outlier ratio** The number of outliers is an output from the DBSCAN algorithm. The number of outliers is divided by the number of cities.

11) **Ratio of cities near the edge of a cluster** The number of edge cities corresponds to the points in DBSCAN that are neither core points or outliers. The number of edge points is divided by the number of cities.

12) **Number of clusters** This is output from DBSCAN algorithm.

13) **Mean radius of the clusters** The average distance from each city to its cluster centroid, and this is averaged across all clusters.

14) **X coordinate of the depot**

15) **Y coordinate of the depot**

16) **Standard deviation of demand** For each problem, the demand for each city is transformed into a proportion of the vehicle capacity. The standard deviation of these proportions is used.

17) **Ratio of total demand to total capacity** The total demand of a problem divided by the total vehicle capacity. If the number of trucks is a decision variable, the minimum number of trucks required is used.

18) **Ratio of max cluster demand to vehicle capacity** The maximum demand of a cluster divided by vehicle capacity. If this value is greater than 1, this indicates more than 1 vehicle is required to service this cluster.

19) **Ratio of outlier demand to overall demand** The sum of all outlier demand divided by the total problem demand.

20) **Ratio of single largest city demand to vehicle capacity**

21) **Average number of customers per vehicle route** The total number of customers divided by the number of vehicles. If the number of vehicles is a decision variable, the minimum number of vehicles required is used.

22) **Area of the rectangle the problem lies in** This uses the distance between the max and min points in the x direction by the distance between the max and min points in the y direction.

23) **Minimum number of trucks required** $\lceil \frac{totaldemand}{vehiclecapacity} \rceil$

5.4 SOM for Exploratory Data Analysis and Prediction

The primary goal of exploratory data analysis is to organize and present a set of data 'in a form that is easily understandable but that at the same time preserves as much essential information of the original data set as possible' [26]. Two general categories of how to approach

exploratory data analysis include projecting the multi-dimensional data set on to a lower dimensional plane in order to help visualize the differences in the data; as well as finding an appropriate clustering of the data in such a way that provides an understanding of the similarities amongst the data [26]. An example of a method that projects a high-dimensional data set into a lower dimension is Sammon's Mapping [27]. Sammon's approach is to project the high-dimensional data into either a two or three dimensional space while trying to preserves the structure in the original dataset [27]. With regard to clustering in exploratory data analysis, the goal is to identify an underlying structure or similarities within the dataset. Research into clustering spans numerous communities, and a significant number of clustering techniques and approaches exist across many disciplines [28].

For exploratory data analysis, the self-organizing map [29, 30, 31, 32] provides both a method for projecting high-dimensional data into a two dimensional space, as well as a way to visualize the clusters in a data set [26, 33]. For data exploration, the SOM is normally initialized with a two-dimensional array of nodes [26]. Each node in the model has a vector, $\mathbf{m}_i$ associated with it. The initial values of these vectors are either randomly generated, or they are initialized to randomly chosen values from the input vectors (data set). To train the SOM, a set of input vectors with the same dimension as $\mathbf{m}_i$, are presented one at a time. The nodes compete to determine which node is most similar to the input vector $\mathbf{x}$, and the wining node, c is identified according to Equation 42.

$$c(\mathbf{x}) = \text{ArgMin}_i\{\|\mathbf{x} - \mathbf{m}_i\|^2\} \tag{42}$$

The winning node and its neighbors adapt their vectors to become more similar to the input vector. The node update rule at any time t depends on both the input vector $\mathbf{x}(t)$ as well as the winning node, $c(\mathbf{x}(t))$. The update rule is as follows:

$$\mathbf{m}_i(t+1) = \mathbf{m}_i(t) + h_{ci}(t)[\mathbf{x}(t) - \mathbf{m}_i(t)] \tag{43}$$

In Equation 43, $h_{ci}(t)$ is the neighborhood function which determines the influence of the winning node on its neighbors. This neighborhood function is a decreasing function and depends on the distance between a node i and the winning node c, with regard to their (x, y) positioning in the SOM grid. The neighborhood function h_{ci} is chosen in such a way that the neighborhood is large near the beginning of learning, and it decreases slowly as learning progresses [33].

Once trained, the SOM offers an ordered display of the dataset. Each data item is mapped

to the SOM node with the closest reference vector to its own. As a result of the training phase of the SOM, nodes near one another on the SOM will have similar reference vectors, resulting in similar data items mapped [33] to the same region of the trained SOM. A classic example and illustration of the mapping of a high-dimensional dataset into a two-dimensional space with the SOM can be found in [26]. In this study, a 39-dimensional dataset pertaining to welfare and poverty in various countries around the world is explored with the use of the SOM. The resulting mapping of the dataset provides a visual representation of how the countries relate to each other based on these 39 factors.

From a trained SOM, the density of the reference vectors of the nodes provide a representation of the density of the input samples [33], and this provides a means to visualize the clusters in the dataset. Displaying the distance between neighboring nodes allows the visualization of the clusters as the high density area, whereas the space between clusters can be viewed as regions where the distance between neighboring nodes is greater [34, 35, 33]. In addition to using the SOM for data visualization through projection and clustering, the SOM can also be used for prediction. Two methods, supervised and unsupervised, of training a SOM for prediction will be discussed.

The unsupervised SOM is the simplest approach to prediction with the SOM. After a SOM has been trained, a dependent variable for each of the items in the data set is introduced [36]. In the case of a continuous, dependent variable, the prediction for new data item is found by first identifying the SOM node with the closest reference vector to the input item. The prediction is then calculated as the mean of all of the objects previously mapped to this node [36]. In the case of a categorical dependent variable, however, the predicted value associated with a node in the trained SOM uses a 'winner-takes-all' strategy based on all of the objects previously mapped to the node during the training phase [36].

A second method suggested by Kohonen [31] for training a SOM for prediction is to concatenate both the independent and dependent variables so they both may be used for training the SOM. This supervised SOM was formally introduced in [37], and it is referred to as an 'XY-Fused' SOM network, where X refers to the independent variables (problem characteristics for this application), and Y refers to the dependent variables (best performing algorithm for this application).

During the training phase, the distance between two vectors, i and j, is measured with both

an X and Y component as follows:

$$D(i,j) = \alpha D_x(i,j) + (1-\alpha) D_y(i,j) \tag{44}$$

In this equation, the weight of the X component starts high, but as the training progresses, the weighting between the X and Y components become equal.

The distance between the X components of two vectors is measured by Euclidean distance. The distance between the Y components is measured by Euclidean distance when Y is a continuous variable, and the Tanimoto distance is used when Y is a categorical variable. A detailed description of the calculation of the Tanimoto distance, as well as details on how Y is updated during training is found in [36, 37].

Once trained, a new vector is mapped to the SOM using only the X vector. Once mapped, the predicted Y value for this input vector corresponds to the Y value of the best matching node in the SOM.

5.5 Experimental Setup

As previously stated, this research is divided into two parts. Although each part is related to the algorithm selection problem, each section has a slightly different objective. The first part explores the diversity of the existing VRP benchmark problem sets in the context of the 23 problem characteristics/features identified from the literature, whereas the second part aims to assess the effectiveness of using an SOM for prediction of the best performing VRP algorithm. Both an unsupervised and supervised SOM are tested for prediction.

5.5.1 Problem Characteristics and Diversity in Benchmark Problems

In this first section, the diversity in the existing VRP problem sets is explored using an unsupervised SOM, with the 23 problem characteristics previously described. Not only will this research assess the diversity of the existing benchmark problem instances, but it will also explore the effectiveness of the 23 problem characteristics for distinguishing between VRP instances.

The scope of problems examined is limited to the capacitated VRP, and 102 commonly used benchmark problems were identified from the literature. A list of these problems can be found in Appendix A, and were originally published in [38, 39, 40, 41, 42]. These instances were chosen due to the their prevalence in the literature, as well as the fact that the best known solution (including many optimal solutions) are also widely cited in the literature.

According to the No Free Lunch Theorem [2], an algorithm is not expected to perform

well across all problem instances. Therefore, in order to help understand the diversity in the benchmark problem sets, the performance of three CVRP algorithms across all problem instances is considered. The expectation is that if these chosen benchmark problems represent a diverse set of problem instances, each algorithm's performance should vary across all problems. The three CVRP algorithms used in this experiment are the Fuzzy Self Organizing Map presented in Chapter 4, the parallel Clarke and Wright algorithm [43, 44] and the Sweep algorithm [45]. Since the Fuzzy SOM employs a stochastic search, the algorithm is replicated on each problem instance 100 times, and the best solution amongst these replications is reported. The Sweep Algorithm implementation used in this research follows the algorithm outlined in [46]. Once the customers are separated into routes according to the Sweep algorithm, the routes are then improved using a 2-opt heuristic. This Sweep plus 2-opt algorithm was repeated n times, with each city used as the starting vertex once. The best result of these n replications is reported. All algorithms are coded and executed in the *Python* programming language.

Although this first part does not aim to solve the algorithm selection problem, each component of the algorithm selection problem is utilized in order to gain an understanding of the diversity of the existing benchmark problem sets. Each component of the algorithm selection problem is identified as follows:

- The problem space $\mathcal{P}$, is defined as the 102 benchmark problems found in Appendix A, which are readily available in the literature [38, 39, 40, 41, 42].
- The feature space $\mathcal{F}$, consists of the 23 problem features identified in Section 4.2.
- The algorithm space $\mathcal{A}$, consists of the parallel Clarke and Wright algorithm, the Sweep + 2-opt algorithm, and the Fuzzy SOM algorithm.
- The performance space $\mathcal{Y}$, consists of an algorithm's performance measured by the percent deviation above the best known solution.

In order to explore the diversity in the 102 existing benchmark problem sets, three aspects are considered. The first is the range of values for the 23 problem characteristics across all of the benchmark problems; the second is how well these problem characteristics can help distinguish algorithm performance; and the third is how well the unsupervised SOM can uncover clusters in the benchmark problems.

The first aspect investigates the range of values for each of the 23 problem characteristics. The range of values is explored using exploratory data techniques. Histograms are used as a visual means to represent the range of values for each problem characteristic. The coefficient of variation (cv) is the ratio of the standard deviation to the mean $\left(\frac{\sigma}{\mu}\right)$ and is found for each of the 23 problem characteristics. In general, the cv is a measurement of the variability of data, where a higher value indicates more variability, and a lower value indicates less.

The second aspect considered is the performance of each algorithm across all of the benchmark problems. From the perspective of the No Free Lunch Theorem, there should be variability of each algorithm's performance across all problems if this is a diverse problem set. Each algorithm's performance across all problems will be examined to look for a relationship between problem characteristics and algorithm performance. This might lead to the discovery of problem characteristics that affect all algorithms' performance, or perhaps just a relationship between certain problem characteristics and algorithm combinations.

Finally, the third aspect of assessing the diversity of the benchmark problem sets includes the training of an unsupervised SOM. The SOM will be examined with regard to how the data is organized in the map, and how this organization corresponds to the 23 problem characteristics as well as algorithm performance. All data analysis is conducted in the R statistical software, and the *kohonen* package [36] is used to create and analyze the SOM.

5.5.2 Using a SOM for Algorithm Performance Prediction

The second part of this research will assess the performance of both an unsupervised and supervised SOM as a means to predict the best performing algorithm for a CVRP instance. In order to train and test each SOM, 200 CVRP instances are used along with the 23 problem characteristics extracted from each problem instance.

Of the 200 CVRP instances, 102 are the previously described benchmark problem sets, and the remaining 98 problem instances are derived from these original benchmark problems. The purpose of generating the new problem instances from the existing problems is to keep the 23 problem characteristics similar across all problems.

Of the existing benchmark problems, there are varying scales used for the (x, y) coordinates of the cities, as well as the city demands. In order to combine the problem instances to generate new problems, the existing problems are standardized to the same area of the $x-y$ plane according to Equation 41, and the problem demands are transformed into a proportion of the problem's

vehicle capacity as described in 14) of the problem characteristics. This proportional demand for city i is referred to as pd_i.

The first step in generating the 98 new problem instances is to randomly choose two of the existing benchmark problems, these will be referred to as the *parent* problems. The number of cities, N_{new}, and the capacity of the vehicle, Q_{new}, in the new problem instance are generated as a random integer found on the interval bounded by the values in the *parent* problems. Once the number of cities and the vehicle capacity is determined for the new problem instance, the cities and their corresponding demands (pd_i) are randomly chosen from each of the *parent* problem instances proportionally.

For the newly generated problems, the demand for each city is found by multiplying the pd_i by the vehicle capacity of the new problem instance, Q_{new}. This demand is rounded to the nearest integer, and each city is required to have a demand greater than or equal to 1. Therefore, any demand in the new problem that rounds to zero is replaced by the minimum demand of 1.

Each of the 200 problem instances is solved with three CVRP algorithms: the parallel Clark and Wright, the Fuzzy SOM, and the Sweep plus 2-opt. The algorithms were run exactly as described in the previous section. For each of the 200 problem instances, the best performing algorithm for each problem instance is recorded. Algorithm performance is based strictly on solution quality. Therefore, the best performing algorithm for any problem is the algorithm with the lowest cost solution to the CVRP.

Both the unsupervised and supervised SOMs are trained and tested using these 200 CVRP problem instances along with the 23 problem characteristics for each instance. The 200 problem instances are randomly divided into a *training set* and a *test set*. Once each SOM is trained, the prediction for the *test set* of problems is recorded. In order to assess the effectiveness of the SOM as a predictive model, the results of the SOM prediction is compared to the known best performing algorithm for each problem in the *test set*.

This work can be examined in the context of the algorithm selection problem as follows:

- The problem space $\mathcal{P}$, is defined as the 200 problem instances where 102 of these problems are benchmark problems from the literature and can be found in Appendix A; and the remaining 98 problems are generated from random two problem combinations of these 102 benchmark problem sets.

- The feature space $\mathcal{F}$, consists of the 23 problem features identified in Section 4.2.

- The algorithm space $\mathcal{A}$, consists of the parallel Clarke and Wright algorithm, the Sweep + 2-opt algorithm, and the Fuzzy SOM algorithm.
- The performance space $\mathcal{Y}$, consists of the solution quality found by algorithm, where the best performing algorithm is the algorithm that found the lowest cost solution.

The new problem instances are generated and solved with the three algorithms using the *Python* programming language. The SOM is created and analyzed using the *kohonen* package in *R*.

5.6 Experimental Results

The results of the experiments described in the previous section are reported below. The results are again divided into the two distinct focuses of this research: assessing the diversity of the existing benchmark problems, and assessing the effectiveness of an SOM for predicting algorithm performance.

5.6.1 Problem Characteristics and Diversity in Benchmark Problems

The 23 problem characteristics are extracted from the 102 benchmark problem instances taken from the literature. To assess the diversity of these problems with regard to each of the problem characteristics, a histogram for each problem characteristic is generated. The full set of histograms for each problem characteristic can be found in Appendix 6.0.2. Analyzing these histograms provides a basic description of how these characteristics vary across all of the benchmark problems. For example, looking at Figure 28, it is clear that the majority of problems in this benchmark problem set have either zero or one cluster, whereas very few problems in the set had more than six clusters. Looking at Figure 29, however, it is clear that the benchmark problem sets span all of the possible values for the proportion of distinct distances within the distance matrix. The set of benchmark problems consisted of problems with very few or no distinct distances within the distance matrix (ie, problems where the cities are equally spaced), to problems where all of the distances are distinct. In addition to histograms, the coefficient of variation (cv) for each problem characteristic is found. These cv values for the 23 problem characteristics are found in Table 7.

The second method for exploring the diversity of the existing benchmark problems is to examine how the problem characteristics relate to each algorithm's performance. The 102 benchmark problems were solved using the parallel Clark and Wright algorithm, the Sweep plus 2-opt

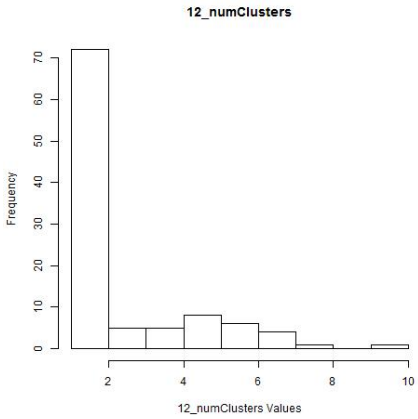


Figure 28: number of clusters

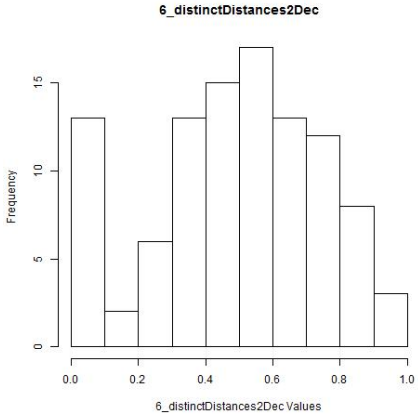


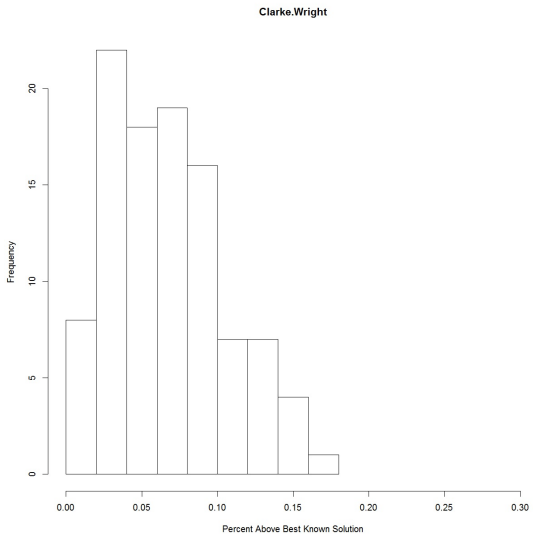
Figure 29: proportion of distinct distances

algorithm, and the Fuzzy SOM algorithm. For these 102 benchmark problems, each algorithm's solution is compared to the best known solution found in the literature, and the percent deviation from this best known solution is recorded, where a higher PDB indicates a lower quality solution. The full results are reported in Table .13 in Appendix A.

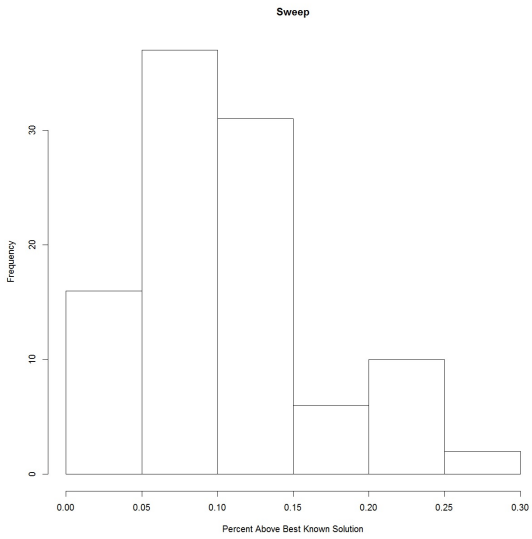
Figure 30 provides a histogram of the PDB for each algorithm across all of the benchmark problems. These histograms provide insight into the overall performance of each algorithm. For example, Figure 30c indicates that more of the Fuzzy SOM algorithm results were within 5% of the best known solution as compared to the other two algorithms, however the worst performance of the Fuzzy SOM algorithm was worse than the worst performance of the Clarke and Wright algorithm.

Each algorithm's performance (measured by PDB) is examined across each problem characteristic. Figure 31 plots the performance of each algorithm against the number of cities. For each of the algorithms, it is clear that their performance, on average, was worse on problems with a greater number of cities. For the Fuzzy SOM, however, there appears to be some linear relationship between algorithm performance and the number of cities. The average PDB appears to increase for problems as the number of cities increases. The full set of scatter plots found in Appendix 6.0.2 offer insight into how each problem characteristic affects each algorithm's performance.

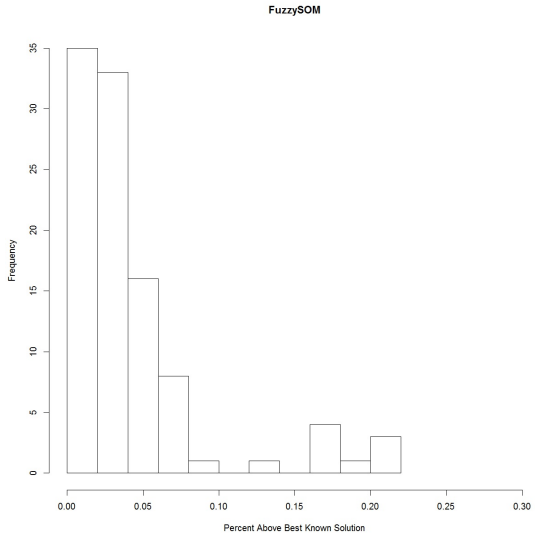
The last part of the investigation into the diversity of the existing benchmark problems uses an unsupervised SOM for exploratory data analysis. The SOM nodes are 23 dimensional vectors which correspond to the 23 problem characteristics extracted from the 102 benchmark



(a) Clarke and Wright Algorithm



(b) Sweep Algorithm



(c) Fuzzy SOM Algorithm

Figure 30: Histograms of each algorithms performance, measured by PDB, across all benchmark problems

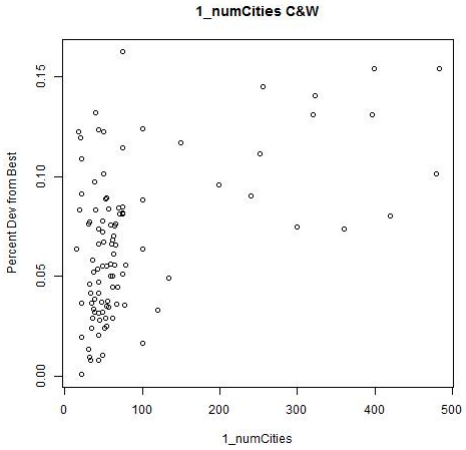
Table 7: Coefficient of variation for problem characteristics across the 102 benchmark problem sets

problem characteristic		mean	std dev	cv
1	Number of cities	93.38	88.46	.95
2	standard deviation of distance matrix	96.26	16.42	.17
3	x-coordinate of problem centroid	193.46	30.71	.16
4	y-coordinate of problem centroid	196.89	25.78	.13
5	radius of problem instance	141.62	23.89	.17
6	fraction of distinct distances in distance matrix (2 decimals)	.59	.29	.50
7	standard deviation of nearest neighbor distances	13.27	6.84	.52
8	coefficient of variation of nearest neighbor distances	.57	.31	.54
9	ratio of number of clusters to number of cities	.041	.036	.88
10	ratio of number of outliers to number of cities	.059	.062	1.05
11	ratio of the number of edge cities to total number of cities	.086	.048	.55
12	number of clusters	2.37	1.86	.79
13	mean radius of the clusters	93.3	49.66	.53
14	x-coordinate of the depot	176.27	87.47	.496
15	y-coordinate of the depot	188.90	94.15	.498
16	standard deviation of demand	.070	.033	.470
17	ratio of total demand to total capacity	.93	.049	.053
18	ratio of max cluster demand to vehicle capacity	6.58	5.72	.87
19	ratio of outlier demand to overall demand	.067	.079	1.18
20	ratio of the maximum city demand to the overall demand	.34	.18	.53
21	average number of customers per route	11.02	6.12	.56
22	area of the rectangle	142855.50	22334.49	.156
23	minimum number of trucks	8.79	5.63	.64

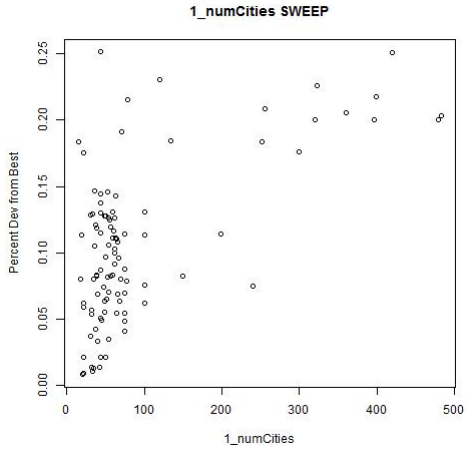
problems. The 102×23 input data set is first scaled so the data corresponding to each problem characteristic has a mean of zero and standard deviation of one. This ensures the equal weighting of each problem characteristic during the training phase.

In accordance with [33], when the data set is relatively small, such is the case with the 102 benchmark problem instances, a general rule of thumb is to set the number of nodes in the SOM equal to the number of observations in the data set. In this research, a (12×8) hexagonal SOM grid was used, as this resulted in the lowest topographical errors in the map.

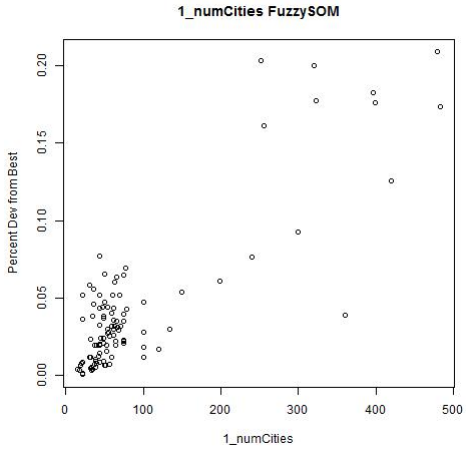
Three topographical error measurements were assessed, and these error measurements are



(a) Clarke and Wright Algorithm



(b) Sweep Algorithm



(c) Fuzzy SOM Algorithm

Figure 31: Scatter plots of algorithm performance (percent deviation above best known solution) versus the number of cities in the problem

calculated with the *kohonen* package in *R*. The first error measurement is the average distance between all pairs of the most similar codebook vectors, where the distance is measured in terms of the euclidean distance between the (x, y) coordinates of the nodes. The second measurement of topographical error is the average distance, in terms of the euclidean distance between the (x, y) coordinates, between the best matching and second best matching nodes for each data point. The last topographical error measurement finds the average difference between a data point and the codebook vector it is mapped to. For each of these three error measurements, a lower error was found for the (12×8) SOM grid than was found in both the (9×10) and (10×10) grid. It is important to note that due to the stochastic nature of the SOM, each training will result in a different output, therefore multiple runs were conducted with each SOM.

Various plots are available to interpret the trained SOM. Figure 32 represents the codebook vectors of the trained SOM nodes. Each node of the SOM consists of a 23-dimensional codebook vector, representing the 23 problem characteristics. The line graphs in each node in Figure 32 illustrates the trained node's value for each of the 23 problem characteristics. Figure 33 represents the average distance between each node and its neighbors, where the distance between neighbors is measured as the euclidean distance between their codebook vectors. In Figure 33, nodes that are very similar to their neighbors are colored red, whereas nodes that are less similar to their neighbors are yellow. How each input vector is mapped to the trained SOM is represented in Figure 34. Each node in Figure 34 is colored to represent the number of input vectors that are mapped to the SOM node. Each input vector is mapped to the SOM node with the most similar codebook vector, and it is clear from Figure 34 that not all SOM nodes have input vectors mapped to them.

Once the SOM is trained, the *k-means* clustering algorithm is used to cluster the SOM nodes. Nodes are compared using their codebook vectors, and different values of k are used to compare the within-cluster sum of squares (WCSS) for each of the k values. The plot corresponding to the WCSS for each value of k is found in Figure 35, and the common standard of finding the 'elbow' in the curve in order to identify the best value of k is used. The SOM is then clustered with $k = 6$ clusters, and the resulting clusters are found in Figure 36.

In order to interpret the results of the trained SOM, it is often useful to map the values of each component of the codebook vector onto the SOM. By looking at each component individually in the trained SOM, it might become apparent how different components (in this case, problem characteristics) affect the organization of the SOM. With the final trained SOM, the values of

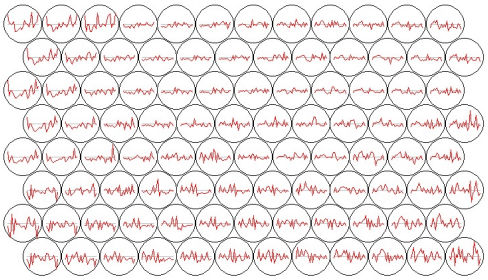


Figure 32: The codebook vectors of the trained SOM based on the 102 benchmark problem sets

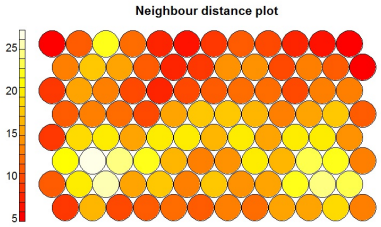


Figure 33: Plot of the average distance between a node and its neighbors

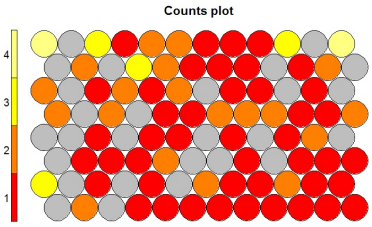


Figure 34: Plot of the number of input vectors that are mapped to each node

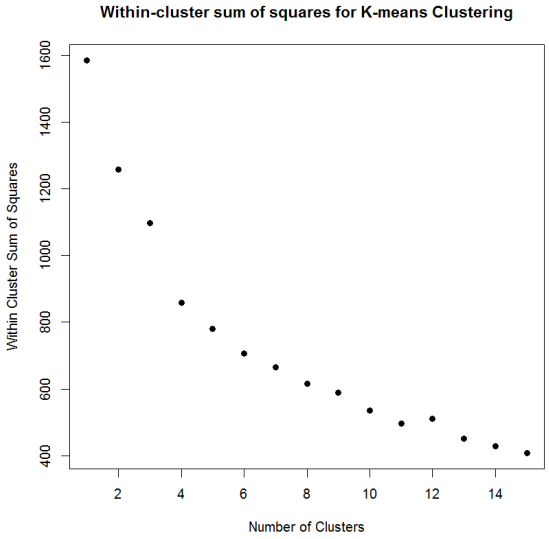


Figure 35: K means plot to identify the 'elbow' in the curve

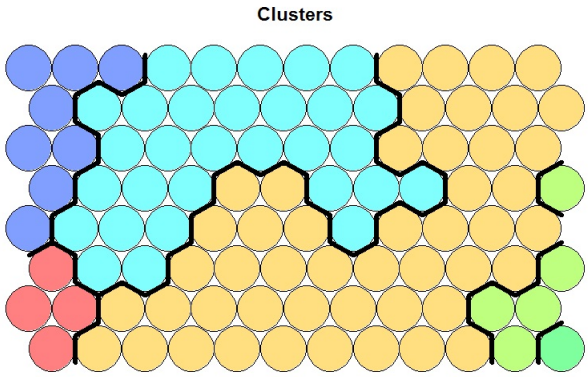


Figure 36: Trained SOM with six clusters

component are overlaid onto the trained grid of nodes. By overlaying the values of one component at a time, the grid can then be inspected to determine how the SOM codebook vectors are clustered according to this one particular component. For instance, looking at Figure 37, it is clear that the problems with the highest *number of cities* were mapped very close to one another on the trained SOM. This indicates that the differences in the number of cities amongst all problems had a strong influence on the SOM training phase. Figures 38 and 39 show that the *number of clusters*, and the *ratio of the number of clusters to the total number of cities* also strongly influenced the SOM training phase. Examining Figure 40, however, the values of the *y-coordinate of the depot* vary across the entire SOM, and there is no obvious clustering amongst the values of this problem characteristic. This lack of clustering of the values of the *y-coordinate of the depot* indicates that this problem characteristic did not strongly influence the training phase.

Finally, two additional pieces of information are mapped to the trained SOM. The 102 benchmark problem instances are broken into 7 sets, {A, B, P, M, F, E, K}. The sets {A,B,P} correspond to the benchmark problems by Augerat et al [41]; the set {M} corresponds to the benchmark problems by Mingozzi et al [39]; the set {F} are the benchmark problems from Fisher [40]; the set {E} refers to the benchmark problems by Christofides and Eilon [38]; and the set {K} refers to the benchmark problems by Golden et al [42]. Each benchmark problem is denoted by the letter of the set it belongs to. These 102 letters are then mapped to the trained SOM.

Figure 41 illustrates the mapping of the problems to the trained SOM. This mapping of the problems allows for an interpretation of the similarities of the problems within a set, as compared to the problems across all sets. Looking at Figure 41, the problems in the {K} set are mapped closely to one another in the upper left corner of the map. This suggests strong similarities amongst these problems. The problems in set {P} , however, appear in various places across the map. This suggests that the problem characteristics within the {P} set have more diverse problem characteristics as compared to the sets which are tightly clustered in the map.

The second piece of information mapped to the trained SOM is the best performing algorithm for each problem. The algorithm performance is measured by the algorithm with the lowest PDB for each of the benchmark problems, therefore, the symbol on the map indicates which algorithm performed best for the benchmark problem mapped to the location of the symbol. Regarding the symbols, the circle indicates the Clarke and Wright algorithm, the triangle indicates the Sweep algorithm, and the plus sign indicates the Fuzzy SOM algorithm. Overlaying the best performing

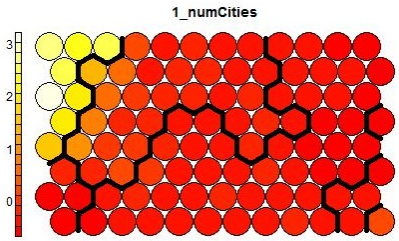


Figure 37: SOM component plane: number of cities

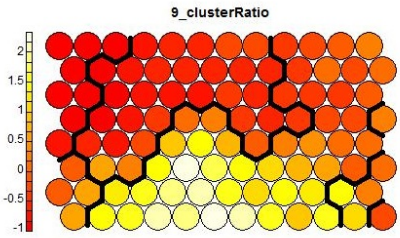


Figure 38: SOM component plane: ratio of the number of clusters to the number of cities

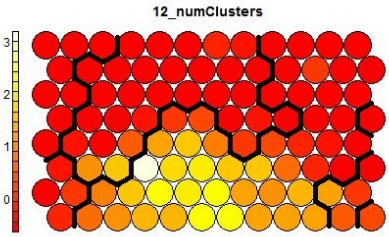


Figure 39: SOM component plane: number of clusters

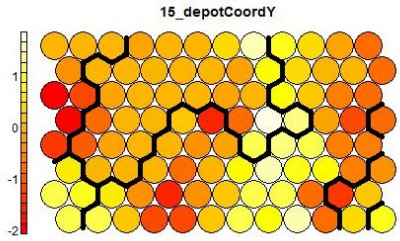


Figure 40: SOM component plane: y-coordinate of the depot

algorithm on the trained SOM offers insight into whether certain algorithms performed better on certain types of problems. Examining the graph, it is useful to look for regions where one algorithm's performance surpassed the others. For instance, looking at the upper left hand corner of Figure 42, it is clear that the Clarke and Wright performed better on these problems as compared to the other two algorithms. From this insight, it is useful to then look at the problems characteristics of the problems mapped to this upper left corner. These problems are the problems from the set of {K} benchmark problems, and looking at Figure 37, it is clear that the problems in this upper left hand corner have the largest number of cities. This suggests that the Clarke and Wright algorithm outperforms both the Sweep and Fuzzy SOM algorithms for problems with a large number of cities.

5.6.2 Using a SOM for Algorithm Performance Prediction

The purpose of generating the 98 new problem instances is to have a large enough data set for training and testing the SOM. The intent of this research is to assess the SOM's prediction performance on problems *similar* to the existing benchmark problem set, hence the use of the existing benchmark problems to generate the new problems. Histograms of each problem characteristic for the new problem set is compared to the histograms for the same problem

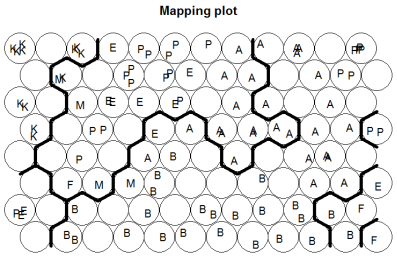


Figure 41: 102 benchmark problems mapped to the trained SOM

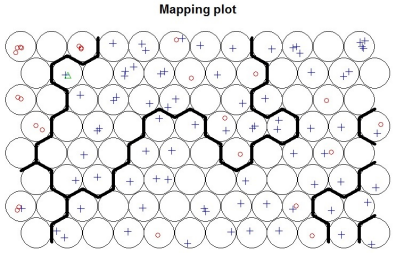


Figure 42: Best performing algorithm for each problem instance mapped to the trained SOM where circle=C& W, triangle=Sweep, and plus sign=Fuzzy SOM

Table 8: Number of Problems an Algorithm Performed Best

Clarke &Wright	Sweep + 2-opt	Fuzzy SOM
45	6	149

characteristics in the benchmark problem set. These histograms are found in Appendix B, and suggest that the set of newly generated problems is similar to the existing benchmark problems.

Each of the 98 new problem instances is solved with the three algorithms: the Clarke and Wright, the Sweep plus 2-opt, and the Fuzzy SOM (see Appendix C), in order to identify the best performing algorithm for each problem. Table 8 summarizes the breakdown of the number of problems where each algorithm was found to be the best performing. This information, as well as the 23 features extracted from these new problem instances is combined with the existing 102 benchmark problems. This set of 200 CVRP instances is used to train and test the accuracy of using a SOM for the prediction of the best performing algorithm based on a problem's characteristics.

To train both the unsupervised and supervised SOMs, the 200 problems are randomly divided into a *training set* and a *testing set*, each consisting of 100 CVRP instances. Prior to training the SOM, the *training set* of data is standardized so each dimension has a mean of zero and standard deviation of one. The *testing set* is then scaled with the same factors used to scale the *training set*. A (12×8) hexagonal SOM grid is again used, as these dimensions resulted in the lowest topographical error, using the three measures previously described.

For the unsupervised SOM, three figures are provided as an initial visualization of the trained SOM. Figure 43 illustrates the codebook vectors for each SOM node, representing how the values

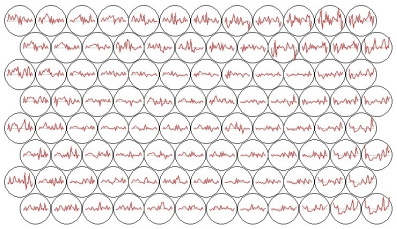


Figure 43: Codebook vectors of the trained SOM

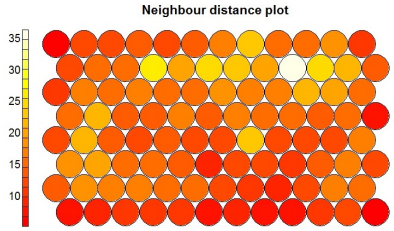


Figure 44: Average distance between a node and its neighboring nodes

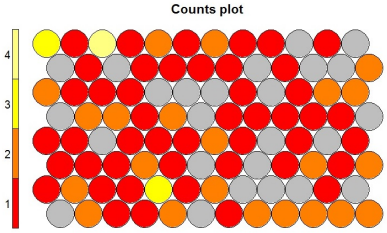


Figure 45: The number of input vectors that are mapped to each node

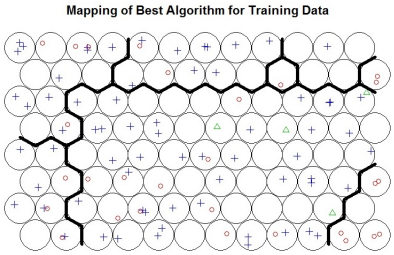


Figure 46: Mapping of the best algorithm for each of the problem instances in the *training set*

for each of the 23 components vary across the nodes. Figure 44 is the neighbor distance plot, where a red node indicates that this node is quite similar to it's neighboring nodes (measured by euclidean distance between codebook vectors), and a yellow or white node indicates that this node is rather different from its neighboring nodes. Finally, Figure 45 illustrates the number of input vectors (from the *training set*) that map to each node of the SOM. As with the SOM in the previous section, not all nodes on the grid have an input vector mapped to it. Additional graphs pertaining to clustering and component planes of this trained SOM are found in Appendix D.

Once trained, the unsupervised SOM is used to predict the best performing algorithm for each of the problem instances in the *testing set*. For each problem in the *training set*, the best performing algorithm is known, and this information is used for the prediction of new problems mapped to the same node. An overlay of the best performing algorithm for the *training set* of data is provided for the trained SOM in Figure 46. The results of the unsupervised SOM predictions for the *testing set* of problems are summarized in Table 9.

The supervised SOM is trained and tested in a similar fashion as the unsupervised SOM. Again, a (12×8) SOM grid is used, and the only difference between the supervised and unsupervised SOMs is the inclusion of the dependent variable Y during the training phase. Once trained,

Table 9: Unsupervised SOM Predicted Best Algorithm Versus the Known Best Algorithm for the *Testing Set*

	Unsupervised SOM - Predicted Best Algorithm		
Known Best Algorithm	Clarke & Wright	Sweep + 2-opt	Fuzzy SOM
Clarke &Wright	10	2	5
Sweep + 2-opt	0	0	2
Fuzzy SOM	29	2	50

Table 10: Supervised SOM Predicted Best Algorithm Versus the Known Best Algorithm for the *Testing Set*

	Supervised SOM - Predicted Best Algorithm		
Known Best Algorithm	Clarke & Wright	Sweep + 2-opt	Fuzzy SOM
Clarke &Wright	10	0	7
Sweep + 2-opt	0	0	2
Fuzzy SOM	14	1	66

however, the supervised SOM only uses the independent variables X to map a new problem instance to the trained supervised SOM map. The results of the supervised SOM predictions for the *testing set* of problems are summarized in Table 9.

Examining Table 9, the unsupervised SOM's prediction of the best performing algorithm on the *testing set* of data was accurate on 60 out of the 100 problems in the *testing set*. However, the supervised SOM's prediction was accurate on 76 out of 100 problems. In order to assess the effectiveness of both the unsupervised and supervised SOM for prediction, it is critical to be explicitly clear about the population of problems for which the trained SOM is making predictions. As previously discussed, generalizing results beyond the scope of a study is likely not accurate.

The SOM's prediction performance is restricted to problems which are similar to the original benchmark problems. Section 5.6.1 provides an overview of the range of problems characteristics found within the 102 benchmark problems, and the graphs in Appendix B show that the newly generated problems have similar characteristics to these known benchmark problems. Therefore, the assessment of this SOM's prediction performance is limited to CVRP problems with similar problem characteristics.

For the 200 problems, the best performing algorithm amongst the three utilized in this research is known. Therefore, if this set of 200 problems is a sample from a larger population of similar CVRP instances, it is reasonable to estimate the proportion of problems in the population

Table 11: Unsupervised SOM Prediction Performance

	22.5%	3%	74.5%
Unsupervised SOM Prediction	Clarke & Wright	Sweep + 2-opt	Fuzzy SOM
picked the best performing algorithm	$\frac{10}{17}$	$\frac{0}{2}$	$\frac{50}{81}$
did not pick best performing	$\frac{7}{17}$	$\frac{2}{2}$	$\frac{31}{81}$

Table 12: Unsupervised SOM Prediction Performance

	22.5%	3%	74.5%
Supervised SOM Prediction	Clarke & Wright	Sweep + 2-opt	Fuzzy SOM
picked the best performing algorithm	$\frac{10}{17}$	$\frac{0}{2}$	$\frac{66}{81}$
did not pick best performing	$\frac{7}{17}$	$\frac{2}{2}$	$\frac{15}{81}$

for which each algorithm is best performing by using the sample data. Table 8 provides the number of problems in the set of 200 where each algorithm performed best. These are used to estimate the population proportion of problems which are solved best by each algorithm.

Table 11 breaks down the estimated percentage of problems for the given population, where each algorithm is the best performing. This percentage is estimated by the sample of 200 problems examined. In addition, Table 11 provides the number of times that the unsupervised SOM predicted correctly and incorrectly for each type of problem (in this case, *type of problem* refers to a problem where it is known that the Clarke and Wright Algorithm performs best, or where the Sweep + 2-opt performs best, etc). Table 12 summarizes the number of times the supervised SOM predicted correctly for each type of problems. These observations from the *testing set* of problems is used to estimate the true probability of the SOM predicting correctly for each problem type.

The law of total probability is used to determine the probability that the SOM makes the correct prediction, given a new problem instance from the described population. This probability is calculated for both the supervised and unsupervised SOMs.

First, for the unsupervised SOM, the event of interest is defined as X_1 : {the unsupervised SOM correctly predicts the best performing algorithm}. Three additional

events are defined for ease of notation:

A :{the Clarke and Wright Algorithm is the best performing algorithm}

B :{the Sweep + 2-opt is the best performing algorithm}

C :{the Fuzzy SOM is the best performing algorithm}

Using the law of total probability, the $Pr(X_1)$ is defined as follows:

$$Pr(X_1) = Pr(X_1|A)Pr(A) + Pr(X_1|B)Pr(B) + Pr(X_1|C)Pr(C)$$

And now using the observations found in Table 11:

$$Pr(X_1) = \frac{10}{17}(.225) + \frac{0}{2}(.03) + \frac{50}{81}(.745) = .592$$

The probability that the unsupervised SOM will correctly predict the best performing algorithm is .592, when given a problem that is similar to the CVRP problems examined in this research.

This same calculation is completed for the supervised SOM. With the unsupervised SOM, the event of interest is defined as X_2 : {the supervised SOM correctly predicts the best performing algorithm}. The three additional events in Equation 5.6.2 are used for the supervised SOM calculation as well. Using the law of total probability, the $Pr(X_2)$ is defined as follows:

$$Pr(X_2) = Pr(X_2|A)Pr(A) + Pr(X_2|B)Pr(B) + Pr(X_2|C)Pr(C)$$

And now using the observations found in Table 12:

$$Pr(X_2) = \frac{10}{17}(.225) + \frac{0}{2}(.03) + \frac{66}{81}(.745) = .740$$

Finally, if predicting the best algorithm is considered as a Bernoulli experiment, X_1 and X_2 are each considered to follow a Bernoulli Distribution. The probability of success for each distribution follows the previously found estimates, $Pr(X_1) = .592$ and $Pr(X_1) = .740$, the expected number of successes (number of times the correct algorithm is chosen) correspond with these proportions: $E(X_1) = .592$ and $E(X_2) = .740$.

5.7 Discussion

The purpose of the first part of this research is to gain an understanding of the diversity of the existing CVRP benchmark problem instances. This is important for two reasons. First,

it is important for algorithm developers and end users to understand what types of problems an algorithm is being tested on. Both the NFL Theorem [2] and the premise of the algorithm selection problem [3] indicate that no one algorithm will outperform all other algorithms across all problems. Therefore, when creating or choosing a CVRP algorithm to use in practice, it is important to understand the characteristics of the problems that the algorithm will be required to solve. This knowledge is vital to ensuring that an appropriate set of problem instances is used for testing the algorithm's performance.

Examining the 23 histograms found in Figures .47 through .69, as well as the coefficients of variation found in Table 7, the 102 benchmark problems included in this study appear to be fairly diverse. There is no standard of what makes a data set diverse, and neither the histograms, nor the coefficients of variation fully explain what is going on within a data set. However, these measurements along with user knowledge of the problem characteristics can lead to an inference about the diversity of a given problem characteristic.

For many of the problem characteristics, the majority of problems fall within a much smaller subset of the entire range. For instance, looking at the histogram in Figure .47, the benchmark problems examined include problem instances with up to 500 cities. However, the majority of all problems in the set have fewer than 100 cities. This histogram clearly illustrates this, and suggests a lack of diversity for this characteristic amongst the problem set. However, the coefficient of variation for the number of cities indicates otherwise. Upon further inspection, however, it is apparent that reason the coefficient of variation for the number of cities is high is due to the fact that the average number of cities across all problems is so low. If the average number of cities was 400, for instance, with the same standard deviation, the coefficient of variation would be significantly lower.

The importance of identifying problem characteristics that might be restricted in some way has to do with the implications this has when testing algorithms. If an algorithm performs well on problems with fewer than 100 cities, this does not necessarily indicate that the algorithm will maintain high performance on bigger problems. With this benchmark set, if a practitioner knows that an algorithm will be routinely required to solve problems with more than 100 cities, the current 102 benchmark problem set is likely not sufficient for assessing the algorithm's performance.

The following characteristics from the original 23, 2) *standard deviation of the distance matrix*, 3), 4) (x, y) *coordinates of the problem centroid*, 5) *radius of the problem instance*, and 22) *area of the rectangle* are all characteristics that have a low coefficient of variation within

the benchmark problem set. Considering that all problems have been standardized to the same (400×400) region of the (x, y) plane, and the fact that these characteristics are significantly affected by the number of cities in the problem, this is not of great concern. In addition to these, the 17) *ratio of total demand to total capacity* also has a very low coefficient of variation. This is expected, however, due to the fact that the total capacity is found by first defining the 23) *minimum number of trucks*. The *minimum number of trucks* for a problem is found by $\lceil \frac{totaldemand}{vehiclecapacity} \rceil$, therefore, it makes sense that the ratio of the total demand to vehicle capacity is rather consistent across all problems. For the problems considered in this research, it is assumed that a CVRP algorithm will attempt to find a solution using the fewest number of vehicles; thus minimizing the amount of unused capacity in the problem.

One additional problem characteristics that appears restricted, is 12) *number of clusters* in a problem. Looking at Figure 28, it is apparent that the majority of problems in the set have fewer than three clusters. This is not necessarily a concern, unless a practitioner seeks to find a CVRP algorithm that performs well on problem instances with a large number of clusters. In this case, this benchmark problem set is likely not sufficient for assessing an algorithm's performance across problems with a high number of clusters.

Another indication of the diversity of the problem set is found by examining algorithm performance across all problems, since it is expected that no algorithm will have superior performance across all problems. Therefore, it is reasonable to expect variation in an algorithm's performance across a diverse set of problems. Examining the histograms in Figure 30, it is evident that no algorithm performed at its best or at its worst across all problems, which is an indication of the diversity in the problem set.

The scatter plots in Figures 31 through .92 offer insight into how the 23 problem characteristics relate to the performance of each algorithm. Inspecting these plots may offer insight into how certain problem characteristics affect an algorithm's performance. Figure 31 plots each algorithm's performance (measured by percent deviation above the best known solution) against the number of cities in each problem. The plots indicate that each algorithm did not perform as well on problems with a large number of cities as compared to the problems with a smaller number of cities. Looking at the results of the Fuzzy SOM specifically, the algorithm's performance seems to have a linear relationship to the problem size. As the number of cities in a problem increased, the performance of the Fuzzy SOM decreased. These plots indicate the negative impact of the problem's size on each algorithm's performance.

Figures .72 and .73 suggest that the location of the problem centroid affects algorithm performance. For each algorithm, it appears as though the more centered the centroid is in the plane, the worse the algorithm performs. With a centrally located centroid, it is indicative of a problem where the cities are more evenly distributed across the plane, which might result in fewer distinct distances in the distance matrix. Looking at Figure .75, for at least the Fuzzy SOM algorithm, fewer distinct distances decreases the algorithm's performance. Although the relationship between distinct distances and the other two algorithms is not as clear, the plots show that each of these algorithms had some of their worst performances on problems with few distinct distances.

For each of the 23 problem characteristic, the scatter plots of algorithm performance offers some insight into the effect of the problem characteristic on each algorithm's performance. Although the relationship between performance and problem characteristic is not always as clear for all algorithms as it is for some, there does appear to be some relationship between each problem characteristic and at least one of the algorithms. Even though the plots of only single characteristics are used, it is likely that further relationships between algorithm performance and the interactions between the problem characteristics exist.

The last part of the analysis of the diversity of the existing benchmark problem sets includes the analysis of the SOM used for exploratory data analysis. Examining the component plane graphs of the trained SOM provides an understanding of how the differences amongst the input vectors (benchmark problems in this case) affect the training of the SOM. For instance, there are multiple problem characteristics that clearly correlate to the trained SOM. Figure 37 shows how all of the problems with the highest number of cities are grouped in the upper left corner of the SOM, which indicates that the number of cities was a distinct characteristic across all problems. Figure .96, however, shows the component plane for the y-coordinate of the city centroid, and there appears to be little to no order of the y-coordinate values in the trained SOM, which indicates that this problem characteristic did not significantly contribute to differentiating between problems. Additional problem characteristics that appear to have significantly influenced the training of the SOM are: 1) *number of cities*, 6) *fraction of distinct distances in the distance matrix*, 9) *ratio of the number of clusters to the number of cities*, 12) *number of clusters*, 13) *mean radius of the clusters*, and 18) *ratio of max cluster demand to vehicle capacity.*

Inspecting Figure 41, it is clear that many of the benchmark problems are most similar to other problems within their own set, than they are to problems across all the sets of benchmark

problems. For example, all of the problems in the set {K} map to the upper left corner of the trained SOM, and for most of these problems in the upper left corner, the nearest problem is another problem from the same set {K}. The mapping of the problems in set {B} is not quite as exclusive as those in set {K}. This is due in part to the sheer number of problems in set {B}, but even though many of the problems in set {B} are mapped near each other, they are spread across a large portion of the map indicating that there are certainly distinguishable differences amongst the problems. The remaining sets, {A,P,E,M,F} are more spread across the trained SOM, indicating a stronger diversity in their problem characteristics.

Finally, Figure 42 illustrates how the best performing algorithm for each benchmark problem corresponds to the trained SOM. Ideally, there will be clear areas of the map where one algorithm outperforms another. This would indicate that the problem characteristics used to train the map are effective at distinguishing between algorithm performance. Although there are some distinct areas where one algorithm outperformed another, it is not strong across the entire map.

The Fuzzy SOM was the best performing algorithm on the majority of the benchmark problems, however, there are a couple of areas on the map where one algorithm appears to have dominated in performance. For instance, the Clarke and Wright algorithm clearly performed best on the problems corresponding to the upper left corner of the map, and from the component planes, this region of the map corresponded to problems with a large number of cities. The Fuzzy SOM performed best on the problems that mapped to the lower right corner, as well as most problems in the upper right corner, however, correlating any specific problem characteristics to these two regions is not as straightforward. The problems in the lower and upper right hand corners appear to correlate most strongly with 8) *coefficient of variation of the Nearest Neighbor* and 16) *standard deviation of demand*. Cross-referencing the graphs like this provides some preliminary indication as to how different problem characteristics help distinguish between performance of the three algorithms.

The second part of this research assesses the effectiveness of both a supervised and unsupervised SOM for the prediction of which algorithm will perform the best given a CVRP instance. Of the 200 CVRP instances included in this experiment, the Fuzzy SOM performed best on 74.5% of the problems, Clarke and Wright performed best on 22.5% of the problems, and the Sweep plus 2-opt algorithm performed best on 3% of the problems. Problems where the Clarke and Wright algorithm is best performing are now referred to as *Clarke and Wright problems*; *Fuzzy SOM problems* refer to the problems where the Fuzzy SOM algorithm was best performing; and

Sweep problems refer to the problems where the Sweep plus 2-opt algorithm was best performing.

Starting with the 102 benchmark problem sets and generating the subsequent 98 problems from these, the pertinent CVRP problem population for this prediction SOM is restricted to the problem instances which are similar to the original benchmark problem set. Therefore, it is reasonable to view the 200 problems as a sample from a larger population, and the proportions of each type of problem was estimated using this problem set.

Because the performance of the Fuzzy SOM dominated the other two algorithms across all 200 problems, if no information about new problem instances is known, a reasonable strategy is to solve all new problem instances with the Fuzzy SOM algorithm. With this strategy, the Fuzzy SOM algorithm will be the best performing algorithm approximately 74.5% of the new problem instances, based on the assumption that the proportions of problems in the sample set are assumed to be reasonable estimates of the proportions in the larger population.

The purpose of training a SOM for prediction, however, is to use information about a new problem to make a better decision about which algorithm to use for solving. For the current experiment, if the prediction model can choose the best algorithm with better performance than the simple strategy of always choosing the Fuzzy SOM, then using the prediction model is the superior strategy.

The results of the unsupervised SOM for prediction indicate that this mapping does not provide a superior strategy. In fact, by using the unsupervised, prediction SOM for algorithm selection, the best algorithm will only be chosen for approximately 59% of the new problem instances. This is significantly lower than the simple strategy of always choosing the Fuzzy SOM algorithm. With regard to the supervised, prediction SOM, however, the results of the experiment indicate that given a new problem instance from the same population as the training set, the supervised SOM will correctly choose the best performing algorithm for approximately 74% of these new problems. This performance metric is nearly identical to the simple strategy of always using the Fuzzy SOM.

Because the result so the supervised SOM and the simple strategy of always using the Fuzzy SOM are very similar, it does not necessarily indicate that these two strategies are interchangeable. In this experiment, the only measure to determine the best performing algorithm is the performance of each algorithm to one another. However, the time each algorithm requires to reach the solution is extremely important. It is possible that one algorithm might perform slightly better than the other two, however, it might require twice as much time. The measure

of algorithm run-time was beyond the scope of this research, but could provide another metric for performance when developing a prediction model for algorithm performance.

It is not clear which aspect of the algorithm selection problem was insufficient in this case. As analyzed in the first part of this section, the 23 problem characteristics seem to offer sufficient distinction amongst the problems. It is more likely that the problem is in the choice of the three algorithms used in this implementation of the algorithm selection problem. The best performing algorithm for each of the benchmark problems was overlayed on the trained SOM in the previous section, and there were not clear distinctions amongst the best performing algorithm across the entire map.

These three algorithms were chosen for their similarity, all being constructive CVRP heuristics. A more robust method for choosing the algorithms to include in the prediction model might be found after careful analysis of problem characteristics and algorithm performance. Good algorithm candidates would be consist of algorithms where there was clear performance differences amongst all of the problems.

5.8 Conclusions and Future Work

The first part of this research investigated the diversity of 102 existing benchmark problem sets from the CVRP literature. From this investigation, the 1) *number of cities*, as well as the 12) *number of clusters* provide clear limitations on range of problems within this problem set. However, due to the fact that many of the other problem characteristics are related to one another, distinct limitations for these problem characteristics are not clear. For CVRP instances with fewer than 500 cities, with only a few clusters, this existing benchmark problem set appears to be diverse. The importance of this diversity is paramount for practitioners who seek to devise a CVRP algorithm that performs well either across a broad range of problems, or perhaps across a vary narrow range of problems. For either case, it is extremely important that the algorithm's performance is tested across an appropriate set of CVRP instances.

Ideally, one seeks to develop an algorithm that consistently performs well across a broad range of problems. However, the NFL Theorem and Algorithm Selection Problem literature indicate that this is not realistic. Identifying the relationship between problem characteristics and an algorithm's performance is a field of research in itself. This is precisely what the second part of this research focused on, testing the Self Organizing Map as a prediction model. The results indicate, based on the given population of problems, and three chosen algorithms, the

supervised SOM will accurately predict the best performing algorithm approximately 74% of the time, whereas the unsupervised SOM will make an accurate prediction only 59% of the time.

Neither of these prediction models is superior to the simple strategy of always using the Fuzzy SOM algorithm for problems from the same CVRP population. However, by incorporating additional measurements of algorithm performance, such as the time required to find the solution, it is likely that the simple strategy of choosing the Fuzzy SOM will not dominate. Future work will include the inclusion of different measurements of algorithm performance.

The 23 problem characteristics investigated in this research appear to provide distinct measures for CVRP instances. This research developed a strategy for extracting all of these problem characteristics given any Euclidean CVRP, therefore, future research is not limited to any specific set of CVRP instances. Future research can examine the statistical significance of these 23 problem characteristics for specific algorithms, by means of a designed experiment. Algorithm performance metrics can also be expanded to include both solution quality and computing time. These results can lead to a better understanding of the relationship between problem characteristics and the performance of specific algorithms, which can then be used to develop a robust prediction model.

List of References

[1] K. Smith-Miles and L. Lopes, "Measuring instance difficulty for combinatorial optimization problems," *Computers & Operations Research*, vol. 39, no. 5, pp. 875–889, 2012.

[2] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[3] J. R. Rice, "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

[4] D. W. Aha, "Generalizing from case studies: A case study," in *Proc. of the 9th International Conference on Machine Learning*, 1992, pp. 1–10.

[5] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.

[6] K. A. Smith-Miles, "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, p. 6, 2008.

[7] L. Kotthoff, "Algorithm selection for combinatorial search problems: A survey," *AI Magazine*, vol. 35, no. 3, pp. 48–60, 2014.

[8] E. P. Tsang, J. E. Borrett, and A. C. Kwan, "An attempt to map the performance of a range of algorithm and heuristic combinations," *Hybrid Problems, Hybrid Solutions*, vol. 27, p. 203, 1995.

[9] E. Horvitz, Y. Ruan, C. Gomes, H. Kautz, B. Selman, and M. Chickering, "A bayesian approach to tackling hard computational problems," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 235–244.

[10] K. Leyton-Brown, E. Nudelman, and Y. Shoham, "Learning the empirical hardness of optimization problems: The case of combinatorial auctions," in *Principles and Practice of Constraint Programming-CP 2002*. Springer, 2002, pp. 556–572.

[11] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, and Y. Shoham, "A portfolio approach to algorithm selection," in *IJCAI*, vol. 1543, 2003, p. 2003.

[12] L. Xu, F. Hutter, H. H. Hoos, and K. Leyton-Brown, "Satzilla-07: the design and analysis of an algorithm portfolio for sat," in *Principles and Practice of Constraint Programming–CP 2007*. Springer, 2007, pp. 712–727.

[13] C. Bierwirth, D. C. Mattfeld, and J.-P. Watson, "Landscape regularity and random walks for the job-shop scheduling problem," in *Evolutionary Computation in Combinatorial Optimization*. Springer, 2004, pp. 21–30.

[14] P. Merz, "Advanced fitness landscape analysis and the performance of memetic algorithms," *Evolutionary Computation*, vol. 12, no. 3, pp. 303–325, 2004.

[15] P. F. Stadler and W. Schnabl, "The landscape of the traveling salesman problem," *Physics Letters A*, vol. 161, no. 4, pp. 337–344, 1992.

[16] P. Cheeseman, B. Kanefsky, and W. M. Taylor, "Where the really hard problems are." in *IJCAI*, vol. 91, 1991, pp. 331–340.

[17] J. N. Hooker, "Testing heuristics: We have it all wrong," *Journal of Heuristics*, vol. 1, no. 1, pp. 33–42, 1995.

[18] K. E. Nygard, P. Juell, and N. Kadaba, "Neural networks for selective vehicle routing heuristics," *ORSA Journal on Computing*, vol. 2, no. 4, pp. 353–364, 1990.

[19] D. Tuzun, M. A. Magent, and L. I. Burke, "Selection of vehicle routing heuristic using neural networks," *International Transactions in Operational Research*, vol. 4, no. 3, pp. 211–221, 1997.

[20] J. I. van Hemert, "Property analysis of symmetric travelling salesman problem instances acquired through evolution," in *Evolutionary Computation in Combinatorial Optimization*. Springer, 2005, pp. 122–131.

[21] J. I. van Hemert, "Evolving combinatorial problem instances that are difficult to solve," *Evolutionary Computation*, vol. 14, no. 4, pp. 433–462, 2006.

[22] K. Smith-Miles, J. I. van Hemert, and X. Y. Lim, "Understanding tsp difficulty by learning from evolved instances." *LION*, vol. 4, pp. 266–280, 2010.

[23] F. Hutter, L. Xu, H. H. Hoos, and K. Leyton-Brown, "Algorithm runtime prediction: Methods & evaluation," *Artificial Intelligence*, vol. 206, pp. 79–111, 2014.

[24] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

[25] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[26] S. Kaski and T. Kohonen, "Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world," in *Neural Networks in Financial Engineering. Proceedings of the Third International Conference on Neural Networks in the Capital Markets*. World Scientific, 1996, pp. 498–507.

[27] J. W. Sammon, "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, no. 5, pp. 401–409, 1969.

[28] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

[29] T. Kohonen, "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

[30] T. Kohonen, "Clustering, taxonomy, and topological maps of patterns," in *Proceedings of the sixth international conference on pattern recognition*. Washington, DC: IEEE Computer Soc. Press, 1982, pp. 114–128.

[31] T. Kohonen, *Self-Organizing Maps*, 3rd ed. Springer-Verlag, 2001.

[32] T. Kohonen, "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

[33] S. Kaski, "Data exploration using self-organizing maps," in *ACTA POLYTECHNICA SCANDINAVICA: MATHEMATICS, COMPUTING AND MANAGEMENT IN ENGINEERING SERIES NO. 82*. Citeseer, 1997.

[34] A. Ultsch, "Self-organizing neural networks for visualization and classification," in *Information and Classification*, ser. Studies in Classification, Data Analysis and Knowledge Organization, O. opitz, B. Lausen, and R. Klar, Eds.

[35] M. Kraaijveld, J. Mao, and A. K. Jain, "A nonlinear projection method based on kohonen's topology preserving maps," *Neural Networks, IEEE Transactionis on*, vol. 6, no. 3, pp. 548–559, 1995.

[36] R. Wehrens and L. Buydens, "Self and super-organzing maps in r: the kohonen package," *Journal of Statistical Software*, vol. 21, no. 5, pp. 1–19, 2007.

[37] W. Melssen, R. Wehrens, and L. Buydens, "Supervised kohonen networks for classification problems," *Chemometrics and Intelligent Laboratory Systems*, vol. 83, no. 2, pp. 99–113, 2006.

[38] N. Christofides and S. Eilon, "An algorithm for the vehicle-dispatching problem," *OR*, pp. 309–318, 1969.

[39] N. Christofides, A. Mingozzi, and P. Toth, "The vehicle routing problem," in *Combinatorial Optimization*, N. Mingozzi, P. Toth, and C. Sandi, Eds. London: John Wiley and Sons, 1979.

[40] M. L. Fisher, "Optimal solution of vehicle routing problems using minimum k-trees," *Operations Research*.

[41] P. Augerat, J. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi, *Computational results with a branch and cut code for the capacitated vehicle routing problem*. IMAG, 1995.

[42] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results," in *Fleet management and logistics*. Springer, 1998, pp. 33–56.

[43] G. u. Clarke and J. W. Wright, "Scheduling of vehicles from a central depot to a number of delivery points," *Operations research*, vol. 12, no. 4, pp. 568–581, 1964.

[44] P. Toth and D. Vigo, *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications, 2002.

[45] B. E. Gillett and L. R. Miller, "A heuristic algorithm for the vehicle-dispatch problem," *Operations research*, vol. 22, no. 2, pp. 340–349, 1974.

[46] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, "Classical and modern heuristics for the vehicle routing problem," *International transactions in operational research*, vol. 7, no. 4-5, pp. 285–300, 2000.

CHAPTER 6

Conclusion and Future Work

6.0.1 Conclusion

For all $\mathcal{NP}$-hard combinatorial optimization problems, there is currently no method for solving these problems in polynomial time. Therefore, much of the research into solving these problems focuses on heuristic methods. Recognizing the importance of heuristics for these combinatorial problems, this research investigated the application of artificial neural networks, which are known to be successful at a myriad of applications, to these combinatorial problems. The specific focus was the application of the Self Organizing Map to the Vehicle Routing Problem.

The SOM is applied to the VRP into two distinct methods in this research, demonstrating the versatility of the algorithm. The first part of this research improves upon the existing SOM-inspired algorithm for solving the CVRP, and demonstrates the competitiveness of this heuristic with similar constructive heuristics. The second part of this research utilizes a SOM in a more traditional way, using the SOM to interpret the problem characteristics of existing benchmark problem sets, as well as assessing the performance of the SOM for predicting the best performing algorithm for a new CVRP instance.

Although the application of the SOM for solving the CVRP has not always been highly regarded in the literature, the performance of the Fuzzy SOM introduced in Chapter 4 demonstrates the effectiveness of this approach as a constructive heuristic. The incorporation of fuzzy logic for parameter control eliminates the need for parameter tuning. The performance of this algorithm suggests that future research can investigate combining the SOM with improvement heuristics, or perhaps combining ideas from the SOM with other CVRP heuristics, in order to find new approaches to solving the CVRP.

Using the SOM for exploratory data analysis of 102 existing CVRP benchmark problem instances provided insight into the significance of the 23 problem characteristics investigated. Interpreting the trained SOM map, combined with the component plane graphs as well as the overlaying of the problem names and algorithm performance, allows a deeper understanding of how each of these problem and algorithm characteristics are related. The visualization that the SOM provides suggests that the 102 problem set is diverse.

The supervised SOM was more effective than the unsupervised SOM for prediction. Since the supervised SOM uses additional information (the dependent variable Y) during the training

phase, it makes sense that this additional information led to superior results. The prediction model developed in Chapter 5 incorporates three CVRP heuristics, where the performance of the Fuzzy SOM dominated the other two algorithms. Based on the sample proportions of CVRP problems where the Fuzzy SOM had the best performance, the simplest strategy for choosing amongst the three algorithms is to always use the Fuzzy SOM. Based on the sample proportion of problems where the Fuzzy SOM was the best performing algorithm, this strategy is expected to result in the best performing algorithm being chosen 74.5% of the time. This is simply because the Fuzzy SOM was the best performing algorithm of 74.5% of the problems instances in the set.

Despite the disparity in algorithm performance (the Fuzzy SOM was the best performing algorithm on most of the problems), the supervised SOM did discern between algorithms when predicting performance. Based on the testing of the prediction model, combined with the estimated proportions of problems in the population of problems, the supervised SOM's expected accuracy of choosing the best performing algorithm is 74% of the time. Although this result is slightly worse than the simple strategy of always choosing the Fuzzy SOM, this result does indicate the ability of the SOM for modeling the relationship between problem characteristics and algorithm performance.

6.0.2 Future Work

Future work into the use of the SOM for solving the CVRP can investigate the hybridization of the algorithm with existing algorithms and improvement heuristics. The fact that the SOM's performance is competitive with existing constructive heuristics, indicates that performance will only be enhanced when coupled with improvement heuristics. Additionally, conducting a time study on the parallelized Fuzzy SOM algorithm can provide insight into how this algorithm compares with other well-known algorithms with regard to solution time.

The application of the SOM to the CVRP is strictly based on the utilization of Euclidean distance in the problem. Updating the SOM requires a measurement of distance, and the Euclidean distance between two points in a CVRP allows for the straightforward application of the SOM to this problem. In order to apply the SOM to additional combinatorial optimization problems, there must be some interpretation of the problem that allows for this distance measurement for the competition between, and updating of nodes in the SOM. Future work can develop methods for applying the SOM to other combinatorial optimization problems.

Future work will continue to explore the CVRP in the context of the algorithm selection

problem. Although the 23 problem characteristics appear to be effective features for understanding algorithm performance, their statistical significance can be investigated by developing a designed experiment and varying these characteristics. Due to the inter-relatedness of many of these features, it is not likely nor possible to vary each one of these characteristics independently, and careful consideration should be give to what features to vary.

The use of the SOM for prediction seems promising. Additional research can compare the effectiveness of the SOM with other prediction methods for algorithm selection with the CVRP. Also, in the proposed prediction model, algorithm performance was solely measured by solution quality, whereas another important measure of algorithm performance is solution time. Incorporating additional measures of algorithm performance will provide another interesting aspect of a prediction model.

Appendix A: Results of Solving the CVRP Benchmark Problems Instances

	% deviation above the best known solution		
Problem Name	Clarke & Wright	Sweep	Fuzzy SOM
E-n22-k4	3.67	6.21	.07
E-n23-k3	9.15	2.12	.13
E-n33-k4	.97	5.34	1.20
E-n51-k5	12.21	2.10	.69
E-n76-k7	8.17	4.11	2.20
E-n76-k8	8.13	5.43	2.07
E-n76-k10	8.47	8.77	3.53
E-n76-k14	5.14	11.42	6.52
E-n101-k8	8.81	6.16	1.82
E-n101-k14	6.36	11.30	4.73
M-n101-k10	1.65	13.07	1.16
M-n121-k7	3.30	23.06	1.72
M-n151-k12	11.67	8.24	5.37
M-n200-k16	9.56	11.43	6.12
A-n32-k5	7.61	12.81	5.84
A-n33-k5	7.72	5.64	2.32
A-n33-k6	4.62	1.37	.44
A-n34-k5	4.17	1.08	.56
A-n36-k5	3.69	7.97	3.82
A-n37-k5	5.80	10.51	5.55
A-n37-k6	2.91	14.67	4.63
A-n38-k5	5.22	12.11	.72
A-n39-k5	9.73	8.20	.95
A-n39-k6	3.86	8.32	.52
A-n44-k6	4.17	12.99	1.96
A-n45-k6	6.62	13.75	3.23
A-n45-k7	4.71	14.46	1.47
A-n46-k7	2.82	4.94	2.39
A-n48-k7	3.71	7.42	4.44
A-n53-k7	8.86	8.18	1.59
A-n54-k7	2.93	14.60	1.99
A-n55-k9	2.50	10.61	2.75
A-n60-k9	5.01	11.12	1.19
A-n61-k9	6.60	11.67	5.22
A-n62-k8	5.03	9.97	2.63
A-n63-k9	4.45	12.64	3.01
A-n63-k10	2.93	10.28	3.56
A-n64-k9	6.13	14.31	3.20
A-n65-k9	5.57	11.03	2.24
A-n69-k9	4.47	6.36	2.92
A-n80-k10	5.56	21.56	4.29
B-n31-k5	1.36	3.68	1.21
B-n34-k5	.80	12.95	.31
B-n35-k5	2.44	1.25	.41
B-n38-k6	3.37	4.22	1.97
B-n39-k5	3.23	11.84	1.05
B-n41-k6	8.33	6.88	1.94

	% deviation above the best known solution		
Problem Name	Clarke & Wright	Sweep	Fuzzy SOM
B-n43-k6	5.39	1.39	1.26
B-n44-k7	3.16	25.14	4.36
B-n45-k5	.82	11.50	7.73
B-n45-k6	7.35	8.67	5.22
B-n50-k7	1.05	6.33	.90
B-n50-k8	3.20	6.36	2.12
B-n51-k7	10.14	12.75	4.71
B-n52-k7	2.40	6.52	.68
B-n56-k7	3.78	12.50	2.82
B-n57-k7	8.37	11.96	.76
B-n57-k9	3.47	8.25	2.52
B-n63-k10	6.83	9.16	4.33
B-n64-k9	7.03	11.12	6.05
B-n66-k9	7.63	6.90	3.52
B-n67-k10	6.58	10.83	6.39
B-n68-k9	3.60	9.59	3.14
B-n78-k10	3.57	7.84	6.94
F-n45-k4	2.07	5.07	2.02
F-n72-k4	8.10	19.09	3.18
F-n135-k7	4.93	18.41	3.02
P-n16-k8	6.39	18.38	.43
P-n19-k2	12.21	8.03	.31
P-n20-k2	8.33	11.36	.66
P-n21-k2	11.94	.81	.81
P-n22-k2	10.88	.87	.86
P-n22-k8	.11	5.87	3.65
P-n23-k8	1.98	17.57	5.18
P-n40-k5	13.18	3.35	.81
P-n45-k5	12.34	2.14	.83
P-n50-k7	7.77	5.51	2.43
P-n50-k8	7.21	5.52	3.85
P-n50-k10	5.51	12.78	3.68
P-n51-k10	6.74	9.65	6.53
P-n55-k7	8.92	3.44	2.75
P-n55-k10	5.50	7.01	3.03
P-n55-k15	3.50	12.64	4.43
P-n60-k10	7.55	8.33	4.06
P-n60-k15	5.63	13.06	3.21
P-n65-k10	7.53	5.43	1.96
P-n70-k10	8.45	8.04	5.21
P-n76-k4	16.21	4.86	2.27
P-n76-k5	11.41	6.93	3.95
P-n101-k4	12.39	7.56	2.83
kelly9	14.47	20.87	16.11
kelly10	14.05	22.63	17.73
kelly11	15.36	21.77	17.61
kelly12	15.39	20.34	17.37
kelly13	11.15	18.34	20.31
kelly14	13.06	20.06	20.01

	% deviation above the best known solution		
Problem Name	Clarke & Wright	Sweep	Fuzzy SOM
kelly15	13.06	20.04	18.27
kelly16	10.12	20.04	20.89
kelly17	9.03	7.48	7.64
kelly18	7.45	17.60	9.29
kelly19	7.35	20.53	3.91
kelly20	8.03	25.12	12.59

Appendix B: Histograms of each problem characteristic

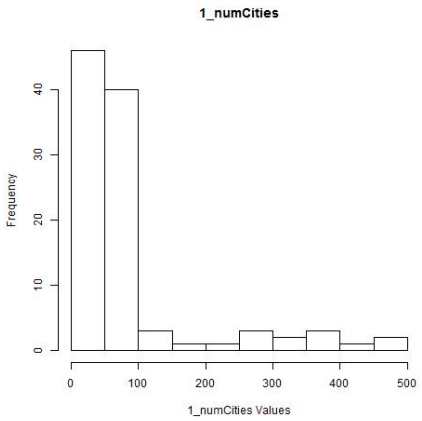


Figure .47: Number of Cities

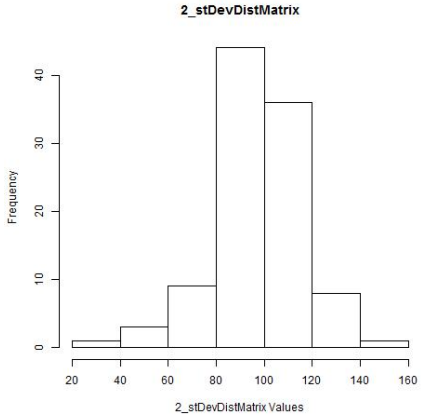


Figure .48: Standard Deviation of the Distance Matrix

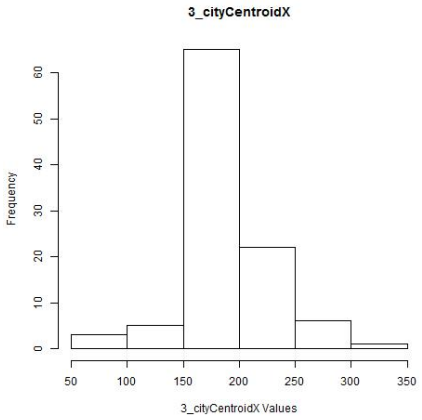


Figure .49: X-coordinate of City Centroid

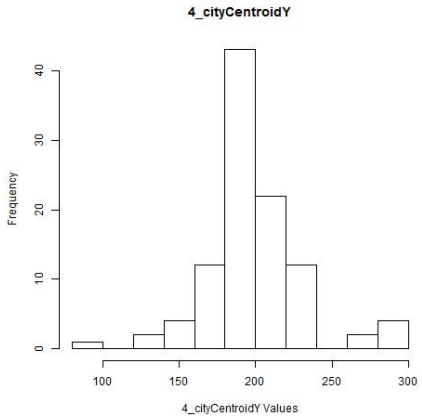


Figure .50: Y-coordinate of City Centroid

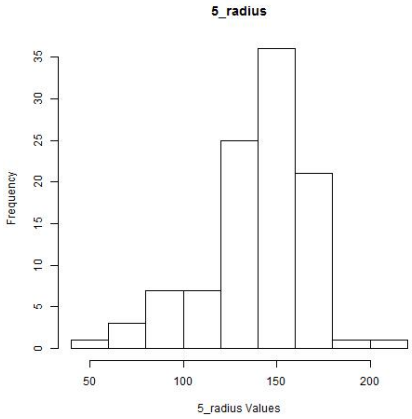


Figure .51: Radius of Problem Instance

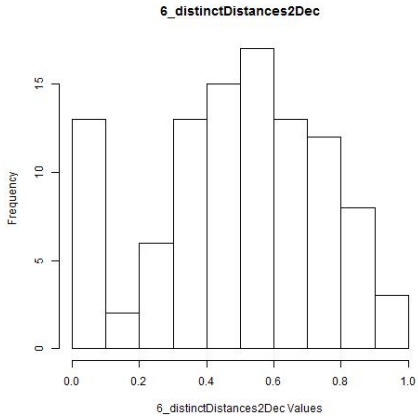


Figure .52: Proportion of Distinct Distances

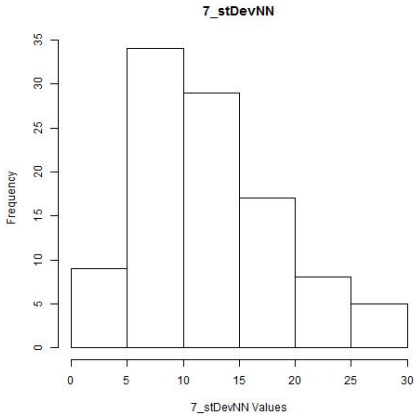


Figure .53: Standard Deviation of the Nearest Neighbor Distances

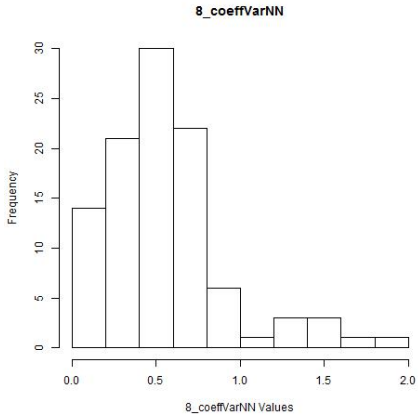


Figure .54: Coefficient of Variation for Nearest Neighbor Distances

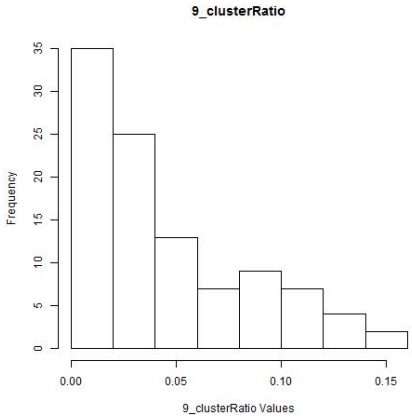


Figure .55: Ratio of the Number of Clusters to the Number of Cities

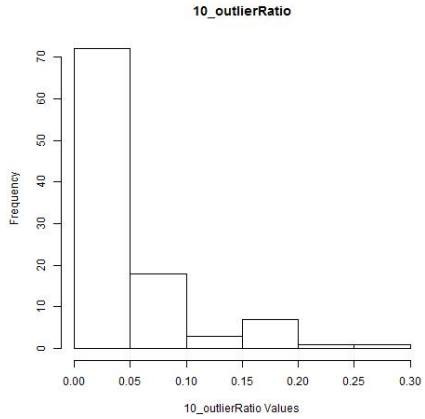


Figure .56: Ratio of the Number of Outliers to the Total Number of Cities

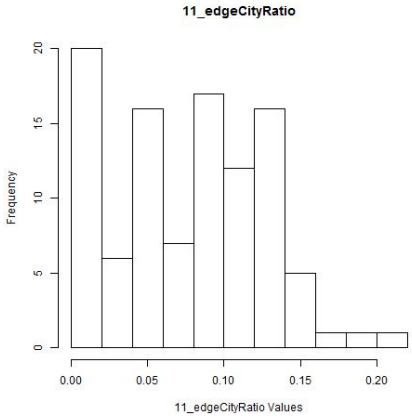


Figure .57: Ratio of the Number of Edge Cities to Total Number of Cities

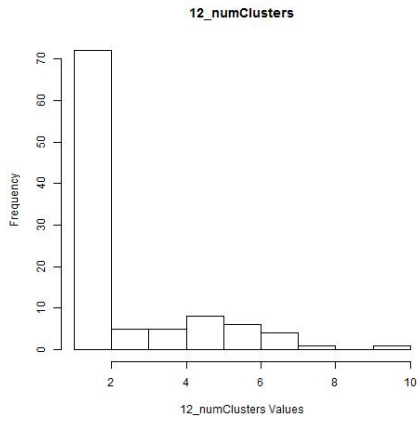


Figure .58: Number of Clusters

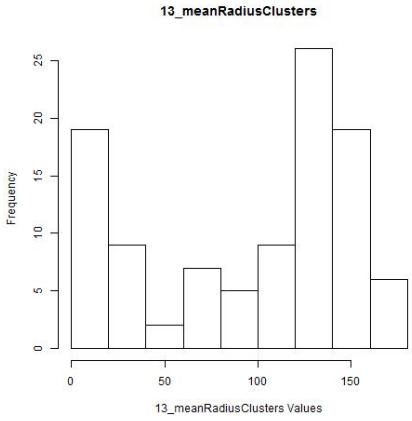


Figure .59: Mean Radius of the Clusters

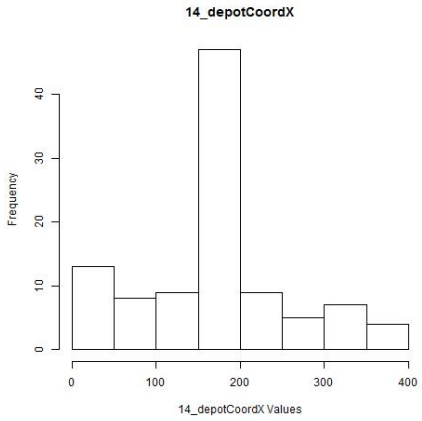


Figure .60: X-coordinate of the Depot

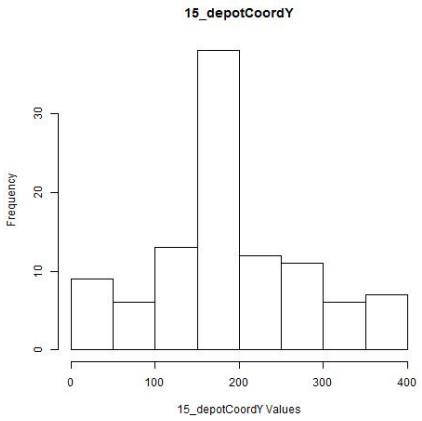


Figure .61: Y-coordinate of the Depot

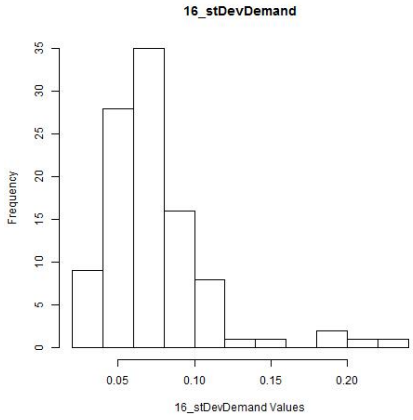


Figure .62: Standard Deviation of Demand

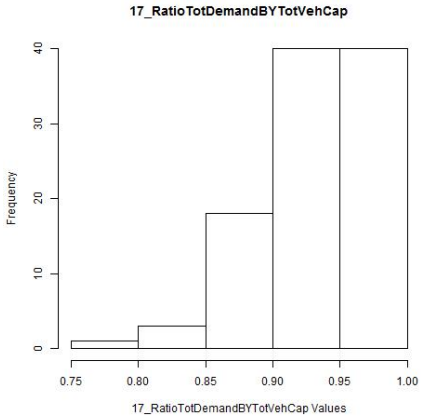


Figure .63: Ratio of Total Demand to Total Vehicle Capacity

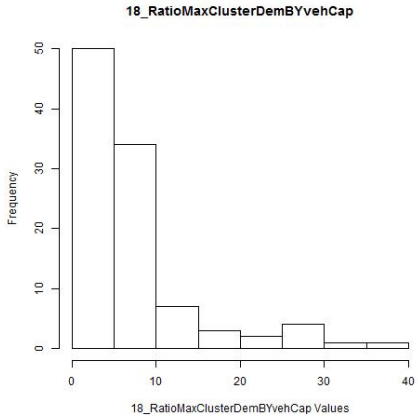


Figure .64: Ratio of the Max Cluster Demand to Vehicle Capacity

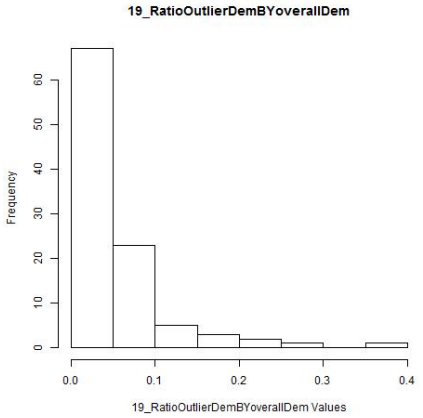


Figure .65: Ratio of Outlier Demand to Overall Demand

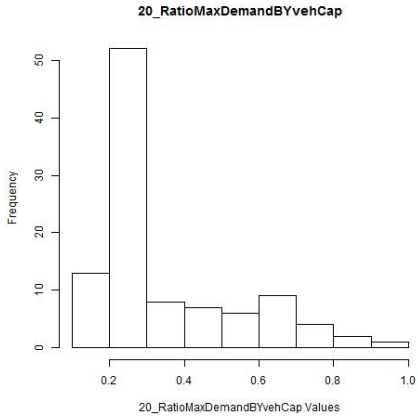


Figure .66: Ratio of the Maximum City Demand to Vehicle Capacity

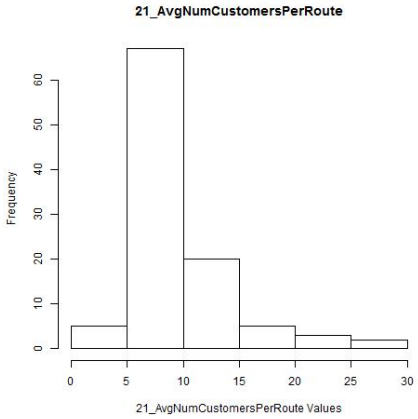


Figure .67: Average Number of Customers Per Route

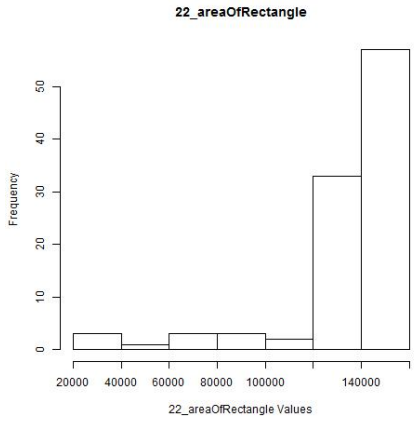


Figure .68: Area of the Rectangle

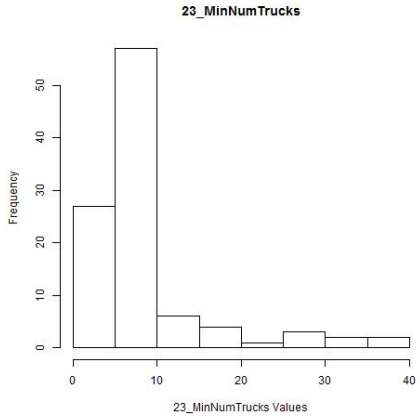
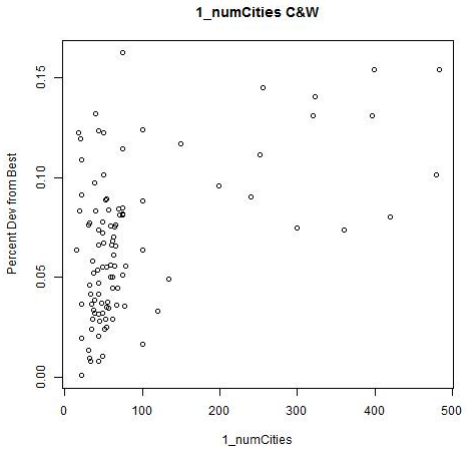
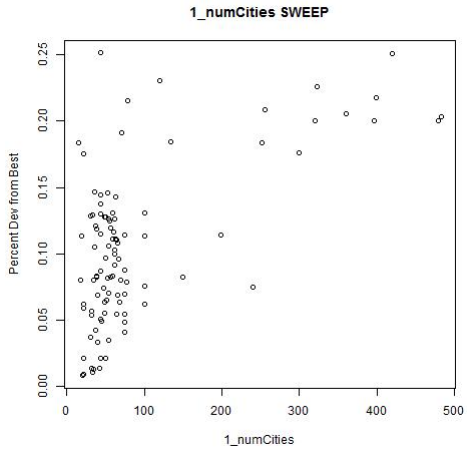


Figure .69: Minimum Number of Trucks

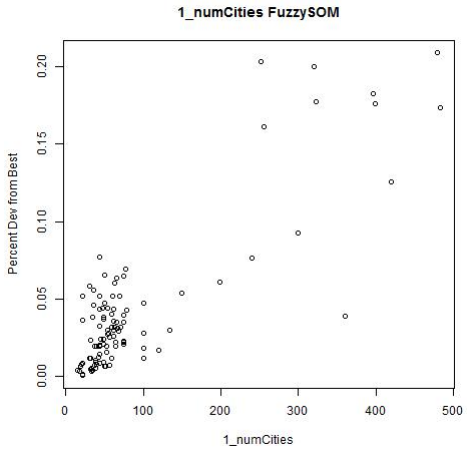
Appendix C: Scatter plots of algorithm performance (percent deviation above best known solution) versus each problem characteristic



(a) Clarke and Wright Algorithm

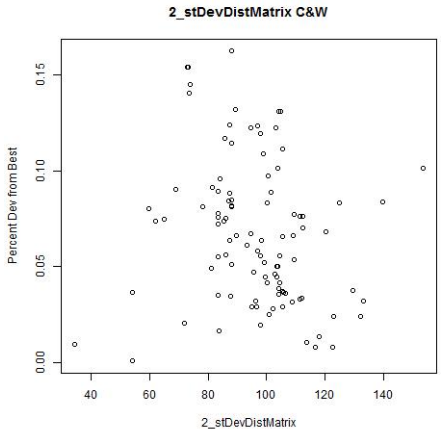


(b) Sweep Algorithm

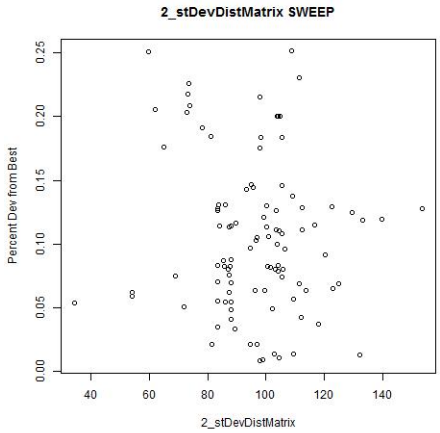


(c) Fuzzy SOM Algorithm

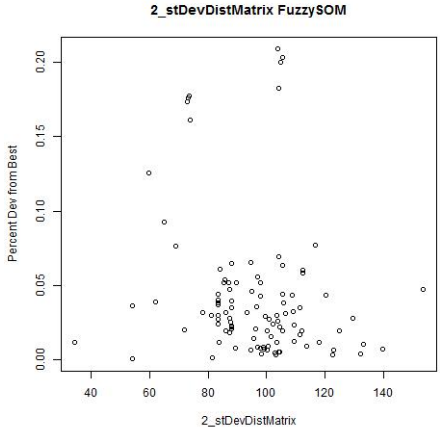
Figure .70: Scatter plots of algorithm performance (percent deviation above best known solution) versus the number of cities in the problem



(a) Clarke and Wright Algorithm

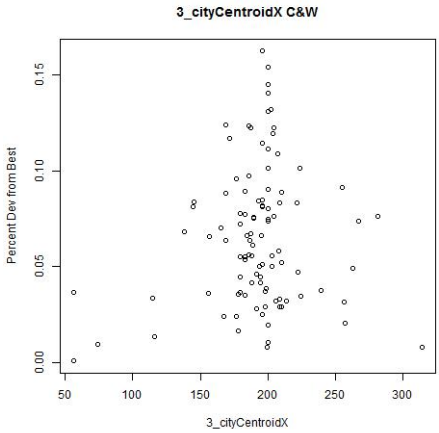


(b) Sweep Algorithm

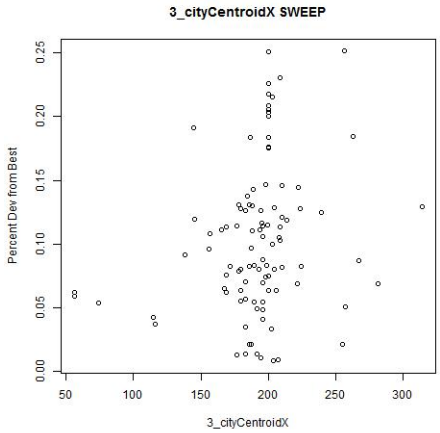


(c) Fuzzy SOM Algorithm

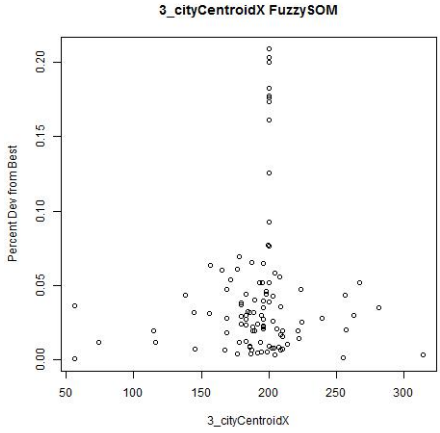
Figure .71: Scatter plots of algorithm performance (percent deviation above best known solution) versus the standard deviation of the distance matrix in the problem



(a) Clarke and Wright Algorithm

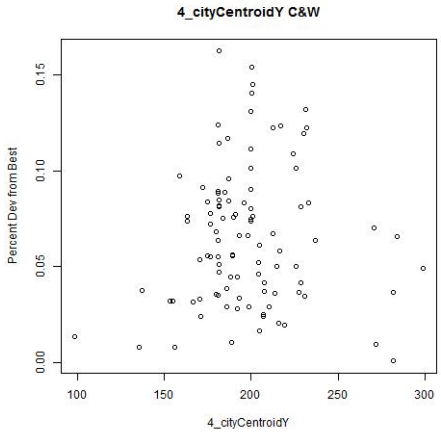


(b) Sweep Algorithm

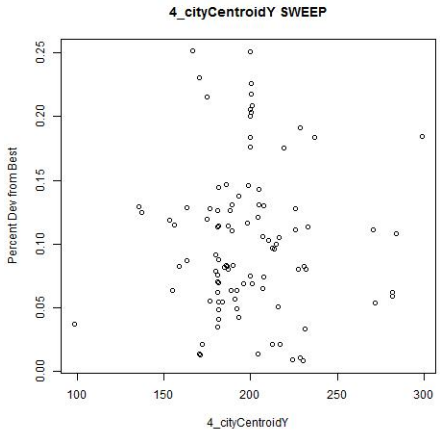


(c) Fuzzy SOM Algorithm

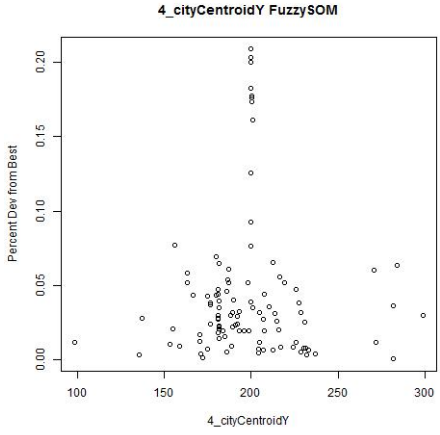
Figure .72: Scatter plots of algorithm performance (percent deviation above best known solution) versus the x coordinate of the instance centroid



(a) Clarke and Wright Algorithm

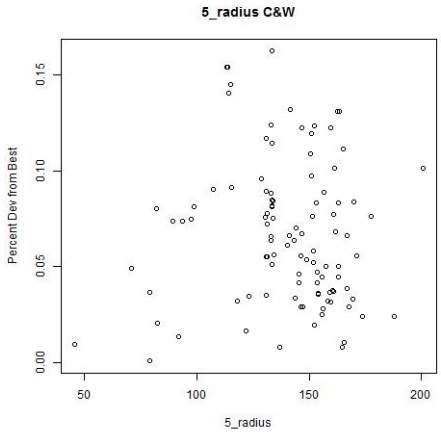


(b) Sweep Algorithm

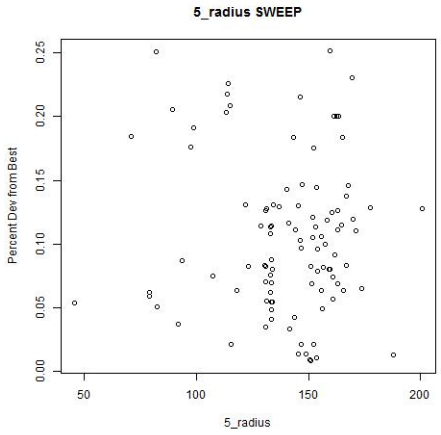


(c) Fuzzy SOM Algorithm

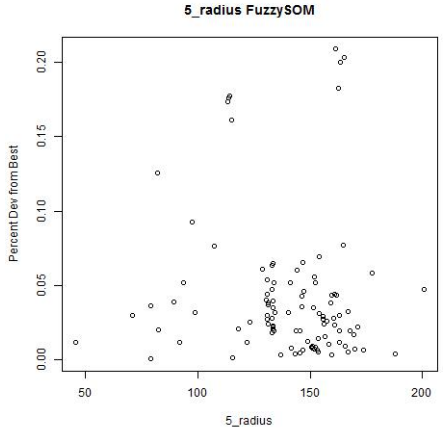
Figure .73: Scatter plots of algorithm performance (percent deviation above best known solution) versus the y coordinate of the instance centroid



(a) Clarke and Wright Algorithm

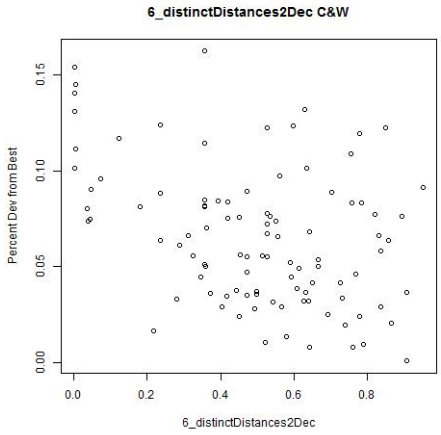


(b) Sweep Algorithm

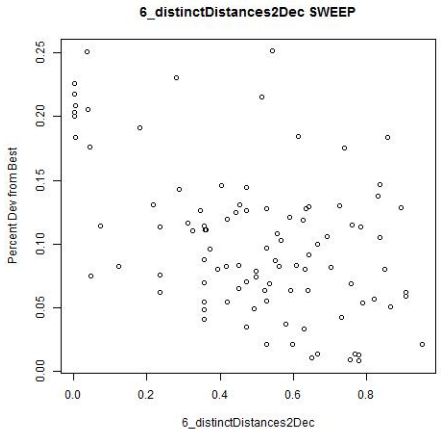


(c) Fuzzy SOM Algorithm

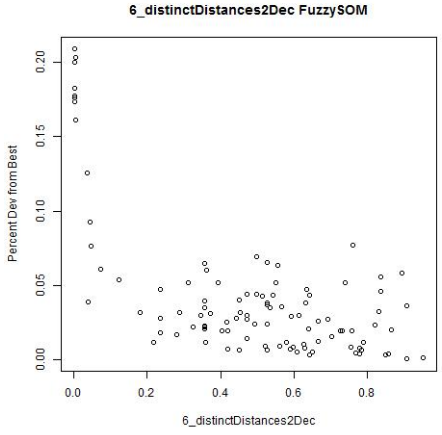
Figure .74: Scatter plots of algorithm performance (percent deviation above best known solution) versus the radius of the problem instance



(a) Clarke and Wright Algorithm

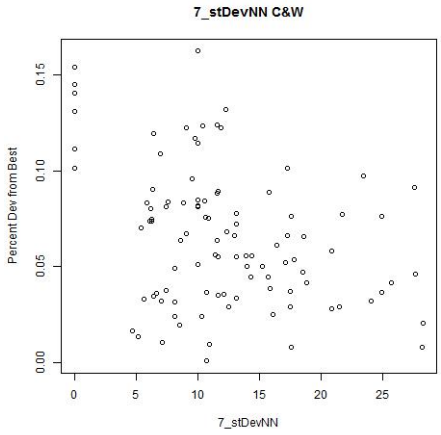


(b) Sweep Algorithm

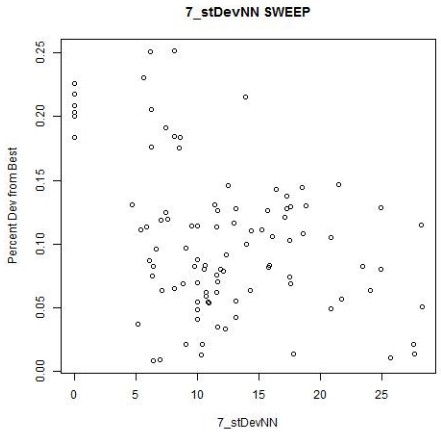


(c) Fuzzy SOM Algorithm

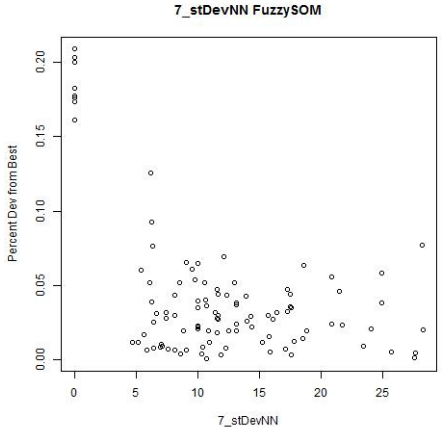
Figure .75: Scatter plots of algorithm performance (percent deviation above best known solution) versus the proportion of distinct distances in the distance matrix (rounded to two decimal values)



(a) Clarke and Wright Algorithm

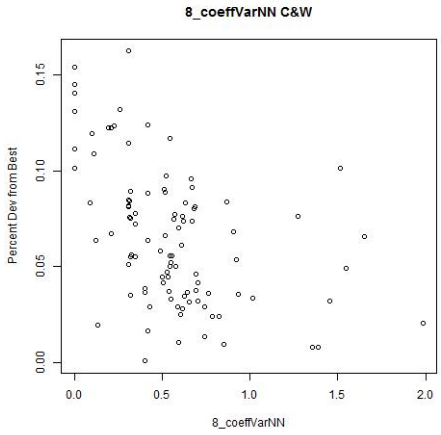


(b) Sweep Algorithm

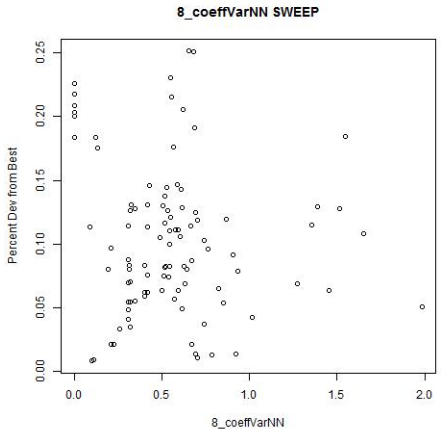


(c) Fuzzy SOM Algorithm

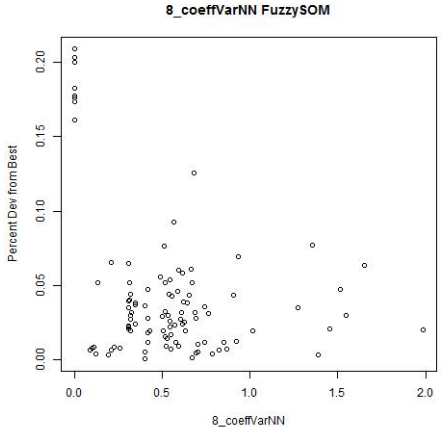
Figure .76: Scatter plots of algorithm performance (percent deviation above best known solution) versus the standard deviation of the nearest neighbors



(a) Clarke and Wright Algorithm

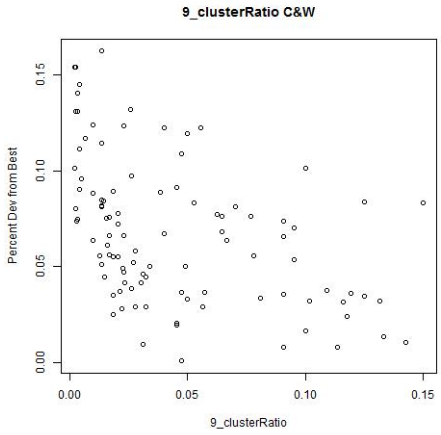


(b) Sweep Algorithm

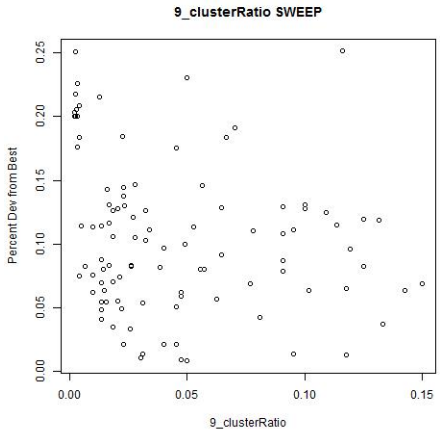


(c) Fuzzy SOM Algorithm

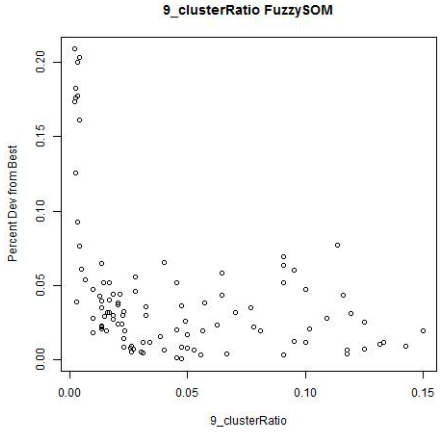
Figure .77: Scatter plots of algorithm performance (percent deviation above best known solution) versus the coefficient of variation of the nearest neighbors



(a) Clarke and Wright Algorithm

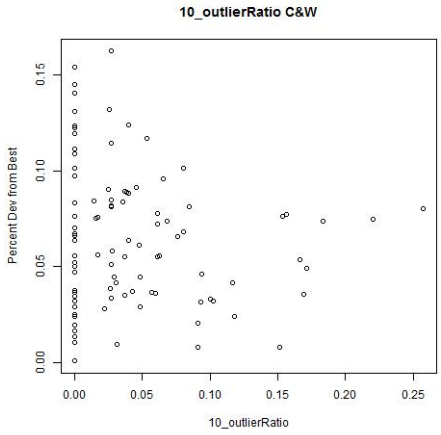


(b) Sweep Algorithm

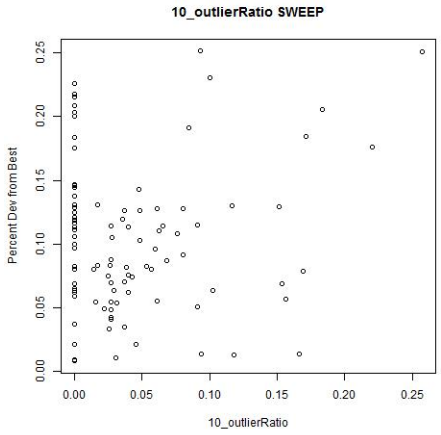


(c) Fuzzy SOM Algorithm

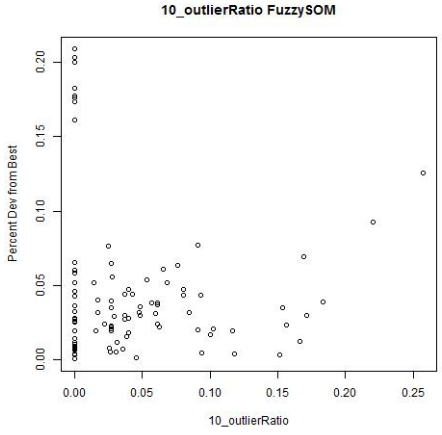
Figure .78: Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of clusters to the total number of cities for each problem instance



(a) Clarke and Wright Algorithm

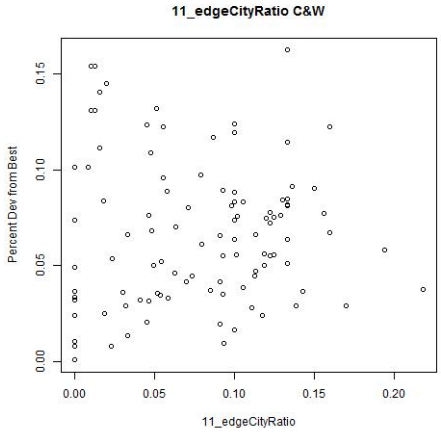


(b) Sweep Algorithm

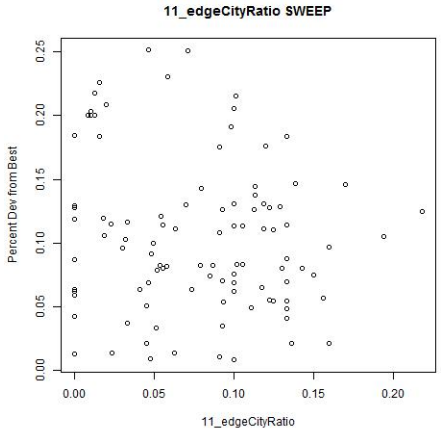


(c) Fuzzy SOM Algorithm

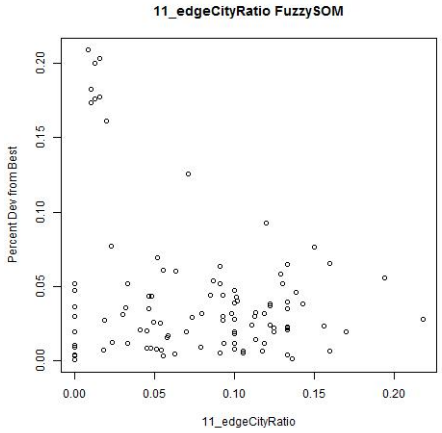
Figure .79: Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of outliers to the total number of cities for each problem instance



(a) Clarke and Wright Algorithm

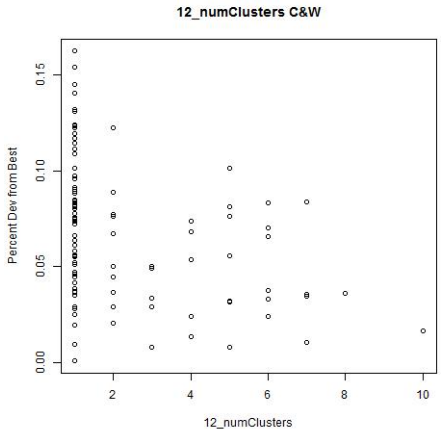


(b) Sweep Algorithm

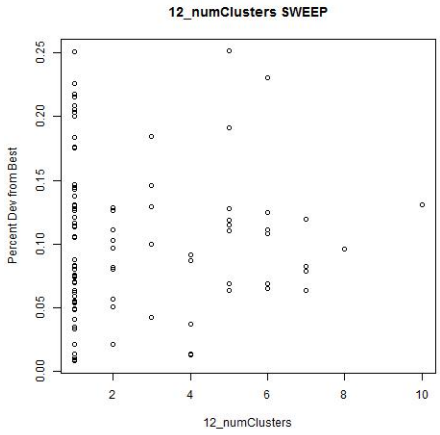


(c) Fuzzy SOM Algorithm

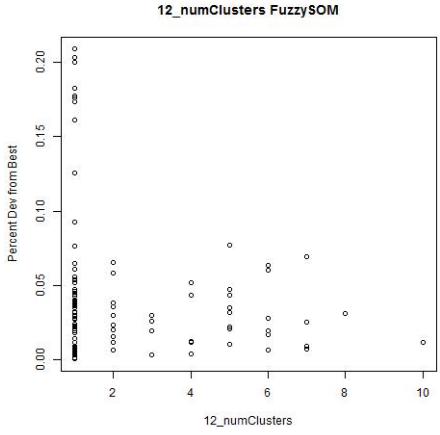
Figure .80: Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of cities at the edge of a cluster (identified by DBSCAN) to the total number of cities for each problem instance



(a) Clarke and Wright Algorithm

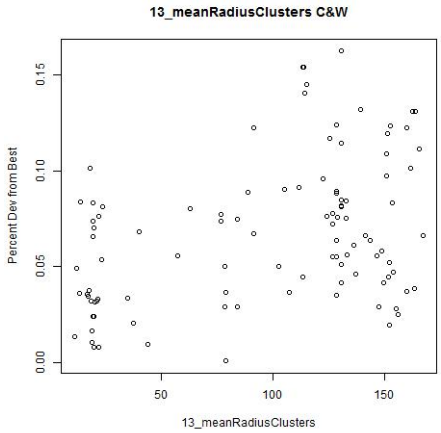


(b) Sweep Algorithm

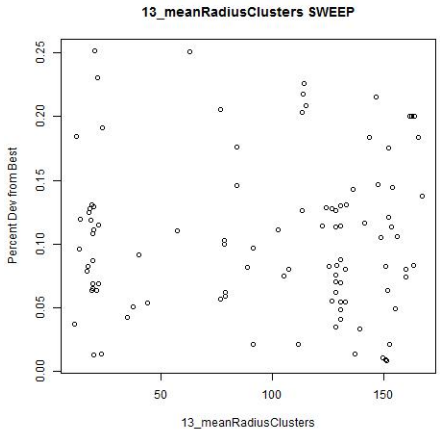


(c) Fuzzy SOM Algorithm

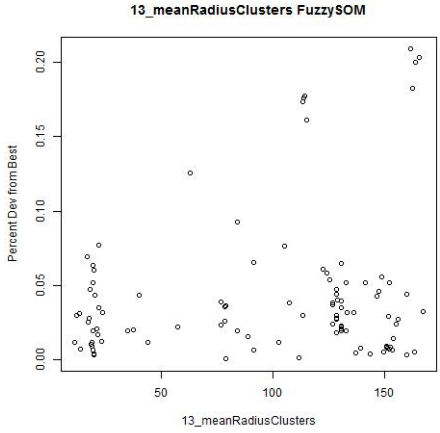
Figure .81: Scatter plots of algorithm performance (percent deviation above best known solution) versus the number of clusters found by DBSCAN for each problem instance



(a) Clarke and Wright Algorithm

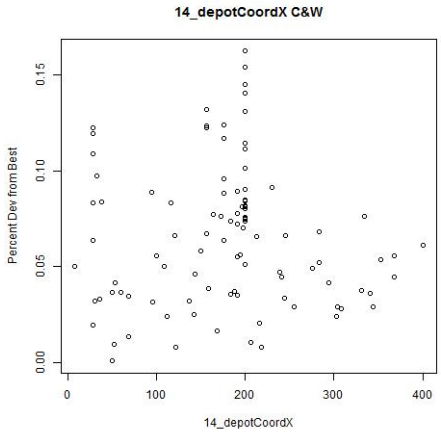


(b) Sweep Algorithm

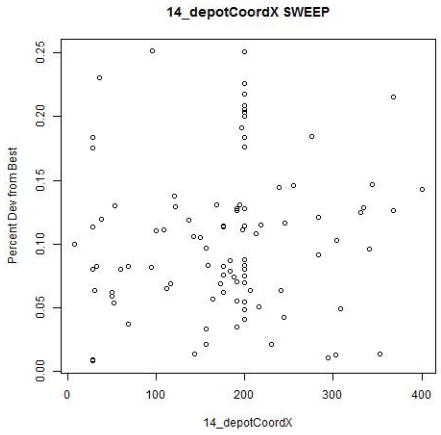


(c) Fuzzy SOM Algorithm

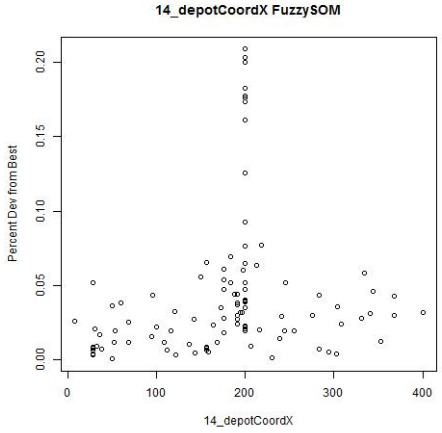
Figure .82: Scatter plots of algorithm performance (percent deviation above best known solution) versus the mean radius of the clusters for each problem instance



(a) Clarke and Wright Algorithm

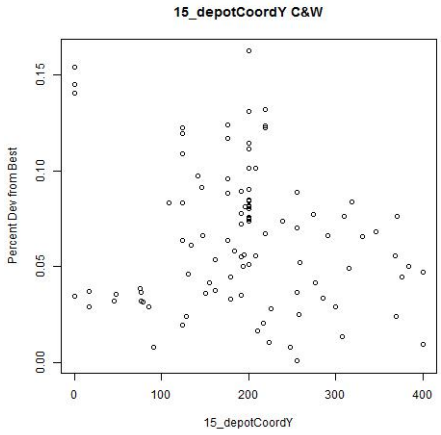


(b) Sweep Algorithm

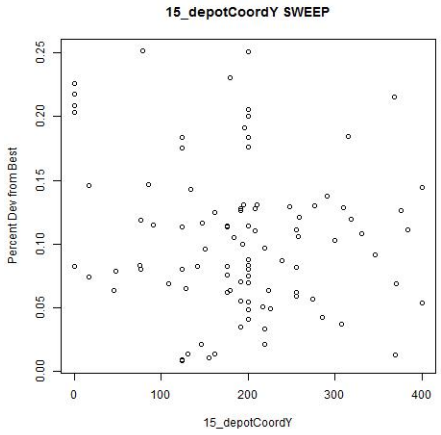


(c) Fuzzy SOM Algorithm

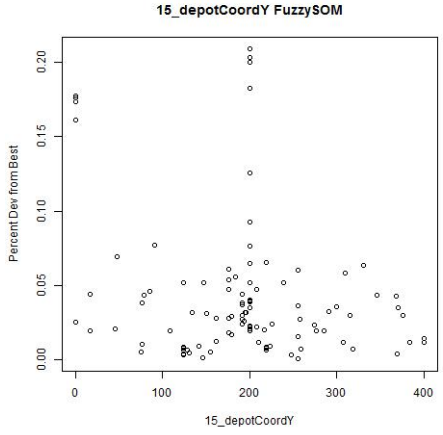
Figure .83: Scatter plots of algorithm performance (percent deviation above best known solution) versus the x coordinate of the depot for each problem instance



(a) Clarke and Wright Algorithm

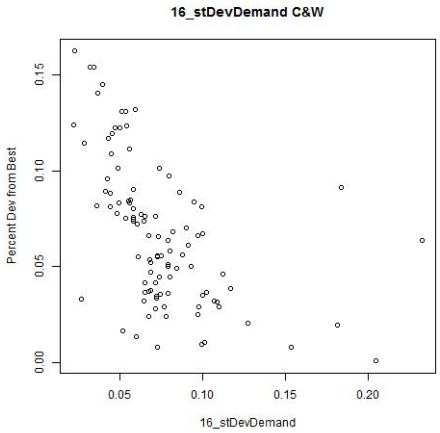


(b) Sweep Algorithm

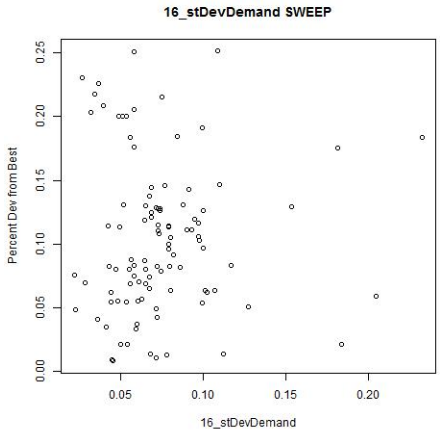


(c) Fuzzy SOM Algorithm

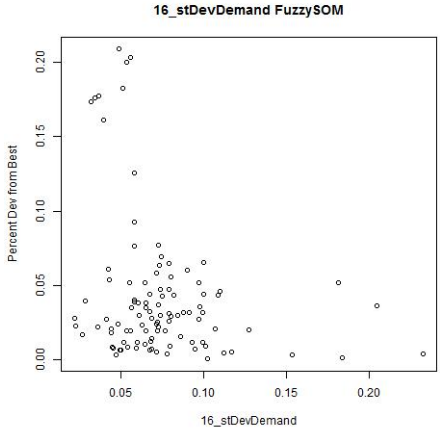
Figure .84: Scatter plots of algorithm performance (percent deviation above best known solution) versus the y coordinate of the depot for each problem instance



(a) Clarke and Wright Algorithm

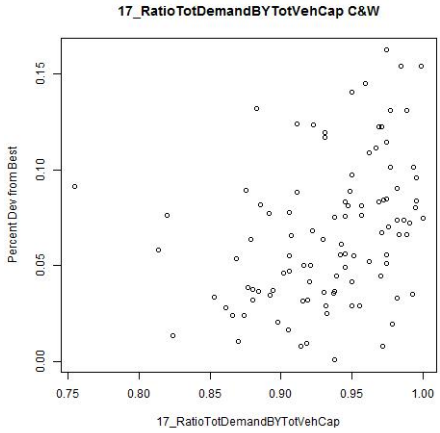


(b) Sweep Algorithm

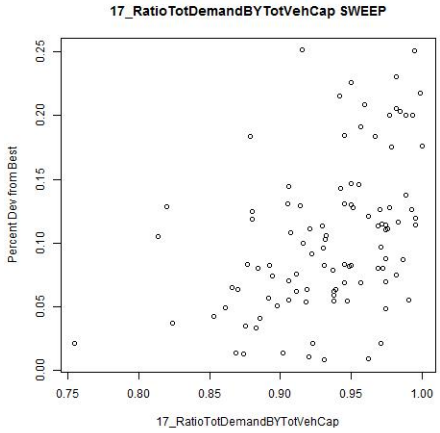


(c) Fuzzy SOM Algorithm

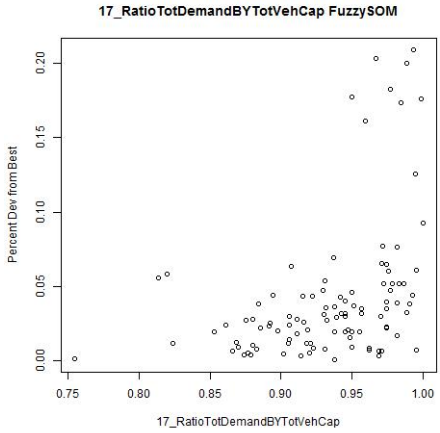
Figure .85: Scatter plots of algorithm performance (percent deviation above best known solution) versus the standard deviation of the demand for each problem instance



(a) Clarke and Wright Algorithm

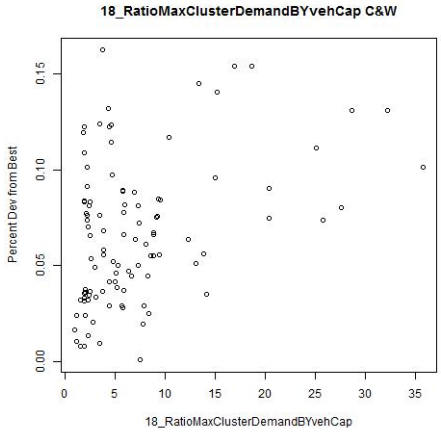


(b) Sweep Algorithm

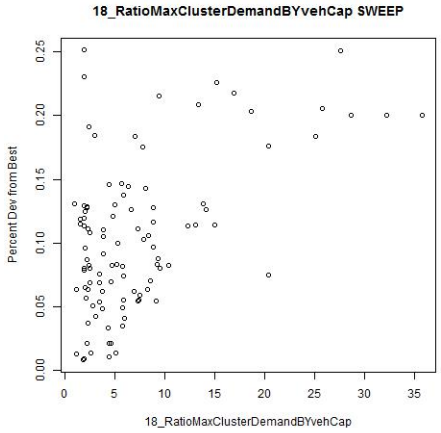


(c) Fuzzy SOM Algorithm

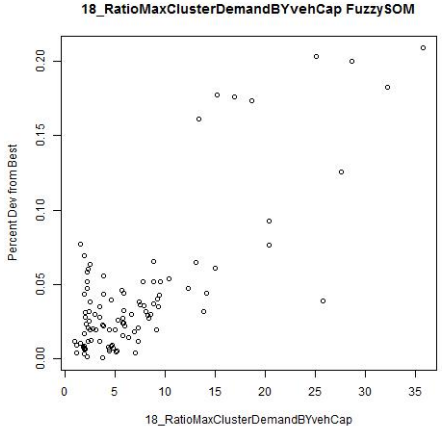
Figure .86: Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of total demand to the available capacity (based on the minimum number of trucks required for the problem) for each problem instance



(a) Clarke and Wright Algorithm

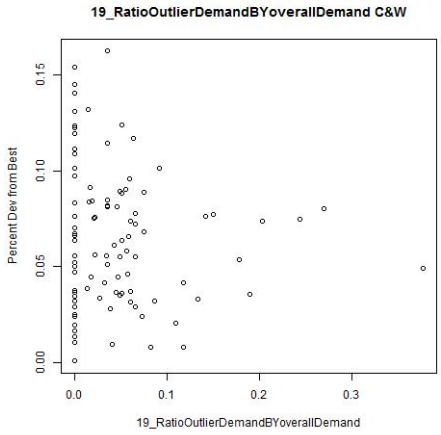


(b) Sweep Algorithm

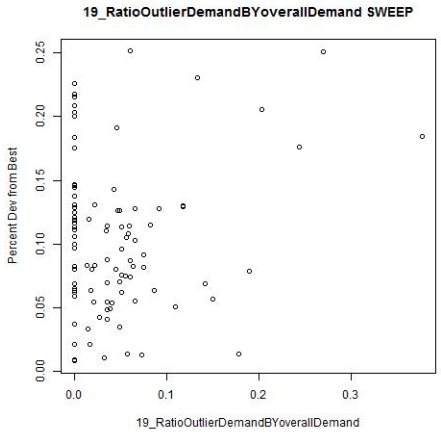


(c) Fuzzy SOM Algorithm

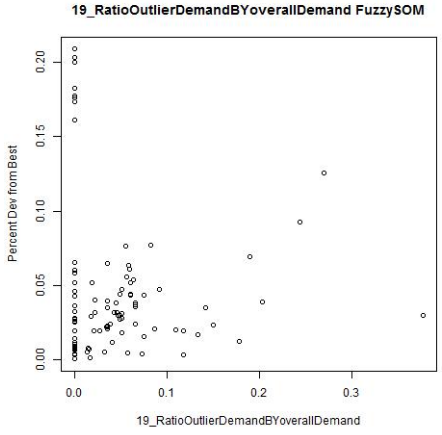
Figure .87: Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of the largest cluster demand to a single vehicle's capacity for each problem instance



(a) Clarke and Wright Algorithm

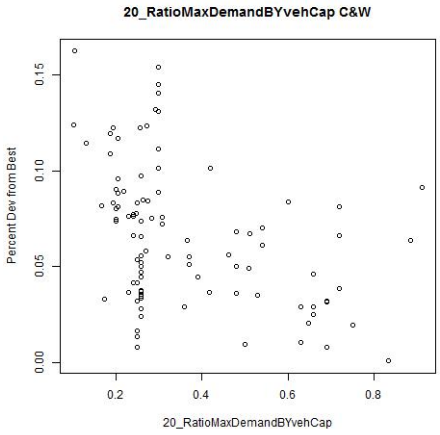


(b) Sweep Algorithm

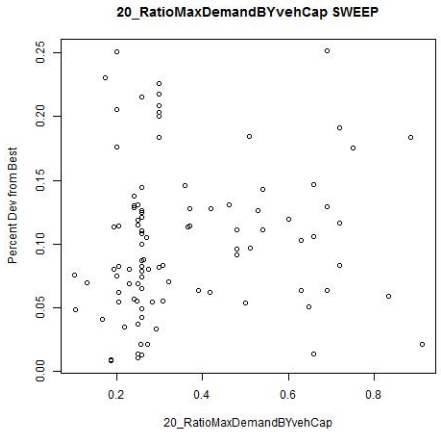


(c) Fuzzy SOM Algorithm

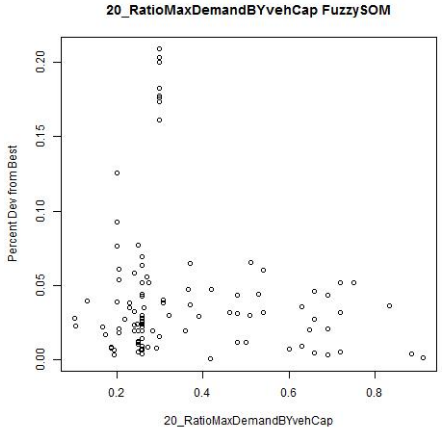
Figure .88: Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of sum of all outlier demands to the overall demand for each problem instance



(a) Clarke and Wright Algorithm

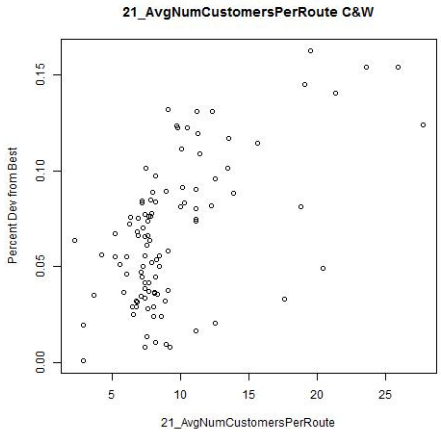


(b) Sweep Algorithm

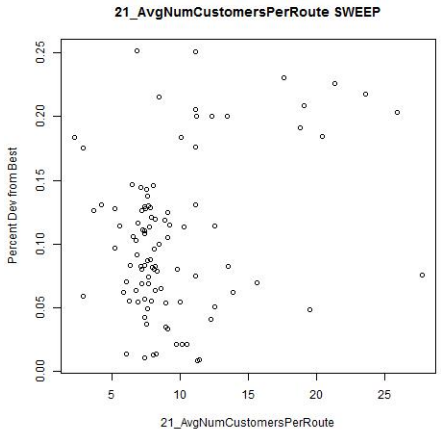


(c) Fuzzy SOM Algorithm

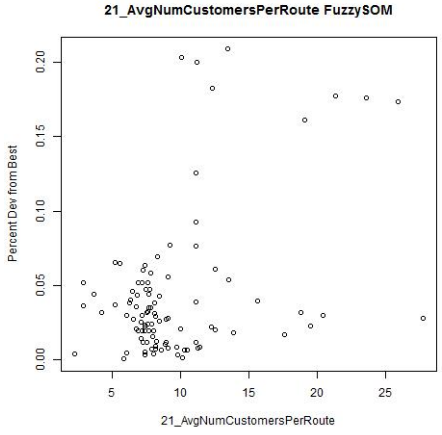
Figure .89: Scatter plots of algorithm performance (percent deviation above best known solution) versus the ratio of the single largest city demand to the vehicle capacity for each problem instance



(a) Clarke and Wright Algorithm

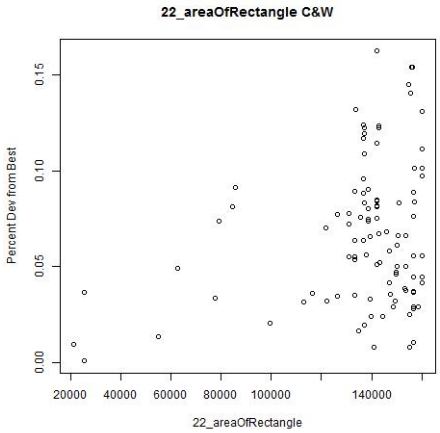


(b) Sweep Algorithm

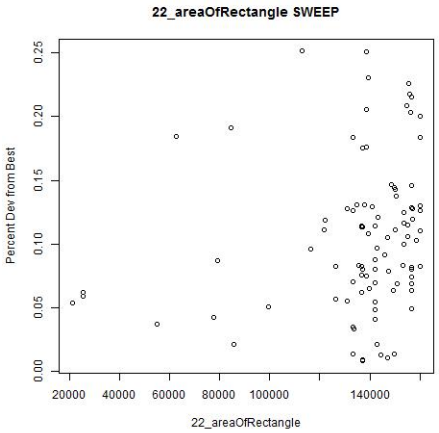


(c) Fuzzy SOM Algorithm

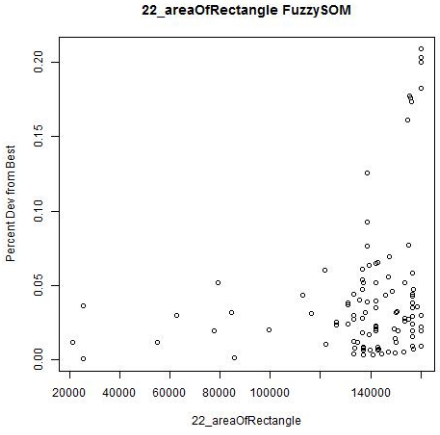
Figure .90: Scatter plots of algorithm performance (percent deviation above best known solution) versus the average number of customers per route for each problem instance



(a) Clarke and Wright Algorithm

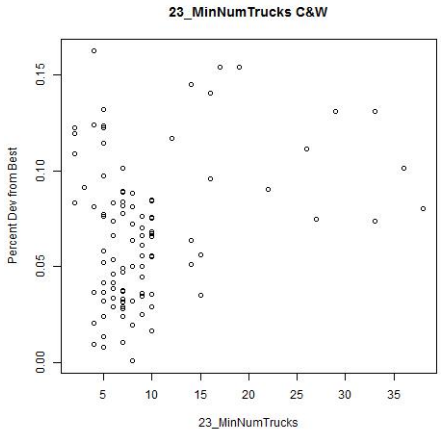


(b) Sweep Algorithm

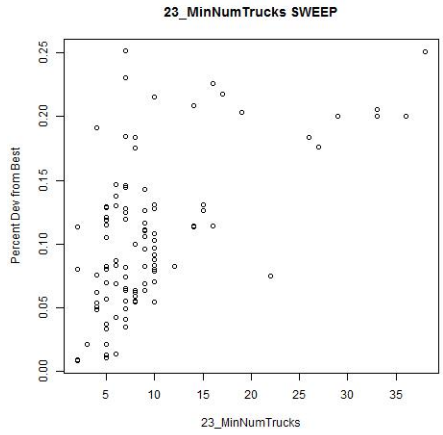


(c) Fuzzy SOM Algorithm

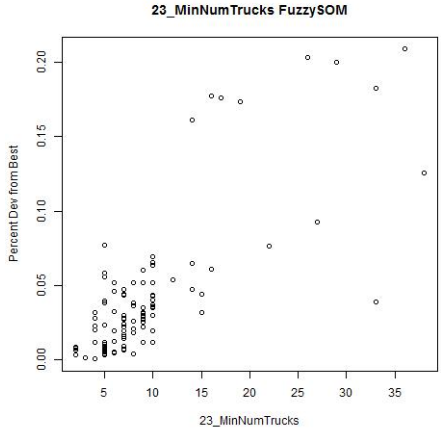
Figure .91: Scatter plots of algorithm performance (percent deviation above best known solution) versus the area of the rectangle that all problems lie within for each problem instance



(a) Clarke and Wright Algorithm



(b) Sweep Algorithm



(c) Fuzzy SOM Algorithm

Figure .92: Scatter plots of algorithm performance (percent deviation above best known solution) versus the minimum number of trucks required for each problem instance

Appendix D: Component Plane Graphs of SOM

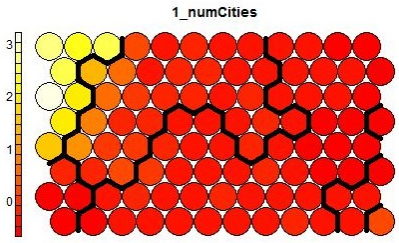


Figure .93: SOM component plane: number of cities

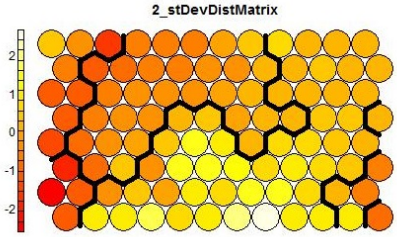


Figure .94: SOM component plane: standard deviation of the distance matrix

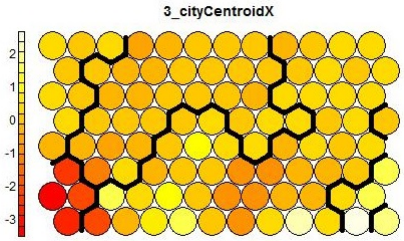


Figure .95: SOM component plane: x-coordinate of city centroid

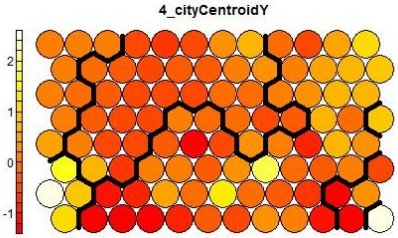


Figure .96: SOM component plane: y-coordinate of city centroid

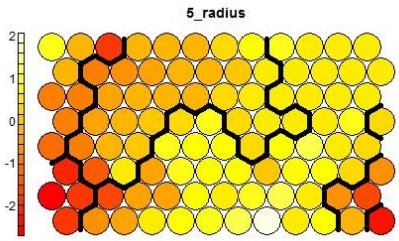


Figure .97: SOM component plane: radius of problem instance

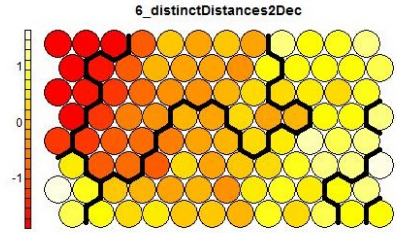


Figure .98: SOM component plane: proportion of distinct distances

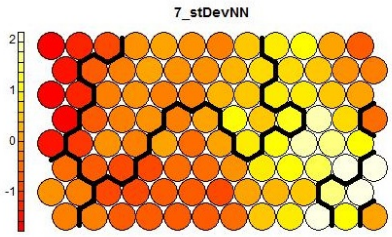


Figure .99: SOM component plane: standard deviation of the nearest neighbor distances

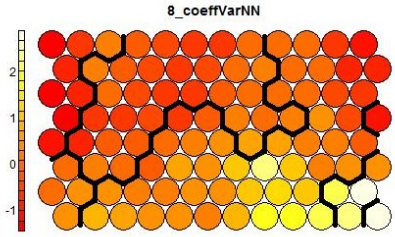


Figure .100: SOM component plane: coefficient of variation for nearest neighbor distances

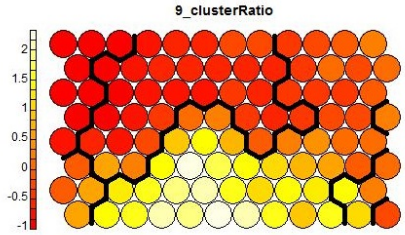


Figure .101: SOM component plane: ratio of the number of clusters to the number of cities

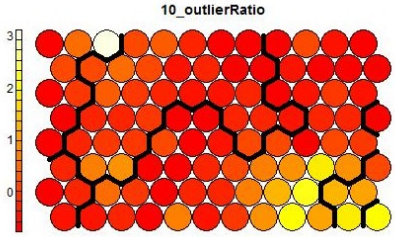


Figure .102: SOM component plane: ratio of the number of outliers to the total number of cities

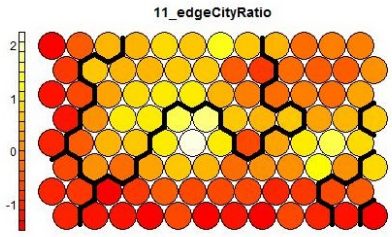


Figure .103: SOM component plane: ratio of the number of edge cities to total number of cities

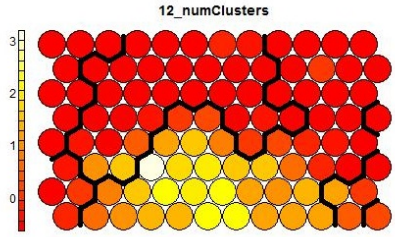


Figure .104: SOM component plane: number of clusters

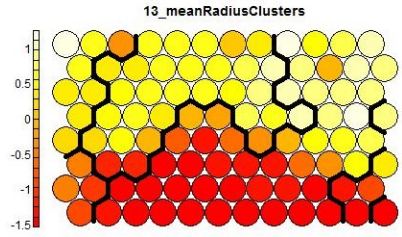


Figure .105: SOM component plane: mean radius of the clusters

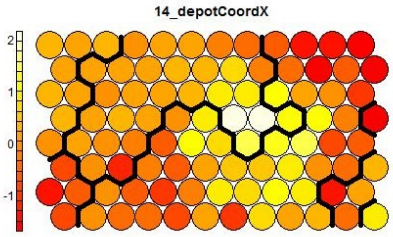


Figure .106: SOM component plane: x-coordinate of the depot

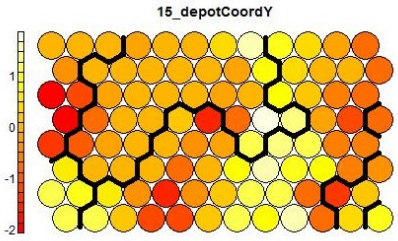


Figure .107: SOM component plane: y-coordinate of the depot

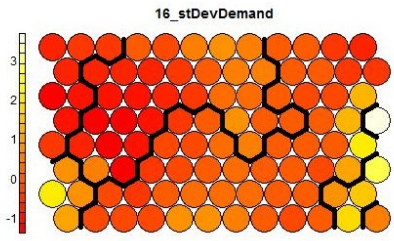


Figure .108: SOM component plane: standard deviation of demand

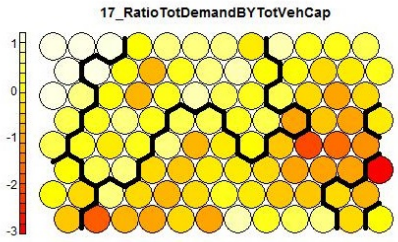


Figure .109: SOM component plane: ratio of total demand to total vehicle capacity

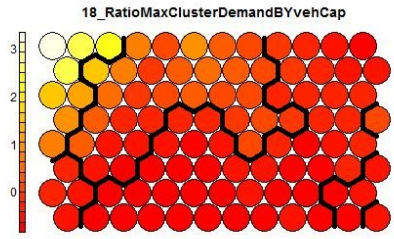


Figure .110: SOM component plane: ratio of the max cluster demand to vehicle capacity

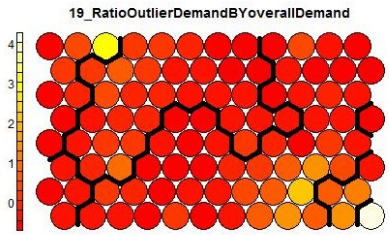


Figure .111: SOM component plane: ratio of outlier demand to overall demand

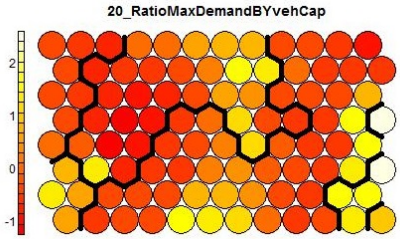


Figure .112: SOM component plane: ratio of the maximum city demand to vehicle capacity

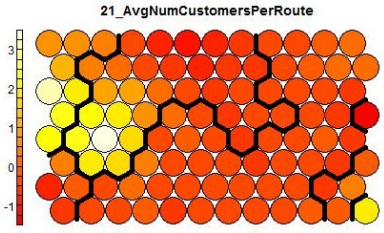


Figure .113: SOM component plane: average number of customers per route

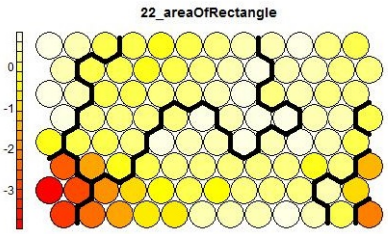


Figure .114: SOM component plane: area of the rectangle

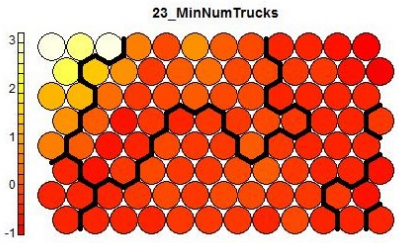
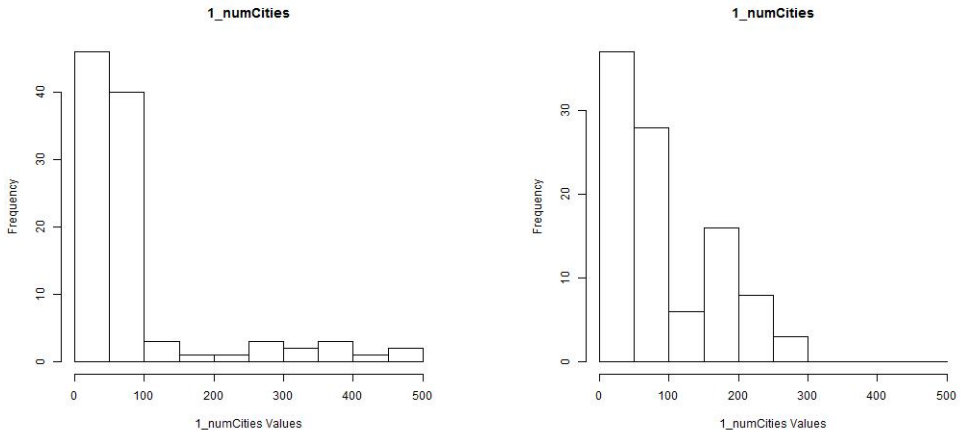


Figure .115: SOM component plane: minimum number of trucks

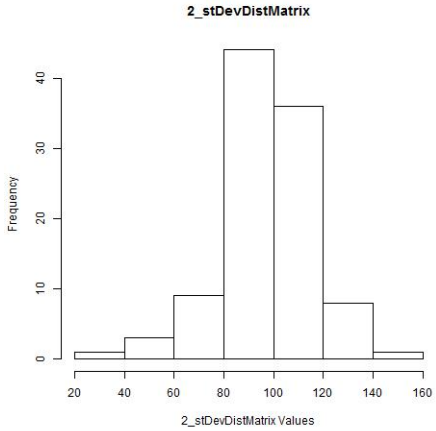
Appendix E: Histograms of the problem characteristics of both the 102 existing benchmark CVRP problems and the 98 generated problems.



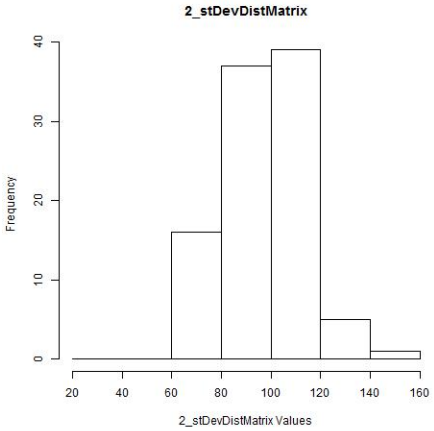
(a) 102 existing benchmark problems

(b) 98 new problems

Figure .116: Histograms of the number of cities in both the existing benchmark problem set and the newly generated problem set

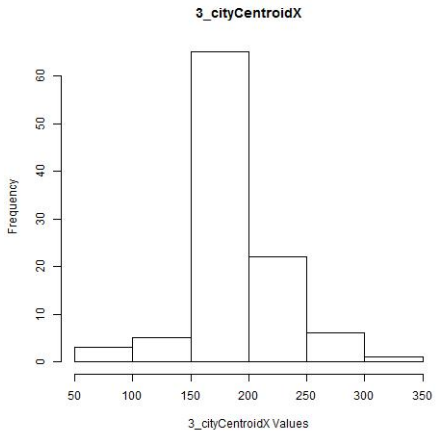


(a) 102 existing benchmark problems

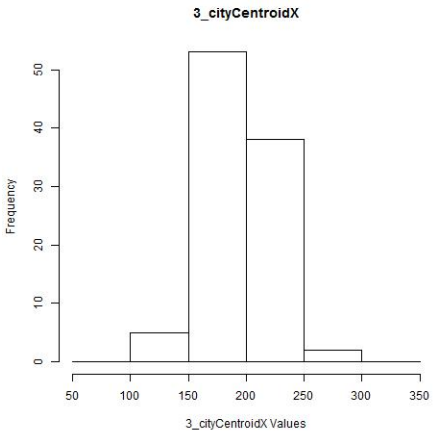


(b) 98 new problems

Figure .117: Histograms of the standard deviation of the distance matrix in both the existing benchmark problem set and the newly generated problem set

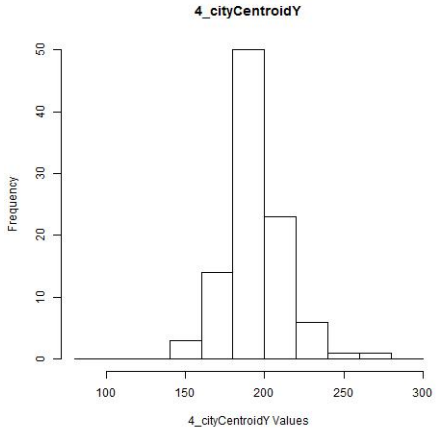


(a) 102 existing benchmark problems

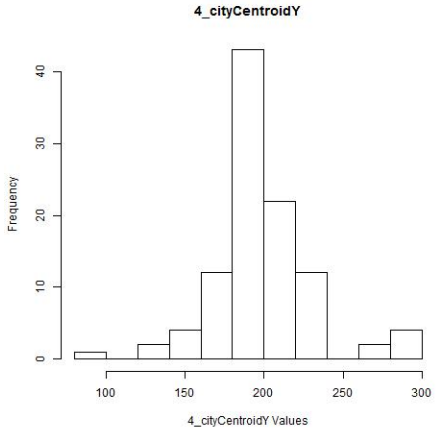


(b) 98 new problems

Figure .118: Histograms of the x-coordinate of the centroid in both the existing benchmark problem set and the newly generated problem set

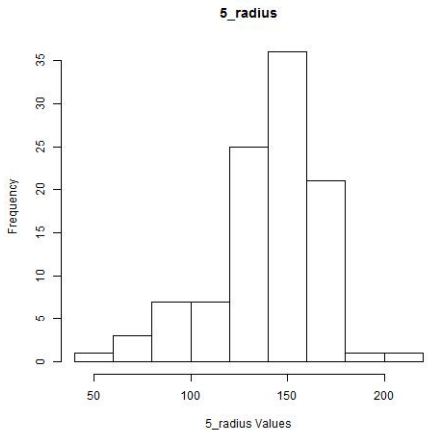


(a) 102 existing benchmark problems

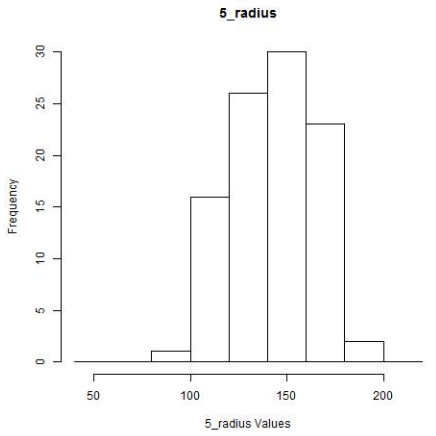


(b) 98 new problems

Figure .119: Histograms of the y-coordinate of the centroid in both the existing benchmark problem set and the newly generated problem set

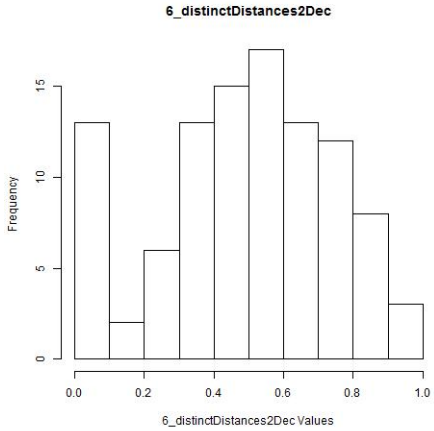


(a) 102 existing benchmark problems

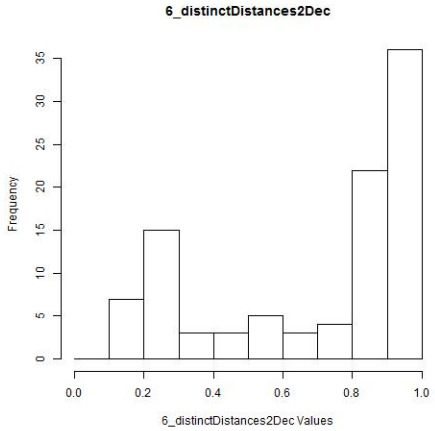


(b) 98 new problems

Figure .120: Histograms of the problem radius in both the existing benchmark problem set and the newly generated problem set

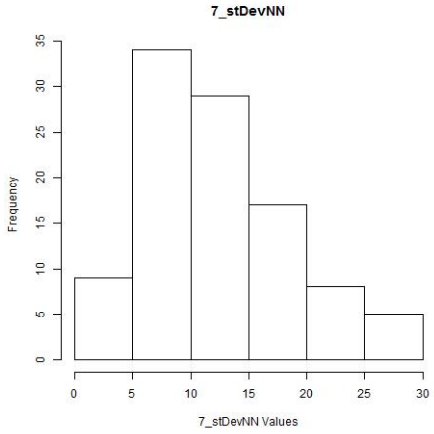


(a) 102 existing benchmark problems

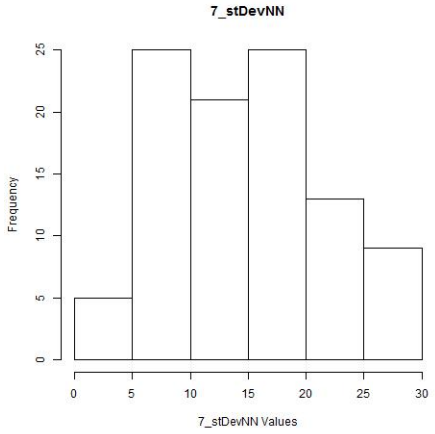


(b) 98 new problems

Figure .121: Histograms of the proportion of distinct distances in the distance matrix in both the existing benchmark problem set and the newly generated problem set

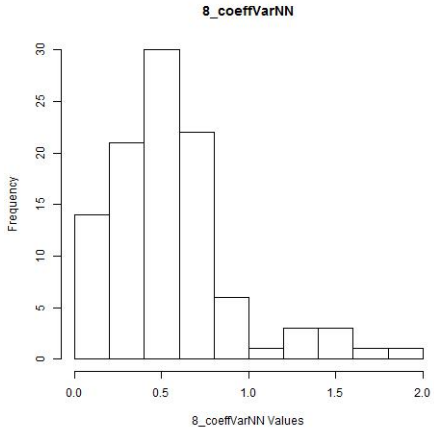


(a) 102 existing benchmark problems

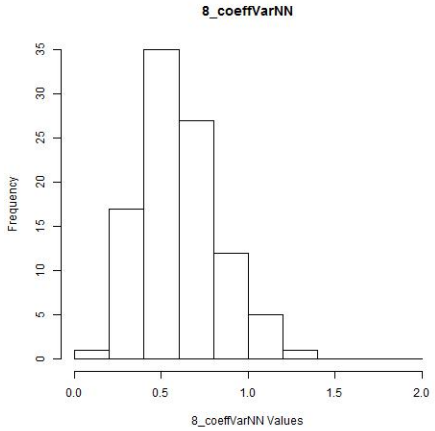


(b) 98 new problems

Figure .122: Histograms of the standard deviation of the nearest neighbors in both the existing benchmark problem set and the newly generated problem set

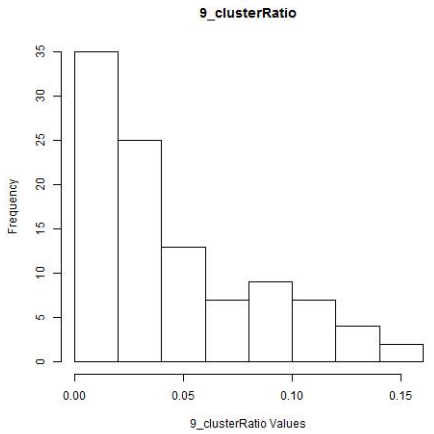


(a) 102 existing benchmark problems

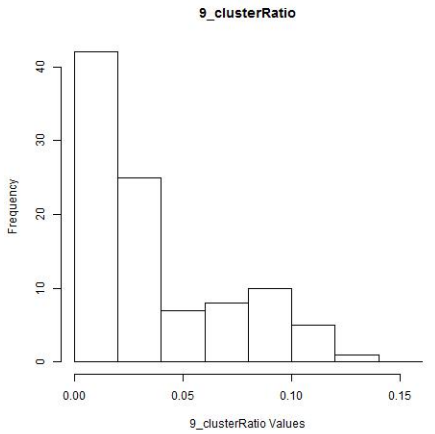


(b) 98 new problems

Figure .123: Histograms of the problem radius in both the existing benchmark problem set and the newly generated problem set

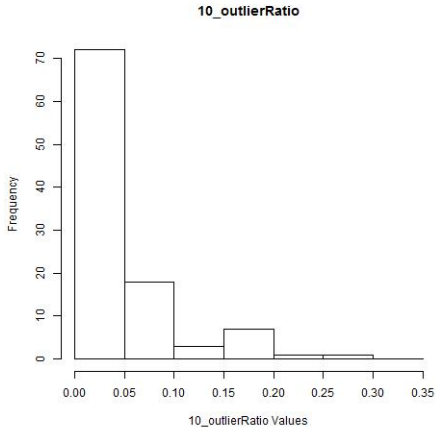


(a) 102 existing benchmark problems

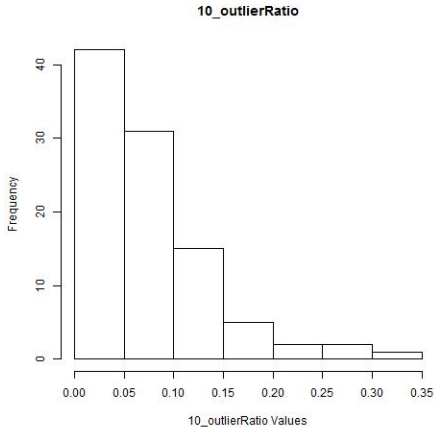


(b) 98 new problems

Figure .124: Histograms of the ratio of the number of clusters to the number of cities in both the existing benchmark problem set and the newly generated problem set

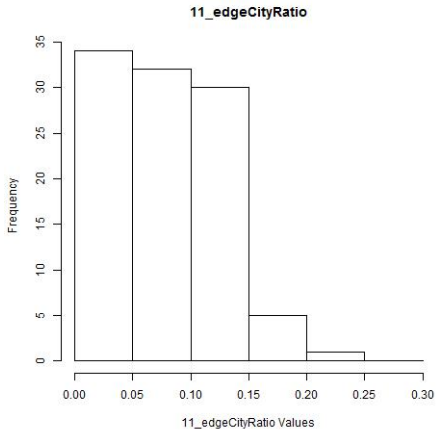


(a) 102 existing benchmark problems

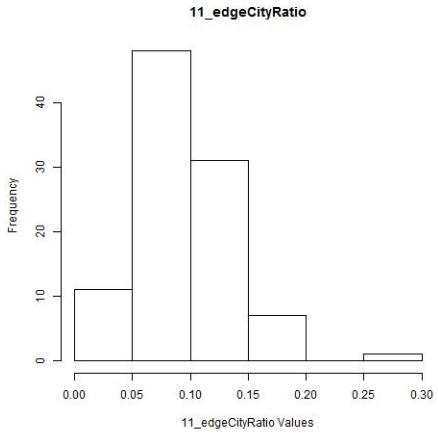


(b) 98 new problems

Figure .125: Histograms of the ratio of the number of outliers to the number of cities in both the existing benchmark problem set and the newly generated problem set

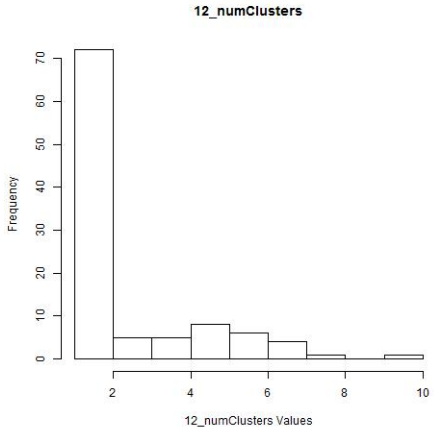


(a) 102 existing benchmark problems

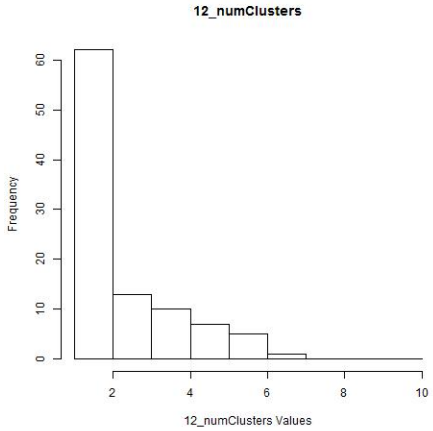


(b) 98 new problems

Figure .126: Histograms of the ratio of the number of cities on the edge of a cluster to the total number of cities in both the existing benchmark problem set and the newly generated problem set

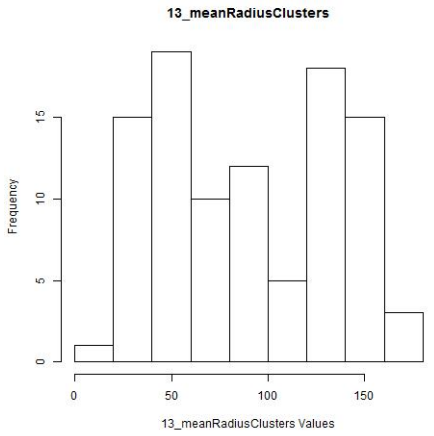


(a) 102 existing benchmark problems

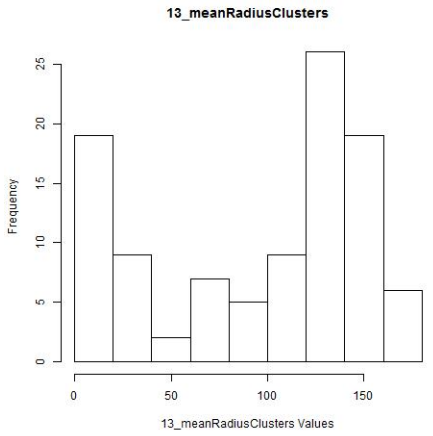


(b) 98 new problems

Figure .127: Histograms of the number of clusters in both the existing benchmark problem set and the newly generated problem set

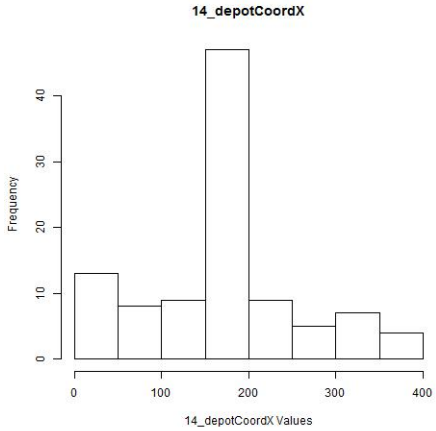


(a) 102 existing benchmark problems

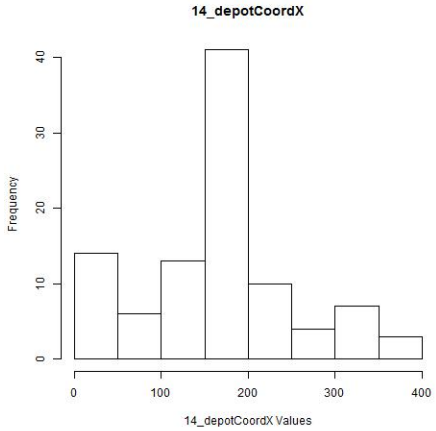


(b) 98 new problems

Figure .128: Histograms of the mean radius of the clusters in both the existing benchmark problem set and the newly generated problem set

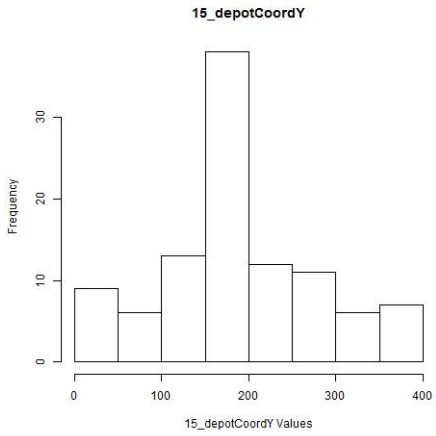


(a) 102 existing benchmark problems

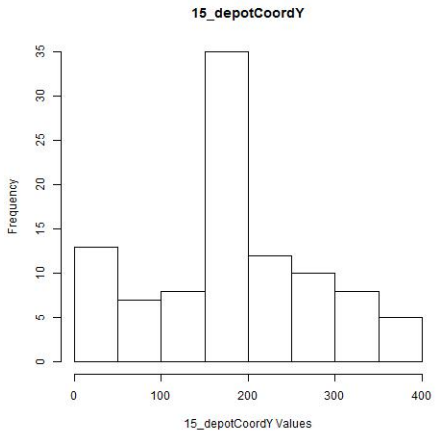


(b) 98 new problems

Figure .129: histograms of the x-coordinate of the depot in both the existing benchmark problem set and the newly generated problem set

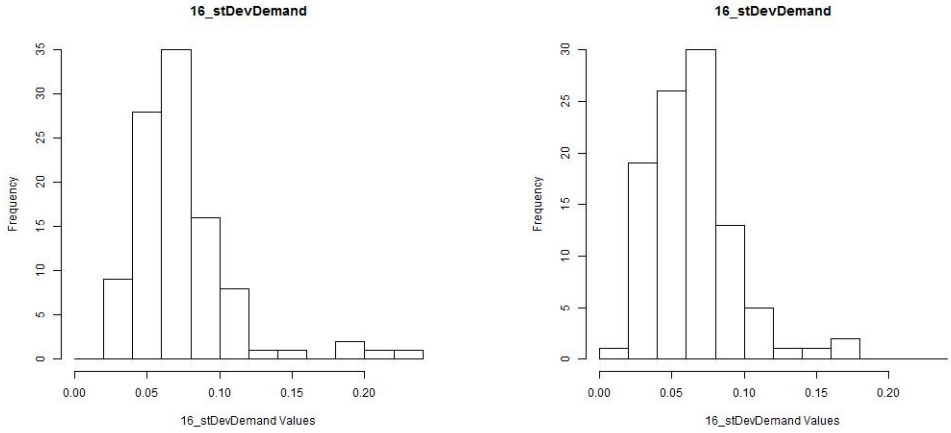


(a) 102 existing benchmark problems



(b) 98 new problems

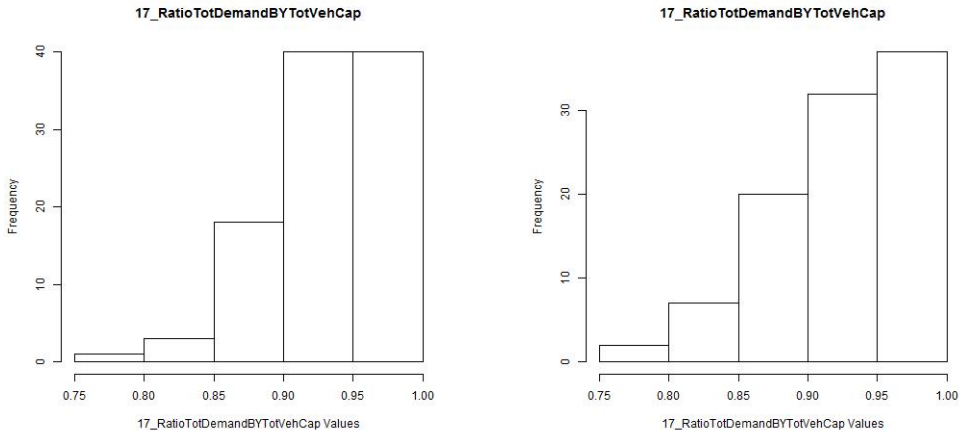
Figure .130: Histograms of the y-coordinate of the depot in both the existing benchmark problem set and the newly generated problem set



(a) 102 existing benchmark problems

(b) 98 new problems

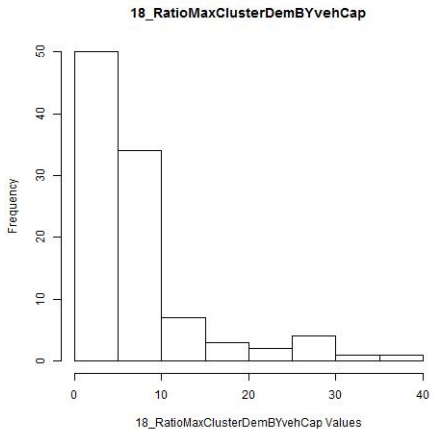
Figure .131: Histograms of the standard deviation of demand in both the existing benchmark problem set and the newly generated problem set



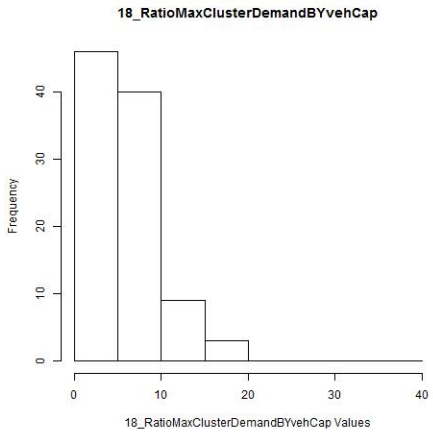
(a) 102 existing benchmark problems

(b) 98 new problems

Figure .132: Histograms of the ratio of the total demand to the total vehicle capacity in both the existing benchmark problem set and the newly generated problem set

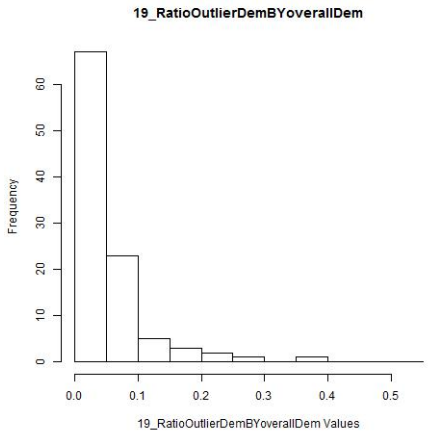


(a) 102 existing benchmark problems

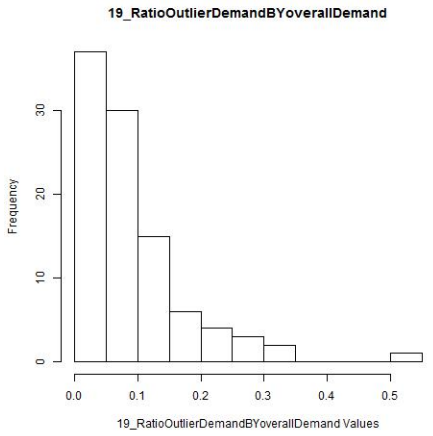


(b) 98 new problems

Figure .133: Histograms of the ratio of the maximum cluster demand to vehicle capacity in both the existing benchmark problem set and the newly generated problem set

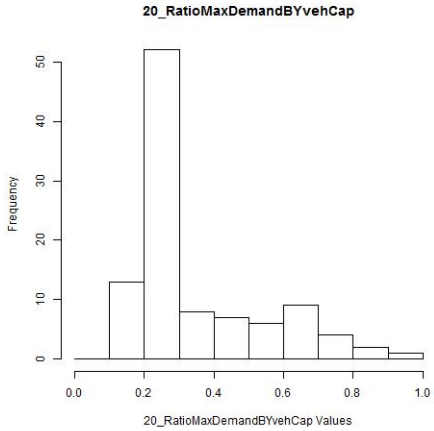


(a) 102 existing benchmark problems

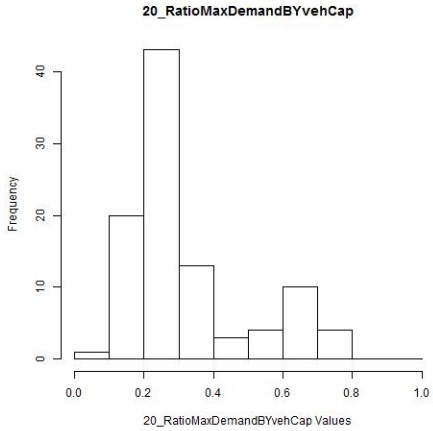


(b) 98 new problems

Figure .134: Histograms of the ratio of the outlier demand to the total demand in both the existing benchmark problem set and the newly generated problem set

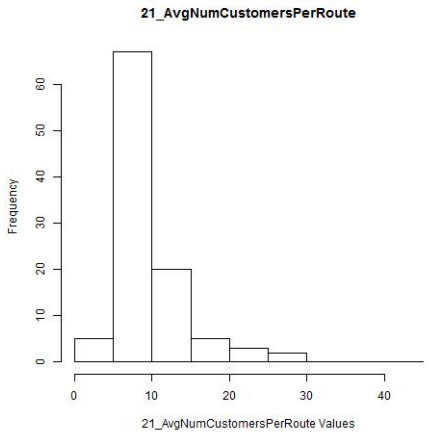


(a) 102 existing benchmark problems

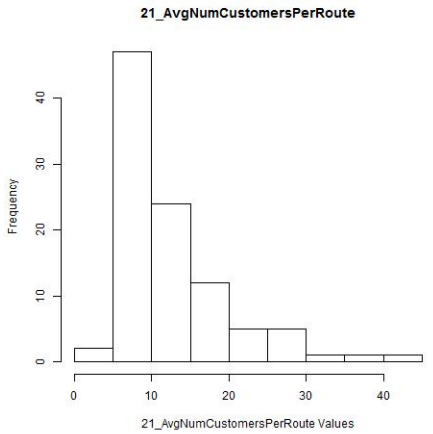


(b) 98 new problems

Figure .135: Histograms of the ratio of the maximum demand to the vehicle capacity in both the existing benchmark problem set and the newly generated problem set

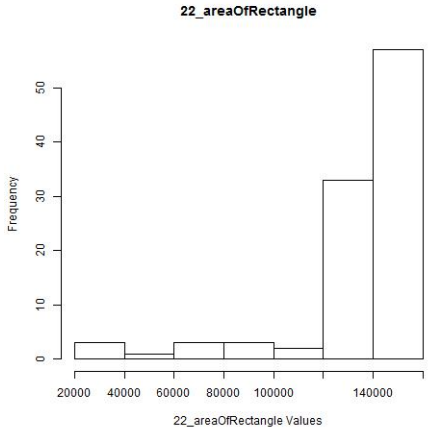


(a) 102 existing benchmark problems

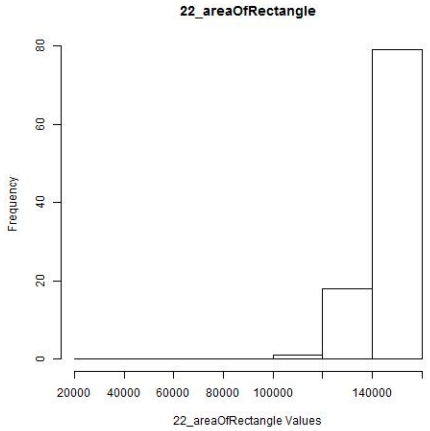


(b) 98 new problems

Figure .136: Histograms of the average number of customers per route in both the existing benchmark problem set and the newly generated problem set

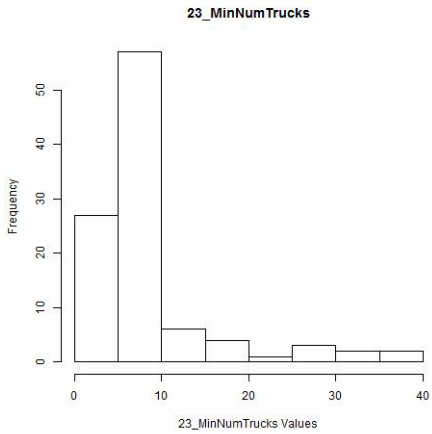


(a) 102 existing benchmark problems

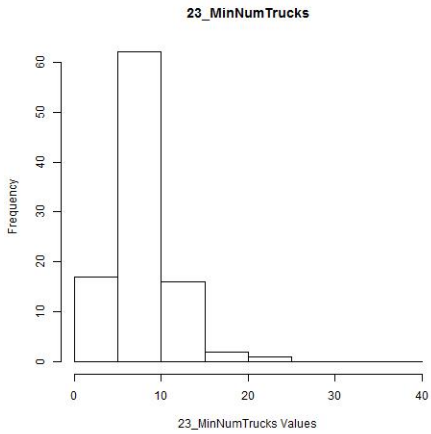


(b) 98 new problems

Figure .137: Histograms of the area of the rectangle the cities lie within for both the existing benchmark problem set and the newly generated problem set



(a) 102 existing benchmark problems



(b) 98 new problems

Figure .138: Histograms of the minimum number of trucks required in both the existing benchmark problem set and the newly generated problem set

Appendix F: Results of Solving the Newly Generated Problem Instances

Problem Name	Clarke & Wright	Sweep	Fuzzy SOM
A-n32-k5-M-n151-k12	4255.71	4185.32	4024.98
A-n33-k6-A-n37-k6	3805.00	4253.12	3830.42
A-n34-k5-B-n66-k9	4288.08	4688.12	4599.45
A-n37-k5-B-n31-k5	3064.27	3116.87	3015.32
A-n38-k5-kelly11	6626.50	6462.52	6041.57
A-n39-k5-A-n54-k7	4187.44	4581.55	4024.83
A-n39-k5-F-n45-k4	2202.47	2418.82	2322.15
A-n39-k5-M-n151-k12	4113.57	3990.93	3889.91
A-n39-k5-P-n45-k5	3088.54	3084.66	2968.59
A-n39-k6-B-n34-k5	3815.74	3834.21	3553.95
A-n39-k6-B-n52-k7	3953.89	3638.58	3503.51
A-n39-k6-E-n51-k5	3421.36	3500.57	3437.10
A-n45-k6-kelly10	6983.67	7082.62	6426.46
A-n45-k6-kelly15	7569.78	7172.67	6906.85
A-n45-k7-kelly19	5166.40	5140.06	4955.18
A-n46-k7-A-n37-k6	3855.44	4263.63	3963.75
A-n46-k7-P-n40-k5	3368.40	3392.04	3366.60
A-n53-k7-A-n34-k5	3478.07	3678.87	3465.84
A-n55-k9-kelly10	7664.66	8438.15	7504.14
A-n60-k9-A-n62-k8	5957.78	6113.33	5685.37
A-n60-k9-P-n55-k10	4307.00	4460.56	4466.00
A-n63-k10-kelly17	5971.62	6025.89	5762.62
A-n63-k9-kelly13	7937.26	7804.87	7390.36
A-n63-k9-P-n19-k2	3043.28	3156.18	3259.12
A-n65-k9-kelly15	8644.66	8491.08	8551.66
A-n65-k9-P-n50-k8	4324.34	4484.32	4302.24
B-n35-k5-B-n41-k6	3193.02	3256.97	2952.94
B-n35-k5-E-n22-k4	2252.33	2364.88	2285.13
B-n43-k6-A-n63-k9	5401.90	5842.53	5276.43
B-n43-k6-F-n72-k4	2694.13	2721.09	2559.84
B-n44-k7-A-n53-k7	4865.94	4830.80	4600.28
B-n44-k7-B-n57-k9	5554.80	6022.38	5593.25
B-n45-k6-B-n50-k7	3058.46	3483.04	3043.05
B-n50-k7-A-n55-k9	4205.62	4285.28	4020.01
B-n50-k7-B-n43-k6	3721.16	4021.35	3719.65
B-n50-k8-P-n20-k2	2666.34	2924.24	2820.09
B-n50-k8-P-n55-k15	4062.54	4153.89	4057.67
B-n51-k7-B-n39-k5	3648.60	3896.13	3476.68
B-n52-k7-M-n121-k7	4087.35	4216.31	4039.34
B-n56-k7-kelly9	6014.49	6250.02	5747.77
B-n57-k9-A-n37-k5	2938.50	3075.99	2876.00
B-n64-k9-A-n65-k9	5075.41	5091.34	4947.28
B-n64-k9-B-n38-k6	3361.36	3692.88	3287.79
B-n66-k9-A-n39-k5	5169.55	5098.30	4985.52
B-n67-k10-B-n45-k5	4370.82	4517.10	4156.90
B-n78-k10-A-n32-k5	4201.12	4275.11	4037.74
E-n101-k8-B-n31-k5	4261.42	4220.22	4076.44
E-n23-k5-P-n55-k7	2294.55	2345.01	2353.93

Problem Name	Clarke & Wright	Sweep	Fuzzy SOM
E-n33-k4-B-n57-k7	2797.58	3350.57	2784.81
E-n33-k4-M-n200-k16	4224.02	4008.70	4082.46
E-n33-k4-P-n60-k10	2789.57	2873.61	2799.58
E-n76-k10-B-n78-k10	6099.76	6677.04	6042.14
E-n76-k10-P-n40-k5	3709.04	3646.42	3593.53
E-n76-k14-B-n50-k8	6750.06	7750.38	7039.22
E-n76-k14-P-n70-k10	5052.06	5312.29	5052.99
E-n76-k7-A-n45-k6	4161.05	4301.63	4009.50
kelly9-A-n44-k7	6584.05	6589.92	6333.80
kelly10-B-n39-k5	5055.62	4882.12	4776.79
kelly10-E-n76-k10	6394.92	6101.53	6006.00
kelly11-B-n35-k5	7102.80	6666.75	6816.94
kelly11-P-n101-k4	5521.02	5390.00	5165.94
kelly14-B-n45-k5	6968.76	6820.68	6443.48
kelly14-P-n55-k15	9250.32	10019.70	8990.52
kelly16-A-n33-k6	8161.59	7858.96	7740.98
kelly16-A-n37-k5	7044.69	6998.37	6725.89
kelly17-B-n34-k5	4541.59	4462.25	4313.26
kelly18-E-n76-k8	5890.72	5720.18	5603.61
kelly19-A-n32-k5	5949.51	6218.15	5720.82
kelly19-B-n51-k7	5695.29	5736.99	5414.53
kelly19-M-n200-k16	8131.60	8763.60	8026.81
M-n121-k7-B-n50-k7	4271.45	4171.58	4005.78
M-n121-k7-B-n50-k7	4271.45	4171.58	4005.78
M-n200-k16-P-n22-k8	5836.37	6102.05	5561.27
P-n101-k4-P-n76-k5	3714.56	3634.49	3809.89
P-n16-k8-B-n35-k5	4468.02	4892.76	4318.37
P-n16-k8-B-n57-k7	4790.88	5022.17	4631.74
P-n20-k2-P-n21-k2	2122.59	2066.47	1964.38
P-n21-k2-A-n62-k8	3287.86	3342.76	3071.35
P-n21-k2-B-n38-k6	2278.45	2180.95	2229.43
P-n23-k8-A-n54-k7	5397.15	6149.02	5360.71
P-n23-k8-kelly10	7261.04	8189.00	7003.24
P-n40-k5-A-n64-k9	4607.02	5224.13	4486.94
P-n40-k5-B-n39-k5	2850.48	3208.03	2896.70
P-n45-k5-kelly15	6998.39	6809.51	6457.63
P-n45-k5-kelly15	6998.39	6809.51	6497.78
P-n45-k5-P-n50-k10	3697.59	3571.84	3522.39
P-n50-k7-A-n45-k6	3907.01	4071.22	3927.60
P-n50-k7-P-n60-k15	3792.42	3917.50	4043.92
P-n50-k8-F-n135-k7	3566.76	3992.79	3644.55
P-n50-k8-kelly14	7596.99	7626.23	7361.75
P-n51-k10-B-n50-k8	5698.89	6372.67	5972.24
P-n51-k10-kelly9	7631.13	8637.01	7577.01
P-n55-k15-P-n51-k10	5277.94	5324.65	5175.38
P-n55-k7-A-n45-k7	4770.47	5295.88	4827.11
P-n55-k7-kelly10	6537.32	6766.74	6415.24
P-n55-k7-M-n121-k7	4401.87	4707.71	4373.13
P-n60-k15-kelly14	9569.40	10307.64	9277.37
P-n65-k10-M-n151-k12	5196.67	5037.95	4978.94

Problem Name	Clarke & Wright	Sweep	Fuzzy SOM
P-n65-k10-M-n200-k16	5484.40	5779.94	5573.82
P-n76-k5-A-n32-k5	3796.64	3917.08	3696.99

Appendix G: Additional Plots for SOM Used for Prediction

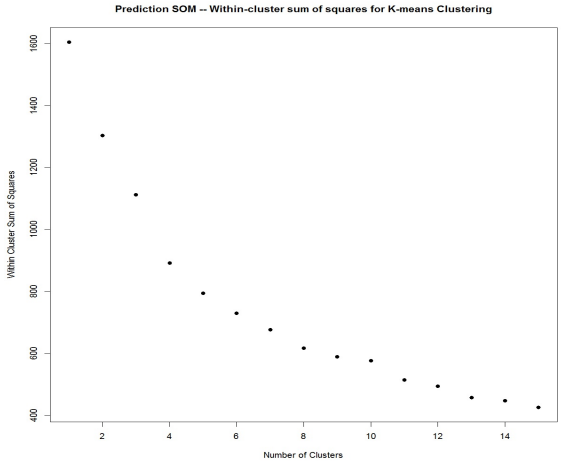


Figure .139: K-means plot for the trained SOM for prediction

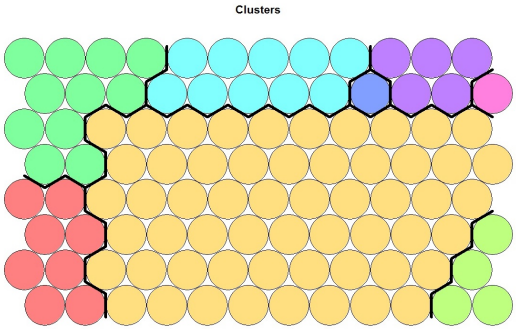


Figure .140: 8 clusters in the trained SOM for prediction

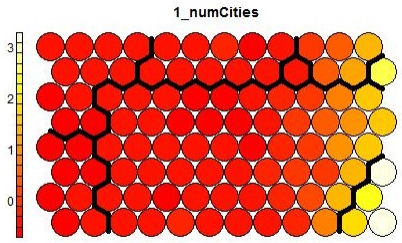


Figure .141: SOM component plane: number of cities

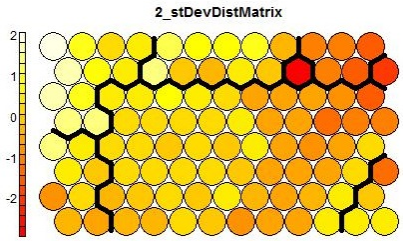


Figure .142: SOM component plane: standard deviation of the distance matrix

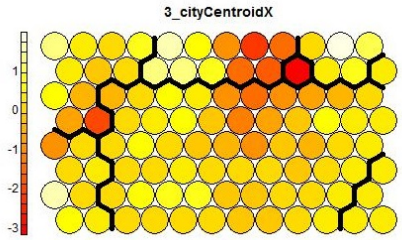


Figure .143: SOM component plane: x-coordinate of city centroid

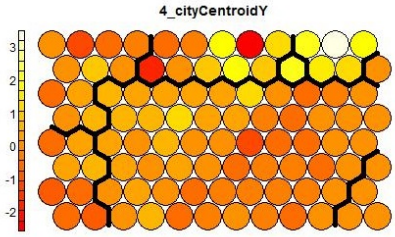


Figure .144: SOM component plane: y-coordinate of city centroid

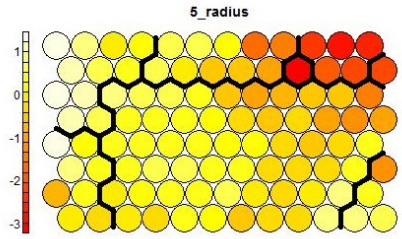


Figure .145: SOM component plane: radius of problem instance

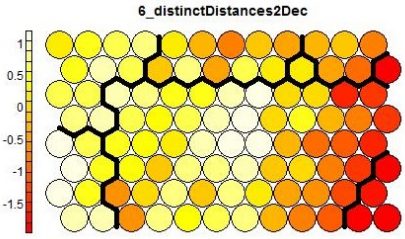


Figure .146: SOM component plane: proportion of distinct distances

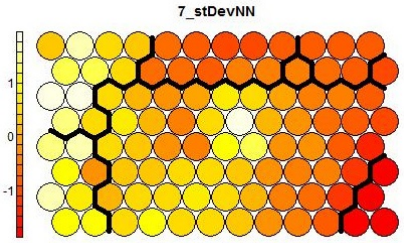


Figure .147: SOM component plane: standard deviation of the nearest neighbor distances

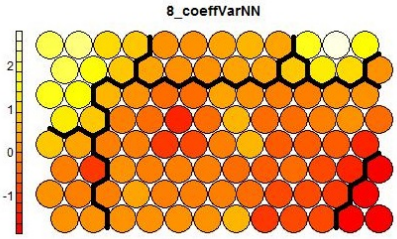


Figure .148: SOM component plane: coefficient of variation for nearest neighbor distances

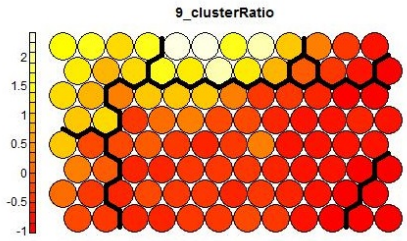


Figure .149: SOM component plane: ratio of the number of clusters to the number of cities

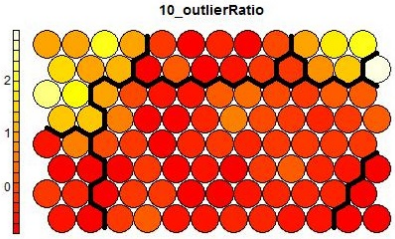


Figure .150: SOM component plane: ratio of the number of outliers to the total number of cities

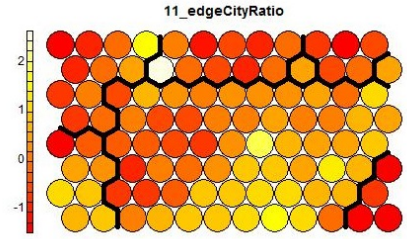


Figure .151: SOM component plane: ratio of the number of edge cities to total number of cities

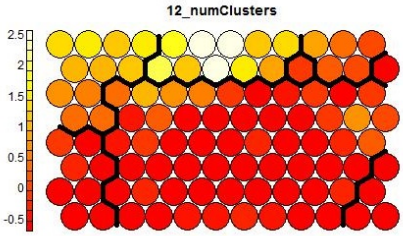


Figure .152: SOM component plane: number of clusters

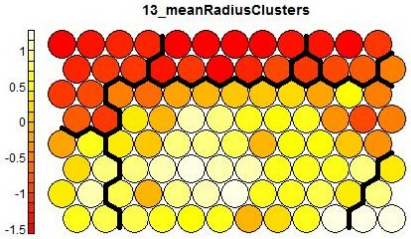


Figure .153: SOM component plane: mean radius of the clusters

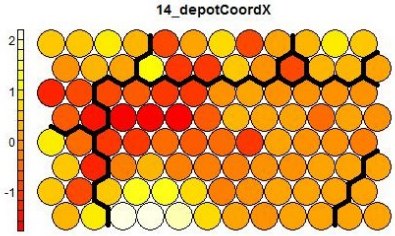


Figure .154: SOM component plane: x-coordinate of the depot

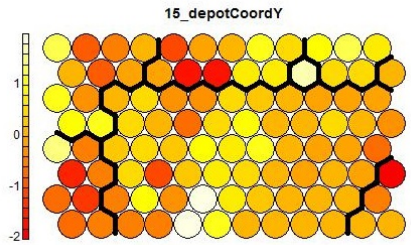


Figure .155: SOM component plane: y-coordinate of the depot

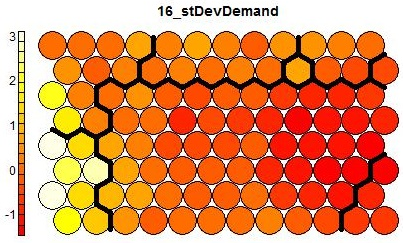


Figure .156: SOM component plane: standard deviation of demand

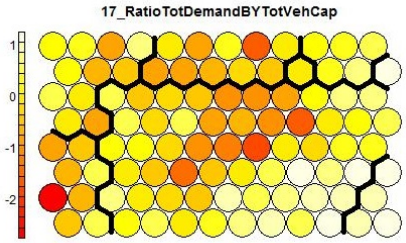


Figure .157: SOM component plane: ratio of total demand to total vehicle capacity

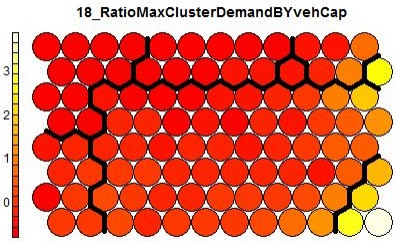


Figure .158: SOM component plane: ratio of the max cluster demand to vehicle capacity

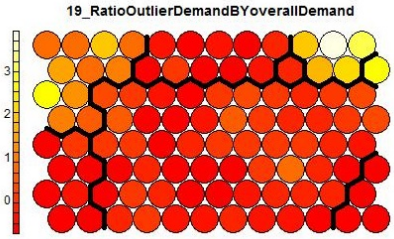


Figure .159: SOM component plane: ratio of outlier demand to overall demand

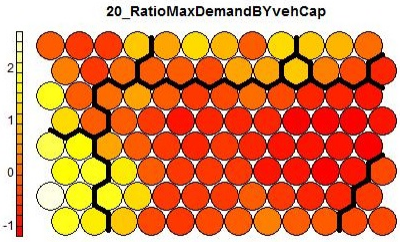


Figure .160: SOM component plane: ratio of the maximum city demand to vehicle capacity

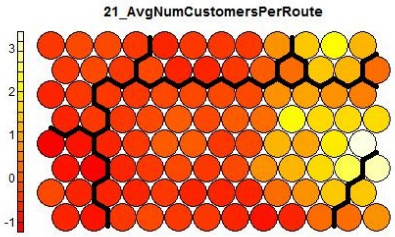


Figure .161: SOM component plane: average number of customers per route

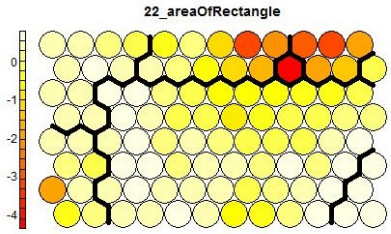


Figure .162: SOM component plane: area of the rectangle

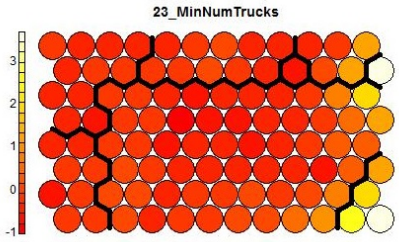


Figure .163: SOM component plane: minimum number of trucks

BIBLIOGRAPHY

Aha, D. W., "Generalizing from case studies: A case study," in *Proc. of the 9th International Conference on Machine Learning*, 1992, pp. 1–10.

Alba, E. and Dorronsoro, B., "Computing nine new best-so-far solutions for capacitated vrp with a cellular genetic algorithm," *Information Processing Letters*, vol. 98, no. 6, pp. 225–230, 2006.

Alexander, S., "On the history of combinatorial optimization (till 1960)," *Handbooks in Operations Research and Management Science: Discrete Optimization*, vol. 12, p. 1, 2005.

Altinkemer, K. and Gavish, B., "Parallel savings based heuristics for the delivery problem," *Operations Research*, vol. 39, no. 3, pp. 456–469, 1991.

Angeniol, B., de La Croix Vaubois, G., and Le Texier, J.-Y., "Self-organizing feature maps and the travelling salesman problem," *Neural Networks*, vol. 1, no. 4, pp. 289–293, 1988.

Augerat, P., Belenguer, J., Benavent, E., Corberán, A., Naddef, D., and Rinaldi, G., *Computational results with a branch and cut code for the capacitated vehicle routing problem.* IMAG, 1995.

Babin, G., Deneault, S., and Laporte, G., "Improvements to the or-opt heuristic for the symmetric travelling salesman problem," *Journal of the Operational Research Society*, vol. 58, no. 3, pp. 402–407, 2007.

Baker, B. M. and Ayechew, M., "A genetic algorithm for the vehicle routing problem," *Computers& Operations Research*, vol. 30, no. 5, pp. 787–800, 2003.

Baldacci, R., Hadjiconstantinou, E., and Mingozzi, A., "An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation," *Operations Research*, vol. 52, no. 5, pp. 723–738, 2004.

Baldacci, R., Toth, P., and Vigo, D., "Recent advances in vehicle routing exact algorithms," *4OR*, vol. 5, no. 4, pp. 269–298, 12 2007, copyright - Springer-Verlag 2007; Last updated - 2012-06-29.

Baldacci, R., Toth, P., and Vigo, D., "Exact algorithms for routing problems under vehicle capacity constraints," *Annals of Operations Research*, vol. 175, no. 1, pp. 213–245, 2010.

Balinski, M. L. and Quandt, R. E., "On an integer program for a delivery problem," *Operations Research*, vol. 12, no. 2, pp. 300–304, 1964.

Bektas, T., "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

Bierwirth, C., Mattfeld, D. C., and Watson, J.-P., "Landscape regularity and random walks for the job-shop scheduling problem," in *Evolutionary Computation in Combinatorial Optimization.* Springer, 2004, pp. 21–30.

Brownlee, J., *Clever algorithms: nature-inspired programming recipes.* Jason Brownlee, 2011.

Bullnheimer, B., Hartl, R. F., and Strauss, C., "Applying the ant system to the vehicle routing problem," in *Meta-Heuristics.* Springer, 1999, pp. 285–296.

Bullnheimer, B., Hartl, R. F., and Strauss, C., "An improved ant system algorithm for the vehicle routing problem," *Annals of operations research*, vol. 89, pp. 319–328, 1999.

Burke, L., "conscientious neural nets for tour construction in the traveling salesman problem: the vigilant net," *Computers & operations research*, vol. 23, no. 2, pp. 121–129, 1996.

Cheeseman, P., Kanefsky, B., and Taylor, W. M., "Where the really hard problems are." in *IJCAI*, vol. 91, 1991, pp. 331–340.

Chen, A.-l., Yang, G.-k., and Wu, Z.-m., "Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem," *Journal of Zhejiang University Science A*, vol. 7, no. 4, pp. 607–614, 2006.

Christofides, N., Mingozzi, A., and Toth, P., "The vehicle routing problem," in *Combinatorial Optimization*, Mingozzi, N., Toth, P., and Sandi, C., Eds. London: John Wiley and Sons, 1979.

Christofides, N. and Eilon, S., "An algorithm for the vehicle-dispatching problem," *OR*, pp. 309–318, 1969.

Clarke, G. u. and Wright, J. W., "Scheduling of vehicles from a central depot to a number of delivery points," *Operations research*, vol. 12, no. 4, pp. 568–581, 1964.

Cochrane, E. and Beasley, J., "The co-adaptive neural network approach to the euclidean travelling salesman problem," *Neural Networks*, vol. 16, no. 10, pp. 1499–1525, 2003.

Colorni, A., Dorigo, M., and Maniezzo, V., "Distributed optimization by ant colonies," in *Proceedings of the first European conference on artificial life*, vol. 142. Paris, France, 1991, pp. 134–142.

Cook, W., *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, 2012.

Cordeau, J.-F., Laporte, G., Savelsbergh, M. W., and Vigo, D., "Vehicle routing," *Transportation, handbooks in operations research and management science*, vol. 14, pp. 367–428, 2007.

Cordeau, J.-F., Gendreau, M., Hertz, A., Laporte, G., and Sormany, J.-S., "New heuristics for the vehicle routing problem," in *Logistics Systems: Design and Optimization*, Langevin, A. and Riopel, D., Eds. Springer US, 2005, pp. 279–297. [Online]. Available: http://dx.doi.org/10.1007/0-387-24977-X_9

Cornuejols, G. and Harche, F., "Polyhedral study of the capacitated vehicle routing problem," *Mathematical Programming*, vol. 60, no. 1-3, pp. 21–52, 1993.

Créput, J.-C. and Koukam, A., "The memetic self-organizing map approach to the vehicle routing problem," *Soft Computing*, vol. 12, no. 11, pp. 1125–1141, 2008.

Dantzig, G., Fulkerson, R., and Johnson, S., "Solution of a large-scale traveling-salesman problem," *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

Dantzig, G. B. and Ramser, J. H., "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

Desrochers, M. and Verhoog, T., "A matching based savings algorithm for the vehicle routing problem," *Cahiers du GERAD*, 1989.

Dorigo, M. and Gambardella, L. M., "Ant colony system: a cooperative learning approach to the traveling salesman problem," *Evolutionary Computation, IEEE Transactions on*, vol. 1, no. 1, pp. 53–66, 1997.

Dorigo, M., Maniezzo, V., and Colorni, A., "Ant system: optimization by a colony of cooperating agents," *Systems, Man and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 1, pp. 29–41, 1996.

Dror, M. and Levy, L., "A vehicle routing improvement algorithm comparison of a greedy and a matching implementation for inventory routing," *Computers & Operations Research*, vol. 13, no. 1, pp. 33–45, 1986.

Durbin, R. and Willshaw, D., "An analogue approach to the travelling salesman problem using an elastic net method," *Nature*, vol. 326, no. 6114, pp. 689–691, 1987.

Eilon, S., Watson-Gandy, C., and Christofides, N., *Distribution management*. Griffin London, 1971.

Ergun, Ö., Orlin, J. B., and Steele-Feldman, A., "Creating very large scale neighborhoods out of smaller ones by compounding moves," *Journal of Heuristics*, vol. 12, no. 1-2, pp. 115–140, 2006.

Ester, M., Kriegel, H.-P., Sander, J., and Xu, X., "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Kdd*, vol. 96, no. 34, 1996, pp. 226–231.

Fahrion, R. and Wrede, M., "On a principle of chain-exchange for vehicle-routeing problems (1-vrp)," *Journal of the Operational Research Society*, pp. 821–827, 1990.

Finke, G., Claus, A., and Gunn, E., "A two-commodity network flow approach to the traveling salesman problem," *Congressus Numerantium*, vol. 41, no. 1, pp. 167–178, 1984.

Fisher, M. L., "Optimal solution of vehicle routing problems using minimum k-trees," *Operations Research*.

Fisher, M. L. and Jaikumar, R., "A generalized assignment heuristic for vehicle routing," *Networks*, vol. 11, no. 2, pp. 109–124, 1981.

Fort, J., "Solving a combinatorial problem via self-organizing process: an application of the kohonen algorithm to the traveling salesman problem," *Biological Cybernetics*, vol. 59, no. 1, pp. 33–40, 1988.

Fukasawa, R., Longo, H., Lysgaard, J., Aragão, M. P. d., Reis, M., Uchoa, E., and Werneck, R. F., "Robust branch-and-cut-and-price for the capacitated vehicle routing problem," *Mathematical programming*, vol. 106, no. 3, pp. 491–511, 2006.

Garey, M. R. and Johnson, D. S., *Computers and Intractability*. wh freeman, 2002, vol. 29.

Garvin, W., Crandall, H., John, J., and Spellman, R., "Applications of linear programming in the oil industry," *Management Science*, vol. 3, no. 4, pp. 407–430, 1957.

Gaskell, T., "Bases for vehicle fleet scheduling," *OR*, pp. 281–295, 1967.

Gavish, B. and Graves, S., "The traveling salesman problem and related problems," 1979.

Gavish, B. and Graves, S., "Scheduling and routing in transportation and distributions systems: Formulations and new relaxations." 1982.

Gendreau, M., Hertz, A., and Laporte, G., "New insertion and postoptimization procedures for the traveling salesman problem," *Operations Research*, vol. 40, no. 6, pp. 1086–1094, 1992.

Gendreau, M., Laporte, G., and Potvin, J.-Y., "Metaheuristics for the capacitated vrp," in *The Vehicle Routing Problem*, Toth, P. and Vigo, D., Eds. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications, 2002, pp. 129–154.

Gendreau, M. and Potvin, J.-Y., "Metaheuristics in combinatorial optimization," *Annals of Operations Research*, vol. 140, no. 1, pp. 189–213, 2005.

Gendreau, M., Potvin, J.-Y., Bräumlaysy, O., Hasle, G., and Løkketangen, A., "Metaheuristics for the vehicle routing problem and its extensions: A categorized bibliography," in *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*, Golden, B. L., Raghavan, S., and Wasil, E. A., Eds. Springer, 2008, pp. 143–169.

Ghaziri, H., "Supervision in the self-organizing feature map: Application to the vehicle routing problem," in *Meta-Heuristics*. Springer, 1996, pp. 651–660.

Ghaziri, H. and Osman, I. H., "Self-organizing feature maps for the vehicle routing problem with backhauls," *Journal of Scheduling*, vol. 9, no. 2, pp. 97–114, 2006.

Ghaziri, H. E., "Solving routing problems by a self-organizing map," in *Artificial Neural Networks, 1991, International Conference on Artificial Neural Networks*. ICANN, 1991, pp. 829–834.

Gillett, B. E. and Miller, L. R., "A heuristic algorithm for the vehicle-dispatch problem," *Operations research*, vol. 22, no. 2, pp. 340–349, 1974.

Glover, F., "Future paths for integer programming and links to artificial intelligence," *Computers & operations research*, vol. 13, no. 5, pp. 533–549, 1986.

Golden, B. L., Assad, A. A., and Wasil, E. A., "Routing vehicles in the real world: applications in the solid waste, beverage, food, dairy, and newspaper industries," *The vehicle routing problem*, vol. 9, pp. 245–286, 2002.

Golden, B. L., Magnanti, T. L., and Nguyen, H. Q., "Implementing vehicle routing algorithms," *Networks*, vol. 7, no. 2, pp. 113–148, 1977.

Golden, B. L., Raghavan, S., and Wasil, E. A., *The Vehicle Routing Problem: Latest Advances and New Challenges: latest advances and new challenges*. Springer Science & Business Media, 2008, vol. 43.

Golden, B. L., Wasil, E. A., Kelly, J. P., and Chao, I.-M., "The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results," in *Fleet management and logistics*. Springer, 1998, pp. 33–56.

Goldstein, M., "Self-organizing feature maps for the multiple traveling salesmen problem," in *Proceedings of the IEEE International Conference on Neural Networks*. IEEE, 1990, pp. 258–261.

Hassoun, M. H., *Fundamentals of artificial neural networks*. MIT press, 1995.

Herrera, F. and Lozano, M., "Adaptation of genetic algorithm parameters based on fuzzy logic controllers," *Genetic Algorithms and Soft Computing*, vol. 8, pp. 95–125, 1996.

Holland, J. H., *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, Ann Arbor, 1975.

Hooker, J. N., "Testing heuristics: We have it all wrong," *Journal of Heuristics*, vol. 1, no. 1, pp. 33–42, 1995.

Hopfield, J. J., "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

Hopfield, J. J., "Neurons with graded response have collective computational properties like those of two-state neurons," *Proceedings of the national academy of sciences*, vol. 81, no. 10, pp. 3088–3092, 1984.

Hopfield, J. J. and Tank, D. W., "neural computation of decisions in optimization problems," *Biological cybernetics*, vol. 52, no. 3, pp. 141–152, 1985.

Horvitz, E., Ruan, Y., Gomes, C., Kautz, H., Selman, B., and Chickering, M., "A bayesian approach to tackling hard computational problems," in *Proceedings of the Seventeenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 2001, pp. 235–244.

Hsu, C.-Y., Tsai, M.-H., and Chen, W.-M., "A study of feature-mapped approach to the multiple travelling salesmen problem," in *Circuits and Systems, 1991., IEEE International Sympoisum on*. IEEE, 1991, pp. 1589–1592.

Hutter, F., Xu, L., Hoos, H. H., and Leyton-Brown, K., "Algorithm runtime prediction: Methods & evaluation," *Artificial Intelligence*, vol. 206, pp. 79–111, 2014.

Jain, A. K., Mao, J., and Mohiuddin, K., "Artificial neural networks: A tutorial," *Computer*, vol. 29, no. 3, pp. 31–44, 1996.

Jain, A. K., Murty, M. N., and Flynn, P. J., "Data clustering: a review," *ACM computing surveys (CSUR)*, vol. 31, no. 3, pp. 264–323, 1999.

Kaski, S. and Kohonen, T., "Exploratory data analysis by the self-organizing map: Structures of welfare and poverty in the world," in *Neural Networks in Financial Engineering. Proceedings of the Third International Conference on Neural Networks in the Capital Markets*. World Scientific, 1996, pp. 498–507.

Kaski, S., "Data exploration using self-organizing maps," in *ACTA POLYTECHNICA SCANDINAVICA: MATHEMATICS, COMPUTING AND MANAGEMENT IN ENGINEERING SERIES NO. 82*. Citeseer, 1997.

Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P., *et al.*, "Optimization by simulated annealing," *science*, vol. 220, no. 4598, pp. 671–680, 1983.

Kohonen, T., "Essentials of the self-organizing map," *Neural Networks*, vol. 37, pp. 52–65, 2013.

Kohonen, T., "Clustering, taxonomy, and topological maps of patterns," in *Proceedings of the sixth international conference on pattern recognition*. Washington, DC: IEEE Computer Soc. Press, 1982, pp. 114–128.

Kohonen, T., "Self-organized formation of topologically correct feature maps," *Biological Cybernetics*, vol. 43, no. 1, pp. 59–69, 1982.

Kohonen, T., *Self-Organizing Maps*, 3rd ed. Springer-Verlag, 2001.

Kotthoff, L., "Algorithm selection for combinatorial search problems: A survey," *AI Magazine*, vol. 35, no. 3, pp. 48–60, 2014.

Kraaijveld, M., Mao, J., and Jain, A. K., "A nonlinear projection method based on kohonen's topology preserving maps," *Neural Networks, IEEE Transactionis on*, vol. 6, no. 3, pp. 548–559, 1995.

Lamagna, E. A., "Infeasible computation: Np-complete problems," *ABACUS*, vol. 4, no. 3, pp. 18–33, 1987.

Laporte, G., "What you should know about the vehicle routing problem," *Naval Research Logistics (NRL)*, vol. 54, no. 8, pp. 811–819, 2007.

Laporte, G., "Fifty years of vehicle routing," *Transportation Science*, vol. 43, no. 4, pp. 408–416, 2009.

Laporte, G., Gendreau, M., Potvin, J.-Y., and Semet, F., "Classical and modern heuristics for the vehicle routing problem," *International transactions in operational research*, vol. 7, no. 4-5, pp. 285–300, 2000.

Laporte, G. and Nobert, Y., "A branch and bound algorithm for the capacitated vehicle routing problem," *Operations-Research-Spektrum*, vol. 5, no. 2, pp. 77–85, 1983.

Laporte, G. and Nobert, Y., "Exact algorithms for the vehicle routing problem," *North-Holland Mathematics Studies*, vol. 132, pp. 147–184, 1987.

Laporte, G., Nobert, Y., and Desrochers, M., "Optimal routing under capacity and distance restrictions," *Operations research*, vol. 33, no. 5, pp. 1050–1073, 1985.

Laporte, G. and Semet, F., "Classical heuristics for the capacitated vrp," in *The Vehicle Routing Problem*, Toth, P. and Vigo, D., Eds. Philadelphia: SIAM Monographs on Discrete Mathematics and Applications, 2002, pp. 109–128.

Lawler, E., Lenstra, J., Rinnooy Kan, A., and Shmoys, D., *The Traveling Salesman Problam*, 1st ed. John Wiley and Sons Ltd., 1985.

Lee, C. C., "Fuzzy logic in control system: Fuzzy logic in controller part i," *Systems, Man and Cybernetics, IEEE Transactions on*, vol. 20, no. 2, pp. 404–418, 1990.

Leyton-Brown, K., Nudelman, E., Andrew, G., McFadden, J., and Shoham, Y., "A portfolio approach to algorithm selection," in *IJCAI*, vol. 1543, 2003, p. 2003.

Leyton-Brown, K., Nudelman, E., and Shoham, Y., "Learning the empirical hardness of optimization problems: The case of combinatorial auctions," in *Principles and Practice of Constraint Programming-CP 2002*. Springer, 2002, pp. 556–572.

Lin, S., "Computer solutions of the traveling salesman problem," *Bell System Technical Journal, The*, vol. 44, no. 10, pp. 2245–2269, 1965.

Lin, S. and Kernighan, B. W., "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

Lysgaard, J., Letchford, A. N., and Eglese, R. W., "A new branch-and-cut algorithm for the capacitated vehicle routing problem," *Mathematical Programming*, vol. 100, no. 2, pp. 423–445, 2004.

Matsuyama, Y., "Self-organization via competition, cooperation and categorization applied to extended vehicle routing problems," in *Neural Networks, 1991., IJCNN-91-Seattle International Joint Conference on*, vol. 1. IEEE, 1991, pp. 385–390.

McClelland, J. L., Rumelhart, D. E., Group, P. R., *et al.*, "Parallel distributed processing," *Explorations in the microstructure of cognition*, vol. 2, 1986.

McCulloch, W. S. and Pitts, W., "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

Melssen, W., Wehrens, R., and Buydens, L., "Supervised kohonen networks for classification problems," *Chemometrics and Intelligent Laboratory Systems*, vol. 83, no. 2, pp. 99–113, 2006.

Merz, P., "Advanced fitness landscape analysis and the performance of memetic algorithms," *Evolutionary Computation*, vol. 12, no. 3, pp. 303–325, 2004.

Mester, D. and Bräysy, O., "Active-guided evolution strategies for large-scale capacitated vehicle routing problems," *Computers & Operations Research*, vol. 34, no. 10, pp. 2964–2975, 2007.

Michael, A. and Takagi, H., "Dynamic control of genetic algorithms using fuzzy logic techniques," in *Proceedings of the Fifth International Conference on Genetic Algorithms*. Citeseer, 1993, pp. 76–83.

Minsky, M. and Papert, S., "Perceptron: An introduction to computational geometry," *The MIT Press, Cambridge, expanded edition*, vol. 19, p. 88, 1969.

Mladenović, N. and Hansen, P., "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

Modares, A., Somhom, S., and Enkawa, T., "A self-organizing neural network approach for multiple traveling salesman and vehicle routing problems," *International Transactions in Operational Research*, vol. 6, no. 6, pp. 591–606, 1999.

Nelson, M. D., Nygard, K. E., Griffin, J. H., and Shreve, W. E., "Implementation techniques for the vehicle routing problem," *Computers & Operations Research*, vol. 12, no. 3, pp. 273–283, 1985.

Nygard, K. E., Juell, P., and Kadaba, N., "Neural networks for selective vehicle routing heuristics," *ORSA Journal on Computing*, vol. 2, no. 4, pp. 353–364, 1990.

Or, I., *Traveling salesman-type combinatorial problems and their relation to the logistics of regional blood banking*. Xerox University Microfilms, 1976.

Osman, I. H. and Laporte, G., "Metaheuristics: A bibliography," *Annals of Operations Research*, vol. 63, no. 5, pp. 511–623, 1996.

Osman, I. H., "Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem," *Annals of operations research*, vol. 41, no. 4, pp. 421–451, 1993.

Padberg, M. and Rinaldi, G., "Optimization of a 532-city symmetric traveling salesman problem by branch and cut," *Operations Research Letters*, vol. 6, no. 1, pp. 1–7, 1987.

Paessens, H., "The savings algorithm for the vehicle routing problem," *European Journal of Operational Research*, vol. 34, no. 3, pp. 336–344, 1988.

Papadimitriou, C. H. and Steiglitz, K., *Combinatorial Optimization: Algorithms and Complexity*. Courier Dover Publications, 1998.

Pornsing, C., "A particle swarm optmiization for the vehicle routing problem," Ph.D. dissertation, University of Rhode Island, 2014.

Potvin, J.-Y., "State-of-the-art surveythe traveling salesman problem: A neural network perspective," *ORSA Journal on Computing*, vol. 5, no. 4, pp. 328–348, 1993.

Potvin, J.-Y., "State-of-the art review- evolutionary algorithms for vehicle routing," *INFORMS Journal on Computing*, vol. 21, no. 4, pp. 518–548, 2009.

Potvin, J.-Y., Kervahut, T., Garcia, B.-L., and Rousseau, J.-M., "The vehicle routing problem with time windows part i: tabu search," *INFORMS Journal on Computing*, vol. 8, no. 2, pp. 158–164, 1996.

Potvin, J.-Y. and Robillard, C., "Clustering for vehicle routing with a competitive neural network," *Neurocomputing*, vol. 8, no. 2, pp. 125–139, 1995.

Prins, C., "A simple and effective evolutionary algorithm for the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 12, pp. 1985–2002, 2004.

Pureza, V. and Franca, P., "Vehicle routing problems via tabu search metaheuristic," 1991.

Rego, C. and Roucairol, C., "A parallel tabu search algorithm using ejection chains for the vehicle routing problem," in *Meta-Heuristics*. Springer, 1996, pp. 661–675.

Reimann, M., DOerner, K., and Hartl, R. F., "D-ants: Savings based ants divide and conquer the vehicle routing problem," *Computers & Operations Research*, vol. 31, no. 4, pp. 563–591, 2004.

Reimann, M., Stummer, M., and Doerner, K., "A savings based ant system for the vehicle routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. Morgan Kaufmann Publishers, Inc., 2002, pp. 1317–1326.

Rice, J. R., "The algorithm selection problem," *Advances in Computers*, vol. 15, pp. 65–118, 1976.

Ritter, H. and Schulten, K., "Kohonen's self-organizing maps: Exploring their computational capabilities," in *Neural Networks, 1988., IEEE International Conference on*. IEEE, 1988, pp. 109–116.

Rochat, y. and Taillar, É. D., "Probabilistic diversification and intensification in local search for vehicle routing," *Journal of heuristics*, vol. 1, no. 1, pp. 147–167, 1995.

Rojas, R., *Neural networks: a systematic introduction*. Springer, 1996.

Ropke, S. and Pisinger, D., "An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows," *Transportation science*, vol. 40, no. 4, pp. 455–472, 2006.

Rosenblatt, F., "The perceptron: A probabilistic model for information storage and organization in the brain." *Psychological review*, vol. 65, no. 6, p. 386, 1958.

Rosenblatt, F., *Principles of Neurodynamics*. Spartan Book, 1962.

Sammon, J. W., "A nonlinear mapping for data structure analysis," *IEEE Transactions on computers*, no. 5, pp. 401–409, 1969.

Sander, J., Ester, M., Kriegel, H.-P., and Xu, X., "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data mining and knowledge discovery*, vol. 2, no. 2, pp. 169–194, 1998.

Savelsbergh, M., "Computer aided routing," Ph.D. dissertation, Erasmus University, The Netherlands, 1988.

Schwardt, M. and Dethloff, J., "Solving a continuous location-routing problem by use of a self-organizing map," *International Journal of Physical Distribution & Logistics Management*, vol. 35, no. 6, pp. 390–408, 2005.

Smith, K. A., "Neural networks for combinatorial optimization: A review of more than a decade of research," *INFORMS Journal on Computing*, vol. 11, no. 1, pp. 15–34, 1999.

Smith-Miles, K. and Lopes, L., "Measuring instance difficulty for combinatorial optimization problems," *Computers & Operations Research*, vol. 39, no. 5, pp. 875–889, 2012.

Smith-Miles, K., van Hemert, J. I., and Lim, X. Y., "Understanding tsp difficulty by learning from evolved instances." *LION*, vol. 4, pp. 266–280, 2010.

Smith-Miles, K. A., "Cross-disciplinary perspectives on meta-learning for algorithm selection," *ACM Computing Surveys (CSUR)*, vol. 41, no. 1, p. 6, 2008.

Stadler, P. F. and Schnabl, W., "The landscape of the traveling salesman problem," *Physics Letters A*, vol. 161, no. 4, pp. 337–344, 1992.

Syed, M. N. and Pardalos, P. M., "Neural network models in combinatorial optimization," in *Handbook of Combinatorial Optimization*. Springer, 2013, pp. 2028–2087.

Tan, K., Tang, H., and Ge, S., "On parameter settings of hopfield networks applied to traveling salesman problems," *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 52, no. 5, pp. 994–1002, 2005.

Thompson, P. M. and Psaraftis, H. N., "Cyclic transfer algorithm for multivehicle routing and scheduling problems," *Operations research*, vol. 41, no. 5, pp. 935–946, 1993.

Torki, A., Somhon, S., and Enkawa, T., "A competitive neural network algorithm for solving vehicle routing problem," *Computers & industrial engineering*, vol. 33, no. 3, pp. 473–476, 1997.

Toth, P. and Vigo, D., *The vehicle routing problem*. SIAM Monographs on Discrete Mathematics and Applications, 2002.

Tsang, E. P., Borrett, J. E., and Kwan, A. C., "An attempt to map the performance of a range of algorithm and heuristic combinations," *Hybrid Problems, Hybrid Solutions*, vol. 27, p. 203, 1995.

Tuzun, D., Magent, M. A., and Burke, L. I., "Selection of vehicle routing heuristic using neural networks," *International Transactions in Operational Research*, vol. 4, no. 3, pp. 211–221, 1997.

Ultsch, A., "Self-organizing neural networks for visualization and classification," in *Information and Classification*, ser. Studies in Classification, Data Analysis and Knowledge Organization, opitz, O., Lausen, B., and Klar, R., Eds.

University of Malaga. "Neo: Networking and emerging optimization group." Jan. 2013. [Online]. Available: http://neo.lcc.uma.es/vrp/

Vakhutinsky, A. I. and Golden, B., "Solving vehicle routing problems using elastic nets," in *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, vol. 7. IEEE, 1994, pp. 4535–4540.

Van Breedam, A., *An Analysis of the Behavior of Heuristics for the Vehicle Routing Problem for a Selectrion of Problems with Vehicle-related, Customer-related, and Time-related Constraints*. RUCA, 1994.

van Hemert, J. I., "Property analysis of symmetric travelling salesman problem instances acquired through evolution," in *Evolutionary Computation in Combinatorial Optimization*. Springer, 2005, pp. 122–131.

van Hemert, J. I., "Evolving combinatorial problem instances that are difficult to solve," *Evolutionary Computation*, vol. 14, no. 4, pp. 433–462, 2006.

Vilalta, R. and Drissi, Y., "A perspective view and survey of meta-learning," *Artificial Intelligence Review*, vol. 18, no. 2, pp. 77–95, 2002.

Wacholder, E., Han, J., and Mann, R., "A neural network algorithm for the multiple traveling salesmen problem," *Biological Cybernetics*, vol. 61, no. 1, pp. 11–19, 1989.

Wehrens, R. and Buydens, L., "Self and super-organzing maps in r: the kohonen package," *Journal of Statistical Software*, vol. 21, no. 5, pp. 1–19, 2007.

Wen, U.-P., Lan, K.-M., and Shih, H.-S., "A review of hopfield neural networks for solving mathematical programming problems," *European Journal of Operational Research*, vol. 198, no. 3, pp. 675–687, 2009.

WERBOS, P., "Beyond regression: New tools for prediction and analysis in the behavioral sciences," *PhD thesis, Harvard University*, 1974.

Willard, J., "Vehicle routing using r-optimal tabu search," 1989.

Wilson, G. and Pawley, G., "On the stability of the travelling salesman problem algorithm of hopfield and tank," *Biological Cybernetics*, vol. 58, no. 1, pp. 63–70, 1988.

Wolpert, D. H. and Macready, W. G., "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

Xu, J. and Kelly, J. P., "A network flow-based tabu search heuristic for the vehicle routing problem," *Transportation Science*, vol. 30, no. 4, pp. 379–393, 1996.

Xu, L., Hutter, F., Hoos, H. H., and Leyton-Brown, K., "Satzilla-07: the design and analysis of an algorithm portfolio for sat," in *Principles and Practice of Constraint Programming–CP 2007*. Springer, 2007, pp. 712–727.

Yellow, P., "A computational modification to the savings method of vehicle scheduling," *Operational Research Quarterly*, pp. 281–283, 1970.