Meta-Heuristics for the Multiple Trip Vehicle Routing Problem with Backhauls

A THESIS SUBMITTED TO
THE UNIVERSITY OF KENT
IN THE SUBJECT OF MANAGEMENT SCIENCE
FOR THE DEGREE
OF DOCTOR OF PHILOSOPHY

By
Naveed Ahmed Wassan
May 2016

Abstract

With the growing and more accessible computational power, the demand for robust and sophisticated computerised optimisation is increasing for logistical problems. By making good use of computational technologies, the research in this thesis concentrates on efficient fleet management by studying a class of vehicle routing problems and developing efficient solution algorithms.

The literature review in this thesis looks at VRPs from various development angles. The search reveals that from the problem modelling side clear efforts are made to bring the classical VRP models closer to reality by developing various variants. However, apart from the real VRP applications (termed as 'rich' VRPs), it is also noticeable that these classical VRP based variants address merely one or two additional characteristics from the real routing problem issues, concentrating on either operational (fleet management) or tactical (fleet acquisition) aspects. This thesis certainly hopes to add to one of those good efforts which have helped in bringing the VRPs closer to reality through addressing both the operational as well as the tactical aspects.

On the solution methodologies development side, the proposed research noted some considerable and impressive developments. Although, it is well established that the VRPs belong to the *NP*-hard combinatorial class of problems, there are considerable efforts on the development of exact methods. However the literature is full of a variety of heuristic methodologies including the classical and the most modern hybrid approaches. Among the hybrid approaches, the most recent one noted is *mat-heuristics*

that combine heuristics and mathematical programming techniques to solve combinatorial optimisation problems. The *mat-heuristics* approaches appear to be comparatively in its infant age at this point in time. However this is an exciting area of research which seeks more attention in the literature. Hence, a good part of this research is devoted to the development of a hybrid approach that combines heuristics and mathematical programming techniques.

When reviewing the specific literature on the VRP problems focused in this thesis, the vehicle routing problem with backhauls (VRPB) and the multiple trip vehicle routing problem (MT-VRP), there is not sufficient development on the problem modelling side in terms of bringing these two problems closer to the reality. Hence, to fill the gap this thesis introduces and investigates a new variant, the multiple trip vehicle routing problem with backhauls (MT-VRPB) that combines the above two variants of the VRP. The problem is first described thoroughly and a new ILP (Integer Linear Programming) mathematical formulation of the MT-VRPB along with its possible variations is presented. The MT-VRPB is then solved optimally by using CPLEX along with providing an illustrative example showing the validation of the mathematical formulation. As part of the contribution, a large set of MT-VRPB data instances is created which is made available for future benchmarking.

The CPLEX implementation produced optimal solutions for a good number of small and medium size data instances of the MT-VRPB and generated lower bounds for all instances. The CPLEX success may be considered as modest, but the produced results proved very important for the validation of the heuristic results produced in the thesis.

To solve the larger instances of the MT-VRPB, a two level VNS algorithm called '*Two-Level VNS'* is developed. It was noticed from the literature that the choice of using VNS

for the VRPs has increased in recent literature due to its simplicity and speed. However our initial experiments with the classical VNS indicated that the algorithm is more inclined towards the intensification side. Hence, the *Two-Level VNS* is designed to obtain a maximum balance of the diversification and the intensification during the search process. It is achieved by incorporating a sub-set of neighbourhood structures and a sus-set of local search refinement routines and hence, a full set of neighbourhood structures and a full set of local search refinement routines at two levels of the algorithm respectively. The algorithm found very encouraging results when compared with the solutions found by CPLEX. These findings in this thesis demonstrate the power of VNS yet again in terms of its speed, simplicity and efficiency.

To investigate this new variant further, we developed an algorithm belonging to the new class of the hybrid methodologies, i.e., *mat-heuristics*. A hybrid collaborative sequential *mat-heuristic* approach called the CSMH to solve the MT-VRPB is developed. The exact method approach produced in Chapter 4 is then hybridised with the *Two-Level VNS* algorithm developed in Chapter 5. The overall performance of the CSMH remained very encouraging in terms of the solution quality and the time taken on average compared with the CPLEX and the *Two-Level VNS* meta-heuristic.

To demonstrate the power and effectiveness of our methodologies, we tested the designed algorithms on the two special versions of the VRP (i.e., VRPB and MT-VRP) to assess whether they are efficient and dynamic enough to solve a range of VRP variants. Hence the *Two-Level VNS* and the CSMH algorithms developed to solve the MT-VRPB are adapted accordingly and implemented to solve the two above variants separately. The algorithms produced very competitive results for the benchmark data sets when compared to the best known solutions from the literature. The successful

implementations of these algorithms on the three VRP models with only minor amendments prove their generalizability and their robustness.

The results in this research show that significant cost savings could be obtained by choosing the right fleet size and better vehicle utilisations with multiple trips and backhauling. Hence, the research proved the justification of studying this interesting combination. Moreover, the problem modelling, efficient algorithm design and implementation, and the research results reveal some vital information and implications from the managerial point of view in terms of making the tactical (fleet acquisition) and the operational (fleet management) decisions in a more informative manner.

Acknowledgements

I would like to express my gratitude to my supervisors Dr Gabor Nagy and Professor Said Salhi for their invaluable guidance and supervision throughout the PhD project. I have benefitted greatly from their scientific knowledge.

Needless to say, the guidance and support throughout all these years from my brother Dr Niaz Wassan has been inestimable. I would also like to thank all friends who have helped and supported me.

Last, but not least, the support, love and prayers of my family are deeply cherished, especially my mother whose unconditional prayers have always been with me.

Table of Contents

Table of Contents viii

List of Tables xii

List of Figures xiv

1. Introduction 1

1.1. Introduction and Motivation 1

1.2. The Multiple Trip Vehicle Routing Problem with Backhauls 4

1.2. Aims and Objectives of the Thesis 7

1.3. Outline of the Thesis 7

2. **The Vehicle Routing Problem: Models and Solution Methods 10**

2.1. The Vehicle Routing Problem and its Variants 10

2.1.1. The Evolution of the Vehicle Routing Problem 10

2.1.2. Definition of the Vehicle Routing Problem: 11

2.1.3. VRP Variants 11

2.1.3.1. The Periodic VRP 13

2.1.3.2. The Multiple Trip VRP 14

2.1.3.3. The Multiple-Depot VRP 14

2.1.3.4. The Mix Fleet VRP 15

2.1.3.5. The VRP with Time Windows 15

2.1.3.6. The Split Delivery VRP 15

2.1.3.7. The classical VRP with Backhauls 16

2.1.3.8. The VRP with Mixed Deliveries and Pickups 16

2.1.3.9. The VRP with Simultaneous Deliveries and Pickups 17

2.1.4. Future of the VRP 17

2.2. Combinatorial Optimisation: Problems and Algorithms 18

2.2.1. Combinatorial Optimisation Problems 18

2.2.2. The term Algorithm 19

2.3. Solution Techniques for the Vehicle Routing Problems: An Overview 21

2.3.1. Exact Methods 22

2.3.2. Heuristic Methods 22

2.4. Examples of Exact Methods 27

2.4.1. The Branch-and-Bound Method 27

2.5. Examples of Heuristic Methods 29

2.5.1. Construction-based Heuristics 30

2.5.1.1. The Sweep Algorithm 30

2.5.1.2. Other Construction-based Heuristic for VRPs ... 31
2.5.2. Intra- and Inter-Route Improvement Heuristics ... 32
2.5.2.1. Transfer Heuristics ... 32
2.5.2.2. Swap Heuristics ... 33
2.5.2.3. Other Improvement Heuristics for VRPs ... 33
2.6. Examples of Metaheuristic Methods ... 33
2.6.1. Variable Neighbourhood Search ... 34
2.6.2. Large Neighbourhood Search ... 36
2.6.3. Other Metaheuristic Methods for VRPs ... 37
2.7. Hybrid Methods ... 38
2.8. Summary ... 39
3. Literature Review of the VRPB and the MT-VRP ... 41
3.1. An Overview of the VRPB ... 41
3.2. Solution Methods for the VRPB ... 43
3.2.1. Exact Methods ... 43
3.2.2. Heuristic Methods ... 44
3.2.3. Metaheuristic Methods ... 47
3.2.4. Studies in VRPB-related areas ... 53
3.3. An Overview of the MT-VRP ... 53
3.4. Solution methods for the MT-VRP ... 55
3.4.1. Exact Methods ... 55
3.4.2. Heuristic Methods ... 56
3.4.3. Metaheuristic Methods ... 58
3.4.4. Studies in MT-VRP related areas ... 62
3.4.5. Studies in which VRPB and MT-VRP are addressed in a combined way ... 63
3.5. Summary ... 63
4. The Multiple Trip Vehicle Routing Problem with Backhauls: Formulation and Analysis ... 65
4.1. The Multiple Trip Vehicle Routing Problem with Backhauls ... 65
4.1.1. Description of the MT-VRPB ... 66
4.1.2. Graph theoretical definition of the MT-VRPB ... 68
4.2. Exact methods options for the MT-VRPB ... 69
4.3. Mathematical Formulation of the MT-VRPB ... 71
4.3.1. Formulation of the basic case ... 71
4.3.2. Model complexity ... 73
4.3.3. Model variants and restricted problems ... 74
4.4. Significance of the MT-VRPB ... 75
4.5. Utility of IBM ILOG CPLEX optimisation studio ... 76
4.6. Validation of the MT-VRPB formulation ... 77
4.7. Generation of a new data set for the MT-VRPB ... 79

4.8. CPLEX Results and Analysis *82*
4.8.1. Relevance of the results 91
4.9. Summary *95*

5. A Two-Level Variable Neighbourhood Search Algorithm for the Multiple-Trip Vehicle Routing Problem with Backhauls 96

5.1. Two-Level VNS Algorithm: An Overview *96*
5.1.1. An overview of the algorithm 97
5.2. Initial solution (Phase I) *102*
5.2.1. Solving the Assignment Problem 104
5.3. Neighbourhoods used in the Two-Level VNS Algorithm (Phase II) *107*
5.4. Multi-layer local search optimiser framework (local search stage) *112*
5.5. Solving the Bin Packing Problem (Phase III) *114*
5.6. Computational Experience *118*
5.6.1. Introduction and Computer Details 118
5.6.2. Results and analysis 120
5.6.2.1. Search diversification and intensification analysis 126
5.7 Summary *133*

6. Solving the MT-VRPB using a Collaborative Sequential Mat-heuristic approach 134

6.1. The Mat-heuristic Approaches. *134*
6.1.1. Matheuristics for VRPs: Brief Literature Review 136
6.2. The Collaborative Sequential Approach for the MT-VRPB *140*
6.3. Computational Experience *146*
6.3.1. Data Set 146
6.3.2. The CSMH execution times 146
6.3.3. The CSMH algorithm performance 147
6.3.4. Comparison of the CSMH vs CPLEX results 149
6.3.5. Comparison of the CSMH and the *Two-Level VNS* results 151
6.4. Summary *166*

7. Adaptation of the *Two-Level VNS* and Mat-heuristic to the VRPB and the MT-VRP 167

7.1. The case of the VRPB *167*
7.1.1. VRPB Formulation 168
7.1.2. The *Two-Level VNS* Algorithm for the VRPB 171
7.1.2.1. Details of the VRPB Computations and the Data sets 174
7.1.2.2. Two-Level VNS VRPB Results and Analysis 176
7.1.3. Solving the VRPB with Mat-heuristic (CSMH algorithm) 181
7.1.3.1. CSMH VRPB Results and Analysis 182
7.2. The case of the MT-VRP *187*

7.2.1. Formulation of the Basic Case 187
7.2.2. The *Two-Level VNS* methodology for the MT-VRP 189
7.2.2.1. Details of MT-VRP Computations and the Data sets 194
7.2.2.2. Two-Level VNS MT-VRP Results and Analysis 195
7.2.3. Solving the MT-VRP with the Mat-heuristic (CSMH algorithm) 201
7.2.3.1. CSMH MT-VRP Results and Analysis 201

7.3. Summary 205

8. Conclusions 208

8.1. Research Summary 208

8.2. Future Research 215

Bibliography 218

Appendix A: 242

Connecting CPLEX with Microsoft Visual Studio 242

Figure a.1: Concert technology libraries for different operating systems Source: User's Manual for CPLEX V12.5 243

C++ code of the MIPstart 243

Contributions to the subject knowledge 244

List of Tables

Table 2.1: Methods used in tackling the VRP and its Variants 24

Table 4.1: The MT-VRPB data *set-1* with original conventions and z* found with free fleet ...80

Table 4.2: The details of the MT-VRPB data *set-1* 80

Table 4.3: CPLEX solutions for data *set-1* with $\boldsymbol{T_1}$ (2-hours running time) 83

Table 4.4: CPLEX solutions for data *set-1* with $\boldsymbol{T_2}$ (2-hours running time) 85

Table 4.5: Vehicle utilisation cost comparison of the free fleet VRPB and the MT-VRPB solutions for $\boldsymbol{T_1}$ 87

Table 4.6: Vehicle utilisation cost comparison of the free fleet VRPB and the MT-VRPB solutions for $\boldsymbol{T_2}$ 88

Table 4.7: Summary CPLEX results and average time for $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$ 89

Table 4.8: Comparison of CPLEX with 2 hours vs CPLEX with 15 hours for $\boldsymbol{T_1}$.....91

Table: 4.9: Comparison of CPLEX with 2 hours vs CPLEX with 15 hours for $\boldsymbol{T_2}$ 91

Table 4.10: Comparison of the free fleet VRPB and the MT-VRPB solutions in terms of vehicle savings (for small and medium instances $\boldsymbol{T_1}$).....93

Table 4.11: Comparison of the free fleet VRPB and the MT-VRPB solutions in terms of vehicle savings (for small and medium instances $\boldsymbol{T_2}$).....94

Table 5.1: The comparison of the *Two-Level VNS* with CPLEX (data *set-1*: $\boldsymbol{T_1}$ & $\boldsymbol{T_2}$).....121

Table 5.2: Detailed comparison of the *Two-Level VNS* with CPLEX for the data *set-1* ($\boldsymbol{T_1}$) ...122

Table 5.3: Detailed comparison of the *Two-Level VNS* with CPLEX for the data *set-1* ($\boldsymbol{T_2}$) ...124

Table 5.4: The number of times each neighbourhood leads towards better quality solution on average for each instance 127

Table 6.1: Summary of the CSMH algorithm solutions (data *set-1*: $\boldsymbol{T_1}$ & $\boldsymbol{T_2}$).....148

Table 6.2: Summary comparison of the CSMH vs CPLEX (data *set-1*: $\boldsymbol{T_1}$ & $\boldsymbol{T_2}$).....150

Table 6.3: Comparison of the CSMH vs the *Two-Level VNS* (data *set-1*: $\boldsymbol{T_1}$ & $\boldsymbol{T_2}$).....151

Table 6.4: Comparison of CPLEX, *Two-Level VNS* and CSMH (data *set-1*: $\boldsymbol{T_1}$ & $\boldsymbol{T_2}$).....152

Table 6.5: Detailed results of the CSMH algorithm for the data *set-1* ($\boldsymbol{T_1}$) 153

Table 6.6: Detailed results of the CSMH algorithm solutions for the data *set-1* ($\boldsymbol{T_2}$) 155

Table 6.7: Detailed comparison results of the CSMH vs CPLEX for the data *set-1* ($\boldsymbol{T_1}$) 157

Table 6.8: Detailed comparison results of the CSMH vs CPLEX for the data *set-1* ($\boldsymbol{T_2}$) 159

Table 6.9: Comparison of the lower bounds produced by CPLEX and CSMH for $\boldsymbol{T_1}$ and $\boldsymbol{T_2}$ .161

Table 6.10: Detailed comparison results of the CSMH vs *Two-Level VNS* for the data *set-1* ($\boldsymbol{T_1}$) 162

Table 6.11: Detailed comparison results of the CSMH vs *Two-Level VNS* for the data *set-1* ($\boldsymbol{T_2}$) 164

Table 7.1: Processor used and the number of run for the published algorithms and the proposed *Two-Level VNS* 177

Table 7.2: Comparison of the best VRPB algorithms with *Two-Level VNS* (data *set-2*).....178

Table 7.3: Comparison of the *Two-Level VNS* with the best algorithms (data *set-3*) 178

Table 7.4: Detailed results of the *Two-Level VNS* vs the Best-known (data *set-2*).....179

Table 7.5: Detailed results of the *Two-Level VNS* vs the Best-known (data *set-3*).....180

Table 7.6: Comparison of the CSMH with the *Two-Level VNS* and the best VRPB algorithms in the literature (data *set-2*) 182

Table 7.7: Comparison of the CSMH with the *Two-Level VNS* and the best VRPB algorithms in the literature (data *set-3*) 183

Table 7.8: Detailed results of the CSMH algorithm (data *set-2*) 184

Table 7.9: Detailed results of the CSMH algorithm (data *set-3*) 185

Table 7.10: Detailed results for 42 instances in G1 (data *set-4*) 198

Table 7.11: Detailed feasible results for 56 instances in G2 (data *set-4*).....199

Table 7.12: Detailed non-feasible results for 5 instances in G3 (data *set-4*) 200

Table 7.13: Average time (in seconds) for the problem classes of *set-4* 200

Table 7.14: Detailed results of the CSMH for 42 instances in G1 (data *set-4*).....203

Table 7.15: Comparison of the CSMH with some best algorithms for 42 instances in G1 (data *set-4*) 204

List of Figures

Figure 1.1: An example of the MT-VRPB 6

Figure 2.1: An illustrative example of the VRP 12

Figure 2.3: A visual representation of the sweep procedure 31

Figure 3.1: An illustrative example of the VRPB 42

Figure 3.2: An illustrative example of the MT-VRP 54

Figure 4.1: An example of the MT-VRPB 68

Figure 4.2: The numerical test instance data 77

Figure 4.3: The CPLEX solution for test instance 78

Figure 4.4: All feasible solutions for test instance 78

Figure 5.1: Algorithmic steps of the *Two-Level VNS* for MT-VRPB 99

Figure 5.2: Pseudo code for the *Two-Level VNS* 101

Figure 5.3: Pseudo code for the BPP 102

Figure 5.4: An illustrative example of sweep procedure for the MT-VRPB 103

Figure 5.5: LH and BH open-ended routes (Problem instance eil22_50 of data *set-2*) 104

Figure 5.6: Distance matrix of end nodes 105

Figure 5.7: Optimal matching obtained by CPLEX 106

Figure 5.8: Combined LH+BH routes (problem instance no: eil22_50) 106

Figure 5.9: An illustrative example of the 1-insertion (intra-route) refinement routine 108

Figure 5.10: An illustrative example of the 1-insertion (inter-route) refinement routine 109

Figure 5.11: An illustrative example of the 1-1 swap refinement routine 109

Figure 5.12: An illustrative example of the 2-2 swap refinement routine 110

Figure 5.13: An illustrative example of the 2-0 shift refinement routine 110

Figure 5.14: An illustrative example of the 2-1 swap refinement routine 111

Figure 5.15: The multi-layer local search optimiser framework flow chart 113

Figure 5.16: An illustrative example of data structure S_p.....115

Figure 5.17: An illustrative example of special data structure Sol_k 116

Figure 5.18: BPP flow chart.....117

Figure 5.19: An illustrative example of the Bisection Method.....118

Figure 5.20: Neighbourhoods diversification vs Intensification solution cost for data instance eil22_66_1_t1 128

Figure 5.21: Neighbourhoods diversification vs Intensification solution cost for data instance eil30_80_1_t1 129

Figure 5.22: Neighbourhoods diversification vs Intensification solution cost for data instance eil51_50_1_t1 130

Figure 5.23: Neighbourhoods diversification vs Intensification solution cost for data instance eilA76_50_1_t1.....131

Figure 5.24: Neighbourhoods diversification vs Intensification solution cost for data instance eilA76_50_1_t1.....132

Figure 6.1: The CSMH approach phases for the MT-VRPB.....141

Figure 6.2: MIPstart construction for the MT-VRPB test instance.....145

Figure 7.1: Algorithmic steps of the *Two-Level VNS* for VRPB 171

Figure 7.2: Algorithmic steps of the *Two-Level VNS* for the MT-VRP.....189

Figure 7.3: An illustrative example of the sweep procedure for the MT-VRP.....192

Figure 7.4: Illustration of all the refinement routines implemented for the MT-VRP.....193

Figure a.1: Concert technology libraries for different operating systems Source: User's Manual for CPLEX V12.5 243

Figure a.2: C++ code for the MIPstart 243

Chapter 1

Introduction

1.1. Introduction and Motivation

The discipline of logistics and supply chain management has seen a continuous and rapid development in recent years due to its importance in the economies of organisations and countries. Typically the role of supply chain management is perceived by most companies as an activity that adds value to their markets, hence it has become very significant to their strategic decision making. Due to evolving customers' demand, the companies want efficient delivery service without compromising the customer service quality and yet having profitable business. On the other hand, issues around the management of the operational physical distribution and collection activities are also being seen from the environmental perspectives, especially by big organizations as a part of corporate social responsibility, and governments and public service institutions (such as councils) as a part of their political agenda. Hence these institutions would like to see less traffic on the roads, meaning less pollution. These evolving demands have put constant pressure on logistics operations to be more efficient to satisfy these agendas. As a result, researchers around the globe are inspired to address these important economic and logistical issues more and more efficiently.

The main findings of the recent estimated figures published by the UK department of transport (DFT) show continuous significant increase in the past two years in all types of traffic, especially in light goods vehicles (LGV). Comparison trends of 2014 and 2015 show that "all motor vehicle traffic increased by 1.8 % to 312.4 billion vehicle miles, car traffic increased by 1.3 %, LGV traffic increased by 5.1 %, reaching a new peak of 45.5 billion vehicle miles, traffic volumes increased across all road classifications, minor rural road traffic increased the most, rising 4.9 % to reach 44.1 billion vehicle miles", (DFT, 2015).

Interestingly the DFT estimates of GDP in UK show an increase in the year ending March 2015. In particular, the four goods traffic related industrial groupings in the economy, i.e., production, construction, services, and agriculture, showed increases in their output over the same period. The above information shows that there is a positive correlation between the growth and increasing traffic volumes. The above findings become very vital if the GDP vs traffic relationship is associated with the emerging developing countries like China and India whose economies are growing much faster than the UK. The above statistics pinpoint the importance of this growing global issue and triggers a need to address the problem even more.

This thesis is yet another part towards the efforts that are being put into place to design more advanced and efficient algorithms to tackle these issues collectively. Vehicle routing as a physical distribution problem is considered to be one of the important modes of logistics; hence it has been studied enormously in the literature. However there is still a wide gap between the assumptions based theoretical studies carried out in academia and the reality of the industry. The research carried out around vehicle routing is concentrated mainly on the fuel costs, meaning reducing total distance travelled by a

fleet of vehicles while fulfilling customer demands. Other operating costs such as fleet and driver expenses are often ignored which can be vital from the management point of view to maintain competitive pricing advantage and retain profitability.

A number of software (e.g., CPLEX or Gurobi) exists that can solve small to medium size instances of vehicle routing logistics; however these are not capable enough to tackle complex and large size problems efficiently. The fact that the exact methods are unable to solve large instances of these well-established hard problems efficiently and the design of heuristics is being concentrated as problem-specific. Therefore, there is a strong desire in the research community to develop more generalised algorithms. The research in this thesis is an attempt to address some of those issues and gaps highlighted above by studying this crucial mode of logistics even further through modelling the backhauling aspect of the reverse logistics within existing vehicle routing problem (VRP) variants known as the multiple trips VRP; and to design efficient algorithms that are dynamic in terms of adaptability to be implementable to a range of VRP variants instances. In the multi-trip VRP a vehicle may be used more than once in planned period of time, hence the model can also be mapped with the light goods vehicle types that are increasingly used by online delivery companies.

Hence, the focus is to be on the issues highlighted above, i.e., economic and environmental costs gains, bridging the gap between the academia and the industry and developing new algorithms that are capable of solving instances from a range of VRP variants.

1.2. The Multiple Trip Vehicle Routing Problem with Backhauls

In this research the Multiple Trip Vehicle Routing Problem (MT-VRP) model is extended to include the backhauling aspect which we call The Multiple Trip Vehicle Routing Problem with Backhauls (MT-VRPB). The MT-VRPB combines the characteristics of the classical versions of two problems studied in the literature, i.e., the MT-VRP in which a vehicle may perform several routes (trips) within a given time period; and the vehicle routing problem with backhauls (VRPB) in which a vehicle may pick up goods to bring back to the depot once the deliveries are made. Therefore in the MT-VRPB a vehicle may not only make more than one trip in a given planning period but it can also collect goods at each trip. Since the MT-VRP and the VRPB have been studied independently in the literature, we first provide a brief description of these problems.

MT-VRP: The MT-VRP model is an extension of the classical VRP in which a vehicle may perform several routes (trips) within a given time period. Along with the typical VRP constraints an additional aspect is included in the model which involves the assignment of the optimised set of routes to the available fleet (Taillard *et al*. 1996).

VRPB: The VRPB is also an extension to the classical VRP that involves two types of customers, deliveries (linehauls) and pickups (backhauls). Typical additional constraints include: (i) each vehicle must perform all the deliveries before making any pickups; (ii) routes with only backhauls are disallowed, but routes with only linehauls can be performed. The reason behind this is that, in reverse logistics the linehaul (delivery) customers are considered more profitable (Goetschalckx and Jacobs-Blecha, 1989).

Both the MT-VRP and the VRPB are considered to be more valuable than the classical VRP in terms of cost savings and placing fewer numbers of vehicles on our roads. The MT-VRP saves a considerable number of vehicles by using the same vehicles more than once in a given planning time (Taillard *et al*. 1996). Whereas, in VRPB a vehicle is used to serve backhaul customers only once it has served the linehaul customers rather than using a separate vehicle to serve backhauls (Toth and Vigo, 1996, 1997, 1999).These features are very important from both the managerial and the ecological perspectives.

We believe, by combining the aspects of the above two models into the MT-VRPB adds even further value to the practice of the vehicle routing especially when it comes to the need to optimise a fixed or limited available fleet and utilizing fully the driver time to achieve strategic competitive advantage. To our knowledge, this is the first time this variant is being studied in the literature. However, there is one study that deals with time windows MT-VRPB-TW by Ong and Suprayogi (2011) where an ant colony optimisation algorithm is implemented. Below we present a detailed description of our MT-VRPB model.

MT-VRPB: The MT-VRPB can be described as a VRP problem with the additional possibilities of having vehicles involved in backhauling and multiple trips in a single planning period. The objective is to mimimise the total cost by reducing the total distance travelled and the number of vehicles used.

Problem characteristics:

1. A given set of customers is divided into two subsets, i.e., delivery (linehaul) and pickup (backhaul).
2. A homogenous fleet of vehicles.

3. A vehicle may perform more than one trip in a single planning period.
4. All delivery customers are served before any pickup ones on a route.
5. Vehicles are not allowed to service only backhauls on any route; however linehaul only routes are allowed.
6. Vehicle capacity constraints are imposed.
7. Note - The *route length* constraint is not imposed in this study, however the model is flexible to add this constraint if needed.

The above characteristics are established in the literature for the MT-VRP and VRPB (Taillard et al. (1996), Toth and Vigo (1996, 1997, 1999)). However, these characteristics are application dependent. For instance, heterogeneous fleet can be considered instead of homogeneous and a vehicle can be allowed to serve backhaul customers only.

Figure 1 presents a graphical example of the proposed MT-VRPB with three homogeneous types of vehicles and a planning period T; *Vehicle 1* performs two trips whereas vehicles 2 and 3 one trip each.

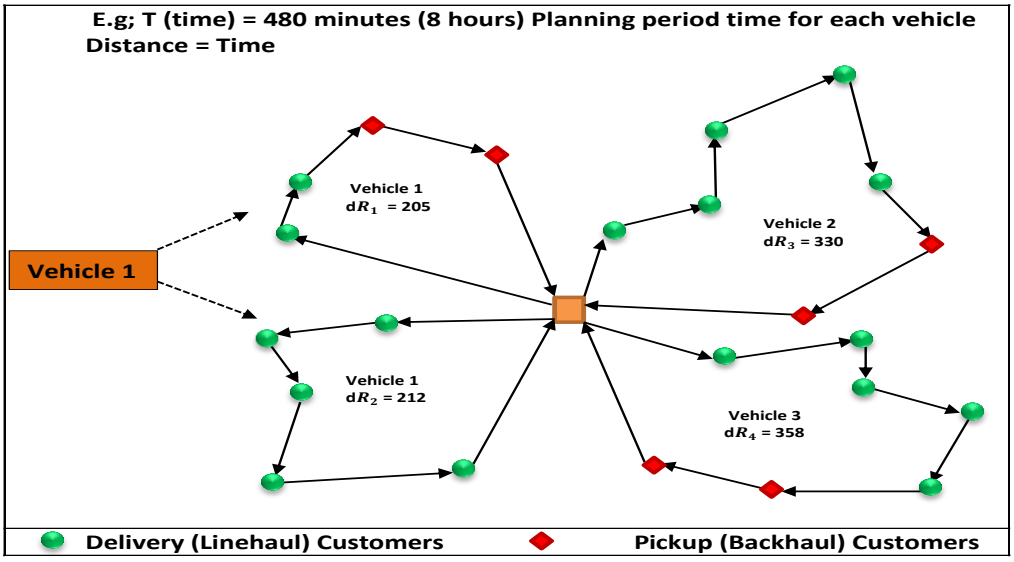

Figure 1.1: An example of the MT-VRPB

1.2. Aims and Objectives of the Thesis

As mentioned in the introduction, the aim is to study the vehicle routing problem (VRP) in terms of reducing the gap between the current assumption based on theoretical VRP models conducted by academics and the actual practices at the industry by developing more realistic models and efficient algorithms. Hence, this research provides insights in regard to the power and efficiency of solution methods, to address the issues (e.g., routing cost, maximising the fleet usage, less vehicles on roads and environmental etc.) which are of growing importance to the industry, governments and other relevant sectors. To achieve the aims of the thesis the following objectives are set.

- To study existing VRP models and methodologies meticulously to gain a better understanding of the issues and the subject area.
- To develop a mathematical model for a VRP variant, i.e., vehicle routing problem with multiple trips by incorporating backhauling (MT-VRPB) aspect of the reverse logistics.
- To design and implement new efficient and robust meta-heuristic and mat-heuristic algorithms that are able to solve instances of a range of VRP variants including the new MT-VRPB model. Moreover, to generate more realistic MT-VRPB test instances data set and conduct tests and analysis to provide in-depth understanding of the issues and discuss limitations.

1.3. Outline of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 presents a general review of the Vehicle Routing Problem (VRP) and some of its main variants models along with their historical developments. It also presents descriptions of various VRP models along with some useful references for the readers. The chapter also reviews other methodologies including exact, classical heuristics and metaheuristics developed around the subject and discuss their pros and cons in terms of their implementation.

Chapter 3 presents a focussed literature review of the two VRP variants models. Since the MT-VRPB is modelled by blending two existing VRP models, i.e., the Vehicle Routing Problem with Backhauls (VRPB) and the Multi-trip Vehicle Routing Problem (MT-VRP), the review of these problems will help provide better understanding of the newly introduced problem. The VRPB and the MT-VRP are studied independently in the literature, their reviews are presented separately.

Chapter 4 introduces a new variant of the VRP being studied in this thesis i.e., the Multiple Trip Vehicle Routing Problem with Backhauls (MT-VRPB) and the exact method options to solve the model. The details of the MT-VRPB including the graph theoretical definition and mathematical formulation along with possible variations are presented. An illustrative example showing validation of the formulation is provided before the details of our CPLEX solution implementation. The chapter also provides details of a newly created large set of MT-VRPB data instances along with the results and analysis.

Chapter 5 presents a *Two-Level VNS* algorithm developed to solve the MT-VRPB. An overview of the algorithm is first provided followed by the details of various components including a multi-layer local search approach that is embedded within the

Two-Level VNS methodology. Details of an adapted *sweep-first-assignment-second* approach to produce an initial solution for the MT-VRPB are also provided. Finally detail of the Bin Packing Problem (BPP) that resolves the multiple trip aspect of the MT-VRPB is presented followed by the results and analysis.

Chapter 6 presents a hybrid collaborative sequential mat-heuristic (CSMH) approach developed to solve the MT-VRPB. Combining mathematical programming techniques with heuristic methods to solve CO problems is a recent development in the literature. These approaches are recognised as a new class of the hybrid methodologies and are termed as '*mat-heuristics*'. The mathematical model developed in Chapter 4 is hybridised with the *Two-Level VNS* algorithm developed in Chapter 5. The *Two-Level VNS* uses three phases, i.e., initial solution by a modified *sweep-first-assignment-second* approach, improved solution by VNS, and packed solution by the BPP. Here in fourth phase, a mathematical formulation is incorporated with the *Two-Level VNS* algorithm to find optimal/better solution for the MT-VRPB.

Chapter 7 presents our study for two classical versions of the VRP, i.e., the Vehicle Routing Problem with Backhauls (VRPB) and the Multi-trip Vehicle Routing Problem (MT-VRP). The *Two-Level VNS* and the CSMH algorithms developed for MT-VRPB in Chapter 5 and 6 are further investigated and implemented to solve the VRPB and the MT-VRP. The results are produced using the benchmark instances of these problems from the literature. The *Two-Level VNS* and CSMH algorithms results are analysed and compared with the best published solutions.

Finally, Chapter 8 provides the conclusions and some future research directions.

Chapter 2

The Vehicle Routing Problem: Models and Solution Methods

This chapter presents a brief review of the historical development of the Vehicle Routing Problem (VRP) and some of its main variants. Short descriptions of various VRP models are presented along with some useful references for the readers. The chapter also reviews the methodologies developed around the subject and discuss their strengths and weaknesses in terms of their implementation.

2.1. The Vehicle Routing Problem and its Variants

2.1.1. The Evolution of the Vehicle Routing Problem

The evolution story of the VRP starts with the generalization of the classical Travelling Salesman Problem (TSP) by Dantzig and Ramser (1959). The TSP is typically described as a salesman who has to start a tour from his/her home city and visits all customers at different locations before returning back to his/her home city. The problem is to find the order in which the salesman is to visit all customers to minimise the total distance travelled. (Lawler *et al.* (1985), Hahsler and Hornick (2006), Bai *et al*. (2005), Gendreau *et al*. (1992), and Gamboa *et al*. (2006)). Special cases of the TSP arise in

terms of its applications [e.g., Chinese Postman Problem where it is not necessary for salesman to returns home (Eiselt *et al*., 1995)]. For instance, the problem may have special properties where the distance between the pairs of nodes is assumed to be asymmetric (not the same in both directions). The story then moves to the extension of the TSP, i.e., the Multiple Travelling Salesman Problem (*m*TSP), which involves the use of exactly *m* salesmen (Lawler *et al*., 1985, Bodin *et al*., 1983, Bektas, 2006). The extension of the *m*TSP model then took the shape of the classical vehicle routing problem (VRP) in the work of Dantzig and Ramser in 1959 with the incorporation of some additional aspects such as vehicle capacity restrictions.

2.1.2. Definition of the Vehicle Routing Problem:

The VRP is a general name devoted to a whole set of problems. In its simplest form, the VRP involves a set of customers with deterministic demands, a fleet of vehicles (normally homogeneous in physicality and unlimited in number) and a depot. The problem is to design such a set of routes (starting and ending at the depot) to serve all the customers at minimum cost while satisfying the vehicle capacity and (in some cases) route-length constraints. Figure 2.1 shows an illustrative example of the classical VRP. For detailed information on the subject of VRP see Toth and Vigo (2002), Mester and Braysy (2007), Bin *et al*. (2009), Fleszar *et al*. (2009).

2.1.3. VRP Variants

In the field of transportation and distribution logistics, the vehicle routing problem has evolved as a pivotal problem since Dantzig and Ramser (1959) first introduced it as the Truck Dispatching Problem. Since then and especially in the past three decades the VRP has emerged to be one of the most studied problem in the area of combinatorial

optimisation. Numerous variants of the VRP have been introduced and hundreds of papers have been written on this subject with new efficient solution methodologies in the literature.

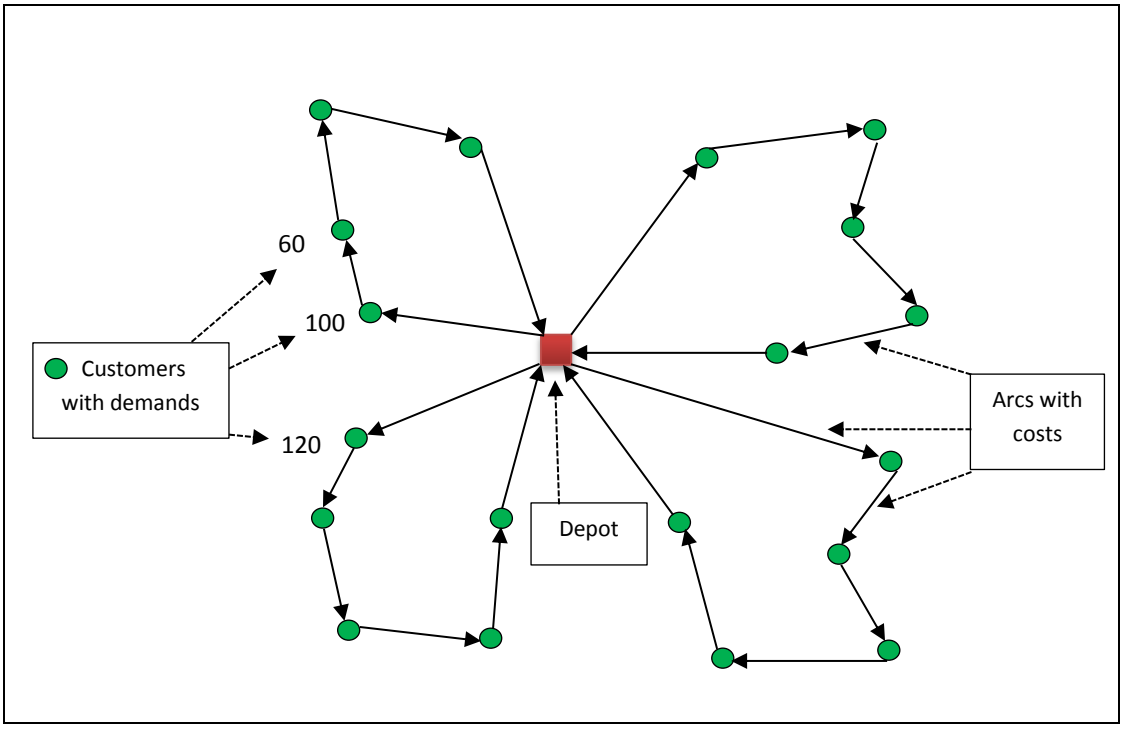

Figure 2.1: An illustrative example of the VRP

The primary objective behind the development of various variants of the VRP and proposed methodologies is to bring the problem closer to the real world applications requirements. Consequently, by taking advantage of the studies around the VRP and its different versions, public or private transportation companies in the real-world can save substantial transportation costs (for example, combining delivery and pickup operations, using a mix of smaller and bigger vehicles, serving from more than one depot, etc.). Ganesh *et al*. (2007) presented a broad review of the Vehicle Routing Problem, its variants, solution approaches and the applications. It has been reported that on average the transportation of goods or material takes the highest proportion of logistics costs.

The authors argue that the VRP has been assumed to be a deterministic and static problem traditionally, however, in present day context, VRP takes account of collecting and processing information and take decisions accordingly within certain time span. However it should be noted that around the time when the study of Ganesh *et al*. (2007) was published another class of the VRP emerged and referred to as "rich" VRPs inspired by real applications (Gribkovskaia *et al*., 2006). A large number of such studies exist in the literature, the reader is referred to Goel and Gruhn, 2008; Vidal *et al*. 2014, and for a recent review Lahyani, *et al*., (2015). On the other hand the counter argument to the study of Ganesh *et al*. (2007) and the fact behind the evolution of VRP variants is that these are inspired by real-life operations. The literature on the VRP shows a clear trend towards bringing it closer to the reality. We believe the research work in this thesis is yet another step to bring the VRP closer to the reality by addressing multiple use of fleet with backhauling in a time span which is very much in practice.

In the following subsections some of the main variants of the VRP and those which are relevant to this study are briefly described and useful references are provided.

2.1.3.1. The Periodic VRP

The Periodic Vehicle Routing Problem (PVRP) is a generalization of the classical VRP that addresses the planning period aspect of the problem. Hence in this problem, the planning period is extended to *M* days as opposed to the classical VRP where a single-day planning period is considered. The objective is to find the minimum cost set of routes over *M* days (Christofides and Beasley, 1984). The PVRP is found in many real-world applications, e.g., waste collection, elevator repair and maintenance and recycling collections. For further details and applications of PVRP, see Russell and Igo (1979),

Beasley (1983), Baptista *et al*. (2002), Blakely *et al*. (2003) and Hemmelmayr *et al*. (2009).

2.1.3.2. The Multiple Trip VRP

The Multiple Trip Vehicle Routing Problem (MT-VRP) is an extension of the classical VRP in which a vehicle may perform several routes (trips) within a given planning period (Taillard *et al*., 1996). As mentioned in Chapter 1, the MT-VRP is one of the variants of the VRP that are intended to be investigated in this thesis; hence further discussion is provided in subsequent chapters.

2.1.3.3. The Multiple-Depot VRP

In the Multiple-Depot Vehicle Routing Problem (MDVRP), customers are served from more than one depot as opposed to the classical VRP where customers are served from a single depot. The objective of this problem is to minimise the number of vehicles used and the total distance travelled while serving all customers. This variant is related to some real-world applications where a company might want to serve its customers from several depots, as their customers may be clustered around the depots, and it would be less costly to serve a respective customer from its nearest depot. In this type of scenario, the problem is either tackled as a set of individual vehicle routing problems or in the case where customers and depots are somehow intermingled then the problem is tackled as a multiple-depot vehicle routing problem. For more details on this problem and its extensions readers are referred to Bodin *et al*. (1983), Renaud *et al*. (1996) and Salhi *et al*. (2014).

2.1.3.4. The Mix Fleet VRP

The Mix Fleet Vehicle Routing Problem (MFVRP) is an extension of the VRP. The MFVRP considers a heterogeneous fleet as opposed to the VRP where a homogeneous fleet is used. Hence in the MFVRP, each vehicle's characteristics differ in terms of capacity, fixed cost and variable travel cost. The objective is to find a set of mix fleet routes with a minimum total cost while serving all customers. For the details of the MFVRP and its further versions readers are referred to Golden *et al*. (1984), Taillard (1999), Salhi and Sari (1997), Wassan and Osman, (2002), Tarantilis *et al*. (2004), Yaman (2006) and Imran *et al.* (2009).

2.1.3.5. The VRP with Time Windows

The Vehicle Routing Problem with Time Windows (VRPTW) addresses time window aspect where each customer specifies his/her service time periods. There are two variations of time windows considered in the literature, (1) hard time windows, in which the customer must be served in the stated time window and (2) soft time windows, where the time window can be violated at an additional cost added to the objective function in order to compensate the customer for the inconvenience. The VRPTW is studied intensively in the literature. For further details on the VRPTW, see Desrochers *et al*. (1992), Halse (1992), Potvin and Bengo (1996), Taillard *et al*. (1997) Toth and Vigo (2002), Yeun *et al.* (2008) and the survey of Bräysy and Gendreau (2005a, 2005b).

2.1.3.6. The Split Delivery VRP

The Split Delivery Vehicle Routing Problem (SDVRP) variant allows a customer to be serviced by two different vehicles if it reduces the overall cost. This relaxation can be

very important if the sizes of the customers' orders are bigger than the capacity of a vehicle and it becomes compulsory to visit a customer more than once. However the objective of the problem stays the same as the VRP. The literature on the SDVRP has seen a big gap since it was first introduced by Dror and Trudeau (1989, 1990). However considerable attention has been paid towards the SDVRP more recently. See the following references for more information on the SDVRP, Belenguer *et al*. (2000), Ho and Haugland (2004), Archetti *et al*., (2008), Jin *et al*. (2007) Jin *et al*. (2008), Aleman (2009), Bolduc *et al*. (2010), Derigs *et al*. (2010), Mohamed (2012) and Nagy *et al*. (2015).

2.1.3.7. The classical VRP with Backhauls

The Vehicle Routing Problem with Backhauls (VRPB) as described in Section 1.2 involves two types of customers, i.e., linehaul (delivery) and backhaul (collection). In this problem a vehicle can deliver goods to the customers and make collections to bring back to depot (Goetschalckx and Jacobs-Blecha (1989), Toth and Vigo (1996, 1997, 1999, 2002), Osman and Wassan (2002). There are some versions of the VRPB that are modelled and studied in the literature. Since the VRPB will be studied in this thesis, a literature review and our investigations will be provided respectively in Chapter 3 and Chapter 7. In this chapter we present brief descriptions of those relevant VRPB variants, along with some useful references, which are not investigated in this thesis.

2.1.3.8. The VRP with Mixed Deliveries and Pickups

The Vehicle Routing Problem with Mixed Deliveries and Pickups (VRPMDP) is another backhauling version in which the order of the pickup and delivery customers is not important when it comes to serve their demand. That is linehaul and backhaul

customers can be mixed freely within a route in a way that customers are either delivery or pickup locations. The VRPMDP is studied further with the extensions such as "multi-depot" and "time windows", see Zhong and Cole (2005), Jarpa *et al*. (2010). The following studies can be a useful start for understanding this version of backhauling VRPs; Halse (1992), Nagy and Salhi (1999, 2005), Salhi and Wade (2001), Wade and Salhi (2002), Ropke and Pisinger (2006), Tutuncu *et al*. (2009), Lin and Tao (2011). Moreover, a recent paper of Wassan and Nagy (2014) provides a comprehensive discussion on the modelling issues and the meta-heuristics developments around this problem.

2.1.3.9. The VRP with Simultaneous Deliveries and Pickups

The Vehicle Routing Problem with simultaneous Deliveries and Pickups (VRPSDP) was introduced by Min (1989). In VRPSDP, a vehicle can serve a linehaul customer only, a backhaul customer only or it can serve a customer both with linehaul and backhaul demands simultaneously. Taking into account the fact that serving customers simultaneously can lead to a problem of rearranging the load on a vehicle, it is assumed that the physical design of a vehicle is designed in such way that it can be accessed from several sides in order to accommodate the load. For more information on the VRPSDP, see Salhi and Nagy (1999), Nagy and Salhi (2005), Chen and Wu (2006), Ganesh and Narendran (2007), Wassan *et al*. (2008a and 2008b), Gajpal and Abad (2009), Zachariadis *et al*. (2010), Wassan and Nagy (2014) and Nagy *et al.* (2015).

2.1.4. Future of the VRP

The above descriptions of the VRP models show that the distance/cost minimization has been the key factor in those models, besides the maximization of fleet utility and

service. Nonetheless, it is noticeable that researches have been continuously trying to develop models that are closer to the real applications of the vehicle routing. It can also be seen that there are still gaps between the existing models and the reality that need to be bridged by developing more integrated models that fulfil the contemporary demands of the industry. More recently, a good development and increasing interest in the real VRP modelling and applications has been noted in the literature. These models are being termed as "Rich" VRP models (Battara *et al*., 2009). However, so far these models seem to be specific to individual applications. The main difficulty to design the VRP models and to solve them in an integrated way by considering various real life routing requirements is the complex nature of those instances of the problem, and the fact that these models belong to the category of hard combinatorial optimisation (CO) problems. We shall discuss the solution methods separately in the remainder of this chapter by first providing a description of *CO problems* and introducing the term *algorithm*.

2.2. Combinatorial Optimisation: Problems and Algorithms

2.2.1. Combinatorial Optimisation Problems

Combinatorial Optimisation (CO) problems arise in many areas, including management, e.g. vehicle routing and scheduling, production, finance, technology, facility location, etc., (Hoffman, 2000). The term "Combinatorial Optimisation" deals with those areas of mathematical programming that find the solution of optimisation problems, normally being termed as combinatorial or discrete (Christofides *et al.*, 1979). In simple words, combinatorial optimisation can be defined as a process of finding one or more best (optimal) solutions in a well-defined discrete problem space. One of the primary issues

that arise for most combinatorial problems is the computational burden associated with various solution approaches when formulating and seeking a solution to these problems. In 1970s computational complexity results were discovered by Cook (1971). It was discovered that many CO problems belong to NP-hard category of problems (for more information for this area we refer to the reader to an excellent study of Garey and Johnson (1979)). Hence the attention was turned to develop more efficient heuristics.

2.2.2. The term Algorithm

The word *algorithm* is derived from the Latin word *Algoritmi*; Latinized from a Persian mathematician's family name who is named Abu Abdullah Muhammad ibn Musa al-Khwarizmi (in Arabic: محمد بن موسى الخوارزمي). He was born in either Khwarizmi or Baghdad and lived approximately between the years A.D.780 to A.D.850. He wrote his first book on systematic solutions to linear and quadratic equations named "*al-Kitāb al-mukhtaṣar fī ḥisāb al-jabr wa-l-muqābala*". The word *algebra* is derived from the word *al-jabr*. As a result, he is considered to be the father of algebra and algorithms. The word algorithm originally meant the rules that govern arithmetic; it was not until the 18th Century, when it evolved to include all procedures and formulae for problem solving.

The origins of algorithm root back to the works of a Hellenistic mathematician known as Diophantus of Alexandria (Greek: Διόφαντος ὁ Ἀλεξανδρεύς), who lived around the time between A.D.200 and A.D.298 in Alexandria, Egypt. According to historic findings, he wrote thirteen Greek books named *Arithmetica*, of which six survived till today. Evidence from Arabic sources show that some of their problems may have originated from these manuscripts, known as Diophantine problems. Moreover, an

Arabic manuscript discovered in 1968 apparently shows a translation of four of the seven lost *Arithmetica* books (Sesiano, 1982). His original Greek manuscripts show an unusually syncopated notation that matches the way al-Khwarizmi's algebras were developed at a much later date. Hence he also shares the title of the father of algebra.

Some quotes from Kowalski (1979) (Kleene 1991, first published in 1952) and Markov (1954) (Knuth 1973, first published in 1968), show how others have defined the word "algorithm" and in the light of those quotes, our understanding of the general definition of an algorithm is a specific finite set of procedures, methods, techniques or formulae that accomplishes a set of tasks within a reasonable and finite amount of time, with the requirement of a given initial state and a user-defined end state, to solve a problem and conclude with a definite logical answer.

We note that although many algorithms are designed to find an exact or optimal solution, with hard combinatorial problems such as the VRPs, often the bigger the problem size, the harder it is to find an optimal solution due to its non-deterministic polynomial nature. In order to find a solution within a reasonable amount of time, approximation algorithms are implemented where the solution is an approximation that is close to the optimal solution. Hence we recognize that the word *solution* possesses different meanings in different situations, where the best solution to a problem could either be an optimal solution or a feasible solution. A feasible solution is an improved better quality solution as compared to the initial solution and may not necessarily represent an optimal solution. Therefore, depending on the end state criteria and time restrictions, this may be taken as the best feasible solution and thus the desired solution rather than the exact or optimal solution.

2.3. Solution Techniques for the Vehicle Routing Problems: An Overview

Over the past few decades various solutions approaches have been developed to solve the VRPs. Among these are the exact and the heuristic/meta-heuristic approaches. The literature reveals that not many authors have proposed exact methods to tackle the VRP and its variants due to their *non-deterministic polynomial hard* (NP-hard) nature, which leads to an exponential amount of time needed to solve the problem to optimality. Therefore, most solutions are achieved by using non-mathematical programming methods to find a near-optimal solution – a good problem solution that may be achieved within a reasonable amount of time – these methods are termed as heuristic methods. Unfortunately, these suffer from inflexibility to changes in the formulation of problems. Moreover, as the problem size increases, it becomes more and more problematic to find high-quality solutions; in many cases, heuristics tend to get trapped within *local optima*, i.e. they tend to find an optimal solution within its neighbouring space, which in most cases do not represent the *global optima*; the optimal solution of the whole solution space. Researchers identified these defects of heuristic methods and produced high-level procedures based on generic principles of heuristics, these types of advanced heuristic algorithms are named as meta-heuristic, or metaheuristic, methods. Metaheuristic methods are also known as artificial intelligence (A.I.) algorithms and are capable of solving a large range of problems more efficiently and effectively than simple heuristics do. They are designed to be flexible, hence easily adapted to different problems and criteria with just a few minor modifications, and do not get hindered from being trapped within local optima.

Exact and heuristic methods both have their pros and cons. There is a compromise between using one or the other; hence users must justify which one is more suitable for

tackling their problem given the constraints that restrains them to find a solution within a set amount of time. Although recent trend shows that the possibility of combining both methods to produce highly competitive solutions is feasible, we shall leave this to the latter part of this chapter. We attempt to give a brief introduction of each method and allow the reader to distinguish between the two methodologies.

2.3.1. Exact Methods

Normally, the exact approaches are developed on the mathematical formulation of the problem. These methods provide guaranteed optimal solutions, but at a very high computational effort (Halse, 1992). These approaches work through the problem intelligently and efficiently and obtain optimal solution for the combinatorial optimisation problems.

Although, the exact approaches have proved their efficiency by solving combinatorial problems of moderate size. However, when the problem is complex and large in size, it may not be a good choice to use exact approaches. Because when engaging with complex and large-size problems, these approaches may lead to some implementation issues and may require too much computational effort. On the other hand, recent advancements and the power in the computer technology has made possible to solve moderate-size problems and in some cases problems of large-size in an acceptable amount of time using exact methods.

2.3.2. Heuristic Methods

The term *heuristic* is originally derived from the Greek word “heurisko” (Greek: ευρίσκω), meaning “I find” or “I discover”. The term was introduced approximately

around A.D.340 by a Hellenized mathematician named Pappus of Alexandria (Greek: Πάππος ὁ Ἀλεξανδρεύς), who was born in Alexandria, Egypt. The original definition of heuristic is a technique to learn, discover, or problem-solve using a simple set of procedures. Heuristics were popularized by a Hungarian mathematician named George Pólya (Pólya, 1945). He wrote a book titled "How to Solve It" (Pólya, 1945) that consolidates ideas about heuristics, ways of understanding a problem, planning how to tackle it and revising the solution method to seek for improvements.

In the context of operational research, heuristic methods use a set of procedures to search approximated solutions for a problem in hand without any guarantee of optimality (Reeves, 1993). These procedures are aimed at finding solutions as near to optimality as possible in a reasonable amount of computational time by searching through the most promising regions of solution choices, rather than performing a time-consuming complete enumeration in the search of the optimal solutions. The down-side of heuristics is the fact that they lack precision and accuracy. This led researchers around the world to investigate what are the most effective ways to deal with problems and how they may be improved by combining different forms of algorithms or algorithmic principles.

In real-life problems, it is better to be able to find an approximate solution to the real problem rather than finding the optimal solution to an approximation model of the problem. As we have mentioned, in real-life applications problem sizes are usually enormous. Hence, generally, it is impractical to attempt finding the optimal solution, which leads to the preference of heuristic methods. Table 2.1 shows a list of methods used to tackle the VRP and its variants. The table includes a categorized list of exact, heuristic, metaheuristic and hybrid methods that researchers have chosen to utilise. Each

method is referenced to author, or authors, that are identified as the accredited founder, where applicable. Note that due to the amount of papers that have been published in the context of the VRP and its variants, it is not possible to include and categorize every method that has been used and therefore we have only included some of the most acknowledged and discussed methods. However, methods for tackling the VRP and its variants are not limited to the ones mentioned.

Table 2.1: Methods used in tackling the VRP and its Variants

Exact Methods:

B&B – Branch-and-Bound (Land and Doig, 1960)

- Carpaneto-Toth B&B for Non-Integer Linear Programming (NILP) (Carpaneto and Toth, 1980)
- *k*-Shortest Spanning Tree, *q*-Paths (Christofides, Mingozzi and Toth, 1981b)
- Branch-and-Cut (Laporte, Nobert and Desrochers, 1985)
- Branch-and-Cut-and-Price (Fukasawa *et al*., 2003)
- Dynamic Programming with State Space Relaxation (Christofides, Mingozzi and Toth, 1981a)
- Two-Commodity Network Flow Formulation (Baldacci, Hadjiconstantinou and Mingozzi, 2004)

Heuristic Methods:

Construction-based Heuristics

- Clarke-Wright savings (Clarke and Wright, 1964)
- NNH – Nearest Neighbour
- Sweep (Gillett and Miller 1974)
- Cluster-First, Route-Second (Fisher and Jaikumar, 1981)
- Route-First, Cluster-Second (Beasley, 1983)
- Petal (Ryan *et al.*, 1983)
- Insertion Heuristics (Flood, 1956)

    - Christofides-Mingozzi-Toth Sequential Insertion Heuristic (Christofides, Mingozzi, Toth, 1979)
    - Parallel Insertion Heuristic
    - Christofides-Mingozzi-Toth Parallel Insertion Heuristic (Christofides, Mingozzi, Toth, 1979)
- GENI (Gendreau *et al*., 1992)

Intra- and Inter-Route Improvement Heuristics

- Transfer Heuristics
    - Or-Opt (Or, 1976) (Inter-/Intra-Route Improvement)
    - 1-0 Exchange (Salhi and Rand, 1987; Water, 1987)
- Swap Heuristics
    - *r*-Opt approx 5% from optimum (Croes, 1958)
    - 2-Opt (Croes, 1958)
    - 3-Opt
    - 4-Opt
    - 1-1 Exchange (Salhi and Rand, 1987; Waters, 1987)
    - λ-Interchange (Osman, 1993)
    - Edge Exchange Scheme (Kindervater and Savelsbergh, 1997)
- Composite Move Heuristics
    - Ejection Chain Process (Thompson and Baraftis, 1989; Rego and Roucard, 1996)

Metaheuristic Methods:

Local Search (LS) Methods

- TS – Tabu Search (Glover, 1989,1989,1990)
    - Taburoute (Gendreau, Hertz and Laporte, 1994)
    - UTSH – Unified TS Heuristic (Cordeau, Laporte and Mercier, 2001; 2004)
    - RTS – Reactive TS (Osman and Wassan, 2002)
    - Granularity Principle – Granular TS (Toth and Vigo, 2003)
- SA – Simulated Annealing (Kirkpatrick, Gelatt and Vecchi, 1983; Černý,1985)

- DA – Deterministic Annealing (Dueck 1993)
    - Threshold-Accepting (Dueck, 1990)
    - Record-to-Record Travel (Dueck, 1993)
- LNS – Large Neighbourhood Search (Shaw, 1997) [Inter-route Improvement]
    - VLNS – Very LNS (Ergun *et al*., 2002)
    - ALNS – Adaptive LNS (Røpke and Pisinger, 2004)
- VNS – Variable Neighbourhood Search (Mladenović and Hansen, 1997)

Population Search / Solution Recombination Methods

- EA – Evolutionary Algorithm
    - EP – Evolutionary Programming (Fogel, Owens and Walsh, 1966)
    - ES – Evolutionary Strategies (Rechenberg, 1973)
- GA – Genetic Algorithm (Holland, 1975)
    - GP – Genetic Programming (Koza, 1992)
    - AMP – Adaptive Memory Programming (Rochat and Taillard, 1995)
    - Population Mechanism (Prins, 2004)

Learning Methods

- ACO – Ant Colony Optimisation (Moyson and Manderick, 1988)
    - *D*-Ants Savings Algorithm (Reinmann, Doerner and Hartl, 2004)
- NN – Neural Networks (Hopfield and Tank, 1985)
- Swarm Intelligence
    - PSO – Particle Swarm Optimisation (Kennedy and Eberhart, 1995)
- CE – Cross-Entropy

Hybrid / Composite Methods:

- Parallel TS/Ejection Chain Algorithm (Glover, 1991; 1992; Rego and Roucard, 1996)
- BoneRoute: hybrid of AMP and LS (Tarantilis and Kiranoudis, 2002)
- Memetic Search: hybrid of GA and LS (Moscatto and Cotta, 2003)
- AGES – Active Guided Evolution Strategy: hybrid of ES and LS (Mester and Bräysy, 2007)

- Hierarchical/Algorithm Hybrid MILP – Mixed Integer Linear Programming (Dondo and Cerdá, 2006)
- RTAMP – Reactive Tabu Adaptive Memory Programming Search (Wassan, 2007)

2.4. Examples of Exact Methods

In this section, we attempt to give a few examples of exact methods. Some of these methods may require the problem to be formulated as an integer linear programming (ILP) problem. The exact methods discussed below are ones that have already been used in solving the VRP and its variants.

2.4.1. The Branch-and-Bound Method

The *branch-and-bound* (B&B) algorithm belongs to the class of *implicit enumeration methods* and was first proposed by Land and Doig (1960) to solve pure integer linear programming (ILP) problems. The general idea may be graphically described in terms of finding the minimal value of a function *f(x)* over a set of solution values within the feasible region of the argument *x*. The name of the B&B algorithm itself automatically suggests that it consists of two parts to form the whole algorithm; *branching* and *bounding*. Branching is a method to finding candidate solutions by covering all the feasible regions and splitting into sub-regions yielding sub-problems. Branching on each sub-region only terminates when it cannot find a feasible and promising candidate solution, or else it is repeated recursively. This branching procedure inevitably forms a *tree structure* and is termed as a *search tree*, also known as a *branch-and-bound-tree*. When further branching of the sub-problem cannot yield any useful information, we say

that the sub-problem is *fathomed*. Bounding is the part where upper and lower bounds for the optimal solution are found within a feasible sub-region. When performing the B&B, constructed sub-regions are referred to as *nodes*. During branching and bounding, a process called *pruning* is performed to search for a better candidate solution. The pruning process observes the lower bound of a currently searched sub-region A and compares it to the upper bound for any other examined sub-region B, if A's lower bound is greater than the upper bound of B, A is discarded from the search. If the upper bound of A matches the lower bound of B, this value becomes the minimum of the function within the subsequent sub-region; in this case, we say that the sub-region is *solved*, but maybe further pruned as the search proceeds.

Two general approaches are used during the search process, namely the *backtracking* and the *jumptracking*. Backtracking (known as depth-first search) leads the search tree by branching down one side of the search tree and quickly finds a candidate solution. It then backtracks up to the top of the other side of the tree. Jumptracking (also known as breadth-first search) solves all the sub-problems created by the branching. It then branches again on the *z*-value found from each sub-region to create further sub-problems. Jumptracking often jumps from one side of the search tree to the other; hence it creates more sub-problems than backtracking and thus requires comparatively more computer storage.

The procedure terminates when all the nodes on the search tree are pruned or solved, where the non-pruned sub-regions have their upper and lower bounds equaling the global minimum of the function. In practice, the procedure is usually terminated after a given time or number of iterations with a range of values that contain the global minimum amongst the non-pruned sub-regions.

The main concern is the efficiency and effectiveness of the B&B algorithm used, as the efficiency is directly affected by the effectiveness of the branching and bounding algorithm used; a bad algorithm could result in non-pruned repeated branching until the sub-regions become very small. In such a case, we refer to this as an *exhaustive enumeration*, which becomes impractical even with relatively small problems.

Carpaneto and Toth (1980) devised a B&B algorithm specifically to tackle asymmetric travelling salesman problem up to $n = 5000$. The problem solution starts off by setting all $c_{ii} = \infty$. The assignment problem is solved to find a lower bound. If there are no sub-tours, then the TSP is solved. Whereas, if there are sub-tours, an upper bound feasible solution is found by using Karp's (1978) *patching algorithm* by eliminating specific edges on sub-tours and reconnecting the sub-tours to form a TSP. These sub-tours are then eliminated using branching by "forcing the sub-tours" one-by-one into the main tour to form TSP solutions. Each corresponding sub-tour is used to solve the assignment problem and any branches whose value is greater than the upper bound is deemed infeasible. The Branch and Bound methods have been used to tackle VRP and some of its variants with a reasonable success, e.g., Christofides and Eilon (1969), Yano *et al*. (1987), Laporte *et al*. (1987), Laporte (1992), Fisher (1994), Mingozzi *et al*. (1996), Toth and Vigo (1997, 2002) and Ralphs (2003).

2.5. Examples of Heuristic Methods

This section briefly introduces some of the most common heuristic methods used for finding optimal or near optimal solutions for small size problems, or for improving (local search methods) on initial solutions found via initial solution construction methods. We note that classical improvement heuristics have two properties; these are

that the solution never deteriorates and always remains feasible. The quality of a heuristic method is assessed on four criteria namely speed, flexibility, accuracy and simplicity (Cordeau *et al*., 2001). Note that in the following sub-sections we are going to describe only those construction and improvement heuristic methods that are directly used in this study. For others, relevant references are provided in sub-section 2.5.1.2.

2.5.1. Construction-based Heuristics

Construction-based heuristics are heuristic methods that create an initial solution from raw data. With small problems, it is possible to find the optimal solution and hence not necessarily have to go through an improvement stage. However, as the problem size becomes larger, these construction-based heuristics could only provide a reasonable initial solution, thus requiring other improvement heuristics to improve the quality of solution.

In the following parts, a brief description of the most well-known construction-based heuristics is introduced with a brief assessment of the quality of the heuristic.

2.5.1.1. The Sweep Algorithm

The *sweep algorithm* was first introduced by Gillett and Miller (1974). It first selects a starting customer, shoots a "beam" from the depot to the starting customer and rotates clockwise or counterclockwise adding customers one-by-one to form a tour. If the capacity constraint is violated, a new route is initiated until all customers have been assigned. Figure 2.3 shows the visual representation in a clock-wise format.

Like the Clarke-Wright savings and the NNH, the sweep algorithm has a relatively high speed and simplicity. The accuracy is only mediocre and it is relatively inflexible. For

more information and the various ways it has been used to the VRP and its variants, readers are referred to Gillet and Miller (1974), Laporte *et al*. (2000), Renaud and Boctor (2002), Salhi, Wassan and Hajarat (2013). We shall elaborate more on the sweep methodology in the later chapters of this thesis since it is adopted in our algorithm implementations.

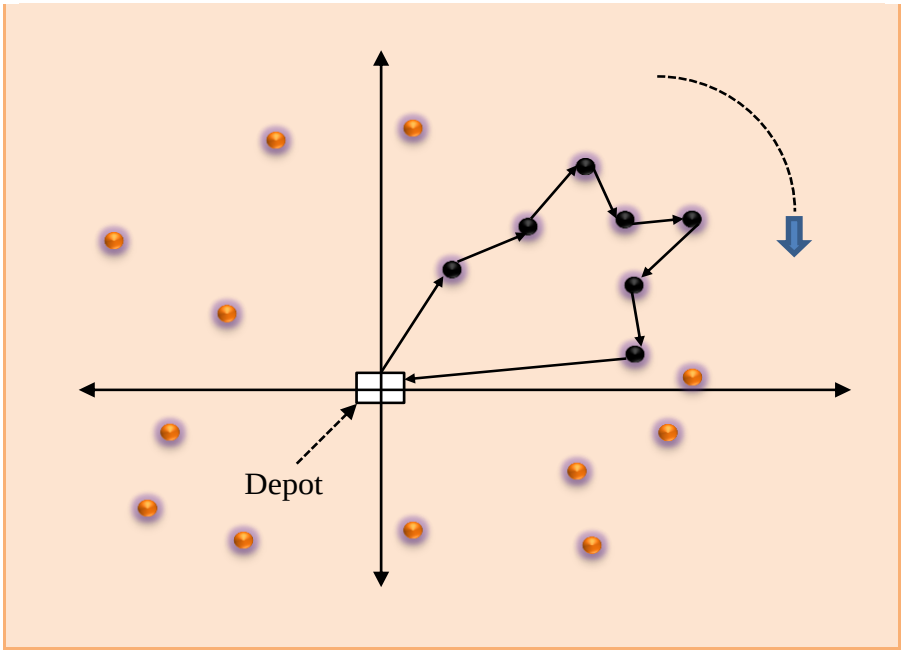

Figure 2.3: A visual representation of the sweep procedure

2.5.1.2. Other Construction-based Heuristic for VRPs

There are several other construction-based heuristics methods that are successfully used for VRPs. Among these some popular methods are; the savings algorithm proposed by Clark-Wright(1964), for more information on the savings algorithm and its enhancements, readers are referred to Laporte *et al*. (2000), Toth and Vigo (2002), Altinel and Oncan, (2005). The *nearest neighbour heuristic* (NNH) method, for more

details of Nearest-Neighbour heuristic and its applications on VRPs, readers are referred to Rosenkrantz *et al*. (1977), Golden *et al.* (1980), Solomon (1987), Fisher (1994), Ganesh *et al*., (2007). The two-phase methods (e.g., *cluster-first, route-second* method introduced by Fisher and Jaikumar (1981), and the *route-first, cluster-second* method introduced by Beasley (1983)). For more details about these methods and their different types/extensions, see Christofides *et al*. (1979), Renaud *et al*. (1996) and Toth and Vigo (2002). The *insertion method* introduced by Flood (1956); for details, see Salhi and Sari (1999). The *GENI* (Genius) heuristic proposed by Gendreau *et al*. (1992), for more details on this method, see Gendreau *et al.* (1994).

2.5.2. Intra- and Inter-Route Improvement Heuristics

Intra- and/or inter-route improvement heuristics are used for improving initial solutions generated from construction heuristics. There are various common improvement heuristics that have been investigated and we shall briefly introduce those which are directly used in this study in the following sub-sections and references are provided for other types of improvement heuristics in sub-section 2.5.2.3.

2.5.2.1. Transfer Heuristics

Transfer heuristics work by removing customer *i* from its initial position of route *I* and reinserting it into a different position in the same route *I* for intra-route optimisation. Whereas, for inter-route optimisation, customer *i* is removed from its initial position of route *I* and reinserting it into a new position in route *J*. Provided, the transfer of customer *i* results in an overall cost minimization without violating vehicle capacity constraints, the solution routes are updated. Two of the more commonly used transfer heuristics are briefly introduced in the following sections.

2.5.2.2. Swap Heuristics

Swap heuristics swap customers *i* and *j* in route *I* for the intra-route optimisation case, and; customer *i* from route *I* with customer *j* from route *J* for the inter-route optimisation case. The new solution resulted from the swap is kept if the new solution reduces the overall cost without violating vehicle capacities constraints (Waters, 1987). This can be extended for swapping several customers, as seen in the following sections.

2.5.2.3. Other Improvement Heuristics for VRPs

In the VRP literature there exist several other improvements/local search heuristics that have been used to improve the solution of VRP and its various extensions. Some of these local search heuristics are: *Or-Optimisation* introduced by Ilhan Or (1976), *r-Optimisation* proposed by Croes (1958), see Bock (1958) and Renaud *et al*. (1996) for different extension of r-optimisation. The *λ-interchange* (lambda-interchange) introduced by Osman (1991, 1993). *Composite Move Heuristics* also known as the *cyclic transfers* algorithms introduced by Thompson and Orlin (1989). One of the most promising *cyclic transfers* algorithms, the Ejection Chain Process was proposed by Glover (1991, 1992), see Thompson and Psaraftis (1993), and Rego and Roucairol (1996) for its VRP implementations.

2.6. Examples of Metaheuristic Methods

The following sections introduce well-known metaheuristic methods that have been used in this study and some of its extensions used for VRPs are also reviewed. For other metaheuristic methods which are widely used for the VRPs, relevant references are provided. Each metaheuristic technique has a characteristic that allows it to be

categorized into either a *local search method*, a *solution recombination method*, or a *learning method*. A good survey on metaheuristics is provided in Boussïd *et al*. (2013).

The metaheuristic method that has been used in this study belongs to the category of the local search methods. A local search method, also known as a neighbourhood search method, searches for an optimal/near optimal solution within its neighbourhood using inter-route and intra-route improvements, yet unlike classical heuristics, it allows the solution to deteriorate including temporary infeasible solutions, so to allow the solution-finding procedure to move out of local optima into unexplored search regions in the attempt of finding a global optima.

2.6.1. Variable Neighbourhood Search

The *variable neighbourhood search* (VNS) was first proposed by Mladenović (1995) and Mladenović and Hansen (Mladenović and Hansen, 1997) and has been quickly adopted and widely implemented. There exist many papers that make use of the VNS or its variations, mainly to enable it to find solutions for larger instances, and hence have proven its recognition as a promising metaheuristics tool.

The VNS may be seen as an extension of the TS algorithm, where it systematically changes between neighbourhoods to find global optima. The VNS searches for solutions for each neighbourhood that has been pre-selected from one type of neighbourhood in varying depth and three simple facts:

1. A local optimum with respect to one neighbourhood does not necessarily mean locally optimal with respect to another neighbourhood structure;
2. A global optima is locally optimal with respect to all neighbourhood structures;

3. In many cases a local optima with respect to one or several neighbourhood structures are relatively close to each other.

According to a survey done by Mladenović and Hansen (1997), the last observation empirically implies that information for finding the global optimum can usually be found within the local optima from the neighbourhood structures. Furthermore, these three facts could be used in three different ways – deterministic, stochastic, or both – which shall be briefly explained below respectively.

The *variable neighbourhood descent* (VND) is a deterministic VNS, where the change of neighbourhoods within the pre-selected neighbourhood set is carried out in a deterministic fashion. Many local search techniques also use a systematic search through the solution space, though these methods only use one or two neighbourhoods in their search. Having mentioned facts (1) and (2) from above, the VNS has an advantage in performing local optimisations on all neighbourhood structures of varying depth, which ultimately covers the whole solution space, while keeping the algorithm simple, effective and flexible. The final solution would eventually be locally optimal with respect to all its neighbourhoods and hence giving it a higher chance to be globally optimal in comparison to just using one or two neighbourhoods within the search.

The *reduced variable neighbourhood search* (RVNS) is the stochastic model of the VNS, where, instead of following a descent, it chooses its points randomly from the neighbourhood set. This random generation of points from the neighbourhood of x is also known as *shaking*. If the point generated through shaking is better than the current point, it becomes the incumbent for the next iteration and the search continues from this new point; otherwise the search continues from the current point. The most common

stopping criterion used is the maximum number of iterations needed between two improvements. The RVNS is good for very large instances, where performing a deterministic local search would be costly. It has also empirically been proven to be at least as good as or better than other methods, showing its efficiency and effectiveness.

The *basic variable neighbourhood search* (i.e. VNS) uses a combination of deterministic and stochastic features to find its final solution. The iterative procedure used for the VNS first performs a shaking, afterwards a local search method is applied and the comparison of the two solutions would give a so-obtained local optima. If this local optimum is better than the incumbent, it becomes the incumbent and the search continues by repeating the procedure; otherwise it continues its search from the incumbent. The iterative procedure is repeated until no further improvement is found or the stopping criterion is met. Stopping conditions may include the maximum number of iterations used in between two improvements, or the maximum computational time is reached, or the maximum number of iterations has been reached. The advantage of the basic VNS is that it avoids cycling by using a shaking procedure, which enables the algorithm to perform a search throughout the neighbourhood set.

Further reading on the survey of VNS variations and applications can be found in Mladenović and Hansen (1997). The details of our implementation of VNS are provided in Chapter 5.

2.6.2. Large Neighbourhood Search

The *large neighbourhood search* (LNS) was first proposed by Shaw (1997). It could be seen as a special case or a variant of the VNS, as the only difference between the LNS and the VNS is that the latter operates on only one type of neighbourhood structure with

varying depth as mentioned in the previous section, whereas the LNS operates on structurally different neighbourhoods by destroying and repairing solutions; i.e., one can imagine that VNS operates on a set of neighbourhoods, where all the neighbourhood starts from the same point and expanding its capsule in all directions within the solution space, hence one neighbourhood is nested within the next; on the other hand, each neighbourhood of the VNS centers around different points of the solution space forming a capsules of neighbourhood within different region of the solution space; hence one neighbourhood and another may or may not overlap with each other, yet it is also possible for one neighbourhood to be nested within another. Otherwise, the basic LNS follows the same procedure as the basic VNS, where the iterative procedures would start off with a shaking, then a search, followed by a comparison for "keep or discard" and then repeating these procedures until the stopping criterion is met. For more on LNS see a survey by Ahuja *et al*. (2002).

2.6.3. Other Metaheuristic Methods for VRPs

Several metaheuristic methods have been proposed and successfully used for the solution of VRP and its various extensions in the literature.

Some metaheuristic methods falling under the category of local search methods are:

- *Tabu Search* (TS) first proposed by Glover (1986), for detailed information, see Glover (1989, 1990), Hertz and de Werra (1990), Cvijovic and Klinowski (1995), Glover and Laguna (1993, 1997), and Piniganti (2014).
- *Simulated Annealing* (SA) founded by Kirkpatrick *et al.* (1983) and Černý (1985), for more details see Reeves (1993).
- *Deterministic Annealing* (DA), see Dueck and Scheuer (1990), Dueck (1993).

One of the metaheuristic methods that belong to the category of Population Search / Solution Recombination Methods is the *genetic algorithm* (GA), first introduced by Holland (1975). For more details on GA, the readers may find Reeves (1993) and Rayward-Smith *et al*. (1996) useful as initial readings.

Some metaheuristic methods falling under the category of learning methods are:

- *Ant Colony Optimisation* (ACO), also known as the *ant systems* (AS), introduced by Moyson and Manderick (1988), Colorni *et al.* (1991) and Dorigo (1992), for more details, see Dorigo and Stützle (2004).
- *Neural Networks* (NN), first used in the TSP by Hopfield and Tank (1985), for more information, see Durbin and Willshaw (1987), Kohonen (1988), Ghaziri, (1991, 1996), Matsuyama (1991), Potvin (1993), and Schumann and Retzko (1995).

2.7. Hybrid Methods

More recently there has been some good progress towards developing hybrid algorithms. The hybridisation of algorithms signifies those designs of the algorithms where either different meta-heuristics are used in conjunction or meta-heuristic features are used in an interconnected manner with mathematical programming techniques or vice versa to approach a problem (Caserta and Voß, 2010). The reasoning behind the hybridisation of diverse algorithmic concepts is to build systems that combine strengths of individual methods in order to approach the problems in a systematic and better way (Raidl, 2006). Blum *et al*. (2011) explains that by combining right elementary

algorithmic concepts, one can achieve top performance in solving various optimisation problems. However, developing such hybrid solution approaches is relatively hard and demands expertise from various fields of optimisation. The hybridisation among the metaheuristics, e.g., TS/SA, TS/SA/GA, Population-based iterated local search, Multilevel techniques etc., has been investigated since 1990s. More recently this idea is being investigated with combining heuristics and exact methods e.g., CP-based large neighbourhood search, ant colony optimisation and constraint programming, dual ascent heuristic and column-and-cut generation etc. The advances in technology and in exact methods have encouraged researchers to design such algorithms where heuristics are combined with mathematical programming models to tackle the problems. While for more information on hybridisation of metaheuristics-to-metaheuristics we refer the reader to the survey of Blum *et al*. (2011). For matheuristics, see Jourdan *et al.* (2009) for taxonomy of hybridising exact and metaheuristic methods and a recent survey on matheuristics by Ball (2011). Furthermore, we shall provide details and a review on heuristic-exact hybrids in Chapter 6 where we have developed such a method for the MT-VRPB.

2.8. Summary

This chapter consists of two parts. In the first part, the evolution of the vehicle routing problem along with its complexity issues is reviewed. The literature around the VRPs modelling appears to be concentrated on developing the variant models that are closer to the reality. Although a lot of progress appears to be made on different types of models, nonetheless, most of the proposed variants of the VRP merely address one or two characteristics from the real life vehicle routing. A slow progress on the development of more complex modelling is noticeable, meaning the modelling gaps needs to be

addressed in terms of bringing the VRP closer to the reality. One of the objectives the this thesis is to add its share in bringing the problem even closer to the reality by extending one the existing models which will be discussed in Chapter 3 onwards.

In the second part of the chapter, some of the well-known exact and heuristics approaches developed for the VRPs are reviewed. We have also compared and contrasted between exact, heuristic and metaheuristic solution methods. Given a description of each of the mentioned solution methods in varying depth, depending on the importance of the solution described that may or may not impact on how we formulate our methodology proposal in solving our research problem. We note that although all the solutions methodologies that have been introduced in this chapter are the most "trendy" techniques, there are still many other less-used solution methods that have been left out from being mentioned, which does not mean they will not be considered for investigation in the future. Furthermore, there are a lot of experimental researches by using hybrid techniques that have empirically proven to be promising. Hence the solution methods described here in this chapter maybe seen as the most basic forms of possible solution methods with a brief idea of how they operate.

Chapter 3

Literature Review of the VRPB and the MT-VRP

The MT-VRPB being introduced in this thesis is a new addition to the version of the VRP models, hence there is no directly related published study in the literature. Since the MT-VRPB is modelled by blending two existing VRP models, i.e., the Vehicle Routing Problem with Backhauls (VRPB) and the Multi-trip Vehicle Routing Problem (MT-VRP), hence this chapter present a review of these problems which will help better understand the newly introduced problem. The VRPB and the MT-VRP are studied independently in the literature; therefore, we present their reviews separately.

3.1. An Overview of the VRPB

As described in Chapter 2, the Vehicle Routing Problem with Backhauls (VRPB) is an extension of the VRP, often termed as the classical VRPB. It is one of the most studied problems among the class of backhauling VRPs in the reverse logistics area. The customers in this variant are divided into two groups known as the linehaul (delivery) and the backhaul (pickup). Hence, in the VRPB the vehicles are also used for picking up goods to bring back to the depot after all the deliveries are made. Figure 3.1 shows an illustrative example of the VRPB.

The objective of the VRPB is to minimise the total (cost) distance travelled while satisfying demands of both types of customers. However, a VRPB solution must satisfy the following main characteristics: (i) a vehicle must perform exactly one route; (ii) each vehicle must make all the deliveries before making any pickups; (iii) the sum of quantity of goods delivered or collected must not exceed separately the vehicle capacity (same capacity vehicles are considered), (iv) no route is constructed with backhaul customers only; though a route with delivery customers only is allowed; (v) all given vehicles must be utilised; (vi) vehicles start and end their journey at the same single depot.

The characteristic (ii) is encouraged by the fact that delivery of goods to the customers is considered to be the most profitable activity in many practical situations, and the fact that some vehicles are rear-loaded and it is difficult to rearrange the delivery load on board in order to adjust the new pickup load. Various definitions and formulations of the VRPB exist in the literature; for details we refer to Goetschalckx and Jacobs-Blecha (1989), Toth and Vigo (1997) and Mingozzi *et al*. (1999).

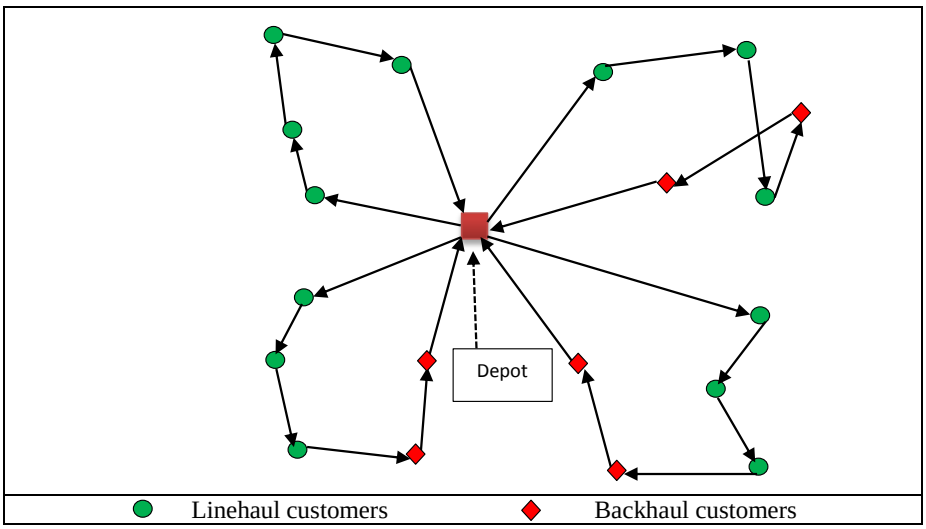

Figure 3.1: An illustrative example of the VRPB

The VRPB arises in many real-life applications such as delivery and pickup of mail to/from customers or post offices, delivery of drink bottles to shops and pickup of empty bottles, delivery of new household appliances and removal of old ones. Another application of the VRPB can be found in grocery distribution industry, where groceries are distributed to stores (considered as linehaul customers) from the distribution centres; whereas, pickups of groceries are carried from the production sites (considered as backhaul customers) to the distribution centres (Ropke and Pisinger, 2006). Moreover, the applications of the VRPB can be found in many other real-world scenarios where return of commodities to the distribution centre is involved, i.e., reverse logistics (Cuervo *et al*. 2014).

3.2. Solution Methods for the VRPB

The classical VRPB has been studied greatly in the literature; many exact and heuristic methodologies have been developed to tackle the problem. We present a review of the VRPB studies in the chronological order of their publication by separating the exact and the heuristic methods.

3.2.1. Exact Methods

There are not many exact methods publications on the VRPB in the literature. We provide a review of those in our knowledge as follows.

Yano *et al*. (1987) developed a set covering based branch and bound approach for a real-life application of the VRP with backhauling. In this problem vehicles were restricted to service a few customers (LH/BH); and found optimal solutions to problem instances involving up to 40 delivery and backhaul customers.

Toth and Vigo (1997) developed a Lagrangean lower bound procedure and a branch and bound (see Section 2.4) algorithm for the VRPB. They tested their methodologies on a range of data instances generated by them and found optimal solutions for instances up to 75 customers, which can be seen as a modest success. They also solved an asymmetric VRPB data set generated from the real-world asymmetric VRP instances described in Fischetti, Toth and Vigo (1994).

Mingozzi and Baldacci (1999) presented a mathematical formulation of the VRPB. They developed two methodologies, called, 'HDS' based on a combination of different heuristic methods to generate lower bounds, and 'EHP' algorithm to find optimal solution for the VRPB. They used CPLEX solver in their HSD and EHP procedures. The algorithms were tested on the instances of size up to 113 customers that were generated in Goetschalckx and Jacobs-Blecha (1989) and in Toth and Vigo (1996). The algorithm performed well in terms of solutions quality; however, it appears that the algorithm could not solve instances in which the number of total customers is higher than 113.

3.2.2. Heuristic Methods

The very first heuristic approach to solve the VRPB is called the DB of Deif and Bodin (1984). The DB heuristic is an extension of the savings method of the Clarke and Wright (1964) (described in Section 2.5.1.1) originally developed for the VRP. The results acquired by the Clarke and Wright method can be greatly affected due to the constraint for 'visiting customers in sequential order' since feasible merging can be reduced due to this constraint. Hence, the DB heuristic modifies the concept of the savings through penalizing the arcs that connect different types of nodes. The authors

experimentally proved that the best solutions for the VRPB can be achieved by delaying the construction of mix routes (i.e., routes with both linehaul and backhaul customers). However, the research published in Toth and Vigo (1999) (explained later in this section) argue that the results found by both methods, i.e., DB of Deif and Bodin (1984) and the Savings of the Clarke and Wright (1964) may remain infeasible in terms of the number of routes used for the final solution. This happens as both algorithms lack control over that aspect and in order to serve all customers in a given VRPB instance may require more routes for the final solution than found by these algorithms, hence resulting in an infeasible solution. Moreover, they argued that looking at DB algorithm from a practical point of view reveals that both the obtained routing cost of the solution and hence probability of solution being feasible are highly related to the number of route merging performed. Therefore, these drawbacks reduce the effectiveness of DB algorithm when it comes to finding the overall cost and obtaining feasible solutions for the VRPB instances. For more details and extension of the Clarke and Wright algorithm, see Golden *et al*. (1985), Casco *et al*. (1988) and Wassan (2007).

Goetschalckx and Jacobs-Blecha (1989) proposed a two-phase composite heuristic (see Section 2.5.2.6.) methodology to solve the VRPB. In the first phase of their heuristic, separate routes for linehaul and backhaul customers are generated based on the idea of space-filling curves. In space-filling curves, linehaul and backhaul vertices are separately transformed into points along a line from points in the plane. These routes are then combined together using space-filling mapping to achieve a set of final LH/BH routes. The initial solution is then further optimised by using the *2-Opt* and *3-Opt* (described in Section 2.5.2.4.) local search refinement routines. The two-phase heuristic produced some modest quality results.

Goetschalckx and Jacobs-Blecha (1993) developed a cluster-first-route-second (see Section 2.5.1.4) algorithm for the VRPB which is based on the generalized assignment methodology similar to the one developed in Fisher and Jaikumar (1981) for the VRP. This approach proved better and produced good quality solutions as compared to their approach in Goetschalckx and Jacobs-Blecha (1989).

Toth and Vigo (1996) developed a cluster-first-route-second heuristic algorithm (see Section 2.5.1.4) and called it TV. The TV algorithm is based on the approach published in Fisher (1994) for the VRPs where the initial solution is obtained by a relaxation approach similar to their published work in Toth and Vigo (1997) described earlier in this section. The TV algorithm used intra-route, i.e., *2-Opt* and *3-Opt* and inter-route, i.e., insertion and swap procedures as post optimisation to improve the final solution. (The intra-route, inter-route, insertion, swap procedures are already explained in Section 2.5.1.4). The TV algorithm was tested on two VRPB data sets, one consists of 62 instances from Goetschalckx and Jacobs-Blecha (1989) and the other one they generated from 11 VRP data instances using the same backhauling percentage conventions of data set one. This algorithm produced better results compared to the published works at that time.

Toth and Vigo (1999) developed a cluster-first-route-second heuristic algorithm (see Section 2.5.1.4.) by studying the VRPB with both symmetric and asymmetric travelling distances. Their cluster-first-route-second algorithm used a new and general clustering method to tackle both symmetric and asymmetric instances. This approach starts by constructing a group of clusters which contain either linehaul or backhaul customers; the clusters are then combined to achieve a (possibly infeasible) set of routes by solving the Assignment Problem. Hence, clusters are combined in such manner that linehaul

clusters are connected with backhaul clusters in order to form mixed routes and any linehaul clusters that are left are connected with the depot. The solution is further improved by using *intra-route* and *inter-route* (see Section 2.5.2) neighbourhood moves as refinement routines. This algorithm produced some good quality results compared to the previously published works, and set new benchmark solutions for asymmetric VRPB.

3.2.3. Metaheuristic Methods

Osman and Wassan (2002) studied the VRPB and developed a tabu search (see Section 2.6.1.1) algorithm. This was the first TS implementation to this problem. Their algorithm used the most sophisticated version of TS called Reactive Tabu Search (RTS). The RTS is believed to be the most efficient and effective among TS procedures as its main objective is to establish a balance between two very important strategies known as intensification and diversification in any TS approach. As oppose to TS, the RTS uses two mechanisms: 1) it performs large number of random moves to get out of local optima and 2) it dynamically increases or decreases *tt* value while evaluating the search process. The RTS algorithm proposed in this study used two savings based methods, the *saving-insertion* and the *saving-assignment*, to construct initial solutions followed by the reactive TS methodology in which two neighbourhood schemes, i.e., 1-interchange and 2-interchange (see Section 2.5.2.5) are used. The 2-interchange neighbourhood scheme moves are conducted by considering consecutive nodes shifts and swaps. In order to record the different values of neighbourhood moves, three data management structures are used. The results obtained by the RTS algorithm were superior quality as compared to the heuristics previously developed including Toth and Vigo (1997, 1999). Moreover a large number of solutions produced by the RTS

algorithm matched the optimal solutions produced by the exact algorithms of Mingozzi *et al*. (1996).

Brandao (2006) developed a tabu search (TS) algorithm for the VRPB. Two methods called *open initial solution* and *K-tree initial solution* are used to generate initial solution. The former involves two steps. In the first step, in order to solve the VRPB, two separate open vehicle routing problems (OVRPs) are solved. This is done because the OVRP is close in structure to the VRPB. Since in OVRP, the vehicles are not required to return to depot at the end of route. The OVRP solution is based on two phases, the *initial phase* and the *improvement phase*. In the *initial phase*, a nearest neighbour (NN) heuristic is used to generate a set of open-ended (i.e., a set of routes consisting of linehaul customers and a set of routes consisting of backhaul customers) routes sequentially. The NN procedure continues until all the customers are routed. Then in the *improvement phase*, tabu search is used. Here for the set of linehaul OVRP routes, TS minimises the overall distance travelled by the vehicles. Whereas for the set of backhaul OVRP routes, TS minimises the number of routes as well as distance travelled. In the second step 2, two Hamiltonian solution paths of OVRP for each LH and BH customers are linked together. Note that four different ways of connecting the end points of linehaul and backhaul routes are evaluated and the link returns the least cost is accepted to form a complete VRPB route. This process is repeated until either the backhaul or the linehaul paths are empty. The *K-tree initial solution* method is based on a lower bound. In this method, linehauls and backhauls are considered as customers only, hence, assuming the VRPB as the VRP. Then the VRP is formulated as a minimum cost *K-tree* as described in Fisher (1994a) with degree *2K* on the depot. Finally, 10 initial solutions are generated from each of 10 K-trees lower bounds. The solution generated by either (i.e., *open initial solution* or *K-tree initial solution*)

methods is improved by their TS implementation. The best performance of the TS algorithm is acquired with the *K-tree initial solution* method.

Ghaziri and Osman (2006) proposed a self-organizing feature maps (SOFM) methodology for the VRP with backhauls which is based on the concept of the Neural Networks. This algorithm is basically an extension of Ghaziri and Osman (2003) algorithm proposed for the Travelling Salesman Problem with backhauls. This algorithm begins by specifying the architecture of the network that comprises of one ring on which artificial neurons are spread spatially. The ring is embedded in the Euclidean space where each neuron is recognized by its position on the ring. Two post-optimisation procedures based on the *2-Opt* procedure are used to improve the solution quality. The technique of type one is used at the end of the algorithm; whereas, the type two is used periodically during the search process. Solutions found by their algorithm are of inferior quality compared to the algorithms of Toth and Vigo (1996, 1999) and Osman and Wassan (2002).

Røpke and Pisinger (2006) proposed a unified heuristic for a large class of vehicle routing problems with Backhauls. The unified heuristic uses large neighbourhood search (LNS) meta-heuristics originally developed in Shaw (1998). The LNS shares similarities with the concept of *Ruin and Recreate* (R&R) which was used in a framework proposed by Schrimpf *et al*. (2000). Various insertion and removal heuristics are used in this framework, some of them as diversification and others for intensification. Røpke and Pisinger embedded three different configurations and called it a unified heuristic methodology. These configurations (strategies) are named as *Standard*, *6R-no learning* and *6R-normal learning*. In the *Standard* configuration, three removal heuristics are used with a learning mechanism; the *6R-normal learning* uses 6

different types of removal heuristics without learning mechanism; and the *6R-normal learning* employs all 6 removal heuristics with learning mechanism (for full detail of the removal heuristics we refer the reader to their paper). The unified heuristic is tested on various data sets belonging to different backhauling variants including the classical VRPB. The unified heuristic performed very well on all data sets in terms of the solutions quality.

Wassan (2007) studied the VRPB and proposed a hybrid meta-heuristic algorithm that combines the processes of the reactive tabu search and adaptive memory programming (AMP). The RTS and AMP are considered as cutting-edge components of TS. The AMP component is based on long term memory structures and it used a wider framework in which strategies such as intensification and diversification are combined together. Both RTS and AMP approaches are coupled and utilised together in this study intelligently in order to obtain high quality solutions. The *savings-insertion* and the *savings-assignment* construction methods developed in Osman and Wassan (2002) are used to construct the initial solution. Solutions are reported for two benchmark VRPB data sets available in the literature. The RTS-AMP algorithm produced better quality solutions (45 new best/optimal) when compared with the best know solutions of two well-known VRPB data sets.

Gajpal and Abad (2009) developed a multi-ant colony system (see Section 2.6.3.1) algorithm called 'MACS' for the VRPB. In this study, the authors have used two types of ants, vehicle-ants and route-ants. In order to construct the feasible solution; two types of trail intensities called the *vehicle trail intensity* and the *route trail intensity* are used. After the initial solution constructed by the ants three types of local search procedures are used. These are *2-Opt*, customer insertion/interchange multi-route scheme and sub-

path exchange multi-route scheme. In order to avoid being trapped in local minima equal importance is given to the elite ants. The MACS algorithm produced competitive results with five new best known solutions compared to the studies published by then. Moreover it has been reported that the CPU time and solution quality of MACS approach can be controlled by varying the number of ants.

Tutuncu, Carreto and Baker (2009) investigated the classical VRPB and two of its extensions known as the mixed and the restricted VRP with backhauls. A decision support system (DSS), which is based on the GRAMPS (Greedy Randomized Adaptive Memory Programming Search, see Ahmadi and Osman (2005)) algorithm. This is basically a visual approach that is based on the work of Fisher and Jaikumars (1981) proposed for vehicle routing and was later extended by Baker in (1992). Their visual approach which they named as CRUISE2 (Computerised Routing Using Interactive Seeds Entry version 2) consists of three stages. The first stage has two phases called the seed selection and proposition phases respectively. At the seed selection phase, using visual representation of the seeds (customers) on the DSS, users can select customers for each vehicle manually or automatically. Whereas at the proposition phase, GRAMPS meta-heuristic construct routes and also performs a local search with learning process at each iteration. Once the classical VRPB solution is obtained at the first stage, the problem modification stage starts where users are optionally permitted to insert backhaul customers before linehaul customers in order to convert the solution into mixed VRPB or restricting backhaul customers' positions in order to make it restricted VRPB. Finally in the stage, the solver (GRAMPS) algorithm is called to obtain the final solution for the mixed and restricted VRPB. The visual DSS framework did not find better solutions when compared to the reactive tabu search algorithm of Osman and

Wassan (2002), however in terms of computational time and overall solution quality, the proposed framework seems quite competitive.

Zachariadis and Kiranoudis (2012) developed a local search heuristic for the classical VRPB that explores rich solution neighbourhoods (i.e., the neighbourhoods which are composed of variable length customer sequences) and makes use of local search moves stored in Fibonacci Heaps (Fibonacci Heaps are basically special types of priority queue structures that allows a program with capabilities such as fast insertion, deletion and retrieval). Moreover, they propose a parameter-free mechanism called "promises" which is based on the aspiration criterion mechanism of tabu search to achieve diversification and avoid cycling. The algorithm is tested on a VRPB data set proposed by Goetschalckx and Jacobs-Blecha (1989). The algorithm outperformed other algorithms in the literature in terms of solution quality.

Recently, Cuervo *et al*. (2013) developed an iterated local search algorithm for the classical VRPB in which an oscillating local search heuristic is used. At each iteration, a broader neighbourhood structure is explored and the information regarding neighbouring solutions is stored in a data structure. At the second stage, a constant transition between feasible and infeasible solution space is achieved by a heuristic while adjusting the transitions by a penalty associated with infeasible solutions dynamically. The iterated local search algorithm is tested on two VRPB benchmark data sets. The algorithm produced high quality solutions when compared with other state of the art algorithms in the literature.

3.2.4. Studies in VRPB-related areas

There are a lot of studies in the literature that are related to the VRPB, however we preclude them since the results could not be compared directly because these studies are the special cases of the VRPB and hence use different data sets. Nevertheless, we provide some references here for the interested readers.

Notable ones are the vehicle routing problem with delivery and backhaul options by Anily (1996); the vehicle routing problem with backhauls and inventory (VRPBI) by Liu and Chung (2009); the mixed vehicle routing problem with backhauls (MVRPB) by Wade and Salhi (2002), Lin and Tao (2011) and Wassan *et al*. (2013); the vehicle routing problem with restricted mixing of deliveries and pickups by Nagy, Wassan and Salhi (2013); the fleet size and mix vehicle routing problem with backhauls by Salhi, Wassan and Hajarat (2013); the vehicle routing problem with divisible deliveries and pickups by Nagy *et al*. (2015). More information on the modelling issues and meta-heuristics solution approaches on the vehicle routing problems involving pickups and deliveries; we refer the reader to Wassan and Nagy (2014).

3.3. An Overview of the MT-VRP

As briefly described in Section 2.1, the MT-VRP is a variant of the classical VRP. The MT-VRP along with the characteristics of the VRP includes a schedule for a vehicle that may serve a subset of routes within a given planning period. This means that an optimised set of routes maybe assigned to a given fleet (Taillard *et al*., 1996). This aspect of the MT-VRP makes it practically important in the context of the operational level where managers have to make driving schedules with a given fixed fleet and with

shorter distance distribution networks on a daily basis. Figure 3.2 shows an illustrative example of the MT-VRP.

A few formulations of the MT-VRP can be found in the literature. Olivera and Viera (2007) were the first to formulate the MT-VRP. Another closely related formulation was then introduced in Azi *et al*. (2010) who developed a branch-and-price model based on a set-packing formulation for the MT-VRP with an additional aspect of the time windows. More recently Mingozzi *et al*. (2013) developed two set-partitioning-like formulations for the MT-VRP.

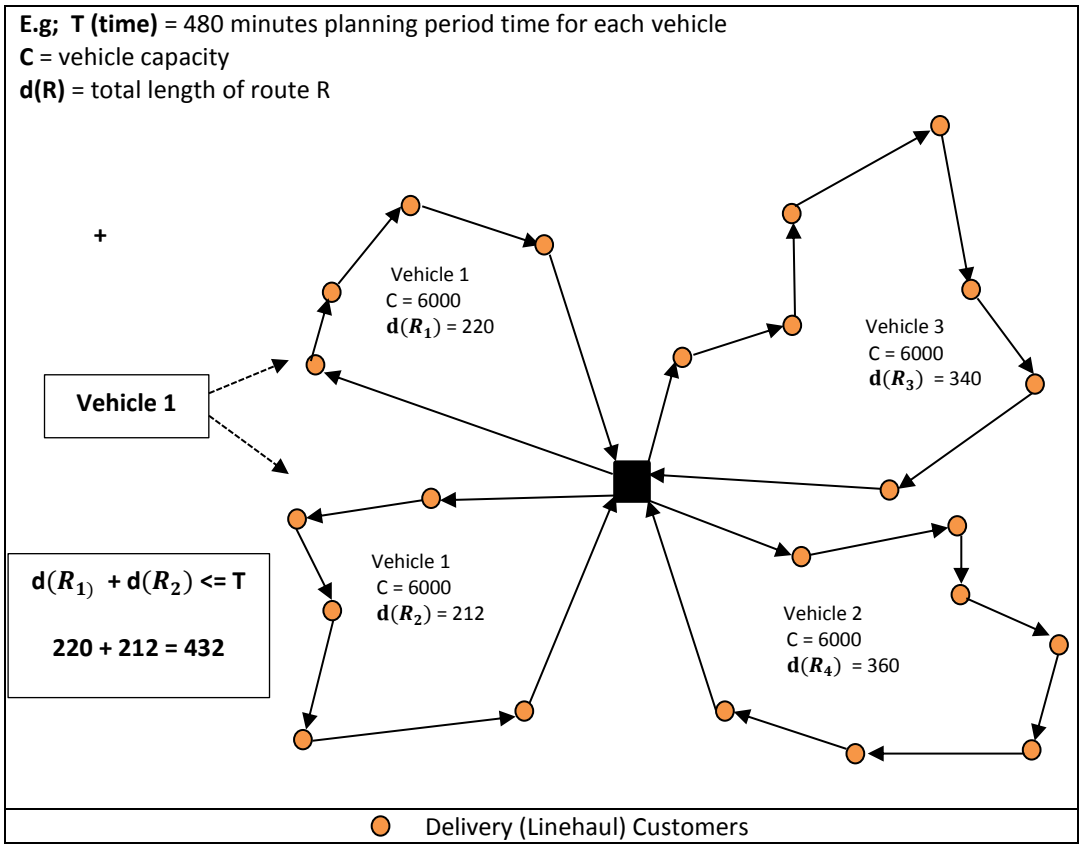

Figure 3.2: An illustrative example of the MT-VRP

In many publications, the circumstances in which a multi-trip scenario may arise and its importance is highlighted. We shall present a brief summary here. From the discussions

so far it has become obvious that in many vehicle routing applications, a vehicle may perform more than one trip in a single working day shift. Battara *et al*. (2009) argue that for a vehicle to perform more than one route arises where the vehicle capacity is small compared to the customer demands; hence fewer customers can be served in each trip. Another possibility is when spread time constraints (the constraints that the hours of any two visits to the same customer must differ by a given time constant) or strict time windows are imposed in a routing application. The importance of MT-VRP arises in many real-life situations where significant cost savings can be obtained by reducing the number of vehicles purchased/hired and hence drivers by taking advantage of multiple scheduling. Applications of the MT-VRP may arise in distribution of goods in urban areas, where travel periods are likely to be small; hence, the vehicles are often reloaded after performing short tours in order to be used again (Petch and Salhi, (2004), Olivera and Viera, (2007), Ahlem *et al*. (2011)).

3.4. Solution methods for the MT-VRP

The MT-VRP has not been studied extensively in the literature as compared to the other variants of the VRP. We present a review of the MT-VRP and its closely related studies in the chronological order of their publication by separating the exact and the heuristic methods as follows.

3.4.1. Exact Methods

There is only one optimal approach attempt in the literature due to Mingozzi, Roberti, and Toth (2013) who developed an exact method based on two set-partitioning-like formulations to tackle the MT-VRP. The first formulation demands a priori generation of all feasible routes; hence for each route and each vehicle, it has a binary variable that

specifies whether a given route is assigned to the schedule of a given vehicle. The second formulation is based on generating all feasible schedules for the vehicles; hence, for each schedule it also has a binary variable that specifies whether a schedule is performed or not. A subset of 52 instances, ranging in size from 50-120 customers, based on the classical MT-VRP benchmark instances, is tested and 42 of them are solved to optimality. For the rest, upper bounds are provided.

3.4.2. Heuristic Methods

The very first research that addresses the multiple trips aspect in the context of vehicle fleet mix is due to Salhi (1987). The study is kept limited to double trips only and a matching algorithm is used to assign routes to vehicles within a refinement process. The next study in the time line appears to be of Fleischmann (1990) who addressed the MT-VRP problem in his working paper. He proposed a modified savings algorithm and used a bin packing heuristic to assign the routes to vehicles.

Petch and Salhi (2004) developed a multi-phase constructive heuristic for the MT-VRP. The initial VRP solution is generated by using savings measure of Yellow (1970). The savings calculations are parameterized in order to obtain the pool of VRP solutions followed by the *2-Opt* and the *3-Opt* arc exchange heuristic procedures to improve the initial VRP solution. In order to obtain the MT-VRP solution, a bin-packing problem (BPP) approach is used to obtain the MT-VRP solution. In BPP, items of varying sizes are supposed to be packed into a finite number of bins with known capacity such that all items are packed into the minimum number of bins without violating the capacity of each bin. In the multiple trip context, the routes are considered as the items with their respective distances as their sizes and vehicles are represented as bins with associated maximum driving time as their respective capacity. Moreover, several tour

improvement procedures and route reassignment to vehicles are used to improve the MT-VRP solution. The bisection approach is used to prescribe the imposed bin sizes. Hence, where solutions are not packed feasibly, overtime is allowed and the solutions are reported with overtime cost. The proposed heuristic is tested on the MT-VRP data set proposed by Taillard *et al*. (1996). When compared with solutions produced by algorithm of Taillard *et al*. (1996) and algorithm of Brandao and Mercer (1997), this approach performs better. In terms of average overtime, this heuristic approach performed 29.59% lower than that of Taillard *et al*. (1996) algorithm and 25.27% higher when compared with the algorithm of Brandao and Mercer (1997). In terms of solution quality this approach performed better especially when compared with the algorithm of Taillard *et al*. (1996).

Ahlem *et al*. (2011) studied and combined two variants of VRP: the profitable VRP and multiple trips vehicle routing problem (MT-VRP). In terms of the profitable VRP, it has been discussed that in many real-life situations it is not possible to satisfy the entire customer's request due to lack of means or of inadequate demand. Therefore it is necessary to give priority to those customers who are more important potentially in the long term or have effective impact on recorded sales turnover. Moreover, this problem is very important practically in those situations where the companies have to face daily distribution schedules with a short course transportation network and have limited vehicle fleet. A mixed integer programming formulation is proposed to solve this problem and the problem primary objective is to maximize the sum of collected profit minus the transportation costs. Two greedy constructive heuristics are used which make use of some local procedure in the algorithm to optimise the solution. This algorithm is implemented in CPLEX and is tested on 20 new randomly generated instances by the authors and on the benchmarks MT-VRP data set of Taillard *et al*. (1996). The

constructive heuristic found optimal solutions for the small size instances within the given computational time limit. However it is reported that the optimal solution cannot be determined where the number of customers is more than 16. Thus for large instances upper and lower bound and their deviation is reported.

3.4.3. Metaheuristic Methods

Taillard *et al*. (1996) were the first researchers to study the MT-VRP. They developed a three phase tabu search heuristic to solve the MT-VRP which is based on the tabu search adaptive memory algorithm of Taillard (1993). In the first phase, a large set of vehicle routes of the classical VRP is produced using the algorithm of Taillard (1993) and routes forming the VRP solution are stored in the list (data structure). Secondly, an enumerative algorithm is used to select a subset of routes generated in the first phase. Finally, a Bin Packing Problem is solved for each VRP solution stored in the list and then the best solution is selected from all the packed solutions. The tabu search algorithm is tested on a number of MT-VRP instances which they generated from the VRP data instances of Christofides *et al*. (1979) and Fisher (1994). Their tabu search algorithm successfully found feasible solutions for most of the instances within reasonable times. Moreover, given the way MT-VRP instances were generated, the authors state that the results show that the feasible solutions (i.e., solutions found without overtime) are on average within 5% to 10% of the best known VRP solutions.

Brandao and Mercer (1997) studied a practical MT-VRP for the British company Burton's Biscuits Ltd. and termed it as the multi-trip vehicle routing and scheduling problem (MTVRSP). Many time related scheduling constraints close to practical world constraints are taken into account in their study. Moreover, as this problem deals with solving the real distribution problems with practical constraints and actual costs, real

distances are used. To solve the problem they developed a tabu search algorithm which consists of three phases. In the first phase, the tabu algorithm generates the initial solution by using nearest neighbour and insertion heuristics. At this initial stage the routes are created in a sequential constructive manner and all the routes are feasible in terms of routing constraints. There may be a possibility that the routes being constructed are infeasible in terms of scheduling constraints but this constraint is not considered at this stage. In the second phase, two objectives, to make the solution feasible (i.e., solution where no overtime is used) in terms of maximum driving time and time windows while decreasing the cost of the solution as much as possible are taken into account simultaneously. In the third phase, a set of swap and insert moves are performed to reduce the solution cost while maintaining feasibility. It has been reported that this algorithm improved over the manual solutions obtained by the company by approximately 20% on average.

In (1998) Brandao and Mercer provided a simplification of the above tabu search algorithm for the MT-VRP. This algorithm has no additional constraints as compared to the above real-world application algorithm. This tabu search algorithm generates the initial solution by using a nearest neighbour insertion heuristic and utilises insertion and swap moves in its search process. Moreover this algorithm takes into account the variable-size tabu list and aspiration criteria. Furthermore infeasible solutions (solutions found with overtime) are also allowed with respect to the maximum overtime permitted. The algorithm is tested on the data set proposed by Taillard *et al*. (1996). The solutions obtained through the proposed tabu search algorithm are compared with the solutions obtained by Taillard *et al*. (1996).

Salhi and Petch (2007) developed a genetic algorithm (GA) based heuristic to solve the MT-VRP (see Section 2.6.2.1). They claim this is the first GA based approach proposed for MT-VRP in the literature. The power of GA lies in that of new solutions can be generated simultaneously. Moreover, classical GAs normally involves binary based chromosome representation. But in practice it is difficult to convert a solution into binary representation. So in this study, the authors have developed a flexible non-binary chromosome structure that is established upon the circle partition concept of Thangiah and Salhi (2001) to address the above hurdle. The initial population of chromosome is obtained by the circle partition scheme that facilitates in providing a base for clustering and finally route generation. In order to maintain the solution quality and population diversity two mechanisms called *Injection* and *Cloning* are used. To generate new chromosome or offspring, the "extraction" and "mutation" operators are used. A savings heuristic is used to solve small VRP sub-problems whereas a bin packing heuristic is used to obtain the final set of vehicle trips. In order to further optimise the trips some post optimisation refinement modules proposed in Petch and Salhi (2004) are used. The algorithm is tested on MT-VRP data set proposed by Taillard *et al*. (1996) in the literature. According to the solutions quality, it appears that the proposed GA approach does not produce better results when compared with other algorithms proposed in the literature for the MT-VRP. However GA found solutions of reasonable quality in short time when compared to the other algorithms.

Olivera and Viera (2007) developed an adaptive memory programming (AMP) approach based on the AMP principle of Rochat and Taillard (1995) to solve the MT-VRP. The authors have also presented the mathematical programming formulation of the problem which is based on the set covering formulation of the VRPTW. The sweep algorithm is used to generate the initial solution by selecting customers randomly each

time. Initial solutions are then improved by using tabu search (TS) algorithm before storing in the memory *M* (data structure), hence storing the top quality solutions. In addition, the data structure in which routes are stored is sorted is ascending order only. This is performed according to the lexicographic criteria to ensure that good routes reside in the first positions of the memory. After that, a new solution *s* is selected from the memory *M* and a bin packing approach is utilised to pack the routes into vehicles while using some local search refinements based on reducing the driver overtime. The memory *M* is updated with new routes while poor solutions are discarded. The AMP algorithm is tested on the 104 benchmark instances proposed by Taillard *et al*. (1996). The AMP algorithm found 98 feasible solutions out of 104 when compared with the algorithms of Taillard *et al*. (1996), Brandao and Mercer (1998) and Petch and Salhi (2004).

Alonso *et al*. (2008) developed a tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. The authors call this problem the site-dependent multi-trip periodic vehicle routing problem (SDMTPVRP). This problem combines some of the characteristics of the VRP, PVRP (for periodic VRP, see Chao *et al*. (1995) and Cordeau *et al*. (1997), SDVRP (for site-dependant VRP, see Nag *et al*. (1988), Chao *et al*. (1999) and Cordeau and Laporte (2001)) and MT-VRP. The tabu search approach used to solve the SDMTPVRP and its particular cases is a modification of the tabu search algorithm developed in Cordeau *et al*. (1997) for the periodic VRP; hence the authors call this algorithm TS-ABB. However this approach differs in many ways that is; the definition of solution attributes, the construction of the neighbourhood, the evaluation of the objective function and finally the construction of the initial solution. According to the authors, the SDMTPVRP is the first problem of its kind so some new data instances are created to test the TS-ABB

algorithm. Moreover the PVRP and the SDVRP test problems are also solved through this approach. The algorithm is tested on the MT-VRP problems proposed by Taillard *et al*. (1996). The computational results obtained show that the TS-ABB algorithm found feasible solutions for most of the MT-VRP problems when compared to those obtained by Taillard *et al*. (1996) while taking approximately the same time.

Cattaruzza *et al*. (2014a) proposed a hybrid genetic algorithm for the MT-VRP that uses some adaptations from the literature. A new local search operator called the combined local search (CLS) is introduced that combines the standard VRP moves and performs the reassignments of trips to vehicles by using a swapping procedure to obtain a better solution. This algorithm produced good quality results. Cattaruzza *et al*. (2014b) extended the model to include time windows aspect and developed an iterated local search methodology to solve the problem.

3.4.4. Studies in MT-VRP related areas

There are a lot of studies in the literature that are related to the MT-VRP, we shall provide some notable references for the interested readers. Battarra *et al*. (2009) developed an adaptive guidance approach to heuristically solve the minimum multiple trip vehicle routing problem (MMTVRP). Azi *et al*. (2010) proposed an exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles (VRPTW). Derigs *et al*. (2011) solved a real-world vehicle routing problem with multiple use of tractors and trailers and EU-regulations for drivers arising in air cargo road feeder service (RFS) and is given a name VRPMTT-EU. Azi *et al*. (2014) proposed an adaptive large neighbourhood search (ALNS) for the vehicle routing problem with multiple trips and time windows (VRPMTW). For more details see Sen and Bulbul (2008).

3.4.5. Studies in which VRPB and MT-VRP are addressed in a combined way

There is one study which addresses the VRPB and MT-VRP with time windows in a combined way; hence it is briefly described as follows.

Ony and Suprayogi (2011) studied the vehicle routing problem with backhaul, multiple trips and time windows (VRPBMTTW). The authors proposed an ant colony optimisation (ACO) algorithm to tackle this problem. The proposed ACO is modified by adding a decoding process which generates solutions based on the VRPBMTTW constraints. However, the sequential insertion method is used as an initial solution generation mechanism. The algorithm is tested on a randomly generated data set; hence it is hard to confirm the quality of the solutions since no other study exists.

3.5. Summary

In this chapter we presented reviews of two important variants of the VRP called the VRPB and the MT-VRP. In Chapter 4 these two variants are merged to create a new VRP variant, also further studied separately in Chapter 7. Here we have presented problem statements and reviews of the methodologies developed to solve them. The methodologies for both the VRPB and the MT-VRP are presented in the chronological order of their publication by separating exact and heuristic methods for ease. We have also provided some important references for the studies of the related problems without going into the details as those could not be compared directly to the problem versions focussed in this thesis.

As for the VRPB, there have been some early attempts in late 90s to solve the problem optimally with some modest success. However, as expected for this kind of hard problems there is an ample material available on heuristics side. While the early studies

of traditional heuristic methods appear to be solving the bigger instances of the problem and producing reasonably good solutions; the more recent metaheuristic based algorithms preformed much better in terms of solution quality but at higher computational costs.

As for the MT-VRP, since its formal inception by Taillard *et al*. (1996) there are some good studies published in the literature. However, as compared to the VRPB it has not drawn tremendous attention. There is one good attempt on the optimal approach side to tackle the MT-VRP; however, several efficient heuristics/meta-heuristics methodologies are reported to solve this problem. Since the MT-VRP is more closely related to the classical VRP which has been studied extensively in the literature, hence we find more relevant works rather than direct comparison studies of MT-VRP. We think this thesis can be an attempt to fill some gap in the literature by studying this problem directly and jointly with the VRPB in the following chapters.

Chapter 4

The Multiple Trip Vehicle Routing Problem with Backhauls: Formulation and Analysis

This chapter focuses on the introduction of a new variant of the VRP being studied in this thesis i.e., the Multiple Trip Vehicle Routing Problem with Backhauls (MT-VRPB). A new mathematical formulation is proposed to solve the problem. The details of the MT-VRPB including the graph theoretical definition along with possible variations are also presented. An illustrative example showing the validation of the formulation is provided followed by the details of our CPLEX solution implementation. The chapter also provides details of a newly created large set of MT-VRPB data instances along with the results and analysis.

4.1. The Multiple Trip Vehicle Routing Problem with Backhauls

The MT-VRPB is created in this thesis by blending the characteristics of two well-studied variants of the VRP, i.e., VRP with Multiple Trips (MT-VRP) and the VRP with Backhauls (VRPB). In the MT-VRP a vehicle may perform several routes (trips) within

a given time period; and in the vehicle routing problem with backhauls (VRPB) a vehicle may pick up goods to bring back to the depot after the deliveries are made. Therefore in the MT-VRPB a vehicle may not only make more than one trip in a given planning period but it can also collect goods in each trip, see Sections 3.1 and 3.3 for the descriptions of the VRPB and the MT-VRP respectively. From the real life applications point of view both the MT-VRP and the VRPB can be even more practical than the classical VRP. In real-life routing applications, vehicles can be used more efficiently; for instance in VRPB, combining delivery and pickup operations can result in saving companies substantial routing costs. Golden *et al*. (1985) reported that grocery stores in USA saved $165 million by taking advantage of backhauling in 1982. On the other hand, maximising the usage of vehicles as it is done in the MT-VRP results in saving the number of vehicle required and hence savings in total distribution costs. Therefore, by combining the aspects from these two routing problems, a new version of the VRPs that we believe will help bridge the gap between theoretical academic studies and the reality is created. The statement of the MT-VRPB is as follows.

4.1.1. Description of the MT-VRPB

The MT-VRPB can be described as a VRP problem with the additional possibilities of having vehicles involved in backhauling and multiple trips in a single planning period. In the MT-VRPB the fleet considered is homogenous, a vehicle (note that a vehicle corresponds to a bin and these two terms are used interchangeably in this study) may perform more than one route (trip) in a single planning period and may serve backhaul (pickup) customers after serving all linehaul (delivery) customers, the fleet is operated from a single depot and the demands of all the customers must be fulfilled; the objective is to minimise the overall cost by reducing the total distance travelled. There are

implicit cost savings attached with the number of vehicles used. The details of the MT-VRPB are as follows.

Problem characteristics and conventions:

1. A given set of customers is divided into two subsets, i.e., delivery (linehaul) and pickup (backhaul).
2. A homogenous fleet of vehicles is located at a single depot.
3. A vehicle may perform more than one trip in a single planning period.
4. All delivery customers are served before any pickup ones.
5. Vehicles routes containing only backhauls are not permitted; however linehaul only routes are allowed.
6. Vehicle capacity constraints are enforced.
7. Note - The route length constraint is not imposed at this stage, however the model is flexible to add this constraint if needed.

The MT-VRPB is to design a set of minimum cost schedules in which each customer (LH/BH) is visited exactly once by the routes (originating and terminating at the same depot) included in the schedules.

Figure 4.1 presents a graphical illustration of the MT-VRPB. Three homogenous vehicles are shown serving a given set of customers with known demands. The distance of dR_3 or dR_4 (where d represents the distance of a respective route) combined with the distance of other three routes cannot be served by the same vehicle in a single planning period T (for example, T could correspond to eight hour working day; i.e., T = 480 minutes for each vehicle); hence two separate vehicles (Vehicle 2 and Vehicle 3) are used to serve these routes. In this study, the terms distance and planning period (travel

time) are used interchangeably. However, Vehicle 1 performs two trips in a single planning period, since the total distance of dR_1 and dR_2 is less than a given planning period time *T*.

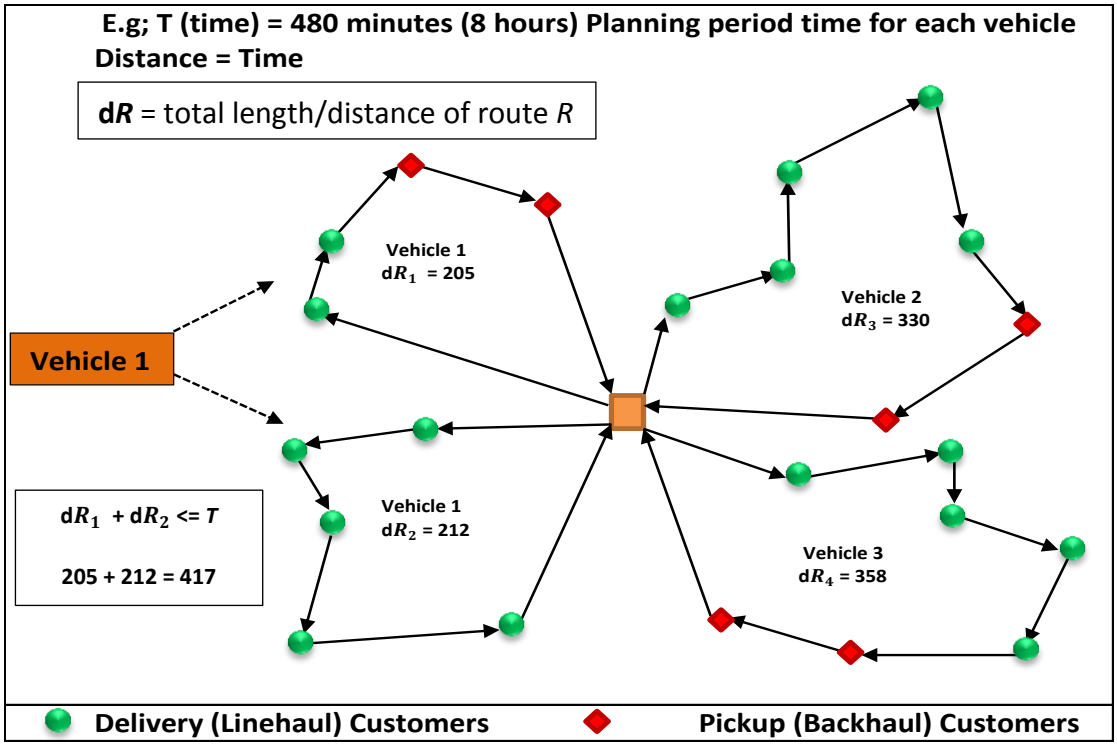

Figure 4.1: An example of the MT-VRPB

4.1.2. Graph theoretical definition of the MT-VRPB

The MT-VRPB can be defined on a graph as follows. Let $G = (N,\ A)$ be an undirected network, where $N = \{0\} \cup L \cup B$ is a set of nodes, $L = \{1, \dots, nl\}$ correspond to the linehaul (delivery) customers and $B = \{nl + 1, nl + 2, \dots nl + nb\}$ correspond to the backhaul (pickup) customers. $A = \{(i, j);\ where\ i, j \in N\}$ is the set of arcs and associated with arc (i, j), there is nonnegative given cost c_{ij} (distance between node *i* and node *j*). Node '0' represents the depot where a fleet $K = \{1, \dots, k\}$ of identical vehicles is located while the other nodes correspond to *L* and *B* customer sets. A non-

negative quantity q_i is associated with each (L/B) node i. Each vehicle has capacity C and maximum driving time T.

A travel cost c_{ij} and a travel time $\mathcal{T}_{ij}$ are associated with each arc $\{i,j\} \in A$. Therefore, a *route* of a vehicle is a least-cost elementary cycle in G that passes through a subset of customers starting and ending at the depot such that the customers visited and their total demand does not exceed the vehicle capacity C. A route cost (duration) is equal to the sum of the travel costs (travel times) of the nodes traversed. A vehicle schedule is a subset of routes whose combined duration is equal to or less than the maximum driving time T. Hence, the MT-VRPB call for the determination of constructing m schedules of least total cost in which each customers is visited exactly once by the routes of the schedules.

In the following section we review briefly the exact methods options for the MT-VRPB.

4.2. Exact methods options for the MT-VRPB

Several exact methods that can be used to solve the VRP and its variants are developed in the literature. These methods can be and have been extended by several researchers to address the additional practical constraints in the VRPs. The exact method approaches for the VRPs can be classified in to one of the following three categories.

Direct tree search methods: This kind of methods involves building VRP routes by means of a branch and bound tree search methodology (Christofides and Eilon (1969), Fisher (1994)).

Dynamic programming (DP): The Dynamic programming is a method that is used to solve a complex problem by breaking it down into a number of sub problems, hence

solving each one of them and storing their solutions. Therefore, these approaches start at an initial stage and go through a number of stages to reach an end stage. The methods can be computationally expensive if care is not taken in terms of the number of positions (Christofides, 1981b).

Integer linear programming (ILP): This approach is noted as being extensive and attracted a lot of attention in the literature. Based on the formulation used it is further divided into three categories (*i*-*iii*) (Laporte *et al*. (1987), Laporte (1992)).

(*i*) *Vehicle flow formulations*: are the most frequently used methods for the versions of the VRP known as two/three-index formulation associated with the decision variables. These methods use integer variables, which are connected with each arc or edge of the graph and hence, resulting in counting the number of times a vehicle traverses the arc or edge (Golden *et al*. (1977), Laporte *et al*. (1985), Toth and Vigo (2002)).

(*ii*) *Commodity flow formulations*: use additional variables v_{ij} that are connected with the arcs or edges and are responsible for representing the flow of the commodities along the routes journeyed by the vehicles (Gavish and Graves (1982), Toth and Vigo (2002)).

(*iii*) *Set-partitioning formulations* (also known as *Set-partitioning problem* (SPP)): usually use an exponential number of binary variables and each one of them is connected with a feasible circuit (Toth and Vigo, 2002).

Our formulation for the MT-VRPB is based on (*ii*) namely a three-index commodity flow formulation. This is provided in the next section.

4.3. Mathematical Formulation of the MT-VRPB

4.3.1. Formulation of the basic case

The MT-VRPB is modelled as an integer linear program. The following formulation is similar to the two-indexed commodity flow formulation of Nagy, Wassan and Salhi (2013). However, the MT-VRPB formulation is a three-index commodity flow formulation. In three-index formulations, variables x_{ijk} specify whether arc (i, j) is traversed by a particular vehicle k or not. On the other hand, in two-indexed formulation, it is not possible to know by variables x_{ij} which vehicle is used on arc (i, j) (Laporte, 1992).

The following notations are used throughout:

Sets

$\{0\}$ the depot (single depot)

L the set of linehaul customers

B the set of backhaul customers

K the set of vehicles (K: upper bound or the # of vehicles)

Input Variables

d_{ij} the distance between customers i and j ($i \in \{0\} \cup L \cup B, j \in \{0\} \cup L \cup B$)

q_i the demand of customer i (such that $i \in L$ for a delivery demand and $i \in B$ for a pickup demand)

Other Parameters

C vehicle capacity

T planning period (maximum driving time)

Decision Variables

$x_{ijk} = \begin{cases} 1, \text{ if vehicle } k \text{ travels from location } i \text{ directly to location } j; \\ 0, \text{otherwise} \end{cases}$

R_{ij} = is the amount of delivery or pickup on board on arc ij

Minimise $Z = \sum_{i\in\{0\}\cup L\cup B} \sum_{j\in\{0\}\cup L\cup B} \sum_{k\in K} d_{ij} x_{ijk}$ (4.1)

Subject to $\sum_{j\in\{0\}\cup L\cup B} \sum_{k\in K} x_{jik} = 1 \quad i \in L \cup B$ (4.2)

$\sum_{j\in\{0\}\cup L\cup B} \sum_{k\in K} x_{ijk} = 1 \quad i \in L \cup B$ (4.3)

$\sum_{j\in\{0\}\cup L\cup B} x_{jik} = \sum_{j\in\{0\}\cup L\cup B} x_{ijk} \quad i \in L \cup B, \ k \in K$ (4.4)

$\sum_{i\in\{0\}\cup L} R_{ij} - q_j = \sum_{i\in\{0\}\cup L\cup B} R_{ji} \quad j \in L$ (4.5)

$\sum_{i\in L\cup B} R_{ij} + q_j = \sum_{i\in\{0\}\cup B} R_{ji} \quad j \in B$ (4.6)

$R_{ij} \leq C \sum_{k\in K} x_{ijk} \quad i \in L \cup B, j \in L \cup B;$ (4.7)

$\sum_{i\in\{0\}\cup L\cup B} \sum_{j\in\{0\}\cup L\cup B} d_{ij} x_{ijk} \leq T \quad k \in K$ (4.8)

$R_{ij} = 0 \quad i \in L, \ j \in B \cup \{0\}$ (4.9)

$x_{ijk} = 0 \quad i \in B, \ j \in L, k \in K$ (4.10)

$x_{0jk} = 0 \quad j \in B, k \in K$ (4.11)

$R_{ij} \geq 0 \quad i \in \{0\} \cup L \cup B, \ j \in L \cup B$ (4.12)

$$x_{ijk} = 0,1 \qquad \begin{matrix} i \in \{0\} \cup L \cup B,\ j \in \{0\} \cup L \cup B \\ k \in K \end{matrix} \qquad (4.13)$$

Equation (4.1) illustrates the objective function representing the total distance travelled. Constraints (4.2) and (4.3) ensure that every customer is served exactly once (every customer has an incoming arc and every customer has an outgoing arc). Constraint (4.4) states that the number of times vehicle k enters into customer i is the same as the number of times it leaves customer i. The vehicle load variation on a route is ensured by Constraints (4.5) and (4.6) for linehaul and backhaul customers respectively. Inequalities (4.7) and (4.8) impose the maximum vehicle capacity constraint and the maximum working day period constraints in which a vehicle is allowed to serve the routes respectively. Constraints (4.19) restricts that a load cannot be carried from a linehaul customer to a backhaul customer or to the depot. Constraints (4.10) and (4.11) impose a restriction that a vehicle cannot travel from a backhaul to a linehaul customer and neither can it travel directly from the depot to a backhaul customer. Inequality (4.12) sets R_{ij} as a non-negative variable. Finally, in (4.13) the decision variable x_{ijk} is set as zero-one variable.

4.3.2. Model complexity

The mathematical model presented above (4.1)–(4.13) has $|L|(|B|+1)$ binary variables, $(|L|+|B|+1)^2|K|$ continuous variables and $3(|L|+|B|)+(|L|+|B|)^2+(|L|+|B|)(|K|)+|K|+|L|(|B|+1)+(|B|)(|L|)(|K|)+(|B|)(|K|)$ constraints.

An illustrative example

In order to check the complexity of our MT-VRPB mathematical model, we selected an instance of size small with 21 customers in total, where the number of linehaul (L)

customers is equal to 11 and the number of backhaul (B) is equal to 10 and the number of vehicle (K) is equal to 2. Hence, by calculating this has 121 binary variables, 968 continuous variables and 909 constraints. As it can be observed that even with an instance of smallest size (i.e., 21 customers in total), the complexity of the model is quite high.

4.3.3. Model variants and restricted problems

The above MT-VRPB formulation may be modified to cater for the following four variants.

a) The MT-VRP: this can be achieved by simply setting the number of backhaul customers equal to zero using equation (4.14).

$$B = \emptyset \quad \text{(setting the number of backhaul customers equal to zero)} \tag{4.14}$$

b) In the above formulation, K is implicitly used as an upper bound though it was observed that in all cases all K vehicles are used. However, the formulation can be extended to cater for the condition where the number of vehicles to be used is exactly as given number K. This imposes that all drivers will be used and this can be achieved by adding the following constraints in (4.15).

$$\sum_{j \in L \cup B} x_{ijk} = K \qquad i \in \{0\};\ i \in L \cup B;\ k \in K \tag{4.15}$$

c) The VRPB: this can be achieved by adding the following constraint (4.16) in the model.

$$\sum_{j \in L \cup B} x_{ijk} \leq 1 \qquad i \in \{0\};\ \ k \in K \tag{4.16}$$

Constraints (4.16) impose restrictions on every vehicle to be used at most once (equal sign may be used if every vehicle must be utilised) and therefore block the use of multiple-trips of vehicles.

d) Finally, the objective function in the above formulation can be changed from reducing the total distance travelled to reducing the number of vehicles. This can be achieved by setting the objective as shown in equation (4.17).

$$\text{Minimise} \qquad Z = \sum_{k \in K} \sum_{j \in L \cup B} x_{0jk} \tag{4.17}$$

Or one could set the objective function shown in equation (4.17) as a primary objective and reducing the total distance travelled as a secondary objective.

4.4. Significance of the MT-VRPB

In the literature both the VRPB and the MT-VRP are considered very important on the operational level since backhauling and multiple scheduling are seen in many real-life applications, where significant cost savings (e.g., operational, fixed costs) can be achieved by reducing the number of vehicles and hence drivers (Wassan (2007), Salhi and Petch (2007) and Ahlem *et al*. (2011)). We believe studying them in a combined way would be even more pragmatic in many real life situations to enhance the overall distribution logistics efficiency. The MT-VRPB appears in many real-world applications such as distribution of groceries, couriers who offer the same day collection and delivery services. Moreover the MT-VRPB especially arises in urban areas where travel times (distances) are rather small and the light load vehicles are reloaded after performing the short tours and used again. The growing examples of those cases arise in online business such as retail markets; i.e., grocery stores, cafes, supermarkets,

restaurants etc. The model may cater largely for those small, medium or large logistics companies that wish to use their limited/fixed fleet or strategically want to reduce the vehicle fleet size. To our knowledge, this is the first time the MT-VRPB is being defined, formulated and studied in such detail.

When it comes to solving this kind of hard complex but important problem efficiently, as indicated in Chapter 2, exact methods have shown a limited success in tackling them. Nevertheless we have used CPLEX for the purpose of formulation validation, optimal solutions for smaller instances and upper/lower bounds for larger ones to check the performance of the meta-heuristic algorithm that will be proposed in the next Chapter. In the following sections we briefly look at the utility of CPLEX software, the details of new data set generation and our CPLEX solution approach for the MT-VRPB.

4.5. Utility of IBM ILOG CPLEX optimisation studio

Technology plays a very fundamental role in almost every sector in this modern era. Especially in the last couple of decades, rapid advancement in the computer and software industry has dramatically changed the way work is carried out in most organizations. Software development organizations such as IBM, Sun, Microsoft and many others have advanced in many fronts and played a major role in developing software packages and tools for countless public and private sectors. By learning and utilizing those software tools and packages we can get our jobs or tasks done in matter of seconds which probably could not be achieved before so easily. In this study we are using a very powerful and efficient package called IBM ILOG CPLEX optimisation studio developed by IBM Corporation to solve the various types of optimisation and business related problems which is briefly described here.

The IBM ILOG CPLEX is an optimisation tool that is implanted with very powerful and reliable solvers (algorithms) that are based on high-performance mathematical programming. These solvers can efficiently handle and solve a variety of problems; i.e., mixed integer programming (MIP), quadratically constrained programming (QCP), linear programming (LP) and quadratic programming (QP) problems. Moreover, CPLEX offers a specific optimiser called CP optimiser particularly for scheduling and combinatorial problems. This optimiser utilises complimentary optimisation technology based on constraint programming. IBM has launched various versions of CPLEX so far but in this study we are employing the CPLEX 12.5 version (User's Manual for CPLEX V12.5).

4.6. Validation of the MT-VRPB formulation

In order to check the validity of our proposed formulation we created a numerical test instance containing 5 customers where nodes 1-3 represent the linehaul customers and nodes 4 and 5 represent the backhaul customers. The maximum driving time T was set to 25, the C (vehicle capacity) is set to 8 units, and the number of bins Tnb (total number of bins, i.e., vehicles) is set to 2. The data of the numerical test instance is illustrated in Figure 4.2.

$\boldsymbol{n} = 5$ [Total customers] $\boldsymbol{b} = 2$ [no. of backhauls] $\boldsymbol{T} = 25$ [maximum driving time] $\boldsymbol{C} = 8$ [vehicle capacity] $\boldsymbol{Tnb} = 2$ [no. of bins (vehicles)]	**Demands** = [6, 5, 7, 7, 2] **Dist. matrix** = $\begin{bmatrix} 0 & 2 & 4 & 3 & 5 & 8 \\ 2 & 0 & 3 & 6 & 4 & 7 \\ 4 & 3 & 0 & 5 & 3 & 1 \\ 3 & 6 & 5 & 0 & 7 & 3 \\ 5 & 4 & 3 & 7 & 0 & 4 \\ 8 & 7 & 1 & 3 & 4 & 0 \end{bmatrix}$

Figure 4.2: The numerical test instance data

This test instance was solved using CPLEX; the optimal solution is shown in Figure 4.3. The optimal solution contains three routes with a total distance of 30 where routes R_1 and R_3 are served by vehicle 2 and route R_2 is served by vehicle 1.

To ensure the CPLEX solution validates our mathematical formulation, we generated all the possible 6 feasible solutions of the test instance enumerating by hand, and found the same solution produced by CPLEX. Figure 4.4 shows all the possible feasible solutions for the test instance.

	Nodes					Cuts/		
	Node	Left	Objective	IInf	Best Integer	Best Bound	ItCnt	Gap
*	0+	0			40.0000	22.0000	6	45.00%
	0	0	30.0000	4	40.0000	30.0000	6	25.00%
*	0+	0			30.0000	30.0000	6	0.00%
	0	0	cutoff		30.0000	30.0000	6	0.00%

The optimal solution routes along with distances:

$\boldsymbol{R_1 = 0 \rightarrow 2 \rightarrow 4 \rightarrow 0}$ $\boldsymbol{length\,(R_1) = 12}$

$\boldsymbol{R_2 = 0 \rightarrow 3 \rightarrow 5 \rightarrow 0}$ $\boldsymbol{length\,(R_2) = 14}$ $\{\boldsymbol{total\ dist.travelled = 30}\}$

$\boldsymbol{R_3 = 0 \rightarrow 1 \rightarrow 0}$ $\boldsymbol{length\,(R_3) = 4}$

Figure 4.3: The CPLEX solution for test instance

$0 \rightarrow 1 \rightarrow 5 \rightarrow 0 = 17$ $0 \rightarrow 2 \rightarrow 4 \rightarrow 0 = 12$ $\{tot.dist = 35\}$ $0 \rightarrow 3 \rightarrow 0 = 6$	$0 \rightarrow 2 \rightarrow 4 \rightarrow 0 = 12$ $0 \rightarrow 3 \rightarrow 5 \rightarrow 0 = 14$ $\{tot.dist = 30\}$ $0 \rightarrow 1 \rightarrow 0 = 4$
$0 \rightarrow 1 \rightarrow 4 \rightarrow 0 = 12$ $0 \rightarrow 2 \rightarrow 5 \rightarrow 0 = 13$ $\{tot.dist = 31\}$ $0 \rightarrow 3 \rightarrow 0 = 6$	$0 \rightarrow 2 \rightarrow 5 \rightarrow 0 = 13$ $0 \rightarrow 3 \rightarrow 4 \rightarrow 0 = 15$ $\{tot.dist = 32\}$ $0 \rightarrow 1 \rightarrow 0 = 4$
$0 \rightarrow 3 \rightarrow 4 \rightarrow 0 = 15$ $0 \rightarrow 1 \rightarrow 5 \rightarrow 0 = 17$ $\{tot.dist = 40\}$ $0 \rightarrow 2 \rightarrow 0 = 8$	**Best solution with minimum total distance:**
$0 \rightarrow 3 \rightarrow 5 \rightarrow 0 = 14$ $0 \rightarrow 1 \rightarrow 4 \rightarrow 0 = 11$ $\{tot.dist = 33\}$ $0 \rightarrow 2 \rightarrow 0 = 8$	$0 \rightarrow 2 \rightarrow 4 \rightarrow 0 = 12$ $0 \rightarrow 3 \rightarrow 5 \rightarrow 0 = 14$ $\{tot.dist = 30\}$ $0 \rightarrow 1 \rightarrow 0 = 4$

Figure 4.4: All feasible solutions for test instance

4.7. Generation of a new data set for the MT-VRPB

To test our model we have generated a set of new MT-VRPB instances, *set-1*, from a set of 21 VRPB instances proposed in Toth and Vigo (1996, 1997). The data *set-1* uses the original VRPB and MT-VRP conventions established in Toth and Vigo (1996, 1997) and in Taillard *et al*. (1996). The data *set-1* contains 168 problem instances by using different values of v (where v is the number of vehicles, (i.e., 1,…,4), starting with an integer between one and the maximum number of vehicles) and T (where T is a maximum driving time). Two values of T are used, T_1 and T_2 for each value of v, where T_1 and T_2 are calculated as follows:

$$T_1 = [1.05\ z^*/v] \qquad T_2 = [1.1\ z^*/v]$$

The resulting values of both T_1 and T_2 are rounded up to the nearest integer (for example, if the resulting value of T is 389.60, then it is rounded up to 390), where z^* represents the VRPB solution obtained by our *Two-Level VNS* algorithm (details provided in Chapter 5) using a free vehicle fleet.

Several MT-VRPB instances are generated from each VRPB problem using T_1 and T_2 with the linehaul percentage of 50, 66, and 80%, respectively. Further details of the data *set-1* are provided in Table 4.1 and Table 4.2. The data sets are made available for researchers to be downloaded from the CLHO website (CLHO, 2015).

Table 4.1: The MT-VRPB data *set-1* with original conventions and *z found with free fleet.**

Problem number	Problem Name	*n*	*L*	*B*	*C*	Orig. fleet	*v* (free-fleet)	*z**
1	eil22_50	21	11	10	6000	3	1,…,3	371
2	eil22_66	21	14	7	6000	3	1,…,3	366
3	eil22_80	21	17	4	6000	3	1,…,3	375
4	eil23_50	22	11	11	4500	2	1,…,3	677
5	eil23_66	22	15	7	4500	2	1,…,3	640
6	eil23_80	22	18	4	4500	2	1,…,2	623
7	eil30_50	29	15	14	4500	2	1,…,2	501
8	eil30_66	29	20	9	4500	3	1,…,3	537
9	eil30_80	29	24	5	4500	3	1,…,3	514
10	eil33_50	32	16	16	8000	3	1,…,3	738
11	eil33_66	32	22	10	8000	3	1,…,3	750
12	eil33_80	32	26	6	8000	3	1,…,3	736
13	eil51_50	50	25	25	160	3	1,…,3	559
14	eil51_66	50	34	16	160	4	1,…,4	548
15	eil51_80	50	40	10	160	4	1,…,4	565
16	eilA76_50	75	37	38	140	6	1,…,6	738
17	eilA76_66	75	50	25	140	7	1,…,7	768
18	eilA76_80	75	60	15	140	8	1,…,8	781
19	eilA101_50	100	50	50	200	4	1,…,5	827
20	eilA101_66	100	67	33	200	6	1,…,6	846
21	eilA101_80	100	80	20	200	6	1,…,7	859

n: number of customers in an instance; ***L***: number of linehauls; ***B***: number of backhauls; ***C***: vehicle capacity; **Orig. fleet**: actual fixed fleet used in base problem; ***v* (free fleet)**: free fleet used by the *Two-Level VNS*; ***z****: free fleet VRPB solution.

Table 4.2: The details of the MT-VRPB data *set-1*

Name	*n*	*L*	*B*	*C*	*v*	*z**	Tnb	T_1	T_2
eil22_50	21	11	10	6000	1,…,3	371	1	390	408
							2	195	204
							3	130	137
eil22_66	21	14	7	6000	1,…,3	366	1	385	403
							2	193	201
							3	129	134
eil22_80	21	17	4	6000	1,…,3	375	1	394	413
							2	197	206
							3	132	138
eil23_50	22	11	11	4500	1,…,3	677	1	711	745
							2	355	372
							3	237	248
eil23_66	22	15	7	4500	1,…,3	640	1	672	704
							2	336	352
							3	224	235
eil23_80	22	18	4	4500	1,…,2	623	1	654	685
							2	327	343
eil30_50	29	15	14	4500	1,…,2	501	1	526	551
							2	264	276
eil30_66	29	20	9	4500	1,…,3	537	1	564	591
							2	282	296
							3	188	197
eil30_80	29	24	5	4500	1,…,3	514	1	540	565
							2	270	283

Name	***n***	***L***	***B***	***C***	***v***	***z****	**Tnb**	T_1	T_2
							3	180	188
eil33_50	32	16	16	8000	1,…,3	738	1	775	812
							2	388	406
							3	258	271
eil33_66	32	22	10	8000	1,…,3	750	1	788	825
							2	394	413
							3	263	275
eil33_80	32	26	6	8000	1,…,3	736	1	773	810
							2	387	405
							3	258	270
eil51_50	50	25	25	160	1,…,3	559	1	587	615
							2	294	308
							3	196	205
eil51_66	50	34	16	160	1,…,4	548	1	576	603
							2	288	302
							3	192	201
							4	144	151
eil51_80	50	40	10	160	1,…,4	565	1	594	622
							2	297	311
							3	198	208
							4	149	156
eilA76_50	75	37	38	140	1,…,6	738	1	775	812
							2	388	406
							3	259	271
							4	194	203
							5	155	163
							6	130	136
eilA76_66	75	50	25	140	1,…,7	768	1	807	845
							2	404	423
							3	269	282
							4	202	212
							5	162	169
							6	135	141
							7	116	121
eilA76_80	75	60	15	140	1,…,8	781	1	821	860
							2	411	430
							3	274	287
							4	206	215
							5	165	172
							6	137	144
							7	118	123
							8	103	108
eilA101_50	100	50	50	200	1,…,5	827	1	869	910
							2	435	455
							3	290	304
							4	218	228
							5	174	182
eilA101_66	100	67	33	200	1,…,6	846	1	889	931
							2	445	466
							3	297	311
							4	223	233
							5	178	187
							6	149	156
eilA101_80	100	80	20	200	1,…,7	859	1	902	945
							2	451	473
							3	301	315
							4	226	237
							5	181	189
							6	151	158
							7	129	135

Name: instance identification name; ***v***: number of vehicles - starting with an integer between one and the maximum number of vehicles; **Tnb**: total number of vehicles in each instance; $\boldsymbol{T_1}$: maximum driving time of type one for each vehicle; $\boldsymbol{T_2}$: maximum driving time of type two for each vehicle.

4.8. CPLEX Results and Analysis

The MT-VRPB model is solved using IBM ILOG CPLEX 12.5 optimiser and it was run on a PC with Intel(R) Core(TM) i7-2600 processor, CPU speed 3.40 GHz and installed memory (RAM) 4.00 GB (2.94 GB usable).

The optimal solutions and upper/lower bounds for the MT-VRPB are reported in Table 4.3 and Table 4.4 for T_1 and T_2, respectively. For each instance the CPLEX time was fixed to 2 hours. A reasonable number of optimal solutions are found for both $T_1$and T_2 groups of instances, ranging in size between 21 and 50 customers along with an instance of size 100 of T_2. Within the allocated time, CPLEX found 60 optimal solutions (i.e., T_1= 24, T_2= 36) out of all the 168 instances. The instances for which CPLEX could not find the solutions or reported as infeasible is due to either the vehicle(s) given time restriction and/or the instances are too large in size. We report upper bound and lower bound for those instances. CPLEX reported infeasibility in four cases where the number of vehicles increases and hence the given driving time decreases for each vehicle. This is due to the fact that the driving time is very small for each vehicle in these instances. Therefore, not even a lower bound could be obtained; hence, CPLEX reported infeasibility.

Table 4.3: CPLEX solutions for data *set-1* with T_1 (2-hours running time)

Name	T_1	Tnb	Optimal Sol.	No. Routes	Actual Time (s)	UB	LB
eil22_50	390	1	**371**	3	1.04	371.0000	367.5294
	195	2	**378**	3	1.17	378.0000	368.0119
	130	3	x	x	x	x	x
eil22_66	385	1	**366**	3	1.01	366.0000	364.9640
	193	2	**382**	4	3.02	382.0000	366.0000
	129	3	x	x	x	x	x
eil22_80	394	1	**375**	3	1.94	375.0000	362.1650
	197	2	**378**	4	2.39	378.0000	364.9665
	132	3	**381**	3	27.13	381.0000	369.0667
eil23_50	711	1	**677**	3	0.33	677.0000	677.0000
	355	2	**698**	3	2.36	698.0000	671.8600
	237	3	x	x	x	x	x
eil23_66	672	1	**640**	3	1.22	640.0000	633.1636
	336	2	**640**	3	1.4	640.0000	635.5000
	224	3	x	x	x	x	x
eil23_80	654	1	**623**	2	1.44	623.0000	618.0870
	327	2	**634**	2	1.59	634.0000	613.3380
Eil30_50	526	1	**501**	2	0.44	501.0000	500.3902
	264	2	x	x	x	x	x
Eil30_66	564	1	**537**	3	2.68	537.0000	511.3725
	282	2	**552**	3	6116	552.0000	537.0000
	188	3	-	-	7200	-	533.7612
Eil30_80	540	1	**514**	3	11.95	514.0000	474.9762
	270	2	-	-	7200	-	459.3289
	180	3	-	-	7200	-	460.3190
eil33_50	775	1	**738**	3	0.51	738.0000	738.0000
	388	2	-	-	7200	-	738.3900
	258	3	-	-	7200	-	740.7581
eil33_66	788	1	**750**	3	2.23	750.0000	732.7999
	394	2	**772**	3	1219.03	772.0000	757.8079
	263	3	-	-	7200	-	746.4629
eil33_80	773	1	**736**	3	121.27	736.0000	733.8901
	387	2	-	-	7200	-	720.3275
	258	3	-	-	7200	-	690.0837
eil51_50	587	1	**559**	3	9.84	559.0000	552.1063
	294	2	-	-	7200	-	550.1111
	196	3	-	-	7200	-	553.0000
eil51_66	576	1	**548**	4	22.23	548.0000	537.7475
	288	2	-	-	7200	-	546.1393
	192	3	-	-	7200	-	542.1467
	144	4	-	-	7200	-	522.9460
eil51_80	594	1	**565**	4	4552.80	565.0000	553.1885
	297	2	-	-	7200	-	555.5726
	198	3	-	-	7200	-	556.1191
	149	4	-	-	7200	-	556.1018
eilA76_50	775	1	-	-	7200	-	708.2119
	388	2	-	-	7200	-	721.9806
	259	3	-	-	7200	-	721.8691
	194	4	-	-	7202	-	711.64.91
	155	5	-	-	7200	-	705.6147
	130	6	-	-	7200	-	708.1701
eilA76_66	807	1	-	-	7200	-	738.1007

Name	T_1	Tnb	Optimal Sol.	No. Routes	Actual Time (s)	UB	LB
	404	2	-	-	7200	-	737.9937
	269	3	-	-	7200	-	734.0403
	202	4	-	-	7200	-	739.9000
	162	5	-	-	7200	-	733.5028
	135	6	-	-	7200	-	739.4740
	116	7	-	-	7200	-	737.0274
eilA76_80	821	1	-	-	7200	-	739.7246
	411	2	-	-	7200	-	726.3083
	274	3	-	-	7200	-	733.6667
	206	4	-	-	7200	-	733.5946
	165	5	-	-	7200	-	732.5992
	137	6	-	-	7200	-	724.3518
	118	7	-	-	7200	-	723.4398
	103	8	-	-	7200	-	718.6787
eilA101_50	869	1	-	-	7200	-	799.5710
	435	2	-	-	7200	-	804.1183
	290	3	-	-	7200	-	802.2318
	218	4	-	-	7200	-	807.1541
	174	5	-	-	7200	-	767.5958
eilA101_66	889	1	-	-	7200	-	829.5004
	445	2	-	-	7200	-	837.3865
	297	3	-	-	7200	-	826.1638
	223	4	-	-	7200	-	815.4809
	178	5	-	-	7200	-	832.78.09
	149	6	-	-	7200	-	816.1044
eilA101_80	902	1	-	-	7200	-	827.3494
	451	2	-	-	7200	-	797.3486
	301	3	-	-	7200	-	790.1850
	226	4	-	-	7200	-	820.9844
	181	5	-	-	7200	-	821.9659
	151	6	-	-	7200	-	799.1573
	129	7	-	-	7200	-	825.4779
# of optimal solutions found			**24**				
Average solution/time			**554.79**		**5165**		
Average CPU time(s) where sol. is found					**417**		

T_1 = Total planning time for a vehicle

Tnb = total number of vehicles in each instance

Optimal Sol.= Optimal solution found by ILOG CPLEX 12.5

No. routes = Total number of routes

Actual time (s) = Actual time taken by ILOG CPLEX to find the optimal solution

UB = Upper bound

LB = Lower bound

x = Infeasible

Table 4.4: CPLEX solutions for data *set-1* with T_2 (2-hours running time)

Name	T_2	Tnb	Optimal Sol.	No. Routes	Actual Time (s)	UB	LB
eil22_50	408	1	**371**	3	0.89	371.0000	370.6087
	204	2	**375**	3	1.67	375.0000	374.0333
	137	3	**378**	3	1.22	378.0000	364.4367
eil22_66	403	1	**366**	3	1.3	366.0000	364.7095
	201	2	**382**	4	1.67	382.0000	366.0000
	134	3	**366**	3	0.59	366.0000	366.0000
eil22_80	413	1	**375**	3	2.72	375.0000	358.9261
	206	2	**378**	4	8.5	378.0000	362.2288
	138	3	**381**	3	24.21	381.0000	364.9274
eil23_50	745	1	**677**	3	0.33	677.0000	677.0000
	372	2	**689**	3	1.98	689.0000	680.0000
	248	3	**716**	3	2.46	716.0000	682.1268
eil23_66	704	1	**640**	3	0.75	640.0000	640.0000
	352	2	**640**	3	1.23	640.0000	631.5000
	235	3	-	-	7200	-	662.4548
eil23_80	685	1	**623**	2	0.91	623.0000	617.8667
	343	2	**631**	2	1.4	631.0000	614.5388
Eil30_50	551	1	**501**	2	0.44	501.0000	500.3902
	276	2	**501**	2	0.73	501.0000	501.0000
Eil30_66	591	1	**537**	3	3.09	537.0000	510.3183
	296	2	**552**	3	3451.24	552.0000	538.0355
	197	3	**538**	3	1.56	538.0000	534.6250
Eil30_80	565	1	**514**	3	10.58	514.0000	482.8207
	283	2	**535**	3	5758	535.0000	525.2368
	188	3	**518**	3	1426.17	518.0000	500.1891
eil33_50	812	1	**738**	3	0.44	738.0000	738.0000
	406	2	**741**	3	2.26	741.0000	736.2820
	271	3	-	-	7200	803.0000	658.5384
eil33_66	825	1	**750**	3	11.7	750.0000	734.5884
	413	2	**767**	3	109.26	767.0000	764.4997
	275	3	-	-	7200	-	746.9500
eil33_80	810	1	**736**	3	136.31	736.0000	716.7393
	405	2	-	-	7200	-	723.4224
	270	3	-	-	7200	-	696.3739
eil51_50	615	1	**559**	3	11.23	559.0000	553.6224
	308	2	**560**	4	67.17	560.0000	550.4380
	205	3	**564**	4	67.49	573.0000	559.6480
eil51_66	603	1	**548**	4	11.87	548.0000	541.1877
	302	2	**548**	4	55.52	548.0000	546.9363
	201	3	-	-	7200	-	521.0965
	151	4	-	-	7200	-	539.9353
eil51_80	622	1	**565**	4	78.13	565.0000	562.5255
	311	2	-	-	7200	-	554.3046
	208	3	-	-	7200	-	553.8339
	156	4	-	-	7200	-	554.7640
eilA76_50	812	1	-	-	7200	-	710.0593
	406	2	-	-	7200	-	722.0668
	271	3	-	-	7201	-	720.4398
	203	4	-	-	7202	-	705.7348
	163	5	-	-	7200	-	706.7157
	136	6	-	-	7200	-	719.6408
eilA76_66	845	1	-	-	7200	-	734.9762

Name	T_2	Tnb	Optimal Sol.	No. Routes	Actual Time (s)	UB	LB
	423	2	-	-	7200	-	741.8414
	282	3	-	-	7200	-	734.1823
	212	4	-	-	7200	-	742.2662
	169	5	-	-	7200	-	738.0464
	141	6	-	-	7200	-	736.3244
	121	7	-	-	7200	-	733.6417
eilA76_80	860	1	-	-	7200	-	741.6530
	430	2	-	-	7200	-	732.6903
	287	3	-	-	7200	-	733.3761
	215	4	-	-	7200	-	733.4002
	172	5	-	-	7200	-	730.9763
	144	6	-	-	7200	-	731.1909
	123	7	-	-	7200	-	722.2782
	108	8	-	-	7200	-	733.8520
eilA101_50	910	1	-	-	7200	-	801.4182
	455	2	-	-	7200	-	813.7763
	304	3	-	-	7200	-	808.5073
	228	4	-	-	7200	-	803.0867
	182	5	-	-	7200	-	781.9759
eilA101_66	931	1	**846**	6	268.45	846.0000	840.8321
	466	2	-	-	7200	-	822.6394
	311	3	-	-	7200	-	831.4000
	233	4	-	-	7200	-	825.1924
	187	5	-	-	7200	-	814.6440
	156	6	-	-	7200	-	835.2673
eilA101_80	945	1	-	-	7200	-	828.6658
	473	2	-	-	7200	-	808.3282
	315	3	-	-	7200	-	819.9952
	237	4	-	-	7200	-	803.4907
	189	5	-	-	7200	-	817.7601
	158	6	-	-	7200	-	812.1149
	135	7	-	-	7200	-	816.7851
# of optimal solutions found			**36**				
Average solution/time			**558.50**		**4251**		
Average CPU time(s) where sol. is found					**313**		

T_2 = Total planning time for a vehicle

Tnb = total number of vehicles in each instance

Optimal Sol.= Optimal solution found by ILOG CPLEX 12.5

No. routes = Total number of routes

Actual time (s) = Actual time taken by ILOG CPLEX to find the optimal solution

UB = Upper bound

LB = Lower bound

x = Infeasible

Moreover, Tables 4.5 and 4.6 present further analysis that is performed to show very large vehicle cost savings due to the multiple use of a given fleet for T_1 and T_2, respectively. Although the comparison analysis is only done for those instances where CPLEX found optimal solution, nevertheless this gives an idea about the importance of the investigation being conducted which is quite significant from both the tactical medium terms and the operational short terms points of views.

Table 4.5: Vehicle utilisation cost comparison of the free fleet VRPB and the MT-VRPB solutions for T_1

Name	*Free Fleet VRPB Solution*		MT-VRPB CPLEX Solution			
	# of Vehicles used	Sol. Cost	Tnb	Optimal Sol.	Extra Cost	# of Vehicles Saved
eil22_50	3	371	1	371	0	2
			2	378	7	1
eil22_66	3	366	1	366	0	2
			2	382	16	1
eil22_80	3	375	1	375	0	2
			2	378	3	1
			3	381	6	0
eil23_50	3	677	1	677	0	2
			2	698	21	1
eil23_66	3	640	1	640	0	2
			2	640	0	1
eil23_80	2	623	1	623	0	1
			2	634	11	0
eil30_50	2	501	1	501	0	1
eil30_66	3	537	1	537	0	2
			2	552	15	1
eil30_80	3	514	1	514	0	2
eil33_50	3	738	1	738	0	2
eil33_66	3	750	1	750	0	2
			2	772	22	1
eil33_80	3	736	1	736	0	2
eil51_50	3	559	1	559	0	2
eil51_66	4	548	1	548	0	3
eil51_80	4	565	1	565	0	3

Free Fleet = Number of vehicles used in the VRPB free fleet solution

Tnb = Total number of given vehicles

Optimal Sol.= Optimal solution found by ILOG CPLEX 12.5

Extra Cost = Extra cost due to overtime/bin packing.

# of Vehicles Saved = The number of vehicle(s) saved due to multiple use of a vehicle.

Table 4.6: Vehicle utilisation cost comparison of the free fleet VRPB and the MT-VRPB solutions for T_2

Name	*Free Fleet VRPB Solution*		MT-VRPB CPLEX Solution			
	# of Vehicles used	Sol. Cost	Tnb	Optimal Sol.	Extra Cost	# of Vehicles Saved
eil22_50	3	371	1	371	0	2
			2	375	4	1
			3	378	7	0
eil22_66	3	366	1	366	0	2
			2	382	16	1
			3	366	0	0
eil22_80	3	375	1	375	0	2
			2	378	3	1
			3	381	6	0
eil23_50	3	677	1	677	0	2
			2	689	12	1
			3	716	39	0
eil23_66	3	640	1	640	0	2
			2	640	0	1
eil23_80	2	623	1	623	0	1
			2	631	8	0
eil30_50	2	501	1	501	0	1
			2	501	0	0
eil30_66	3	537	1	537	0	2
			2	552	15	1
			3	538	1	0
eil30_80	3	514	1	514	0	2
			2	535	21	1
			3	518	4	0
eil33_50	3	738	1	738	0	2
			2	741	3	1
eil33_66	3	750	1	750	0	2
			2	767	17	1
eil33_80	3	736	1	736	0	2
eil51_50	3	559	1	559	0	2
			2	560	1	1
			3	564	5	0
eil51_66	4	548	1	548	0	3
			2	548	0	2
eil51_80	4	565	1	565	0	3
eilA101_66	6	846	1	846	0	5

Free Fleet = Number of vehicles used in the VRPB free fleet solution

Tnb = Total number of given vehicles

Optimal Sol. = Optimal solution found by ILOG CPLEX 12.5
Extra Cost = Extra cost due to overtime/bin packing.

# of Vehicles Saved = The number of vehicle(s) saved due to multiple use of a vehicle.

For further clarity a summary of the results for the two groups of the instances is provided in Table 4.7. For 84 instances of the group T_1, CPLEX found 24 optimal solutions (28%), while 4 instances only were reported as infeasible due to the maximum driving time limit for each vehicle in these instances being too small. For the rest of instances of this group CPLEX found the lower bounds (LB) only.

Table 4.7: Summary CPLEX results and average time for T_1 and T_2

	T_1 (84)	T_2 (84)
# of solutions found (out of 84)	24	37
# of optimal solutions found	24	36
# of incumbent solutions found	0	1
# of instances reported infeasible by CPLEX	4	0
Total average CPU time (s)	5165.91	4248.66
Average CPU time (s) where sol. is found	417	313

For 84 instances of the group T_2, CPLEX found a total of 36 optimal solutions and one incumbent (i.e., feasible solutions for which no overtime is used) solution, while none reported infeasible; and for the rest of the instances in this group CPLEX found only the lower bounds (LB) except one instance where both upper and lower bounds were found.

It was observed in situations where the number of vehicles increases, hence the given time decreases for each vehicle, and CPLEX either could not find a solution or reported infeasibility in few cases.

As for the problem classes, CPLEX performed better on T_2 compared to T_1 since the prior class uses relatively a larger relaxed planning period time for each vehicle, hence better chances of obtaining an optimal or an incumbent feasible solution.

Moreover, to justify and check as to why CPLEX could not find optimal solutions or upper bounds for the majority of the instances in both types (i.e., T_1 and T_2) within 2 hours computational time limit, we ran CPLEX for a longer time (15 hours) on some of those instances where it did not reach either optimal or upper bound levels within 2 hours computational time. For this reason, a small subset of instances containing sizes of 75 and 100 nodes was chosen to run for T_1 and T_2 groups. The comparison of CPLEX runs with different run times (i.e., 2hrs vs 15hrs) results for the two groups of instances is shown separately in Tables 4.8 and 4.9. As it can be seen from the tables the increase in time did not make any difference in terms of optimal solutions or upper bound results for both groups. We believe, the reason behind CPLEX being unable to find the solutions even with extended computational time is due to either the bin(s) given time restriction and/or the instances are too large in size.

In terms of lower bound results, the increase in time made little difference. However, for some instances in both groups, the lower bound is slightly better when CPLEX was run for 15 hours. On the basis of this experiment we decided not to run CPLEX for longer times.

Although CPLEX produced a good number of optimal solutions and upper/lower bounds, it is still a modest success since exact approaches struggle when it comes to larger instances of this kind of hard complex problems. This observation is in line with the literature reviewed in the previous chapters. Nevertheless these results (optimal,

upper/lower bounds) would prove very useful for comparison purposes for our heuristic algorithm approaches in Chapter 5.

Table 4.8: Comparison of CPLEX with 2 hours vs CPLEX with 15 hours for T_1

Name	T_1	Tnb	CPLEX Running for 15 hours			CPLEX Running for 2 hours		
			Sol.	Upper bound	Lower bound	Sol.	Upper bound	Lower bound
eil76_50	775	1	NF	NF	**708.2119**	NF	NF	707.1327
eil76_66	404	2	NF	NF	737.9937	NF	NF	737.9937
eil76_80	821	1	NF	NF	739.7246	NF	NF	739.7246
eilA101_50	290	3	NF	NF	802.2318	NF	NF	802.2318
eilA101_66	223	4	NF	NF	815.4809	NF	NF	815.4809
eilA101_80	902	1	NF	NF	**827.3494**	NF	NF	825.0081

Table: 4.9: Comparison of CPLEX with 2 hours vs CPLEX with 15 hours for T_2

Name	T_2	Tnb	CPLEX Running for 15 hours			CPLEX Running for 2 hours		
			Sol.	Upper bound	Lower bound	Sol.	Upper bound	Lower bound
eil76_50	812	1	NF	NF	**710.0593**	NF	NF	708.0581
eil76_66	423	2	NF	NF	**741.8414**	NF	NF	738.7458
eil76_80	860	1	NF	NF	741.6503	NF	NF	741.6503
eilA101_50	304	3	NF	NF	808.5073	NF	NF	808.5073
eilA101_66	233	4	NF	NF	**825.1924**	NF	NF	824.9404
eilA101_80	945	1	NF	NF	828.6658	NF	NF	828.6938

4.8.1. Relevance of the results

A further analysing of the results provided in Table 4.5 and Table 4.6 is given in Table 4.10 and Table 4.11 for T_1 and T_2 classes of the data instances, respectively. The solutions for most instances (15) of T_1 class appeared consuming no extra time/cost

when using one vehicle for a planning period as against using 2-4 vehicles in free fleet scenarios. For using two vehicles in a planning period there are 8 CPLEX solutions where a small extra cost (11.88 on average) incurs; and for three vehicles instances, CPLEX produced one solution with an extra cost of 6 units only. For the T_2 class, similar results are obtained as shown in Table 4.11

The results provided in the tables show clear advantages for logistics companies and their management decisions. The results demonstrate that a logistics company adopting a multi-trip routing strategy can utilize fully all working hours in a planning period. The results also show that the multi-trip provides clear advantage in reducing fixed costs by reducing the number of vehicles used which can be very much relevant for those companies who depend on hiring a fleet for the distribution and/or reverse logistics reasons. Making deliveries in a given planning period is especially relevant for those companies which are involved in supplying fresh/perishable goods, and in urban area distribution logistics such as online deliveries.

Table 4.10: Comparison of the free fleet VRPB and the MT-VRPB solutions in terms of vehicle savings (for small and medium instances T_1)

T_1 with 1 vehicle used						
Name	***Free Fleet VRPB Solution***		**MT-VRPB CPLEX Solution**			
	# of Vehicles used	**Sol. Cost**	**Tnb**	**Optimal Sol.**	**Extra Cost**	**No. of Vehicles Saved**
eil22_50	3	371	1	371	0	2
eil22_66	3	366	1	366	0	2
eil22_80	3	375	1	375	0	2
eil23_50	3	677	1	677	0	2
eil23_66	3	640	1	640	0	2
eil23_80	2	623	1	623	0	1
eil30_50	2	501	1	501	0	1
eil30_66	3	537	1	537	0	2
eil30_80	3	514	1	514	0	2
eil33_50	3	738	1	738	0	2
eil33_66	3	750	1	750	0	2
eil33_80	3	736	1	736	0	2
eil51_50	3	559	1	559	0	2
eil51_66	4	548	1	548	0	3
eil51_80	4	565	1	565	0	3
Average extra/overtime cost					**0**	
T_1 with 2 vehicle used						
eil22_50	3	371	2	378	7	1
eil22_66	3	366	2	382	16	1
eil22_80	3	375	2	378	3	1
eil23_50	3	677	2	698	21	1
eil23_66	3	640	2	640	0	1
eil23_80	2	623	2	634	11	0
eil30_66	3	537	2	552	15	1
eil33_66	3	750	2	772	22	1
Average extra/overtime cost					**11.88**	
T_1 with 3 vehicle used						
eil22_80	3	375	3	381	6	0
Average extra/overtime cost					**6**	

Table 4.11: Comparison of the free fleet VRPB and the MT-VRPB solutions in terms of vehicle savings (for small and medium instances T_2)

T_2 with 1 vehicle used						
Name	***Free Fleet VRPB Solution***		**MT-VRPB CPLEX Solution**			
	# of Vehicles used	**Sol. Cost**	**Tnb**	**Optimal Sol.**	**Extra Cost**	**No. of Vehicles Saved**
eil22_50	3	371	1	371	0	2
eil22_66	3	366	1	366	0	2
eil22_80	3	375	1	375	0	2
eil23_50	3	677	1	677	0	2
eil23_66	3	640	1	640	0	2
eil23_80	2	623	1	623	0	1
eil30_50	2	501	1	501	0	1
eil30_66	3	537	1	537	0	2
eil30_80	3	514	1	514	0	2
eil33_50	3	738	1	738	0	2
eil33_66	3	750	1	750	0	2
eil33_80	3	736	1	736	0	2
eil51_50	3	559	1	559	0	2
eil51_66	4	548	1	548	0	3
eil51_80	4	565	1	565	0	3
eilA101_66	6	846	1	846	0	5
Average extra/overtime cost					**0**	
T_2 with 2 vehicle used						
eil22_50	3	371	2	375	4	1
eil22_66	3	366	2	382	16	1
eil22_80	3	375	2	378	3	1
eil23_50	3	677	2	689	12	1
eil23_66	3	640	2	640	0	1
eil23_80	2	623	2	631	8	0
eil30_50	2	501	2	501	0	0
eil30_66	3	537	2	552	15	1
eil30_80	3	514	2	535	21	1
eil33_50	3	738	2	741	3	1
eil33_66	3	750	2	767	17	1
eil51_50	3	559	2	560	1	1
eil51_66	4	548	2	548	0	2
Average extra/overtime cost					**7.69**	
T_2 with 3 vehicle used						
eil22_50	3	371	3	378	7	0
eil22_66	3	366	3	366	0	0
eil22_80	3	375	3	381	6	0
eil23_50	3	677	3	716	39	0
eil30_66	3	537	3	538	1	0
eil30_80	3	514	3	518	4	0
eil51_50	3	559	3	564	5	0
Average extra/overtime cost					**8.86**	

4.9. Summary

In this chapter a new variant of the VRP called the Multiple Trip Vehicle Routing Problem with Backhauls (MT-VRPB) is introduced. This is the main focus problem that is being studied in the thesis. The problem is thoroughly described including a graph theoretical definition. A brief review of the exact methodology options for the VRPs is provided followed by an ILP formulation of the MT-VRPB along with its possible variations. An illustrative example showing the validation of the formulation is given along with the details of our CPLEX solution implementation. The chapter also provides details of a newly created large set of MT-VRPB data instances along with the results and analysis. The results show that CPLEX is able to solve small and medium size data instances of the MT-VRPB to optimality and generate upper/lower bounds. Although a good number of optimal solutions and upper/lower bounds are found, the success could not be highlighted more than just modest. However, these results are very important for validation as well as assessing the performance of heuristics results produced in Chapter 5.

The MT-VRPB results show that a large overall cost savings could be obtained by deciding the right fleet size and better vehicle utilizations with multiple trips and backhauling. This can be very vital from the managerial point of view when it comes to making the tactical (acquisition) and fleet management (operational) decisions.

Chapter 5

A Two-Level Variable Neighbourhood Search Algorithm for the Multiple-Trip Vehicle Routing Problem with Backhauls

In this chapter we present a *Two-Level VNS* algorithm developed to solve the MT-VRPB. An overview of the algorithm is first provided followed by the details of various components including a multi-layer local search approach that is embedded with the two level VNS methodology. Details of an adapted *sweep-first-assignment-second* approach to produce an initial solution for the MT-VRPB are also provided. Finally detail of the Bin Packing Problem that resolves the multiple aspect of the MT-VRPB is presented followed by the results and analysis.

5.1. *Two-Level VNS* Algorithm: An Overview

The details of Variable Neighbourhood Search (VNS) approach including its variants and applications are provided in Section 2.3.6.1. Here we present our designed *Two-Level VNS* approach for the MT-VRPB which is motivated by the enhanced features used in the recent paper on VNS by Mladenovic, Todosijevic and Urosevic (2014). In our approach the basic VNS concept is enriched by embedding Sequential Variable

Neighbourhood Descent (SeqVND) along with two shaking steps and a set of neighbourhood schemes to achieve a vigorous diversification during the search process. Moreover, a series of local search routines at two levels of the skeleton of the VNS are used to intensify the search. The merit of the two-level strategy is that it ensures a speedy and continuous balanced intensification and diversification by employing two shaking steps. The details of our VNS algorithm are given in the following sections.

5.1.1. An overview of the algorithm

The algorithm comprises two levels, i.e., outer and inner. We have employed several neighbourhood structures along with associated local search procedures at both levels of the algorithm. For the outer-level we define $N_k^O \ (k = 1, \dots, k_{max})$ as a subset of neighbourhoods (shaking at outer-level) and $LS_k^O \ (k = 1, \dots, k_{max})$ as a subset of local search refinement routines; and at the inner-level $N_l^I \ \ (l = 1, \dots, l_{max})$ as a full set of neighbourhoods (shaking at inner-level) and $LS_l^I \ (l = 1, \dots, l_{max})$ as a full set of local search refinement routines. Note that, "*O*" and "*I"* refer to the neighbourhoods and local search refinement routines used at the outer and the inner levels, respectively. Moreover, a 3-dimentional data structure S_p (detailed description of this data structure is given in Section 5.5) is used to store the initial solution x as well as many other improved solutions during the search process.

At each cycle of the search process, the outer-level of the algorithm generates randomly a transitory solution x' from $N_k^O(x)$. A sub-set LS_k^O of local search refinement routines is then utilised to improve x'. Note that k represents a subset of neighbourhoods and a subset of local search refinement routines used at outer-level. The resulting best solution x'_{best} is then recorded and transferred to the inner level of the algorithm where a

sequential variable neighbourhood descent (SeqVND) is used. At the inner level, both sets of the neighbourhoods and local search refinement routines are utilised and embedded systematically within a multi-layer local search optimiser framework.

Again a transitory solution x'' is generated randomly from $N_l^I(x)$ at the inner-level and transferred to LS_l^I (the multi-layer local search optimiser framework) for improvement. Note that l represents a full set of neighbourhoods and a full set of local search refinement routines used at inner-level. If the solution obtained by the multi-layer local search approach, x''_{best}, is better than the incumbent best solution x'_{best}, then it is updated as $x'_{best} = x''_{best}$ and the process cycles back to the same neighbourhood N_l^I. Moreover, if x''_{best} is found to be the same or worse compared to x'_{best}, then a new x'' is generated using the next neighbourhood $N_{l+1}^I(x'_{best})$ and the multi-layer local search approach is then operated in the same manner. The process continues with the inner-level till $N_{l_{max}}^I$ is reached. At this stage the search process restarts from the outer-level and if x'_{best} is found to be better than the incumbent x then it is updated as $x = x'_{best}$ and the improved solution is stored $S_p = x$; hence, the process of generating a transitional solution restarts from the same neighbourhood N_k^O. But if x'_{best} is found to be the same or worse than the incumbent x, a new transitory x' is generated using the next neighbourhood in $N_{k+1}^O(x)$. Hence, the outer-level is also iterated till $N_{k_{max}}^I$ is reached. The process terminates when the maximum number of iterations $iter_{max}$ is met.

The Bin Packing Problem (BPP) is then solved for a pool of solutions obtained by the *Two-Level VNS*. The BPP starts by sorting the solutions in S_p in the order of lowest to highest cost and initializing a 3-dimentional data structure Sol_k (special data structure

which stores the solutions according to what routes are served by which vehicles – a detailed description of this data structure is provided in Section 5.5). CPLEX optimiser is then called to solve the BPP for each VNS solution in the pool and the packed solutions are stored in the Sol_k data structure. Note that in the cases where a solution could not be packed due to the tight bin capacity then we use the *Bisection Method* (Petch and Salhi, 2004) to increase the bin capacity (i.e., allowing overtime) and the packed solution is reported with overtime. The allocation of overtime is common practice in multiple trip routing and allocation of overtime occurs in a situation where the number of vehicles increases and hence driving time decreases for each vehicle. Therefore, it becomes hard to pack a solution due to tight vehicle's driving time and hence allowing overtime becomes essential. The details of the *Bisection Method* are provided at the end of subsection 5.1.5 in this Chapter. The algorithmic steps of the *Two-level VNS* and BPP are shown in Figure 5.1 with their respective pseudo code presented in Figure 5.2 and Figure 5.3, respectively. The explanation of the main steps will be given next.

Phase I: **Initial solution – *sweep-first-assignment-second* approach**

- Generate LH and BH open-ended routes using *sweep*
- Create a distance matrix of end nodes from open-ended routes
- Solve the assignment problem by calling CPLEX to obtain an initial feasible VRPB free fleet solution x

Phase II: Two-Level VNS **Algorithm**

Initialize the solution pool data structure S_p and add the initial solution x to S_p,

Set: $iter$ = 1 and $iter_{max} = 200$

Repeat the process while $iter \leq iter_{max}$

Start *outer-level*

Let: $LS_k^O =< R_3, R_4, R_5 >$ subset of local search refinement routines for the *outer-level*

Set: $k = 1$

Repeat the process while $k \leq N_{k_{max}}^O$

***a.1*:** Generate a neighbouring solution $x' \in N_k^O(x)$ at random;

***a.2*:** Apply LS_k^O on neighbouring solution x' to improve it

***a.3*:** Assign the resulting solution x' to x'_{best} $[x'_{best} = x']$

a.4*: Start *inner-level using x'_{best}

Let:$LS_l^I =< \{R_1 \& R_6\}, \{R_2 \& R_6\}, \{R_3 \& R_6\}, \{R_4 \& R_6\}, \{R_5 \& R_6\} >$

Multi-Layer local search optimiser framework

Set: $l = 1$

Repeat the process while $l \leq N_{l_{max}}^I$

***a.4*(1):** Generate a neighbouring solution $x'' \in N_l^I(x'_{best})$ at random

***a.4*(2):** Apply LS_l^I on the neighbouring solution x''

***a.4*(3):** Assign the resulting solution x'' to x''_{best} $[x''_{best} = x'']$

***a.4*(4):** If $x''_{best} < x'_{best}$ then $x'_{best} = x''_{best}$; set $l = 1$ and got to ***a.4*(1)**

Else set $l = l + 1$ and got to ***a.4*(1)**

a.5: If $x'_{best} < x$ then $x = x'_{best}$; $S_p = x$; set $k = 1$ and go to ***a.1***

Else set $k = k + 1$ and go to ***a.1***

***Phase III:* Solving the Multiple Trips aspect using the BPP**

Initialize special 3-dimentional data structure Sol_k and let Sol_{max} number of solutions stored in S_p.

Let $iterBM_{max} = 5$.

Set: $iter_{Sol} = 1$

Repeat the process while $iter_{Sol} \leq Sol_{max}$

Step1. Solve the BPP for solution p using CPLEX optimiser ($p = 1, \dots, Sol_{max}$)

Step2. If solution *p* is feasibly packed then go to ***Step4***

Else, go to ***Step3***

Step3. Apply the *Bisection Method* to optimise the bin capacity

Set: $iterBM = 1$

Repeat the process while $iterBM \leq iterBM_{max}$

***Step3*.(1):** Use the *Bisection Method*

***Step3*.(2):** Solve the BPP for solution *p* using CPLEX optimiser

Step4. Store the solution in the special data structure Sol_k according to what routes are served by which bins (vehicles)

Figure 5.1: Algorithmic steps of the *Two-Level VNS* for MT-VRPB

Function *Two-Level VNS* $(x,\ N^O_{k_{max}},\ N^I_{l_{max}},\ iter_{max})$
Let: S_p = be a solution pool data structure
$S_p \leftarrow x$
$iter \leftarrow 1$
while $iter \leq iter_{max}$ do
Let: $LS^O_k =< R_3, R_4, R_5 >$
$k \leftarrow 1$
while $k \leq k_{max}$ do
Select $x' \in N^O_k(x)$ at random; [shake_outer]
$x'_{best} \leftarrow LS^O_k(x')$;
Let: $LS^I_l =< \{R_1 \& R_6\}, \{R_2 \& R_6\}, \{R_3 \& R_6\}, \{R_4 \& R_6\}, \{R_5 \& R_6\} >$
$l \leftarrow 1$
while $l \leq l_{max}$ do
Select $x'' \in N^I_l(x'_{best})$ at random; [shake_inner]
$x''_{best} \leftarrow LS^I_l(x'')$; [Multi-Layer local search framework]
If $f(x''_{best}) < f(x'_{best})$ then
$x'_{best} \leftarrow x''_{best}$; $l \leftarrow 1$;
Else $l \leftarrow l + 1$;
end while
return x'_{best};
If $f(x'_{best}) < f(x)$ then
$x \leftarrow x'_{best}$; $S_p \leftarrow x$; $k \leftarrow 1$;
Else $k \leftarrow k + 1$;
end while
return x;
end while

Figure 5.2: Pseudo code for the *Two-Level VNS*

Function *Bin-Packing* (S_p, $iter_{Sol}$, Sol_{max})

Let: Sol_k = be a 3-D data structure to store the packed solutions

Set: S_p = be a pool of all sorted solution in descending order of cost ($p = 1, \dots, Sol_{max}$)

$iter_{Sol} \leftarrow 1$

while $iter_{Sol} \leq Sol_{max}$ do

a. Select $p \in S_p$ (where $p = 1, \dots, Sol_{max}$)

b. $CPLEX\ (p)$ If feasibly packed, go to ***d***, Else go to ***c***

c. $iterBM \leftarrow 1$

while $iterBM \leq iterBM_{max}$ do

c1. Use the *Bisection Method* (see Section 5.5)

c2. $CPLEX\ (p)$

end while

d. $Sol_k \leftarrow CPLEX\ (p)$

end while

Figure 5.3: Pseudo code for the BPP

5.2. Initial solution (*Phase I*)

The Sweep procedure of Gillett and Miller (1974) is considered to be a simple and an efficient construction method for the VRPs. A *sweep-first-assignment-second* approach is developed to generate the initial solution for the MT-VRPB. The way the sweep method works is that it starts with clustering customers into feasible groups in such a way that those customers who belong to the same group are close to each other geographically and centred to the depot to be served by the same vehicle. We have used the Sweep method in such a way that two sets of open-ended routes are constructed by sweeping through linehaul (LH) and backhaul (BH) nodes separately. To the best of our knowledge, no one has used sweep method in this format before this study for any backhauling version of the VRP. Figure 5.4 shows an illustrative example of the way we have used the Sweep procedure to generate open-ended LH and BH routes.

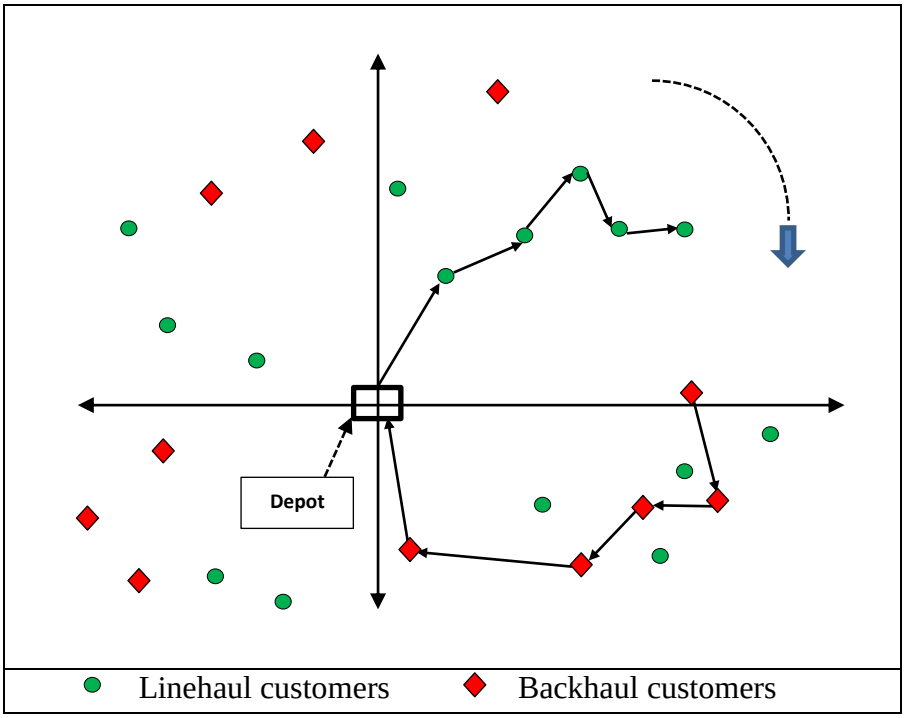

Figure 5.4: An illustrative example of sweep procedure for the MT-VRPB

In Figure 5.5 an illustrative example of the problem instance *eil21_50* is shown, demonstrating the visual features of the open-ended routes. In the cases where the number of open-ended BH routes is less than the number of open-ended LH routes, dummy BH open ended route(s) containing the depot only is created and added to the matrix. This is done to obtain the same number of routes for the purpose of optimal matching. Note that if the solution is not feasible in terms of the precedence backhauling constraints (i.e., all delivery customers are served before any pickup customers; vehicle routes containing only backhaul customers are not allowed) then it can be amended by moving customers among routes before passing it on to the VNS stage of the algorithm. However this situation did not arise in solving the data sets in this thesis.

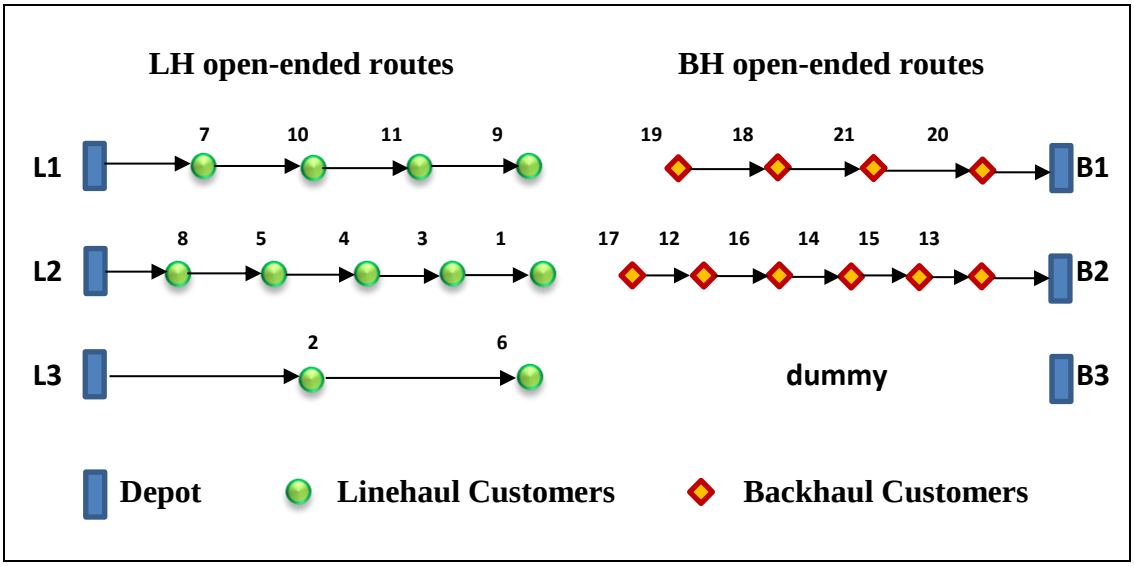

Figure 5.5: LH and BH open-ended routes (Problem instance *eil22_50* of *data set-2*)

Once linehaul and backhaul open-ended routes are constructed, the assignment problem is solved. But before we discuss how it has been solved, it would be useful to understand how CPLEX optimiser can be embedded in different programming languages on different platforms. Since, we have solved an assignment problem and Bin Packing Problem (BPP) described in section 5.1.5 of this Chapter by calling the CPLEX optimiser within C++ programming language in Microsoft Visual Studio environment, a brief description of the procedure is presented in Appendix A.

5.2.1. Solving the Assignment Problem

The following assignment formulation is modelled and implemented in C++ programming language within Microsoft Visual Studio Environment. The designed model calls CPLEX optimiser within Visual Studio Environment to find the optimal matching of both types of routes in order to create a set of routes (i.e., routes with both linehaul and backhaul customers).

In the following assignment formulation nr denotes the number of open LH and BH routes. X_{ij} is a binary decision variable defining whether the i^{th} open linehaul route is connected with the j^{th} open backhaul route.

Minimise Z $\quad \sum_{i \in nr} \sum_{j \in nr} D_{ij} \; X_{ij}$ (5.1)

Subject to $\quad \sum_{i=1}^{nr} X_{ij} = 1 \qquad \forall \, (j = 1, \dots, nr)$ (5.2)

$\sum_{j=1}^{nr} X_{ij} = 1 \qquad \forall \, (i = 1, \dots, nr)$ (5.3)

$X_{ij} \in \{0,1\} \qquad \forall \, (i, j = 1, \dots, nr)$ (5.4)

Where

$$X_{ij} = \begin{cases} 1, & \text{if } i^{th} \text{ open linehual route is connected to } j^{th} \text{ open backhaul route;} \\ 0, & \text{otherwise} \end{cases}$$

A distance/cost matrix D_{ij} that consists of distances between the end points of i^{th} open linehaul routes to the end point of the j^{th} open backhaul routes is then created in order to solve the assignment problem. A dummy route containing the depot is added to the matrix where a number of LH and BH routes are not equal.

An illustrative example:

A matrix containing the actual distances is shown in Figure 5.6.

$$D_{ij} = \begin{array}{c} \\ \text{L1} \\ \text{L2} \\ \text{L3} \end{array} \begin{array}{c} \begin{array}{ccc} \text{B1} & \text{B2} & \text{B3} \end{array} \\ \begin{pmatrix} 17 & 69 & 22 \\ 72 & 9 & 49 \\ 70 & 30 & 42 \end{pmatrix} \end{array}$$

Figure 5.6: Distance matrix of end nodes

To produce combined LH-BH routes, the optimal matching is then obtained by solving an assignment problem using ILOG CPLEX 12.5 optimiser coded with C++ within Microsoft Visual Studio Environment. The optimal assignment matching result for the example problem is illustrated in Figures 5.7 and 5.8.

Optimal assignment with an objective value = 68

$$X_{ij} = \begin{array}{c} \\ \text{L1} \\ \text{L2} \\ \text{L3} \end{array} \begin{array}{c} \begin{array}{ccc} \text{B1} & \text{B2} & \text{B3} \end{array} \\ \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \end{array}$$

Linehaul Route 1 : matches with backhaul route 1 ($X_{00} = 1$)

Linehaul Route 2 : matches with backhaul route 2 ($X_{11} = 1$)

Linehaul Route 3 : matches with backhaul route 3 ($X_{22} = 1$)

Figure 5.7: Optimal matching obtained by CPLEX

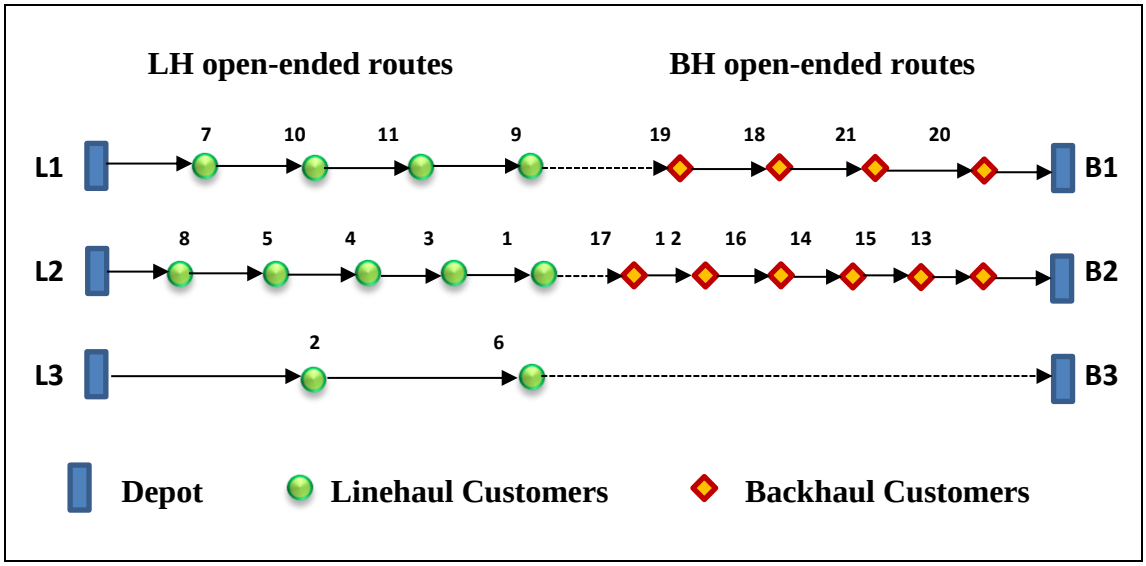

Figure 5.8: Combined LH+BH routes (problem instance no: eil22_50)

5.3. Neighbourhoods used in the *Two-Level VNS* Algorithm (*Phase II*)

The neighbourhood generation is a fundamental part in heuristic search design in general and in the VRPs in particular. In this study six neighbourhoods are used. We first briefly describe these neighbourhoods along with their illustrations provided in corresponding figures, and then provide an explanation as to how we used them in our algorithm.

1-insertion (intra-route): relocates the position of a customer at a non-adjacent arc within the same route as shown in Figure 5.9. The upper part of the figure shows all the positions for linehaul node 1 and backhaul node 4 can possibly be re-located within the same route. The lower parts of the figure demonstrate a 1-insertion move where a linehaul customer (node 2) is removed from its position in route r_1 and inserted at a non-adjacent position in the same route, resulting in a savings in travelling cost. Note that 1-insertion routine moves both linehaul and backhaul customers and it has been implemented in such a way that; if a linehaul customer is selected then it can only be inserted in any non-adjacent linehaul arc and the same is the case with backhaul customers. This is done due to the backhaul precedence constraints (see Subsection 4.1.1) that all delivery customers must be served before any pickups.

1-insertion (inter-route): relocates a customer from one route to another. As shown in Figure 5.10, a linehaul node is removed from route r_2 and inserted in route r_1 to achieve a travelling cost reduction.

1-1 swap: swaps two customers each taken from two separate routes. As shown in Figure 5.11, two linehaul nodes are swapped between route r_1 and r_2 to obtain a savings in travelling cost.

2-2 Swap: swaps two pairs of consecutive customers taken from two separate routes. Figure 5.12 shows two pairs (consecutive) of linehaul/backhaul nodes are swapped between routes r_1 and r_2 to obtain a savings in travelling cost.

2-0 shift: re-locates two consecutive customers from one route to another. Figure 5.13 shows a consecutive pair of backhaul nodes is shifted from route r_1 to route r_2 to gain a reduction in travelling cost.

2-1 swap: swaps a consecutive pair from one route with a single customer from another route. As shown in Figure 5.14, a pair (consecutive) of linehaul nodes from route r_1 is swapped with a linehaul node from route r_2 to obtain a reduction in travelling cost

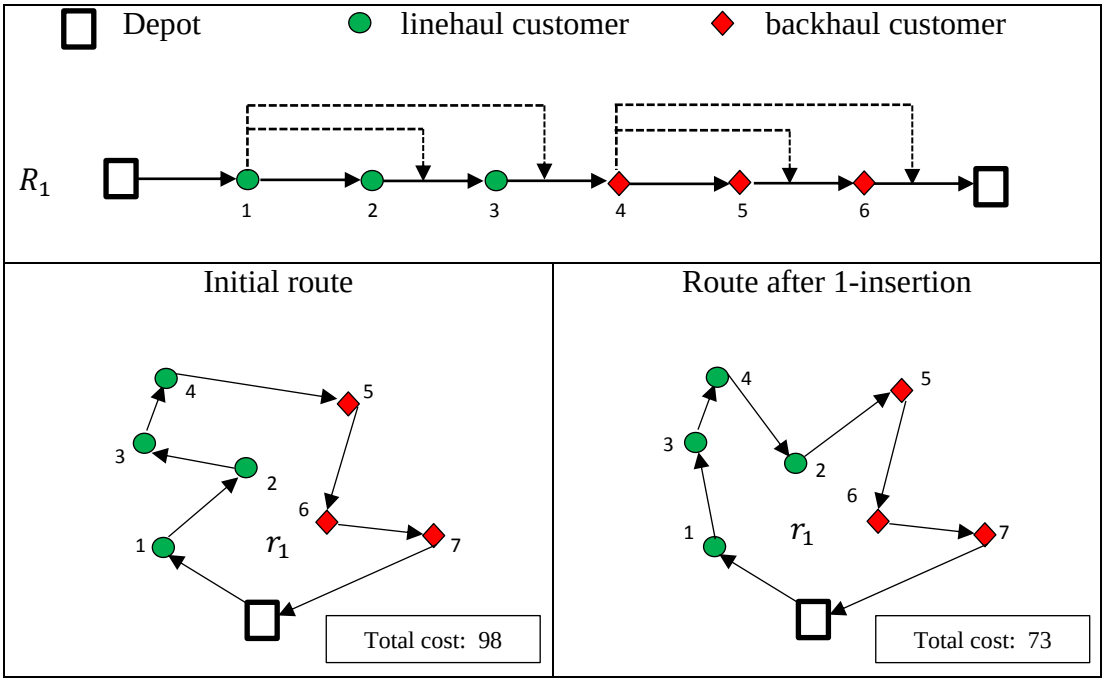

Figure 5.9: An illustrative example of the 1-insertion (intra-route) refinement routine

.

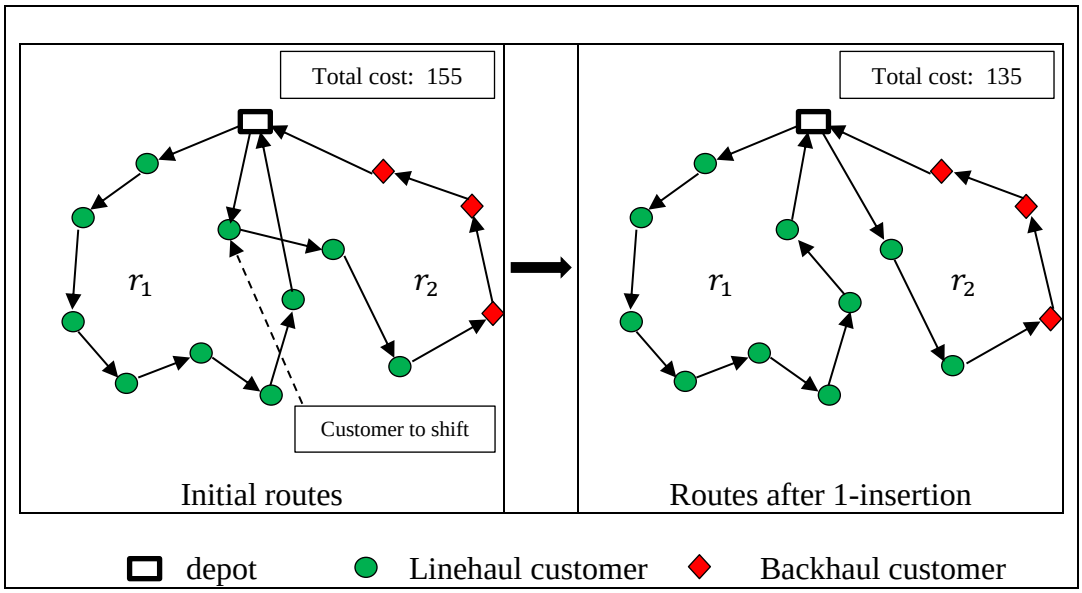

Figure 5.10: An illustrative example of the 1-insertion (inter-route) refinement routine

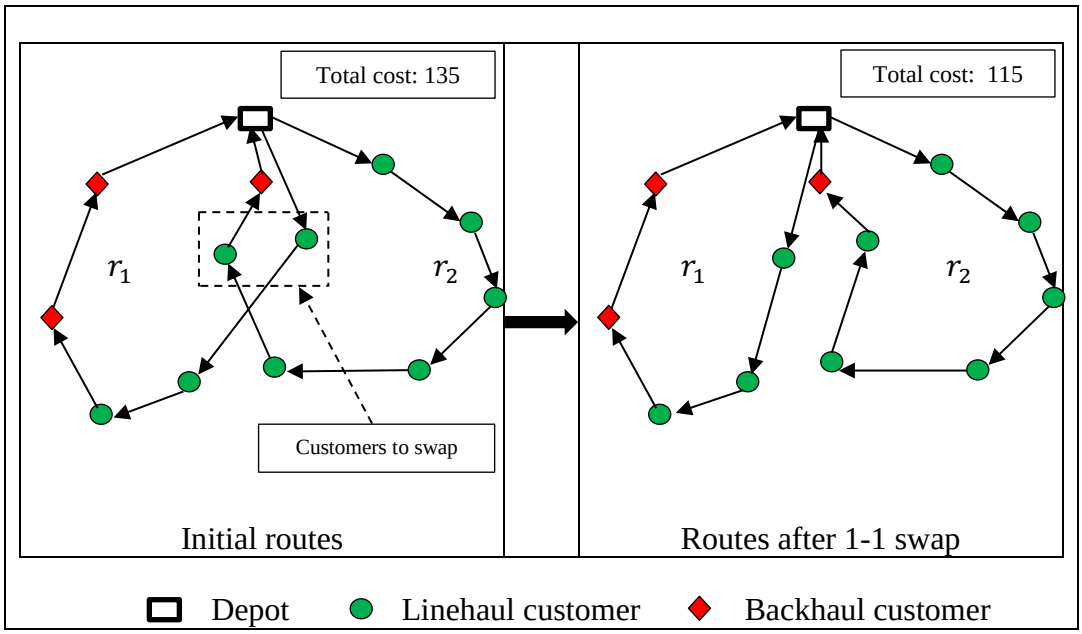

Figure 5.11: An illustrative example of the 1-1 swap refinement routine

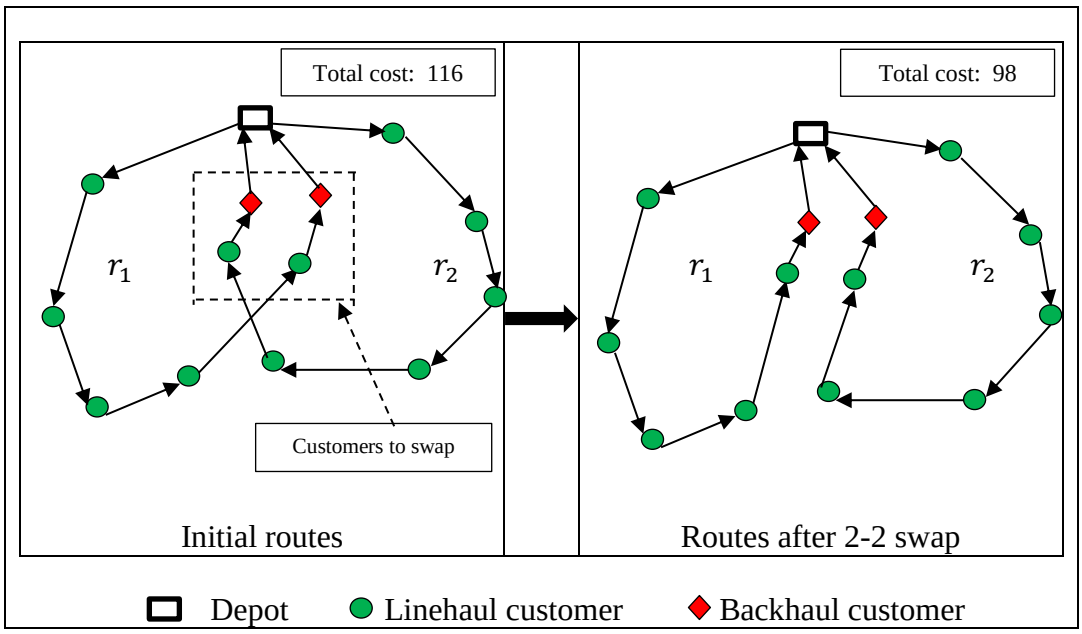

Figure 5.12: An illustrative example of the 2-2 swap refinement routine

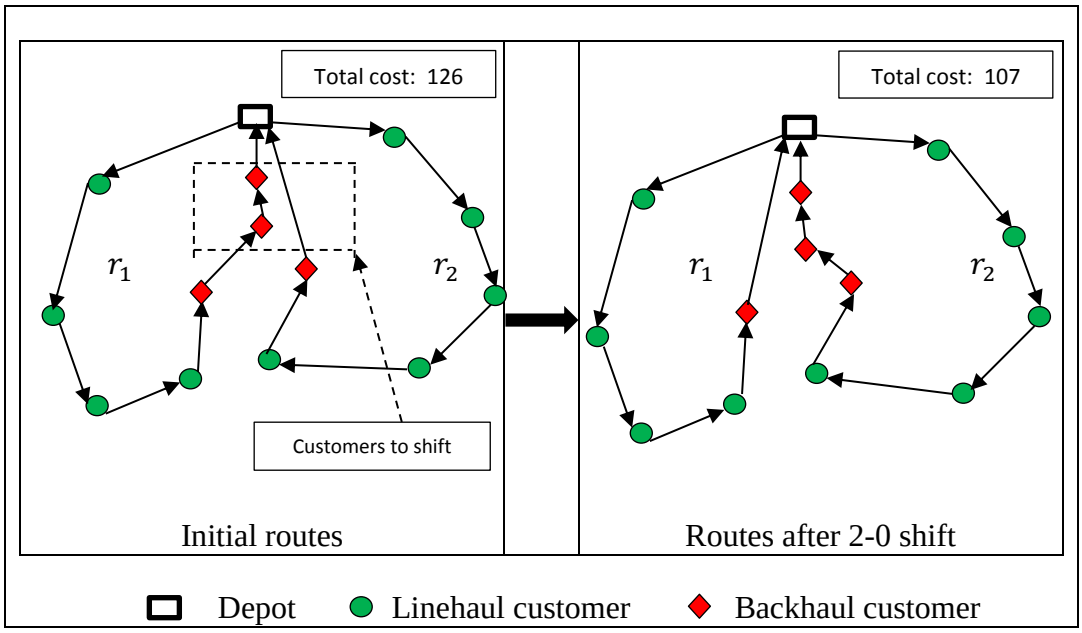

Figure 5.13: An illustrative example of the 2-0 shift refinement routine

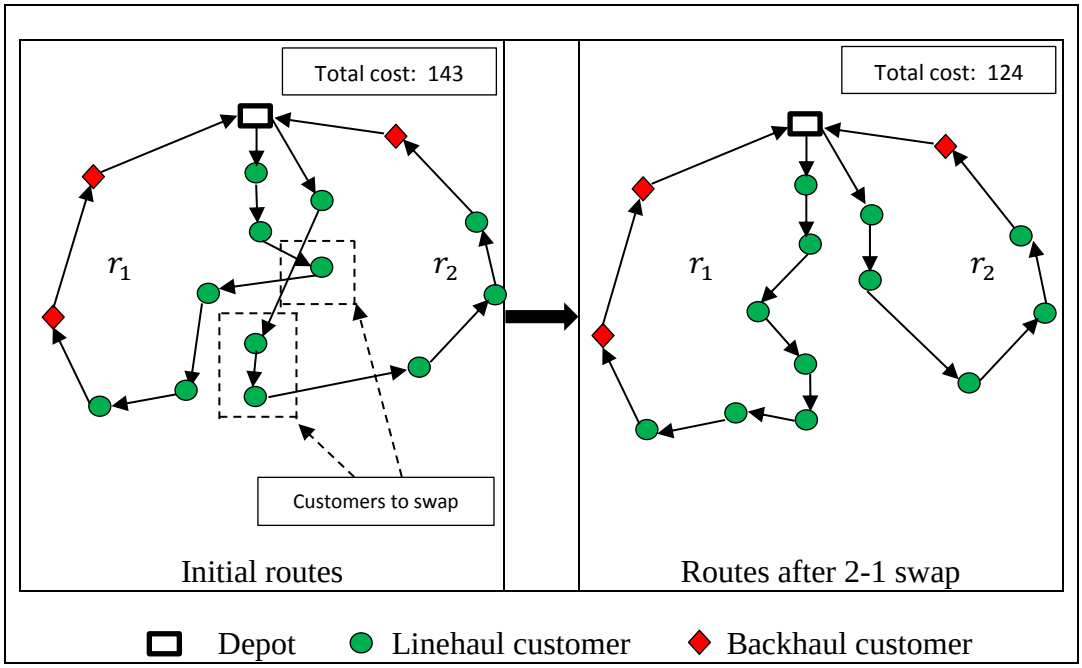

Figure 5.14: An illustrative example of the 2-1 swap refinement routine

Use of neighbourhoods in the Two-Level VNS algorithm:

The moves in all the above neighbourhoods are conducted according to the backhauling constraint conventions described in Section 3.1 of Chapter 3. These neighbourhood schemes are used at the shaking and local search stages of the *Two-Level VNS* algorithm.

Shaking Stage:

All six neighbourhoods are used in the shaking stage of the algorithm in the following order that was found empirically. 1-insertion intra-route N_1, inter-route N_2, 1-1 swap N_3, 2-2 swap N_4, 2-0 shift N_5, 2-1 swap N_6. In the VNS literature, the neighbourhood moves are used in various ways, i.e. systematically, partial systematic manner, complete random manner, etc. In our case, all the neighbourhood moves, i.e., customers re-locate positions and routes are selected randomly. Hence, only feasible moves (in terms of

problem constraints, i.e., vehicle capacity, backhauling precedence conventions set explained in Section 3.1) are accepted in the search process.

5.4. Multi-layer local search optimiser framework (local search stage)

The details of how this group of six neighbourhoods are used as local search refinement routines in the *Two-Level VNS* algorithm are provided here.

Our multi-layer local search optimiser framework can be categorised as a composite heuristic. The multi-layer local search framework uses all six neighbourhood schemes, presented in Subsection 5.1.3, in the form of local search refinement routines. The notion of manipulating the power of several neighbourhood structures as local searches within a local search framework was originally developed by Salhi and Sari (1997) known as multi-level composite heuristic and successfully implemented in Imran, Salhi and Wassan (2009). We have adapted this idea into our *Two-Level VNS* algorithm. The order in which these refinement routines (denoted with R_i, i=1,…,6) are executed is important. The following order is chosen empirically: 1-insertion (inter-route) R_1, 1-1 swap R_2, 2-2 swap R_3, 2-0 shift R_4, 2-1 swap R_5; and 1-insertion (intra-route) R_6 . The last routine R_6 is used as a post-optimiser after each local search refinement routine is executed in the framework.

The multi-layer framework search process starts with a transitory feasible solution x'as explained in Subsection 5.1.1. Each local search routine is then executed in order till a local optimum is reached followed by the post-optimiser routine R_6, i.e., 1-insertion (intra-route). Note that the post-optimiser is used only if the preceding routine in the list improves the solution. The framework of our multi-layer local search optimiser is provided in a flow chart shown in Figure 5.15. Note that this multi-layer local search

framework is similar to VND except here there are several local search refinement routines used instead of 2 or 3 local searches.

Acceptance criteria:

In the literature, these refinement routines are implemented in two solution acceptance criterion strategies, i.e., the *first-improvement* and the *best-improvement*. In the *first-improvement* strategy, the change in the solution is accepted and updated any time during the search process if it improves the current best incumbent solution. In the *best-improvement* strategy, the best of all possible improvements is accepted at the end of the search cycle. We conducted experiments with both strategies in our initial trials and found the *first-improvement* producing better results for the MT-VRPB while being relatively faster. Note that in the original implementation of the multi-level heuristic (Salhi and Sari, 1997), the *best-improvement* strategy is used instead.

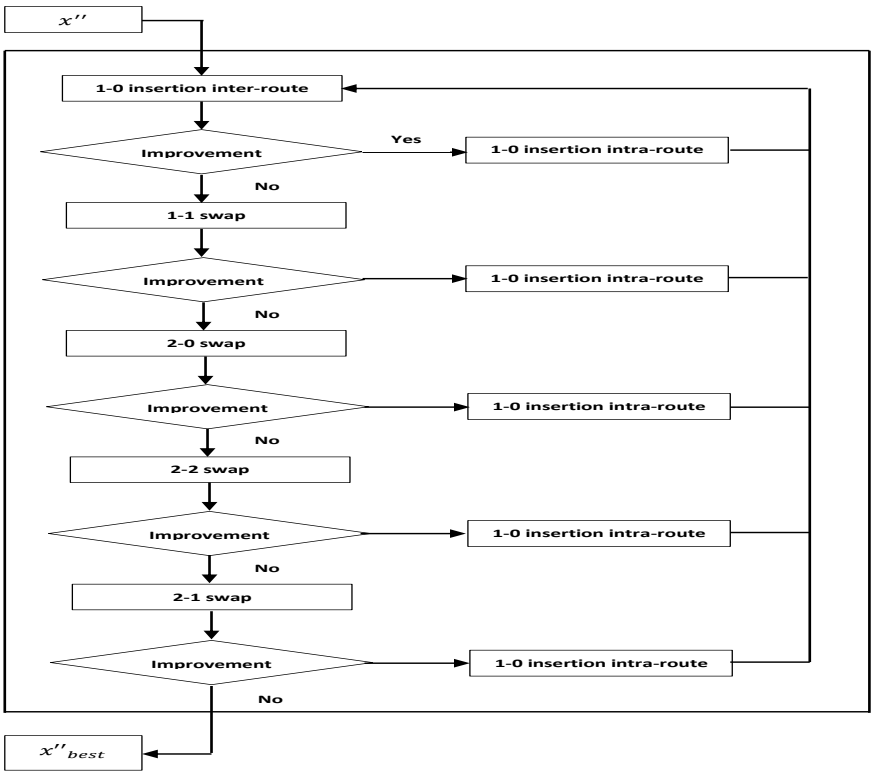

Figure 5.15: The multi-layer local search optimiser framework flow chart

5.5. Solving the Bin Packing Problem (*Phase III*)

In the Bin Packing Problem (BPP) items of different sizes/volumes are to be packed into a finite number of bins/containers with a known capacity *c* such that all items are packed into the minimum number of bins without violating the capacity of each bin. For early classical applications of the bin packing see study of Eilon and Christofides (1971) for vehicle/container loading problem. For a review on a variety of knapsack problems see Wilbaut, Hanafi and Salhi (2008). For recent studies we refer to the studies of Lewis, Song, Dowsland and Thompson (2011) and Song, Lewis, Thompson and Wu (2012). We have solved the following BPP model for the MT-VRPB.

Given k bins (vehicles) $(v_1, \dots, v_k)$ of the same size c (time) and n items (routes) with varying weights $(w_1, \dots, w_n)$.

Minimise $\sum_{i=1}^{k} y_i$ (5.5)

Subject to $\sum_{j=1}^{n} w_j x_{ij} \leq c y_i \qquad i = 1, \dots, k$ (5.6)

$\sum_{i=1}^{k} x_{ij} = 1 \qquad j = 1, \dots, n$ (5.7)

$y_i = 0 \; or \; 1, \qquad i = 1, \dots, k$ (5.8)

$x_{ij} = 0 \; or \; 1, \qquad i = 1, \dots, k, j = 1, \dots, n$ (5.9)

Where

$$y_i = \begin{cases} 1, & \text{if bin } i \text{ is used;} \\ 0, & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1, & \text{if item } j \text{ is assigned to bin } i; \\ 0, & \text{otherwise} \end{cases}$$

Constraint 5.6 ensures the capacity *c* is not violated for each of the bins; whereas, Constraint 5.7 guaranties that each item (route) is assigned to at most one bin.

To solve the above BPP, a pool is created containing different solutions produced by the *Two-Level VNS* algorithm having completed its given number of iterations for each instance. The solutions (candidate list of the new improved solutions appeared during the search process at *Phase II* for an instance) in the pool are stored in a 3-dimensional data structure S_p before solving the BPP. An illustrative example of this data structure is shown in Figure 5.16.

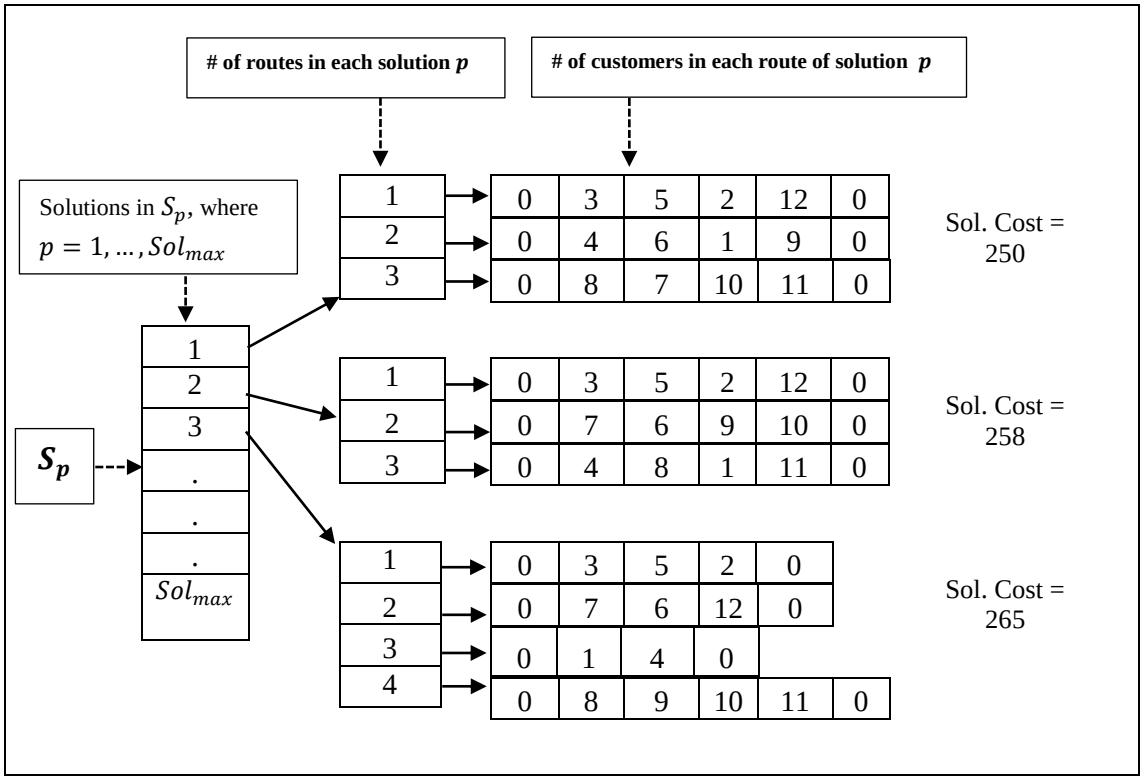

Figure 5.16: An illustrative example of data structure S_p

The BPP model is designed and coded in C++ programming language in Microsoft Visual Studio Environment that calls for the CPLEX optimiser. The BPP model starts with sorting the solutions stored in the 3D data structure S_p in the order of lowest to

highest cost. The process of the packing starts by choosing the lowest cost solution from the ordered solutions pool and solving the BPP by calling CPLEX optimiser. If the chosen solution is feasibly packed (without allowing overtime to any of the bins) into a given number of the bins then it is stored in a separate special 3-dimentional data structure Sol_k (note that data structure Sol_k stores solution according to what routes are packed in what bin/s) as one of the possible solution results for an instance. An illustrative example of special 3-dimentional data structure Sol_k is shown in Figure 5.17. The process is repeated for all the solutions in the pool. In the case where the feasible packing could not be achieved for a solution, we use a repair mechanism known as the *Bisection Method* (Petch and Salhi, 2004) which allows overtime progressively to the given bin(s) of those instances to pack the routes. Figure 5.18 presents a flow chart showing the BPP solution procedure.

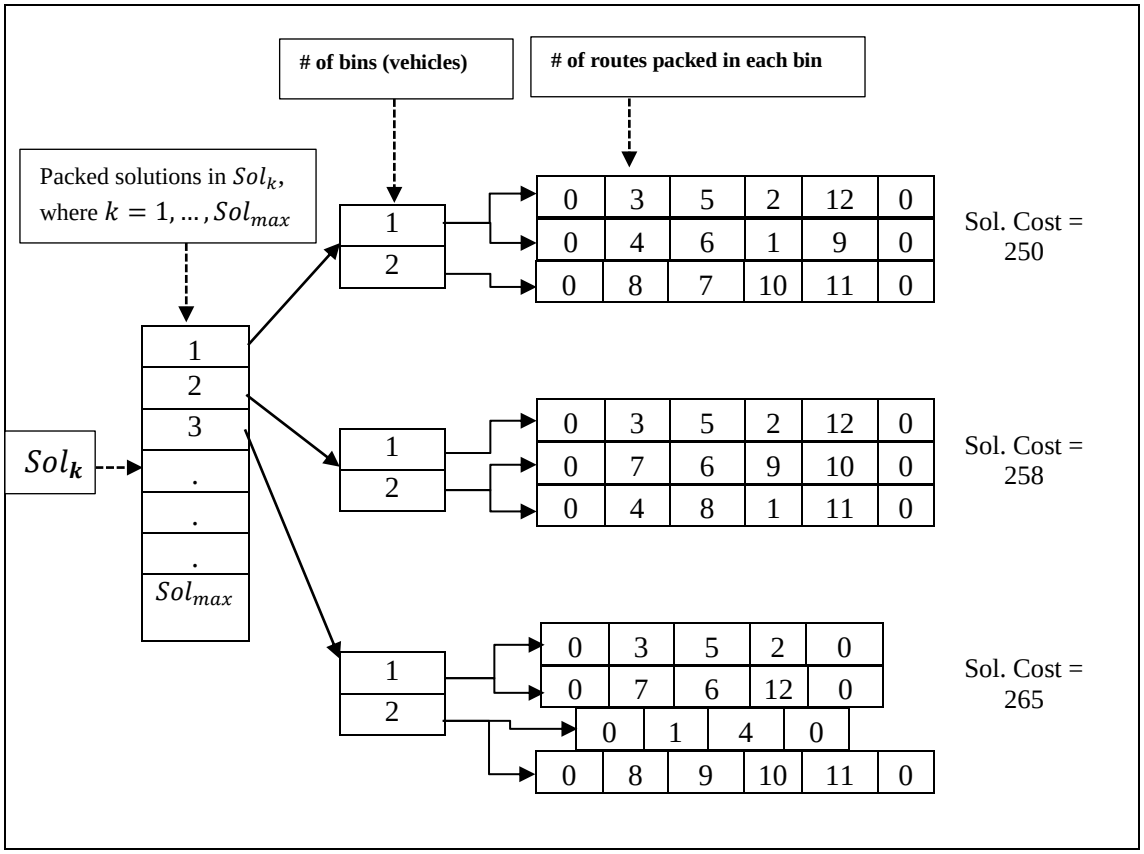

Figure 5.17: An illustrative example of special data structure Sol_k

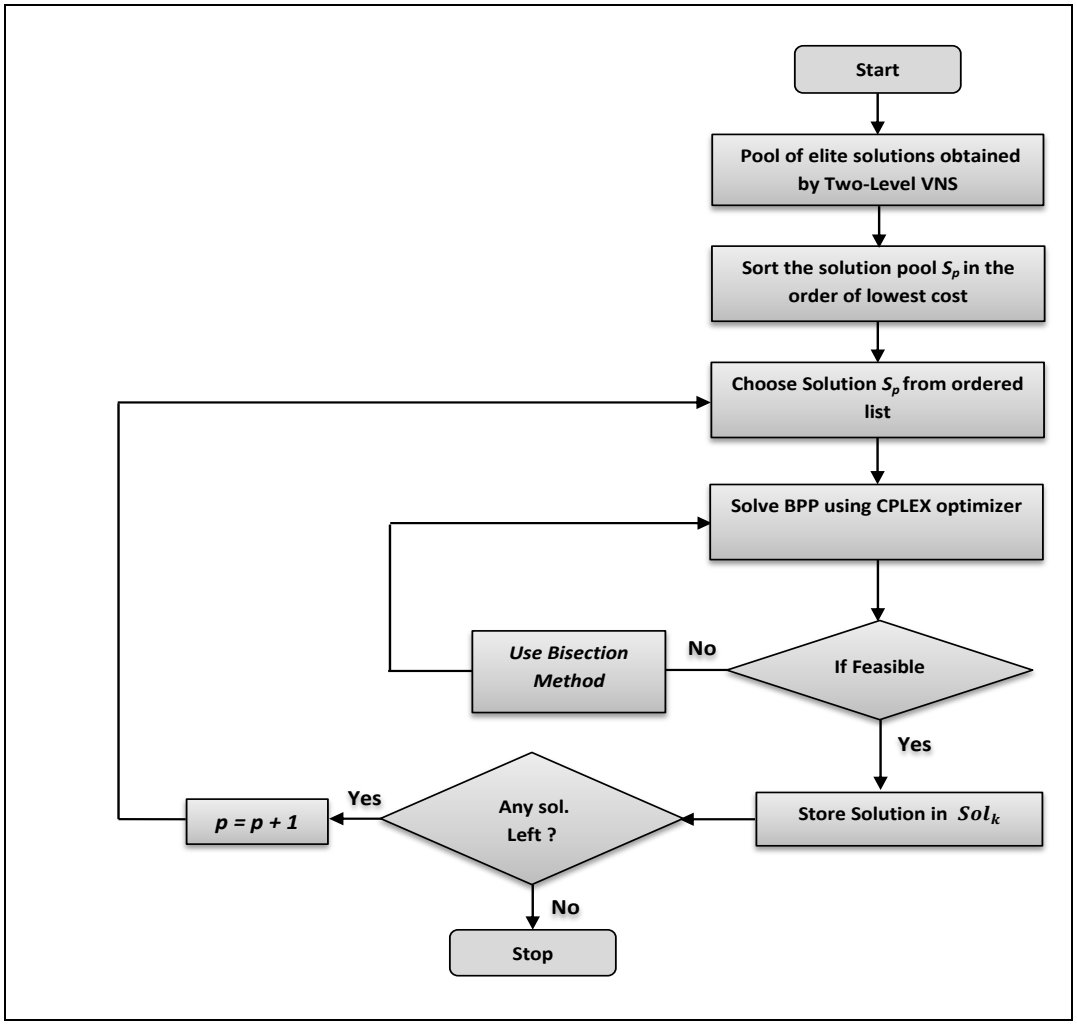

Figure 5.18: BPP flow chart

***Bisection Method (Repair Mechanism)*:** the *Bisection method* works in such a way that it starts increasing the bin(s) capacity by a certain percentage iteratively until routes are packed into bin(s). For instance, as it is shown in Figure 5.19, the bin(s) capacity is increased by 5% at every iteration; and suppose the required capacity of bins is achieved at 25% level increase; it then tries to optimise the bin capacity by using a decreasing/increasing percentage mechanism, say starting from a decreasing percentage of -2.5% which is then decreased/increased by half iteratively (i.e., 1.25, 0.625,…, and so on) till the bin(s) overtime is optimised. In our case the bin capacity increasing percentage is fixed at 5% and the decreasing percentage starts from 2.5%. This is done

because the optimal packing achieved by the BPP say with +25% bin overtime increase might be a bit higher than required. Note that in this study, the *Bisection Method* is used with a fixed number of iterations, $iterBM_{max} = 5$ which was found appropriate for all the instances. Figure 5.19 presents an illustrative example of the *Bisection Method*.

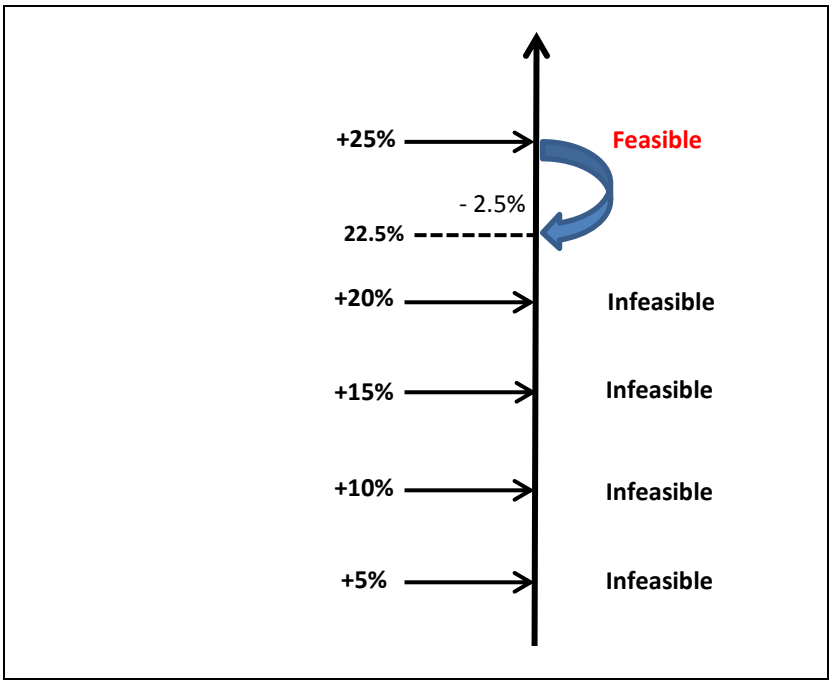

Figure 5.19: An illustrative example of the *Bisection Method*

5.6. Computational Experience

5.6.1. Introduction and Computer Details

The *Two-Level VNS* algorithm including the initial solution design and the BPP model is implemented in C++ programming within the Microsoft Visual Studio Environment. The experiments were executed on a PC with Intel(R) Core(TM) i7-2600 processor, CPU speed 3.40 GHz. The IBM ILOG CPLEX 12.5 optimisers are used to solve the Assignment and BPP problems for the MT-VRPB.

Initial Solution: The *sweep-first-assignment-second* approach is implemented by calling CPLEX optimiser within the Visual Studio Environment to find the optimal matching of LH-BH routes.

Packing route into Bins: The Bin Packing Problem approach is also implemented that calls the CPLEX optimiser within the Visual Studio Environment in order to obtain the optimal packing of routes within Bin(s).

Glossary for tables:

T_1 = Total driving time (type one) for a bin/ vehicle.

T_2 = Total driving time (type two) for a bin/ vehicle.

Tnb = Total number of vehicles (bins) in each instance.

No.R = Number of total routes in solution.

No. of Routes in each Bin = Number of routes served by each bin/vehicle.

X = Infeasible.

- = Not found.

^ = Incumbent solution.

Opt. Sol. = optimal solution found by CPLEX.

Overtime = Overtime (equivalent to per unit distance travelled by a vehicle) allocated to bin(s) where needed to feasibly pack routes within bin(s).

Cost with overtime = Total solution cost including Overtime units.

Time(s) = CPU time in seconds taken to solve each instance.

n = Total number of customers.

RPD = Relative Percentage Deviation = [(VNS Sol. - best known)/ best known * 100].

5.6.2. Results and analysis

Our *sweep-first-assignment-second* approach is very fast in producing initial feasible VRPB solutions, spending less than a second on average.

It is to be noted that the MT-VRPB is being introduced in this thesis hence there are no previously developed benchmarks instances results to compare with. Therefore the performance of the algorithm is compared against the optimal solutions and lower/upper bounds produced by CPLEX. Table 5.1 provides a summary of the *Two-Level VNS* as compared to CPLEX results found in Chapter 4.

The *Two-Level VNS* solved all 168 instances of T_1 and T_2 groups as compared to 61 of CPLEX. It also matched 51 optimal/incumbent solutions out of 61 of CPLEX. The *Two-Level VNS* also proved very efficient in using bin overtime of only 5 to 10 units on average. In terms of speed it used less than 20 seconds on average per instance.

The *Two-Level VNS* found 46 feasible (without overtime) solutions and 38 infeasible (with overtime) for the instances in T_1. The algorithm performed better for T_2 type instances, where 59 feasible solutions are found leaving 25 infeasible solutions in this group.

Table 5.2 and Table 5.3 report the detailed solutions of the *Two-Level VNS* algorithm along with the CPLEX results for the data *set-1* ($T_1$and T_2). The algorithm is run for 200 iterations and, due to the random element, best solution is reported out of 5 runs. For

T_1 the algorithm found a number of good quality (no overtime used) solutions (46 out of 84) and for the remaining 38 it took less than 30 units of overtime in most cases. For T_2, 59 solutions are found without overtime and the rest (apart from a few) the algorithm did not exceed 30 units of overtime.

It can be observed (see Table 5.2 and Table 5.3) that good quality solutions are found when the bin capacity is relatively large and the number of bins is smaller. It can also be seen that the increase in the number of bins increases the likelihood of overtime being used. In summary, the algorithm is able to solve all the instances including 51 optimal solutions at a very low computational cost requiring on average 18 seconds per instance.

Table 5.1: The comparison of the *Two-Level VNS* with CPLEX (data *set-1*: T_1 & T_2)

	T_1		T_2	
	CPLEX	***Two-Level VNS***	**CPLEX**	***Two-Level VNS***
# of solutions found (out of 84)	24	84	37	84
# of feasible solutions found (out of 84)	24	46	37	59
# of optimal solutions found	24	21	36	30
Max overtime (units)	-	58	-	52
Min overtime (units)	-	2	-	1
Average overtime (units)	-	10.24	-	5.33
Average CPU time (s)	5165	18	4248	17

Table 5.2: Detailed comparison of the *Two-Level VNS* with CPLEX for the data *set-1* (T_1)

Name	T_1	Tnb	CPLEX			*Two-Level VNS*				
			Opt. Sol.	No. R.	Time (s)	Actual Cost	Over Time	Cost with overtime	No. R.	Time (s)
eil22_50	390	1	371	3	1	371	0	**371**	3	**2**
	195	2	378	3	1	378	0	**378**	3	**3**
	130	3	x	x	x	380	10	390	4	3
eil22_66	385	1	366	3	1	366	0	**366**	3	5
	193	2	382	4	3	386	10	396	4	4
	129	3	x	x	x	366	4	370	3	3
eil22_80	394	1	375	3	2	375	0	**375**	3	4
	197	2	378	4	2	378	0	**378**	4	5
	132	3	381	3	27	381	0	**381**	3	3
eil23_50	711	1	677	3	1	677	0	**677**	3	3
	355	2	698	3	2	677	34	711	3	2
	237	3	x	x	x	712	13	725	3	5
eil23_66	672	1	640	3	1	640	0	**640**	3	4
	336	2	640	3	1	640	0	**640**	3	4
	224	3	x	x	x	655	47	702	3	3
eil23_80	654	1	623	2	1	623	0	**623**	2	4
	327	2	634	2	2	634	0	**634**	2	4
eil30_50	526	1	501	2	1	501	0	**501**	2	4
	264	2	x	x	x	501	6	507	2	3
eil30_66	564	1	537	3	3	537	0	**537**	3	6
	282	2	552	3	6116	544	21	565	3	6
	188	3	-	-	7200	539	2	541	3	5
eil30_80	540	1	514	3	12	514	0	**514**	3	6
	270	2	-	-	7200	517	23	540	3	7
	180	3	-	-	7200	518	0	518	3	6
eil33_50	775	1	738	3	1	738	0	**738**	3	5
	388	2	-	-	7200	738	28	766	3	6
	258	3	-	-	7200	764	58	822	3	4
eil33_66	788	1	750	3	2	750	0	**750**	3	9
	394	2	772	3	1219	772	0	**772**	3	8
	263	3	-	-	7200	752	40	792	3	5
eil33_80	773	1	736	3	121	736	0	**736**	3	6
	387	2	-	-	7200	756	0	756	3	9
	258	3	-	-	7200	736	30	766	3	5
eil51_50	587	1	559	3	10	559	0	**559**	3	9
	294	2	-	-	7200	568	0	568	3	11
	196	3	-	-	7200	568	6	574	3	10
eil51_66	576	1	548	4	22	548	0	**548**	4	10
	288	2	-	-	7200	552	0	552	4	11
	192	3	-	-	7200	552	25	577	4	11
	144	4	-	-	7200	563	20	583	4	10
eil51_80	594	1	565	4	4553	565	0	**565**	4	13
	297	2	-	-	7200	565	0	565	4	12
	198	3	-	-	7200	582	0	582	5	11
	149	4	-	-	7200	581	11	592	5	11
eilA76_50	775	1	-	-	7200	738	0	738	6	21
	388	2	-	-	7200	738	0	738	6	23
	259	3	-	-	7200	741	0	741	6	22
	194	4	-	-	7202	738	49	787	6	23
	155	5	-	-	7200	747	36	783	6	22

Name	T_1	Tnb	CPLEX			*Two-Level VNS*				
			Opt. Sol.	No. R.	Time (s)	Actual Cost	Over Time	Cost with overtime	No. R.	Time (s)
	130	6	-	-	7200	748	31	779	6	22
eilA76_66	807	1	-	-	7200	768	0	768	7	23
	404	2	-	-	7200	768	0	768	7	21
	269	3	-	-	7200	772	0	772	7	23
	202	4	-	-	7200	784	0	784	8	21
	162	5	-	-	7200	781	36	817	8	23
	135	6	-	-	7200	783	5	788	8	23
	116	7	-	-	7200	771	22	793	8	22
eilA76_80	821	1	-	-	7200	781	0	781	8	23
	411	2	-	-	7200	781	0	781	8	23
	274	3	-	-	7200	784	0	784	8	22
	206	4	-	-	7200	787	0	787	8	23
	165	5	-	-	7200	785	3	788	8	23
	137	6	-	-	7200	800	7	807	9	24
	118	7	-	-	7200	792	24	816	8	23
	103	8	-	-	7200	796	38	834	8	23
eilA101_50	869	1	-	-	7200	827	0	827	5	39
	435	2	-	-	7200	835	0	835	5	42
	290	3	-	-	7200	847	2	849	5	42
	218	4	-	-	7200	849	6	855	5	42
	174	5	-	-	7200	833	30	863	5	41
eilA101_66	889	1	-	-	7200	846	0	846	6	43
	445	2	-	-	7200	846	0	846	6	41
	297	3	-	-	7200	846	0	846	6	42
	223	4	-	-	7200	866	9	875	6	43
	178	5	-	-	7200	846	28	874	6	43
	149	6	-	-	7200	874	32	906	7	42
eilA101_80	902	1	-	-	7200	859	0	859	7	42
	451	2	-	-	7200	859	0	859	7	45
	301	3	-	-	7200	859	0	859	7	45
	226	4	-	-	7200	770	5	775	7	42
	181	5	-	-	7200	869	17	886	7	43
	151	6	-	-	7200	863	23	886	7	42
	129	7	-	-	7200	859	46	905	7	44

Table 5.3: Detailed comparison of the *Two-Level VNS* with CPLEX for the data *set-1* (T_2)

Name	T_2	Tnb	CPLEX			*Two-Level VNS*				
			Opt. Sol.	No. R.	Time (s)	Actual Cost	Over Time	Cost with overtime	No. R.	Time (s)
eil22_50	408	1	371	3	1	371	0	**371**	3	3
	204	2	375	3	2	375	0	**375**	3	4
	137	3	378	3	1	380	2	382	3	3
eil22_66	403	1	366	3	1	366	0	**366**	3	2
	201	2	382	4	2	382	3	385	4	3
	134	3	366	3	1	366	1	367	3	2
eil22_80	413	1	375	3	3	375	0	**375**	3	3
	206	2	378	4	9	378	0	**378**	4	3
	138	3	381	3	24	381	0	**381**	3	4
eil23_50	745	1	677	3	1	677	0	**677**	3	4
	372	2	689	3	2	691	2	693	3	5
	248	3	716	3	2	716	0	**716**	3	4
eil23_66	704	1	640	3	1	640	0	**640**	3	4
	352	2	640	3	1	640	0	**640**	3	4
	235	3	-	-	7200	696	0	696	3	5
eil23_80	685	1	623	2	1	623	0	**623**	2	4
	343	2	631	2	1	631	0	**631**	2	4
eil30_50	551	1	501	2	1	501	0	**501**	2	4
	276	2	501	2	1	501	0	**501**	2	3
eil30_66	591	1	537	3	3	537	0	**537**	3	6
	296	2	552	3	3451	544	8	**552**	3	7
	197	3	538	3	2	538	0	**538**	3	5
eil30_80	565	1	514	3	11	514	0	**514**	3	6
	283	2	535	3	5519	535	0	**535**	3	7
	188	3	518	3	1426	518	0	**518**	3	5
eil33_50	812	1	738	3	1	738	0	**738**	3	4
	406	2	741	3	2	769	0	769	3	8
	271	3	803 ^	-	7200	764	35	799	3	4
eil33_66	825	1	750	3	12	750	0	**750**	3	5
	413	2	767	3	109	767	0	**767**	3	9
	275	3	-	-	7200	754	21	775	3	5
eil33_80	810	1	736	3	136	736	0	**736**	3	8
	405	2	-	-	7200	756	0	756	3	6
	270	3	-	-	7200	736	18	754	3	6
eil51_50	615	1	559	3	11	559	0	**559**	3	10
	308	2	560	4	67	560	0	**560**	4	9
	205	3	564	4	67	568	0	568	3	11
eil51_66	603	1	548	4	12	548	0	**548**	4	10
	302	2	548	4	56	548	0	**548**	4	11
	201	3	-	-	7200	774	0	774	4	10
	151	4	-	-	7200	563	7	570	4	11
eil51_80	622	1	565	4	78	565	0	**565**	4	11
	311	2	-	-	7200	565	0	565	4	10
	208	3	-	-	7200	587	0	587	4	10
	156	4	-	-	7200	579	0	579	5	10
eilA76_50	812	1	-	-	7200	738	0	738	6	21
	406	2	-	-	7200	738	0	738	6	22
	271	3	-	-	7201	738	0	738	6	22
	203	4	-	-	7202	738	29	767	6	22
	163	5	-	-	7200	747	28	775	6	24
	136	6	-	-	7200	747	15	762	6	21

Name	T_2	Tnb	CPLEX			*Two-Level VNS*				
			Opt. Sol.	No. R.	Time (s)	Actual Cost	Over Time	Cost with overtime	No. R.	Time (s)
eilA76_66	845	1	-	-	7200	768	0	768	7	22
	423	2	-	-	7200	768	0	768	7	21
	282	3	-	-	7200	772	0	772	7	22
	212	4	-	-	7200	769	0	769	7	22
	169	5	-	-	7200	777	13	790	8	23
	141	6	-	-	7200	778	5	783	8	22
	121	7	-	-	7200	771	6	777	8	22
eilA76_80	860	1	-	-	7200	781	0	781	8	23
	430	2	-	-	7200	781	0	781	8	22
	287	3	-	-	7200	783	0	783	8	23
	215	4	-	-	7200	783	0	783	8	22
	172	5	-	-	7200	783	0	783	8	22
	144	6	-	-	7200	786	10	796	8	23
	123	7	-	-	7200	792	13	805	8	23
	108	8	-	-	7200	795	46	841	8	22
eilA101_50	910	1	-	-	7200	827	0	827	5	41
	455	2	-	-	7200	827	0	827	5	41
	304	3	-	-	7200	855	0	855	5	43
	228	4	-	-	7200	838	9	847	5	42
	182	5	-	-	7200	838	13	851	5	42
eilA101_66	931	1	846	6	268	846	0	**846**	6	43
	466	2	-	-	7200	846	0	846	6	42
	311	3	-	-	7200	846	0	846	6	43
	233	4	-	-	7200	868	0	868	6	42
	187	5	-	-	7200	848	14	862	6	43
	156	6	-	-	7200	852	52	904	6	44
eilA101_80	945	1	-	-	7200	859	0	859	7	42
	473	2	-	-	7200	859	0	859	7	43
	315	3	-	-	7200	859	0	859	7	46
	237	4	-	-	7200	859	0	859	7	43
	189	5	-	-	7200	863	15	878	7	44
	158	6	-	-	7200	870	13	883	7	45
	135	7	-	-	7200	859	24	883	7	42

5.6.2.1. Search diversification and intensification analysis

A further analysis regarding the search diversification and intensification is carried out. The idea is to check which neighbourhoods are important when compared with others in terms of diversification and to know which neighbourhoods are leading towards better quality solution and to what extent the search is intensifying. To achieve this, a small subset of five instances was selected ranging in size between 21-100 customers from data *set-1*. The algorithm is executed on each instance for 5 iterations and the neighbourhood moves leading towards better quality solution were recorded for each iteration and the average was calculated. However, in terms of intensification, the iteration yielding the best solution was also recorded.

Table 5.4 report the number of times each neighbourhood move leads towards a better quality solution on average including the grand average for each instance. It can be observed from the individual averages that each neighbourhood varies in terms of times they lead towards better solution and they all appear to be important. However, when looking at the grand average it can be noticed that N_1 (1-0 Intra Route) neighbourhood appears to be the most important among all leading 8 times toward better quality solution on the grand average. The second most important move is N_6 (2-1 Swap) leading 6 times towards better quality solution. Although, the number of times other neighbourhoods lead towards better quality solution is slightly lower than these two but their importance cannot be ignored as they are also playing a vital role in terms of search diversification and hence leading towards better quality solutions.

Moreover, we have also provided a graphical representation showing the extent to which the search diversification and intensification is achieved for each instance in Figure 5.20, Figure 5.21, Figure 5.22, Figure 5.23 and Figure 5.24 respectively.

It can be observed from these figures that N_1 (1-0 Intra Route) and N_6 (2-1 Swap) neighbourhoods appear to provide maximum diversification. In terms of intensification, the algorithm is doing really well as it can be seen in the graphs how quickly it improves solution quality. The *Two-Level VSN* algorithm also accepts low quality solutions at *inner-level* in order to get out of local optima and hence improving the solution quality.

Table 5.4: The number of times each neighbourhood leads towards better quality solution on average for each instance

	eil22_66 Average	**eil30_80 Average**	**eil51_50 Average**	**eilA76_50 Average**	**eilA101_80 Average**	**Grand Average**
N_1 : 1-0 Intra Route	2	8	7	8	16	8
N_2 : 1-0 Inter Route	1	5	5	5	6	4
N_3 : 1-1 Swap	1	5	4	6	6	4
N_4 : 2-2 Swap	1	6	5	4	6	4
N_5 : 2-0 Swap	1	5	6	6	8	5
N_6 : 2-1 Swap	1	5	6	6	10	6

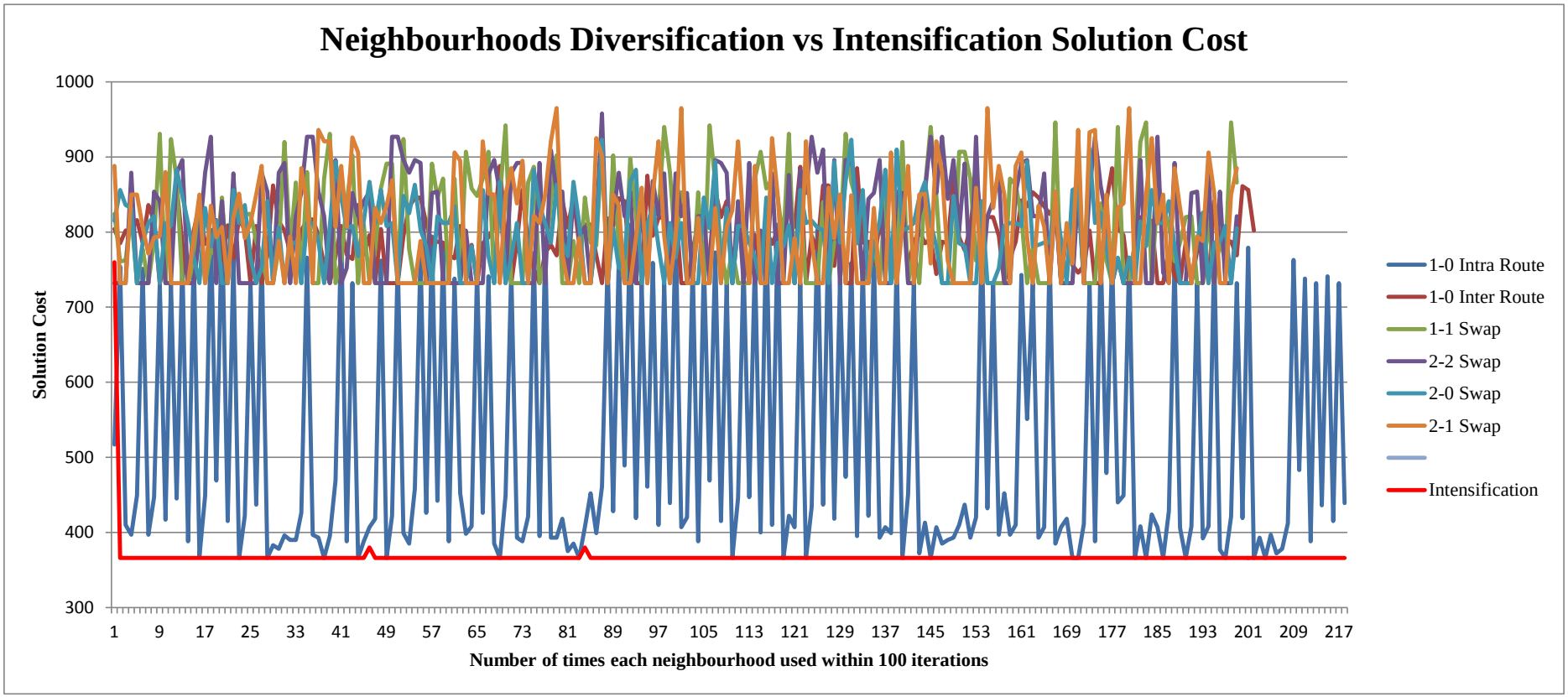

Figure 5.20: Neighbourhoods diversification vs Intensification solution cost for data instance eil22_66_1_t1

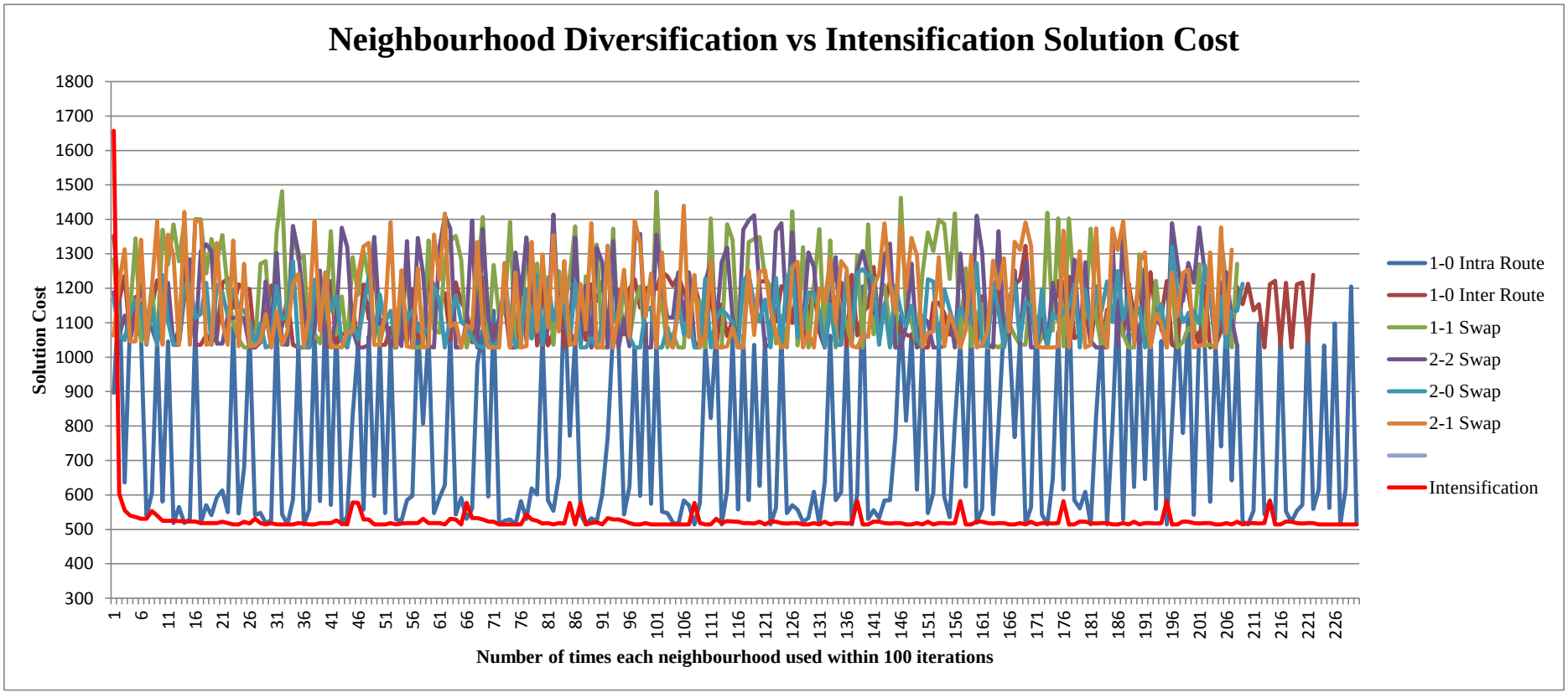

Figure 5.21: Neighbourhoods diversification vs Intensification solution cost for data instance eil30_80_1_t1

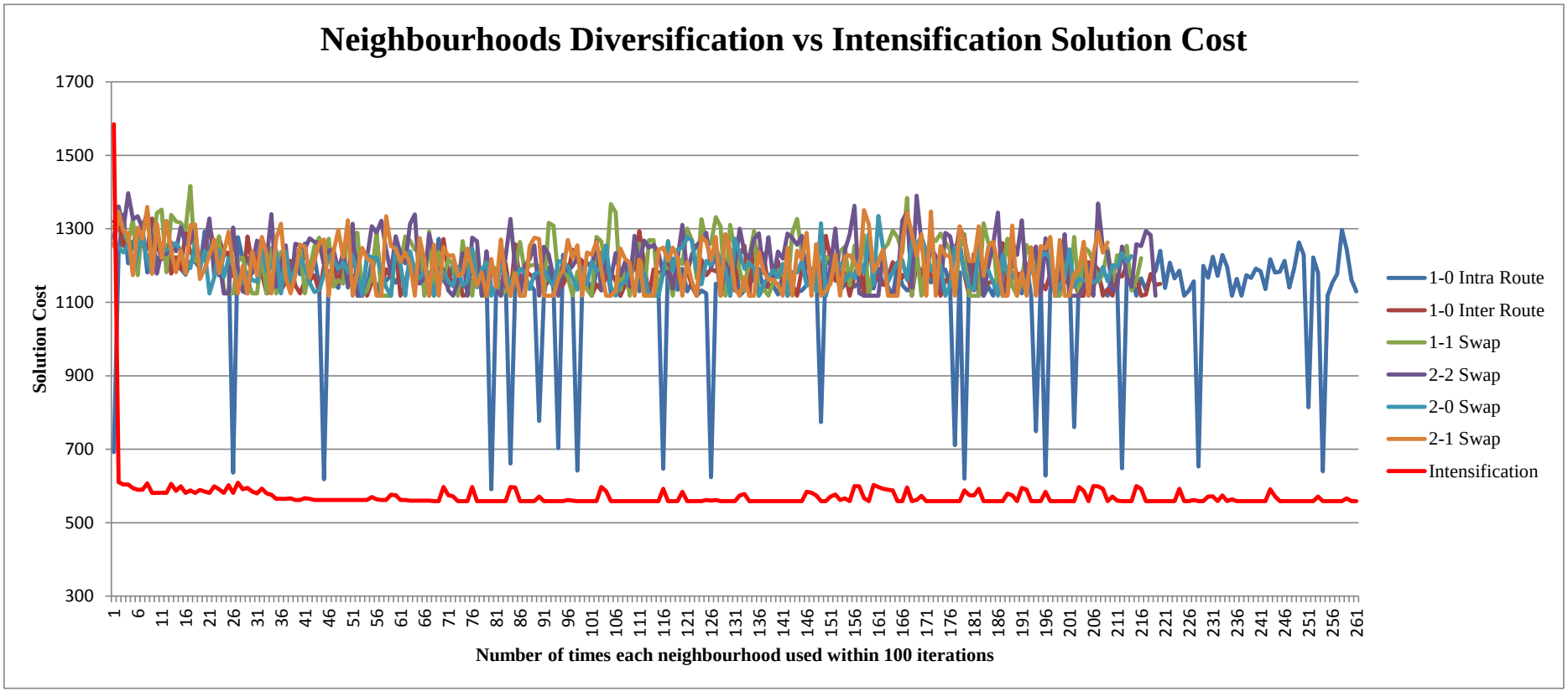

Figure 5.22: Neighbourhoods diversification vs Intensification solution cost for data instance eil51_50_1_t1

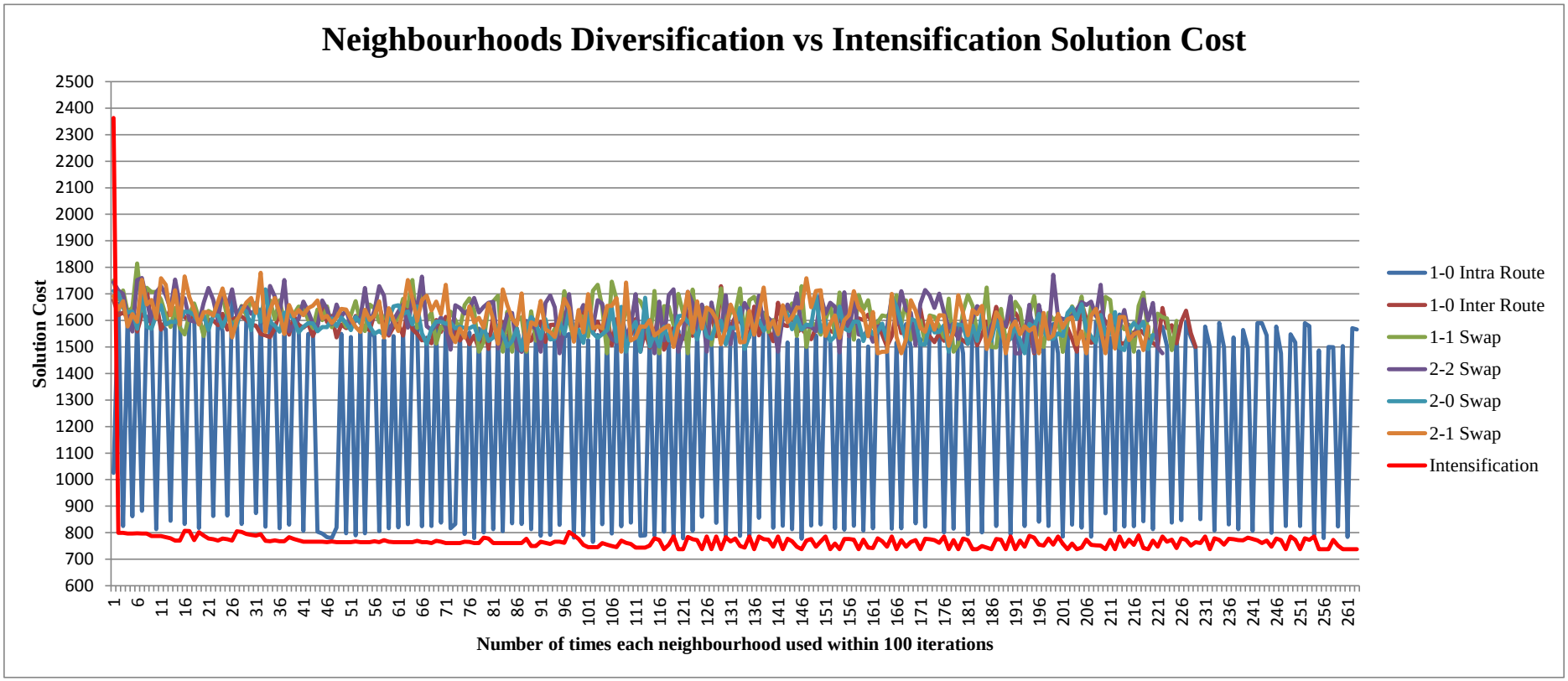

Figure 5.23: Neighbourhoods diversification vs Intensification solution cost for data instance eilA76_50_1_t1

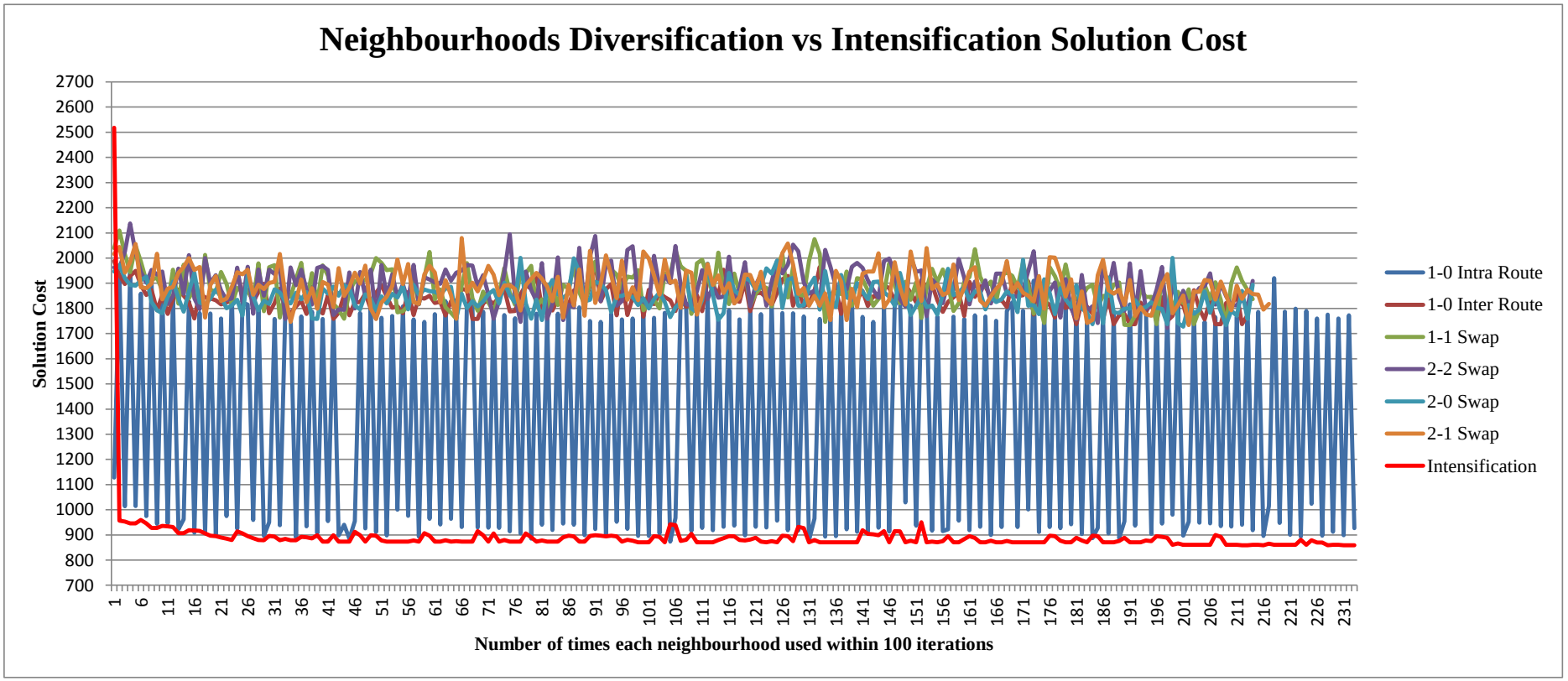

Figure 5.24: Neighbourhoods diversification vs Intensification solution cost for data instance eilA76_50_1_t1

5.7 Summary

In this chapter we designed a new VNS algorithm that uses two levels to solve the MT-VRPB. The *Two-Level VNS* algorithm uses skeletons of the classical VNS and VND methodologies. A number of neighbourhoods and local searches are employed in such a way to achieve diversification at the outer level (basic VNS) of the algorithm and intensification at the inner-level (VND with a multi-layer local search framework). The algorithm found promising solutions when compared with the solutions found by CPLEX. It matched 85% of the optimal solutions obtained by CPLEX ranging in size between 21-50 customers including one instance with 100 customers. The *Two-Level VNS* obviously solved all the 168 instances (105 with no overtime used); and the rest with only 5 and 10 units average overtime for T_2 and T_1, respectively. The speed of the algorithm remained remarkably fast as it requires less than 20 seconds on average per problem instance. It can therefore be said that this study demonstrates the power of VNS yet again in terms of its simplicity, flexibility, efficacy and speed. Moreover, a brief analysis of the algorithm in terms of search diversification and intensification show the importance of neighbourhood moves used during the search process.

Although, the *Two-Level VNS* found a very high number of good feasible solutions, optimality or closeness to optimality could not unfortunately be measured as CPLEX could not find optimal solutions for all instances. In the next Chapter, we explore the use of a Collaborative Sequential Mat-heuristic (CSMH) algorithm for the MT-VRPB to see whether or not the solutions of the *Two-Level VNS* can be either improved or shown to be optimal if possible.

Chapter 6

Solving the MT-VRPB using a Collaborative Sequential Mat-heuristic approach

Combining mathematical programming techniques with heuristic methods to solve Combinatorial Optimisation problems is one of the recent developments in the OR literature. These approaches are recognised as a new class of hybrid methodologies and being termed as '*mat-heuristics*'. In this chapter, a hybrid collaborative sequential *mat-heuristic* approach is developed to solve the MT-VRPB. The mathematical model developed in Chapter 4 is hybridised with the *Two-Level VNS* algorithm developed in Chapter 5 in a sequential manner. The MT-VRPB data set used in the previous Chapters is tested to assess the benefit of combining these two methodologies.

6.1. The *Mat-heuristic* Approaches

The term *mat-heuristics* refers to designing of those optimisation algorithms in which heuristics and mathematical programming techniques are used in conjunction. For more information on the general classification of combining exact and heuristics method for

combinatorial optimisation problems, we refer to Caserta and Voß (2010), Puchinger and Raidl (2005), and Raidl (2006). Two categories of combinations called '*Collaborative Combinations'* and '*Integrative Combinations'* are presented. In the collaborative combinations, the algorithms (exact-heuristic) are combined in such a way that they are not part of each other; hence they can only exchange information. However, the collaborative combined algorithms (exact-heuristic) may be performed either in parallel, interconnected or in a sequential manner. On the other hand, integrative combinations category joins the algorithms (exact-heuristic) in such a way that one method works as an assistant embedded component of another. Therefore one algorithm (either exact or heuristic) works as a master method and the other performs as a slave (subordinate) method. The *Collaborative* and *Integrative* combinations are further categorized into subcategories.

The collaborative combination is divided into '*Sequential execution*' and '*Parallel* or *Intertwined execution*'. In the former either the heuristic technique is executed first followed by the exact technique or vice-versa; whereas in the latter exact and heuristic methods work in parallel or in interconnected style. Both the *Sequential* and the *Parallel* versions have their pros and cons.

The *Integrative* combination is also subcategorized into '*incorporating exact algorithms in heuristics*' (where heuristic works as a master method and the exact algorithm works as a slave method) and '*incorporating heuristics in exact algorithms*' (where the exact algorithm performs as a master component and the heuristic technique performs as an embedded slave component). More information of these components can be found in

Puchinger and Raidl, (2005); whereas, the taxonomy of exact-heuristics hybridisation is provided in Jourdan *et al*. (2009).

We have developed a collaborative sequential *mat-heuristic* that chains our mathematical programming in Chapter 4 and the VNS meta-heuristic in Chapter 5 to solve the MT-VRPB. The details of our approach are provided in the following sections.

6.1.1. Matheuristics for VRPs: Brief Literature Review

One of the early studies is by Foster and Ryan (1976) in which an improvement heuristic that incorporates the solution of a mixed-integer linear programming (MILP) model is proposed for the VRP. In this study, a set partitioning formulation for the VRP is presented first and then a matheuristic algorithm is proposed. At first phase, a set of petal routes is generated using a heuristic construction method known *farthest away cheapest insertion* method. The reason behind calling them petal routes is because of their resemblance to petals as they are rooted at the depot. In the second phase, set partitioning formulation is solved on the set of routes obtained at first phase. The algorithm was tested on data set containing fifteen instances ranging in size from 21 to 100 customers. The computational results show improvements when compared with previously published results.

Fisher and Jaikumar (1981) proposed a cluster-first-route-second method. In this algorithm, heuristic is used to select so-called *seed customers* and then in order to assign the remaining customers to the seed customers, an assignment problem is solved to optimality at first phase. Where each seed customer pinpoints a cluster of customers associated with it. At second phase, a Travelling Salesman Problem (TSP) is solved on

each cluster to obtain the final set of routes. The algorithm was tested on 12 VRP standard problem instances and outperformed previously published studies.

In 1995, Bramel and Simchi-Levi proposed a method similar to that of Fisher and Jaikumar (1981) for the VRP. This methodology is based on the routing problem type formulation as a Capacitated Concentrator Location Problem (CCLP). The basic idea in this algorithm is to identify seed customers in order to estimate the cost of assigning each customer to each seed customer and then solve a CCLP to determine the customer clusters. Hence, after determining the clusters, a TSP is solved on each cluster to obtain the solution. The computational results show that this algorithm outperforms all published heuristics when tested on a set of standard test problems.

Rochat and Taillard (1995) proposed a matheuristic algorithm for the VRP. In the first phase, a heuristic based on local search algorithm is used to solve the VRP. Hence all routes obtained at this phase are stored in a set *P*. At second phase, in order to choose best routes from set *P*, a set partitioning model is solved to optimality. The algorithm was tested on various problem instances from the literature and solutions of 40 instances are improved compared to previous published work.

Kelly and Xu (1999) proposed a set partitioning based heuristic for the VRP. This algorithm is similar to that of Foster and Ryan (1976). In the first phase, different solutions are obtained using a simple and fast construction heuristics. In the second phase, a set partitioning model is solved in order to select the best routes from the set of all routes. The algorithm is tested on VRP benchmark instances. The computational results show that this algorithm found same solutions in most cases when compared with the best known published results.

De Franceschi *et al*. (2006) proposed a new ILP-based refinement heuristic for vehicle routing problems. In this algorithm, the initial solution is constructed by taking the best known solution in the literature. Then chains of customers are removed from the solution; hence, a large number of chains is organized from the removed chains of customers and various insertion points are pinpointed in the partial solution. Then chains of removed customers are inserted in the insertion points by solving the MILP model to optimality. The algorithm is tested on two data sets from the literature. The results presented show that the algorithm found better solutions in some cases when compared with the best known in the literature.

Archetti and Speranza (2008) proposed an optimization-based heuristic for the split delivery vehicle routing problem (SDVRP). An integer program based on the extension of the classical set-covering model is used as master program and the tabu search works as a subordinate (slave) method. That is, tabu search is used once and frequency counters are used to analyse the obtained set of solutions by tabu search. This is done in order to specify the number of times a particular edge was part of a solution and to point out whether particular customers' demand was split. Furthermore, the solutions came across by the tabu search in which a customer that is never or rarely split has been served by a single vehicle in high-quality solution. Likewise the edges which are encountered frequently during tabu search are likely indicated as a part of near-optimal solution. A set of promising routes *R* has been generated by the frequency counters and desirability measures are used in order to sort out this set. Finally, a subset of routes *r* has been taken from the set of promising routes *R* and based on that subset of routes, a set-covering problem is solved iteratively. The computational results show that the

initial solutions obtained by the tabu search are improved by the proposed method in all test instances except one.

Schmid *at el*. (2009) proposed a hybrid solution approach based on the integer multicommodity network flow (MCNF) component and variable neighbourhood search (VNS) for the ready-mixed concrete delivery problem. The proposed hybrid approach belongs to the *collaborative* category of matheuristics. Therefore the information between an integer MCNF component and VNS is exchanged in a bi-directional way in order to obtain the high quality solutions. First the MCNF component is solved, that is initialized with a randomly generated set of patterns. Then VNS is used to further improve the best solution iteratively in order to enrich the pool of patterns used by the MCNF. Finally the MCNF component is used again to obtain even better solutions. Computational experiments are done using a real-life data taken from a concrete company. The obtained computational results show that the proposed hybrid approach performed better when compared with the solution obtained by a commercial approach (based on simulated annealing meta-heuristic) developed specially for this type of problems.

Rei *et al*. (2010) presented a hybrid algorithm the single VRP with stochastic demands. This methodology employs both local branching heuristic and Monte Caro sampling in order to divide the solution space in sub-regions; hence obtaining sub-problems. A MILP model is then used to solve the sub-problems. A sub-set of instances ranging in size from 60 to 90 customers are tested using this methodology. The algorithm proved quite effective in terms of solution quality.

6.2. The Collaborative Sequential Approach for the MT-VRPB

Our collaborative sequential *mat-heuristic* (CSMH) approach for the MT-VRPB belongs to the '*Collaborative Combinations*' category of the *mat-heuristic* approaches in general and more specifically fits to the '*Sequential execution*'. Hence the CSMH approach executes the *Two-Level VNS* heuristic first followed by the exact technique formulation using CPLEX optimiser. The generic phases of the CSMH approach are shown in Figure 6.1.

The ingredients of the first three phases in the CSMH algorithm are already provided and explained in Chapter 5 but we briefly summarise them here as follows for ease of understanding.

Phase I:

In *Phase I* of the CSMH algorithm approach an initial feasible VRPB solution is obtained by using the *sweep-first-assignment-second* methodology. Firstly two sets of open ended routes (one for each LH and BH customers) are generated by using the *sweep* procedure separately on LH and BH customers. The LH and BH routes are then connected by solving the assignment problem; and if needed the backhauling conditions are satisfied by performing some local changes in the combined LH/BH solution before moving to the next phase. The initial solution generation steps are already provided in detail in Section 5.2.

Phase I: **Initial solution –*sweep-first-assignment-second* approach (see Section 5.2)**

Phase II: Two-Level VNS **Algorithm (see Sections 5.3 & 5.4)**

Phase III: **Solving the Multiple Trips aspect using the BPP (see Section 5.5)**

Phase IV: **Solve the mathematical model using CPLEX**

a. Choose the best solution *k* (in terms of feasibility, i.e., solution without overtime) from the data structure Sol_k

b. If overtime is used in the solution *k*, then go to ***d***
Else, go to ***c***

c. Prepare *MIPstart* for CPLEX

- *MIPstart = k* [*MIPstrat* represents a feasible solution]
- Call *CPLEX_model* and run until total allocated time of 2 hours is reached
- Report optimal/incumbent solution

d. Set: $Z_C \leq Z_H$ [where Z_C represents *CPLEX_model* objective value and Z_H best heuristic solution cost with overtime]

- Call *CPLEX_model* and run until total allocated time of 2 hours is reached
- Report optimal/incumbent solution

Figure 6.1: The CSMH approach phases for the MT-VRPB

***Phase II*:**

In *Phase II*, the *Two-level VNS* mechanism is used to improve the initial solution and obtain a pool of solutions. The *Two-Level VNS* is a composite mechanism that comprises two levels, called outer and inner levels. Several neighbourhood structures and local search refinement routines (developed in Section 5.3) are used to achieve a balanced diversification and intensification within the levels during the search process.

For both levels, a subset of neighbourhoods and a subset of local search refinement routines are proposed. These local search refinement routines are embedded within a multi-layer local search framework. For further details, see Section 5.4.

***Phase III*:**

Phase III determines the multiple trip packing of the routes in the solutions by solving the Bin Packing Problem which is based on the pool of solutions obtained in *Phase II*. For each solution in the pool, the BPP is solved by calling CPLEX optimiser within the Microsoft Visual Studio Environment followed by a repair mechanism if necessary known as the *Bisection Method*. Here, bin capacity is gradually increased by a certain percentage iteratively until routes are feasibly packed into bins; see Section 5.5 for more details.

***Phase IV*:**

Phase IV chooses the best MT-VRPB solution from the data structure Sol_k and passes it on to the CPLEX optimiser using a mechanism called mixed integer programming start '*MIPstart*'. For this stage the mathematical formulation model of the MT-VRPB is coded in C++ programming language within the Microsoft Visual Studio Environment that calls the CPLEX optimiser that uses the best packed solution from the *Phase III* as an incumbent solution. The *MIPstart* is explained in the next section.

Use of the *MIPstart* mechanism

The *MIPstart* is a mechanism provided by the IBM ILOG CPLEX Optimisation Studio through which one can provide the CPLEX optimiser with an initial solution. For instance, a first or second integer solution could be from a MIP problem which was

found previously or a feasible solution from a heuristic. The *MIPstart* may include various types of variables such as integer variables, semi-continuous variables and binary variables etc. An MIP starting variable/s can be established using some methods. For Concert Technology application users, the method 'addMIPStart 'is used; whereas for the Callable Library applications method is called 'CPXaddmipstarts'. Since we are using Concert Technology in our application, the former method is used. For more information on the types of variables see User's Manual for CPLEX V12.5.1.

***Preparing the MIP start for CPLEX optimiser*:**

If the chosen solution from the data structure Sol_k is feasible with respect to the given planning period *T*, then the *MIPstart* variables have no problem in working with the formulation of the MT-VRPB. However if the chosen solution is not feasible in terms of *T* then the *MIPstart* will not take it as an input solution. This is because our basic formulation in Chapter 4 does not allow overtime to be used.

Basic Modification:

To overcome this hurdle, we have added constraint (6.1) in our MT-VRPB model formulation which enables the infeasible solutions as workable input bound for the *MIPstart*.

$$Z_C \leq Z_H \tag{6.1}$$

Where, Z_C represents the objective value in our CPLEX model and Z_H the best heuristic solution cost (i.e., best solution with overtime chosen from data structure Sol_k). Note that this constraint is employed automatically for the *False MIPstart* condition (i.e.,

when none of the packed solutions is feasible in terms of maximum driving time and an infeasible (i.e., solution with overtime) solution is chosen).

The C++ programming language code that we have used to add the *MIPstart* in our model is provided in Appendix A. As it can be seen that we have two decision variables in our formulation where R[i][j] represents the amount delivered/picked up on arc (i, j) and X[i][j][k] is a $(0, 1)$ decision variable that represents X[i][j][k] $= 1$ if vehicle k travels arc (i, j), 0 otherwise. At this stage the solution found by the *Two-Level VNS* approach is prepared in a format that is understandable by CPLEX interactive optimiser. Therefore, two multidimensional arrays represented as R_VNS[i][j] (integer in type) and X_VNS[i][j][k] (0, 1 in type) are created that contain the heuristic solution. Then these (i.e., R_VNS[i][j] and X_VNS[i][j][k]) multidimensional arrays are first flattened into one-dimensional arrays (since CPLEX converts all multidimensional arrays into one-dimensional arrays and then starts working on them) and then added to the respective decision variable arrays (i.e., R[i][j] and X[i][j][k]) using 'startVal.add' method.

Transformation to CPLEX and an illustrative example:

Since our MT-VRPB formulation model is coded in C++ programming language, and therefore adding a *MIPstart* to a model that is implemented in the C++ API (Application Programming Interface) needs 'IloCplex::addMIPStart' method provided by ILOG IBM CPLEX 12.5. Moreover, we need to write and pass the heuristic solution in such format that is understandable by the CPLEX optimiser. The idea is to provide a copy of each decision variable with its corresponding values while the decision variables need to be the same type and same dimension. For example if CPLEX uses X_{ij} we need to provide another variable say XX_{ij} with its values.

Test instance characteristics

n=6 (number of customers] (1,…,4, 5, 6) where 1 to 4 are deliveries and last 2 are backhauls
b=2 [number of backhaul customer]
T=195 [planning period (maximum driving time)]
C=2000 [vehicle capacity]
v=2 [number of total Bins/vehicles]

q = [1000,800,500,1200,1500,300] [demand/supply]

Dist= [0,49,42,37,31,28,23]
[49,0,23,21,24,30,40]
[42,23,0,34,33,38,23]
[37,21,34,0,5,11,38] [distance matrix]
[31,24,33,5,0,6,35]
[28,30,38,11,6,0,35]
[23,40,23,38,35,35,0]

Solution routes obtained for test instance above where R_2 and R_3 are served by bin (vehicle) 1 and R_1 is served by bin 2.

Route 1:

		dist	demand/supply
Dep:	0		
Cust:	1	49.00	1500
Cust:	3	21.00	500
Cust:	5	11.00	0
Dep:	0	28.00	1500

C	L_load	B_load	EmptSpace
1500	1500	0	0
1500		1500	0

Cost/Distance = 109.00

$R_1 = 0 \rightarrow 1 \rightarrow 3 \rightarrow 5 \rightarrow 0$

Route 2:

		dist	demand/supply
Dep:	0		
Cust:	2	42.00	800
Cust:	6	23.00	0
Dep:	0	23.00	300

C	L_load	B_load	EmptSpace
1500	800		700
1500		300	1200

Cost/Distance = 88.00

$R_2 = 0 \rightarrow 2 \rightarrow 6 \rightarrow 0$

Route 3:

		dist	demand/supply
Dep:	0		
Cust:	4	31	1200
Dep:	0	31	0

C	L_load	B_load	EmptSpace
1500	1200		300
1500		0	1500

Cost/Distance = 62.00

$R_3 = 0 \rightarrow 4 \rightarrow 0$

Transforming the solution routes to CPLEX understandable format

R_VNS[i][j] =

Depot	0	1	2	3	4	5	6
0	0	1500	800	0	1200	0	0
1	0	0	0	500	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0
5	1500	0	0	0	0	0	0
6	300	0	0	0	0	0	0

X_VNS[i][j][k] =

Depot	0	1	2	3	4	5	6
0	0	1	1	0	1	0	0
1	0	0	0	1	0	0	0
2	0	0	0	0	0	0	1
3	0	0	0	0	0	1	0
4	1	0	0	0	0	0	0
5	1	0	0	0	0	0	0
6	1	0	0	0	0	0	0

Figure 6.2: MIPstart construction for the MT-VRPB test instance

As an example, in our case, we construct a MIPstart as shown in Figure 6.2 which shows the characteristics of the test instance and how R_VNS and X_VNS multidimensional arrays are constructed.

6.3. Computational Experience

The CSMH algorithm is coded in C++ programming language and implemented within the Microsoft Visual Studio Environment (version: 2010). The experiments were executed on a PC with Intel(R) Core(TM) i7-2600 processor, CPU speed 3.40 GHz. The CSMH calls the IBM ILOG CPLEX 12.5 interactive optimiser within the Visual Studio Environment to solve the assignment problem (*Phase I*), the BPP (*Phase III*) and the exact method formulation (*Phase IV*).

6.3.1. Data Set

The computational experiments are reported for the MT-VRPB data *set-1* generated in this thesis with the details provided in Section 4.7. For convenience, all the data sets used in this thesis can also be downloaded from CLHO website (CLHO, 2015).

6.3.2. The CSMH execution times

The CSMH algorithm is run for a maximum CPU time of 2 hours (7200 seconds) for all the four phases. In addition we allow 100 iterations for *Phase II* and 5 iterations for *Phase III* if the *Bisection method* is required.

Glossary for tables:

T_1 = Total driving time (type one) for a bin/ vehicle,

T_2 = Total driving time (type two) for a bin/ vehicle,

Tnb = Total number of vehicles (bins) in each instance,

VNS Sol. = Solution obtained by *Two-Level VNS*,

No.R = Number of total routes in solution,

Optimal Sol. = Optimal solution,

Incum. Sol. = Incumbent solution (feasible solution found without using overtime),

UB = Upper bound,

LB = Lower bound,

%gap = % gap between optimal/incumbent solution and lower bound,

X = Infeasible instance (not even lower bound exist for these instances),

- = Not found,

^ = Incumbent solution (feasible solution, i.e., solution without overtime),

* = Optimal solution,

+ = Solution with overtime,

Time(s) = CPU time in seconds taken to solve each instance.

6.3.3. The CSMH algorithm performance

The performance comparison of the CSMH algorithm is summarised in Tables 6.1 – 6.4, and the detailed results are provided in Tables 6.5 - 6.10. The CSMH performed very well in terms of the solutions quality and the CPU time consumption. Table 6.1 presents the overall summary of the CSMH algorithm. For T_1 the CSMH algorithm found a large number of optimal solutions (38 out of 84) and a good number of incumbent solutions (12 out of 84). Here, the upper bound/integer solutions are reported, however optimality was not achieved in 2 hour time limit. For the remaining 34 instances, no change happened at the end of *Phase IV* highlighting that the heuristic

solution achieved at *Phase III* is very good. Hence, the lower bounds are also reported for those instances. Moreover, for 6 instances *Phase VI* (CPLEX optimiser) found better packed solutions reducing the number of vehicle routes by one for each case. On the other hand, *Phase VI* (CPLEX optimiser) did not increase the number of vehicle routes for any of the tested instances in order to obtain a better solution.

For the T_2 set of instances the CSMH algorithm found a great number (more than half of the instances) of optimal solutions (46 out of 84) and a good number of new best incumbent (feasible) solutions (18 out of 38 non-optimal). For the rest (20 out of 84) no optimal/incumbent solution is obtained, hence the input heuristic solution is retained, and the lower bounds are reported for those instances. Furthermore, for 3 instances *Phase VI* found better packed solutions reducing the number of vehicle routes by one for each instance. However, *Phase VI* (CPLEX optimiser) increased one vehicle route for an instance in order to a obtain better solution. Detailed results are provided in Table 6.5 and Table 6.6 for T_1 and T_2 data instances classes, respectively.

Table 6.1: Summary of the CSMH algorithm solutions (data *set-1*: T_1 & T_2)

	T_1	T_2
# of solutions found (out of 84)	84	84
# of optimal solutions found	38	46
# of new best solutions found	12	18
# of instances where no. of routes decreased at the end of *Phase IV*	6	3
# of instances where no. of routes increased at the end of *Phase IV*	0	1
# of instances reported infeasible by the CSMH algorithm	4	0
Average CPU time (s)	3993	3694

6.3.4. Comparison of the CSMH vs CPLEX results

In this section, the results found by the CSMH algorithm are compared with the results found by CPLEX in Chapter 4. Note that the solutions found by the CSMH algorithm with overtime are not considered in this comparison since the MT-VRPB mathematical model CPLEX solutions do not include overtime. Therefore, here only those instances are compared for which either CPLEX or CSMH algorithm found optimal/incumbent solutions. The run time for both CPLEX and the CSMH algorithm is set to 2-hours.

Table 6.2 presents a summary of comparisons, whereas the detailed solutions comparison is provided in Table 6.7 and 6.8 respectively. For the T_1 group of 84 instances, compared to 24 solutions (all optimal) of CPLEX, the CSMH found 50 solutions, i.e., 38 optimal and 12 incumbent (solutions where both upper and lower bounds are obtained – i.e., a feasible solution however optimality was not proved in given computational time). Hence, in terms of optimal/incumbent solutions, the CSMH found more than 50% additional solutions as compared to CPLEX in this group.

For T_2, the CSMH found 64 (46 optimal, 18 incumbent) solutions as compared to 37 (36 optimal, 1 incumbent) of CPLEX.

Furthermore, the CSMH algorithm produced optimal solutions for the instances ranging in size between 21-100 customers for both groups (T_1 and T_2) compared to CPLEX where the solutions are found ranging in size between 21-50 customers except one instance in T_2 with 100 customers.

Lower bound effects:

Moreover, we did some analysis about the quality of the lower bounds generated by CPLEX and the CSMH by calculating the percentage gaps from the optimal solution found in this study. The details of the comparison and average gaps (%) [(Opt – LB)/Opt * 100] are shown in Table 6.9. The CSMH also proved superior on this front by producing lower gaps of 1.90% and 1.99% as compared to the original formulation with 2.89% and 2.30% for T_1 and T_2 classes respectively, see Table 6.9.

It should be noted that the average time actually used by the CSMH is much lower ($T_2 = 3993$ and $T_2 = 3694$) than the allocated average time of 7200 seconds. On average the CSMH used relatively lower time compared to the time spent by CPLEX in Chapter 4. Moreover, comparing the computational time for only optimal solutions of CPLEX and the CSMH, the latter performed better by spending 117 seconds/instance on average as comparted to 504 seconds/instance of CPLEX for T_1, and respectively 221 seconds/instance and 314 seconds/instance for T_2, see Table 6.2.

Hence, looking at the overall assessment, the CSMH proved superior in terms of the solution quality and speed.

Table 6.2: Summary comparison of the CSMH vs CPLEX (data *set-1*: T_1 & T_2)

	T_1		T_2	
	CPLEX	**CSMH Algorithm**	**CPLEX**	**CSMH Algorithm**
# of solutions found without overtime (out of 84)	24	50	37	64
# of optimal solutions found	24	38	36	46
# of incumbent solutions found	0	12	1	18
Grand average gap (%)	2.89	1.90	2.30	1.99
Average CPU time (s)	5165	3993	4248	3694
Average CPU time (s) for optimal solutions.	504	117	314	221

6.3.5. Comparison of the CSMH and the *Two-Level VNS* results

Table 6.3 presents a summary performance comparison of the CSMH algorithm and the *Two-Level VNS*. The detailed comparison is provided in Table 6.10 and Table 6.11 for T_1 and T_2, respectively.

For T_1 and T_2, both algorithms performed very well in terms of solving all the instances of both groups (84 out of 84). When it comes to the numbers of optimal solutions, the CSMH performed better with 38 and 46 as compared to the *Two-Level VNS* of 33 and 38 for T_1 and T_2, respectively. The CSMH algorithm also found a higher number of incumbent solutions (i.e., feasible solutions for which no overtime is used) than the *Two-level VNS*. However, the *Two-Level VNS* found better solutions for the instances in both groups (T_1 & T_2) in which overtime (i.e., solutions with overtime) is used. This is understandable since the best solutions are reported out of 5 run for the *Two-Level VNS*. In terms of computational time, the *Two-level VNS* is obviously much faster due to the fact that the CSMH algorithm uses CPLEX optimisation technique.

Table 6.3: Comparison of the CSMH vs the *Two-Level VNS* (data *set-1*: T_1 & T_2)

	T_1		T_2	
	Two-Level VNS	**CSMH Algorithm**	***Two-Level VNS***	**CSMH Algorithm**
# of solutions found (out of 84)	84	84	84	84
# of solutions found without overtime (out of 84)	46	50	59	64
# of optimal solutions found	33	38	38	46
# of solutions found with overtime (out of 84)	38	34	25	20
Average CPU time (s)	18	3993	17	3694

Moreover, Table 6.4 shows a summary comparison of the results from CPLEX, *Two-Level VNS* and CSMH.

Table 6.4: Comparison of CPLEX, *Two-Level VNS* and CSMH (data *set-1*: T_1 & T_2)

	T_1			T_2		
	CPLEX	***Two-Level VNS***	**CSMH**	**CPLEX**	***Two-Level VNS***	**CSMH**
# of solutions found (out of 84)	24	84	84	37	84	84
# of optimal solutions found (out of 84)	24	33	38	36	38	46
# of incumbent solutions found (out of 84)	0	46	50	1	59	64
Average CPU time (s)	5165	18	3993	4248	17	3694

Table 6.5: Detailed results of the CSMH algorithm for the Data *set-1* (T_1)

Name	T_1	Tnb	CSMH Algorithm								
			VNS Sol.	No. R.	Optimal Sol.	Incum. Sol.	No. R.	UB	LB	%Gap	Time (s)
eil22_50	390	1	371	3	371	-	3	371.0000	354.5515	4.43%	2
	195	2	396+	3	378	-	3	378.0000	376.9945	0.27%	3
	130	3	392+	3	x	-	x	x	x	x	3
eil22_66	385	1	366	3	366	-	3	366.0000	343.1949	6.23%	3
	193	2	396+	4	382	-	4	382.0000	375.5642	1.68%	7
	129	3	375+	3	x	x	x	x	x	x	5
eil22_80	394	1	375	3	375	-	3	375.0000	364.8797	2.70%	4
	197	2	388	4	378	-	4	378.0000	367.1494	2.87%	10
	132	3	389	4	381	-	3	381.0000	374.0939	1.81%	105
eil23_50	711	1	677	3	677	-	3	677.0000	640.4404	5.40%	3
	355	2	710+	4	698	-	3	698.0000	677.2488	2.97%	11
	237	3	725+	3	x	x	x	x	x	x	8
eil23_66	672	1	640	3	640	-	3	640.0000	612.5018	4.30%	3
	336	2	640	3	640	-	3	640.0000	629.2119	1.69%	3
	224	3	702+	3	x	x	x	x	x	x	4
eil23_80	654	1	623	2	623	-	2	623.0000	599.1210	3.83%	2
	327	2	634	2	634	-	2	634.0000	620.8261	2.08%	3
eil30_50	526	1	501	2	501	-	2	501.0000	501.0000	0.00%	6
	264	2	507+	2	x	x	x	x	x	x	8
eil30_66	564	1	537	3	537	-	3	537.0000	537.0000	0.00%	7
	282	2	599+	3	552	-	3	552.0000	538.0000	2.54%	2302
	188	3	541+	3	-	-	-	-	532.6645	-	7200
eil30_80	540	1	514	3	514	-	3	514.0000	514.0000	0.00%	6
	270	2	559+	3	535	-	3	535.0000	465.5482	12.98%	6172
	180	3	518	3	518	-	3	518.0000	510.8803	1.37%	8
eil33_50	775	1	738	3	738	-	3	738.0000	738.0000	0.00%	5
	388	2	766+	3	-	-	-	-	738.1813	-	7200
	258	3	822+	3	-	-	-	-	740.6625	-	7200
eil33_66	788	1	750	3	750	-	3	750.0000	723.3959	3.55%	4
	394	2	772	3	772	-	3	772.0000	768.0827	0.51%	93
	263	3	792+	3	-	-	-	-	760.4523	-	7200
eil33_80	773	1	736	3	736	-	3	736.0000	730.2669	0.78%	9
	387	2	763	3	756	-	3	756.0000	754.4379	0.21%	1087
	258	3	766+	3	-	-	-	-	702.4513	-	7200
eil51_50	587	1	560	4	559	-	3	559.0000	554.6452	0.78%	14
	294	2	573	4	562	-	4	562.0000	558.9278	0.55%	108
	196	3	605+	4	-	-	-	-	551.0056	-	7200
eil51_66	576	1	551	4	548	-	4	548.0000	547.0163	0.18%	41
	288	2	560	4	552	-	4	552.0000	550.6893	0.24%	171
	192	3	577+	4	-	-	-	-	544.7850	-	7200
	144	4	583+	4	-	-	-	-	549.8806	-	7200
eil51_80	594	1	578	4	565	-	4	565.0000	563.1379	0.33%	159

Name	T_1	Tnb	CSMH Algorithm								
			VNS Sol.	No. R.	Optimal Sol.	Incum. Sol.	No. R.	UB	LB	%Gap	Time (s)
	297	2	565	4	565	-	4	565.0000	563.2845	0.30%	1352
	198	3	582	6	-	578	5	578.0000	560.4578	3.03%	7200
	149	4	606+	4	-	-	-	-	550.8429	-	7200
eilA76_50	775	1	741	6	738	-	6	738.0000	734.9669	0.41%	237
	388	2	738	6	738	-	6	738.0000	717.7974	2.74%	458
	259	3	747	6	-	741	6	741.0000	723.2373	2.40%	7200
	194	4	787+	6	-	-	-	-	712.4578	-	7200
	155	5	784+	6	-	-	-	-	708.2323	-	7200
	130	6	780+	6	-	-	-	-	710.6943	-	7200
eilA76_66	807	1	772	7	768	-	7	768.0000	761.2526	0.88%	2450
	404	2	772	7	768	-	7	768.0000	754.4035	1.77%	6178
	269	3	775	7	-	775	7	775.0000	748.5092	3.42%	7200
	202	4	784	8	-	784	8	784.0000	744.8268	5.00%	7200
	162	5	821+	8	-	-	-	-	738.2245	-	7200
	135	6	800+	7	-	-	-	-	728.2736	-	7200
	116	7	793+	8	-	-	-	-	737.2219	-	7200
eilA76_80	821	1	790	9	-	781	8	781.0000	744.4484	4.68%	7200
	411	2	784	8	-	781	8	781.0000	743.2255	4.84%	7200
	274	3	786	8	-	784	8	784.0000	733.2294	6.48%	7200
	206	4	790	9	-	787	8	787.0000	737.2137	6.33%	7200
	165	5	792+	9	-	-	-	-	733.5592	-	7200
	137	6	811+	9	-	-	-	-	723.2520	-	7200
	118	7	816+	8	-	-	-	-	725.4333	-	7200
	103	8	834+	8	-	-	-	-	720.2287	-	7200
eilA101_50	869	1	835	5	827	-	5	827.0000	825.8372	0.14%	6143
	435	2	854	5	-	842	5	842.0000	816.2896	3.05%	7200
	290	3	864+	5	-	-	-	-	804.0097	-	7200
	218	4	870+	5	-	-	-	-	813.1355	-	7200
	174	5	863+	5	-	-	-	-	807.4466	-	7200
eilA101_66	889	1	858	6	846	-	6	846.0000	842.6713	0.39%	230
	445	2	852	6	846	-	6	850.0000	843.7442	0.27%	6213
	297	3	857	6	846	-	6	846.0000	838.9900	0.83%	6544
	223	4	881+	6	-	-	-	-	810.2531	-	7200
	178	5	874+	6	-	-	-	-	831.4405	-	7200
	149	6	907+	7	-	-	-	-	818.6633	-	7200
eilA101_80	902	1	872	7	-	859	7	859.0000	834.4338	2.86%	7200
	451	2	861	7	-	858	7	858.0000	833.6667	2.84%	7200
	301	3	864	7	-	864	7	864.0000	832.2288	3.68%	7200
	226	4	903+	7	-	-	-	-	831.0961	-	7200
	181	5	886+	7	-	-	-	-	829.0228	-	7200
	151	6	891+	7	-	-	-	-	805.3378	-	7200
	129	7	905+	7	-	-	-	-	826.1179	-	7200

Table 6.6: Detailed results of the CSMH algorithm solutions for the Data *set-1* (T_2)

Name	T_2	Tnb	CSMH Algorithm								
			VNS Sol.	No. R.	Optimal Sol.	Incum. Sol.	No. R.	UB	LB	%Gap	Time (s)
eil22_50	408	1	371	3	371	-	3	371.0000	354.5515	4.43%	2
	204	2	375	3	375	-	3	375.0000	372.2941	0.72%	4
	137	3	385+	3	378	-	3	378.0000	367.7167	2.72%	4
eil22_66	403	1	366	3	366	-	3	366.0000	343.1949	6.23%	3
	201	2	387+	4	382	-	4	382.0000	371.0000	2.88%	7
	134	3	380	3	366	-	3	366.0000	360.6417	1.46%	3
eil22_80	413	1	375	3	375	-	3	375.0000	365.3228	2.58%	4
	206	2	386	4	378	-	4	378.0000	371.6156	1.69%	23
	138	3	382	4	381	-	3	381.0000	372.9394	2.12%	24
eil23_50	745	1	677	3	677	-	3	677.0000	640.4404	8.12%	2
	372	2	693+	4	689	-	3	689.0000	677.0000	1.74%	3
	248	3	716	3	716	-	3	716.0000	704.2018	1.65%	4
eil23_66	704	1	640	3	640	-	3	640.0000	612.5018	4.30%	2
	352	2	640	3	640	-	3	640.0000	624.9952	2.34%	4
	235	3	696	3	694	-	3	694.0000	637.6332	5.12%	3671
eil23_80	685	1	623	2	623	-	2	623.0000	599.1210	3.83%	2
	343	2	631	2	631	-	2	631.0000	622.6453	1.32%	3
eil30_50	551	1	501	2	501	-	2	501.0000	467.5271	6.58%	3
	276	2	501	2	501	-	2	501.0000	489.5667	2.28%	4
eil30_66	591	1	537	3	537	-	3	537.0000	520.5895	3.06%	6
	296	2	569+	3	552	-	3	552.0000	548.3510	0.66%	20
	197	3	543	3	538	-	3	538.0000	526.5343	2.13%	25
eil30_80	565	1	514	3	514	-	3	514.0000	495.0307	3.69%	8
	283	2	546+	3	535	-	3	535.0000	514.6325	3.81%	6452
	188	3	527	3	518	-	3	518.0000	514.6487	0.65%	152
eil33_50	812	1	738	3	738	-	3	738.0000	738.0000	0.00%	4
	406	2	769	3	741	-	3	741.0000	737.5128	0.47%	7
	271	3	799+	3	-	-	-	-	658.3292	-	7200
eil33_66	825	1	750	3	750	-	3	750.0000	721.9751	3.74%	4
	413	2	767	3	767	-	3	767.0000	763.7783	0.42%	44
	275	3	786+	3	-	-	-	-	762.8841	-	7200
eil33_80	810	1	736	3	736	-	3	736.0000	727.3115	1.18%	7
	405	2	759	3	756	-	3	756.0000	754.7114	0.17%	1144
	270	3	768+	3	-	-	-	-	725.7577	-	7200
eil51_50	615	1	559	3	559	-	3	559.0000	553.4257	1.00%	12
	308	2	560	4	560	-	4	560.0000	557.3371	0.48%	90
	205	3	572	3	564	-	4	564.0000	562.5770	0.25%	595
eil51_66	603	1	548	4	548	-	4	548.0000	542.9184	0.93%	14
	302	2	548	4	548	-	4	548.0000	544.2247	0.69%	35
	201	3	574	5	-	772	5	772.0000	546.3256	4.49%	7200
	151	4	585+	5	-	-	-	-	535.1061	-	7200

Name	T_2	Tnb	CSMH Algorithm								
			VNS Sol.	No. R.	Optimal Sol.	Incum. Sol.	No. R.	UB	LB	%Gap	Time (s)
eil51_80	622	1	565	4	565	-	4	565.0000	561.8438	0.56%	72
	311	2	565	4	565	-	4	565.0000	562.5858	0.43%	208
	208	3	590	5	-	578	5	578.0000	565.4189	2.18%	7200
	156	4	579	5	-	579	5	579.0000	555.9473	3.98%	7200
eilA76_50	812	1	748	6	738	-	6	738.0000	735.4884	0.34%	278
	406	2	741	6	738	-	6	738.0000	736.4473	0.21%	940
	271	3	741	6	-	741	6	741.0000	720.5745	2.76%	7200
	203	4	790+	6	-	-	-	-	716.3202	-	7200
	163	5	778+	6	-	-	-	-	708.6608	-	7200
	136	6	766+	6	-	-	-	-	719.9978	-	7200
eilA76_66	845	1	772	7	768	-	7	768.0000	761.2766	0.88%	2412
	423	2	772	7	768	-	7	768.0000	754.4035	1.77%	5345
	282	3	772	7	-	772	7	772.0000	740.8410	4.04%	7200
	212	4	776	8	-	769	7	769.0000	739.0600	3.85%	7200
	169	5	790+	8	-	-	-	-	738.4467	-	7200
	141	6	795+	7	-	-	-	-	740.4613	-	7200
	121	7	779+	8	-	-	-	-	733.8843	-	7200
eilA76_80	860	1	788	8	-	781	8	781.0000	755.6674	3.24%	7200
	430	2	786	8	-	781	8	781.0000	757.3963	3.02%	7200
	287	3	784	8	-	783	8	783.0000	737.4536	5.82%	7200
	215	4	788	8	-	783	8	783.0000	738.0464	5.74%	7200
	172	5	784	8	-	783	8	783.0000	736.3244	5.96%	7200
	144	6	802+	9	-	-	-	-	731.1909	-	7200
	123	7	817+	9	-	-	-	-	722.2782	-	7200
	108	8	843+	8	-	-	-	-	733.8520	-	7200
eilA101_50	910	1	856	5	827	-	5	827.0000	825.6868	0.16%	2209
	455	2	833	5	-	833	5	833.0000	812.7952	2.43%	7200
	304	3	855	5	-	848	5	848.0000	802.2780	6.17%	7200
	228	4	847+	5	-	-	-	-	803.7431	-	7200
	182	5	851+	5	-	-	-	-	782.9959	-	7200
eilA101_66	931	1	867	6	846	-	6	846.0000	843.5580	0.29%	300
	466	2	853	6	846	-	6	846.0000	843.7442	0.27%	5364
	311	3	846	6	846	-	6	846.0000	840.2255	0.68%	7200
	233	4	868	6	-	868	6	868.0000	833.8972	3.93%	7200
	187	5	862+	6	-	-	-	-	815.6880	-	7200
	156	6	904+	6	-	-	-	-	837.2273	-	7200
eilA101_80	945	1	864	7	-	858	7	858.0000	836.8852	2.46%	7200
	473	2	861	7	-	858	7	858.0000	826.0394	3.73%	7200
	315	3	865	7	-	865	7	865.0000	832.3404	3.78%	7200
	237	4	863	7	-	863	7	863.0000	833.7552	3.39%	7200
	189	5	889+	7	-	-	-	-	815.7701	-	7200
	158	6	903+	7	-	-	-	-	814.2284	-	7200
	135	7	903+	7	-	-	-	-	816.6678	-	7200

Table 6.7: Detailed comparison results of the CSMH vs CPLEX for the Data *set-1* (T_1)

Name	T_1	Tnb	CPLEX			CSMH Algorithm		
			Optimal Sol.	No. R.	Time (s)	Sol.	No. R.	Time (s)
eil22_50	390	1	371	3	1	371 *	3	2
	195	2	378	3	1	378 *	3	3
	130	3	x	x	x	x	x	x
eil22_66	385	1	366	3	1	366 *	3	3
	193	2	382	4	3	382 *	4	7
	129	3	x	x	x	x	x	x
eil22_80	394	1	375	3	2	375 *	3	4
	197	2	378	4	2	378 *	4	10
	132	3	381	3	27	381 *	3	105
eil23_50	711	1	677	3	1	677 *	3	3
	355	2	698	3	2	698 *	3	11
	237	3	x	x	x	x	x	x
eil23_66	672	1	640	3	1	640 *	3	3
	336	2	640	3	1	640 *	3	3
	224	3	x	x	x	x	x	x
eil23_80	654	1	623	2	1	623 *	2	2
	327	2	634	2	2	634 *	2	3
eil30_50	526	1	501	2	1	501 *	2	6
	264	2	x	x	x	x	x	x
eil30_66	564	1	537	3	3	537 *	3	7
	282	2	552	3	6116	552 *	3	2302
	188	3	-	-	7200	-	-	7200
eil30_80	540	1	514	3	12	514 *	3	6
	270	2	-	-	7200	**535** *	3	6172
	180	3	-	-	7200	**518** *	3	8
eil33_50	775	1	738	3	1	738 *	3	5
	388	2	-	-	7200	-	-	7200
	258	3	-	-	7200	-	-	7200
eil33_66	788	1	750	3	2	750 *	3	4
	394	2	772	3	1219	772 *	3	93
	263	3	-	-	7200	-	-	7200
eil33_80	773	1	736	3	121	736 *	3	9
	387	2	-	-	7200	**756** *	3	1087
	258	3	-	-	7200	-	-	7200
eil51_50	587	1	559	3	10	559 *	3	14
	294	2	-	-	7200	**562** *	4	108
	196	3	-	-	7200	-	-	7200
eil51_66	576	1	548	4	22	548 *	4	41
	288	2	-	-	7200	**552** *	4	171
	192	3	-	-	7200	-	-	7200
	144	4	-	-	7200	-	-	7200
eil51_80	594	1	565	4	4553	565 *	4	159
	297	2	-	-	7200	**565** *	4	1352
	198	3	-	-	7200	**578** ^	-	7200
	149	4	-	-	7200	-	-	7200
eilA76_50	775	1	-	-	7200	**738** *	6	237
	388	2	-	-	7200	**738** *	6	458
	259	3	-	-	7200	**741** ^	6	7200
	194	4	-	-	7202	-	-	7200
	155	5	-	-	7200	-	-	7200
	130	6	-	-	7200	-	-	7200
eilA76_66	807	1	-	-	7200	**768** *	7	2450

Name	T_1	Tnb	CPLEX			CSMH Algorithm		
			Optimal Sol.	No. R.	Time (s)	Sol.	No. R.	Time (s)
	404	2	-	-	7200	**768** *	7	6178
	269	3	-	-	7200	**775** ^	7	7200
	202	4	-	-	7200	**784** ^	8	7200
	162	5	-	-	7200	-	-	7200
	135	6	-	-	7200	-	-	7200
	116	7	-	-	7200	**781** ^	-	7200
eilA76_80	821	1	-	-	7200	**781** ^	8	7200
	411	2	-	-	7200	**784** ^	8	7200
	274	3	-	-	7200	**787** ^	8	7200
	206	4	-	-	7200	-	8	7200
	165	5	-	-	7200	-	-	7200
	137	6	-	-	7200	-	-	7200
	118	7	-	-	7200	-	-	7200
	103	8	-	-	7200	-	-	7200
eilA101_50	869	1	-	-	7200	**827** *	5	6143
	435	2	-	-	7200	**842** ^	5	7200
	290	3	-	-	7200	-	-	7200
	218	4	-	-	7200	-	-	7200
	174	5	-	-	7200	-	-	7200
eilA101_66	889	1	-	-	7200	**846** *	6	230
	445	2	-	-	7200	**846** *	6	6213
	297	3	-	-	7200	**846** *	6	6544
	223	4	-	-	7200	-	-	7200
	178	5	-	-	7200	-	-	7200
	149	6	-	-	7200	-	-	7200
eilA101_80	902	1	-	-	7200	**859** ^	7	7200
	451	2	-	-	7200	**858** ^	7	7200
	301	3	-	-	7200	**864** ^	7	7200
	226	4	-	-	7200	-	-	7200
	181	5	-	-	7200	-	-	7200
	151	6	-	-	7200	-	-	7200
	129	7	-	-	7200	-	-	7200

Table 6.8: Detailed comparison results of the CSMH vs CPLEX for the Data *set-1* (T_2)

Name	T_2	Tnb	CPLEX			CSMH Algorithm		
			Optimal Sol.	No. R.	Time (s)	Sol.	No. R.	Time (s)
eil22_50	408	1	371	3	1	371 *	3	2
	204	2	375	3	2	375 *	3	4
	137	3	378	3	1	378 *	3	4
eil22_66	403	1	366	3	1	366 *	3	3
	201	2	382	4	2	382 *	4	7
	134	3	366	3	1	366 *	3	3
eil22_80	413	1	375	3	3	375 *	3	4
	206	2	378	4	9	378 *	4	23
	138	3	381	3	24	381 *	3	24
eil23_50	745	1	677	3	1	677 *	3	2
	372	2	689	3	2	689 *	3	3
	248	3	716	3	2	716 *	3	4
eil23_66	704	1	640	3	1	640 *	3	2
	352	2	640	3	1	640 *	3	4
	235	3	-	-	7200	**694** ^	3	3671
eil23_80	685	1	623	2	1	623 *	2	2
	343	2	631	2	1	631 *	2	3
eil30_50	551	1	501	2	1	501 *	2	3
	276	2	501	2	1	501 *	2	4
eil30_66	591	1	537	3	3	537 *	3	6
	296	2	552	3	3451	552 *	3	20
	197	3	538	3	2	538 *	3	25
eil30_80	565	1	514	3	11	514 *	3	8
	283	2	535	3	5519	535 *	3	6452
	188	3	518	3	1426	518 *	3	152
eil33_50	812	1	738	3	1	738 *	3	4
	406	2	741	3	2	741 *	3	7
	271	3	803 ^	-	7200	-	-	7200
eil33_66	825	1	750	3	12	750 *	3	4
	413	2	767	3	109	767 *	3	44
	275	3	-	-	7200	-	-	7200
eil33_80	810	1	736	3	136	736 *	3	7
	405	2	-	-	7200	**756** *	3	1144
	270	3	-	-	7200	-	-	7200
eil51_50	615	1	559	3	11	559 *	3	12
	308	2	560	4	67	560 *	4	90
	205	3	564	4	67	564 *	4	595
eil51_66	603	1	548	4	12	548 *	4	14
	302	2	548	4	56	548 *	4	35
	201	3	-	-	7200	**772** *	5	7200
	151	4	-	-	7200	-	-	7200
eil51_80	622	1	565	4	78	565 *	4	72
	311	2	-	-	7200	**565** *	4	208
	208	3	-	-	7200	**578** ^	5	7200
	156	4	-	-	7200	**579** ^	5	7200
eilA76_50	812	1	-	-	7200	**738** *	6	278
	406	2	-	-	7200	**738** *	6	940

Name	T_2	Tnb	CPLEX			CSMH Algorithm		
			Optimal Sol.	No. R.	Time (s)	Sol.	No. R.	Time (s)
	271	3	-	-	7201	**741** ^	6	7200
	203	4	-	-	7202	-	-	7200
	163	5	-	-	7200	-	-	7200
	136	6	-	-	7200	-	-	7200
eilA76_66	845	1	-	-	7200	**768** *	7	2412
	423	2	-	-	7200	**768** *	7	5345
	282	3	-	-	7200	**772** ^	7	7200
	212	4	-	-	7200	**769** ^	7	7200
	169	5	-	-	7200	-	-	7200
	141	6	-	-	7200	-	-	7200
	121	7	-	-	7200	-	-	7200
eilA76_80	860	1	-	-	7200	**781** ^	8	7200
	430	2	-	-	7200	**781** ^	8	7200
	287	3	-	-	7200	**783** ^	8	7200
	215	4	-	-	7200	**783** ^	8	7200
	172	5	-	-	7200	**783** ^	8	7200
	144	6	-	-	7200	-	-	7200
	123	7	-	-	7200	-	-	7200
	108	8	-	-	7200	-	-	7200
eilA101_50	910	1	-	-	7200	**827** *	5	2209
	455	2	-	-	7200	**833** ^	5	7200
	304	3	-	-	7200	**848** ^	5	7200
	228	4	-	-	7200	-	-	7200
	182	5	-	-	7200	-	-	7200
eilA101_66	931	1	846	6	268	846 *	6	300
	466	2	-	-	7200	**846** *	6	5364
	311	3	-	-	7200	**846** *	6	7200
	233	4	-	-	7200	**868** ^	6	7200
	187	5	-	-	7200	-	-	7200
	156	6	-	-	7200	-	-	7200
eilA101_80	945	1	-	-	7200	**858** ^	7	7200
	473	2	-	-	7200	**858** ^	7	7200
	315	3	-	-	7200	**865** ^	7	7200
	237	4	-	-	7200	**863** ^	7	7200
	189	5	-	-	7200	-	-	7200
	158	6	-	-	7200	-	-	7200
	135	7	-	-	7200	-	-	7200

Table 6.9: Comparison of the lower bounds produced by CPLEX and CSMH for T_1 and T_2

T_1					T_2				
Optimal Sol.	**CPLEX**		**CSMS**		**Optimal Sol.**	**CPLEX**		**CSMH**	
	LB	**%Gap**	**LB**	**%Gap**		**LB**	**%Gap**	**LB**	**%Gap**
371	367.5294	0.94%	354.5515	4.43%	371	370.6087	0.11%	354.5515	4.43%
378	368.0119	2.64%	376.9945	0.27%	375	374.0333	0.26%	372.2941	0.72%
366	364.9640	0.28%	343.1949	6.23%	378	364.4367	3.59%	367.7167	2.72%
382	366.0000	4.19%	375.5642	1.68%	366	364.7095	0.35%	343.1949	6.23%
375	362.1650	3.42%	364.8797	2.70%	382	366.0000	4.19%	371.0000	2.88%
378	364.9665	3.45%	367.1494	2.87%	366	366.0000	0.00%	360.6417	1.46%
381	369.0667	3.13%	374.0939	1.81%	375	358.9261	4.29%	365.3228	2.58%
677	677.0000	0.00%	640.4404	5.40%	378	362.2288	4.17%	371.6156	1.69%
698	671.8600	3.74%	677.2488	2.97%	381	364.9274	4.22%	372.9394	2.12%
640	633.1636	1.07%	612.5018	4.30%	677	677.0000	0.00%	640.4404	5.40%
640	635.5000	0.70%	629.2119	1.69%	689	680.0000	1.31%	677.0000	1.74%
623	618.0870	0.79%	599.1210	3.83%	716	682.1268	4.73%	704.2018	1.65%
634	613.3380	3.26%	620.8261	2.08%	640	640.0000	0.00%	612.5018	4.30%
501	500.3902	0.12%	501.0000	0.00%	640	631.5000	1.33%	624.9952	2.34%
537	511.3725	4.77%	537.0000	0.00%	694	662.4548	4.55%	637.6332	8.12%
552	537.0000	2.72%	538.0000	2.54%	623	617.8667	0.82%	599.1210	3.83%
514	474.9762	7.59%	514.0000	0.00%	631	614.5388	2.61%	622.6453	1.32%
535	459.3289	14.14%	465.5482	12.98%	501	500.3902	0.12%	467.5271	6.58%
518	460.3190	11.14%	510.8803	1.37%	501	501.0000	0.00%	489.5667	2.28%
738	738.0000	0.00%	738.0000	0.00%	537	510.3183	4.97%	520.5895	3.06%
750	732.7999	2.29%	723.3959	3.55%	552	538.0355	2.53%	548.3510	0.66%
772	757.8079	1.84%	768.0827	0.51%	538	534.6250	0.63%	526.5343	2.13%
736	733.8901	0.29%	730.2669	0.78%	514	482.8207	6.07%	495.0307	3.69%
756	720.3275	4.72%	754.4379	0.21%	535	468.6333	12.40%	514.6325	3.81%
559	552.1063	1.23%	554.6452	0.78%	518	500.1891	3.44%	514.6487	0.65%
562	550.1111	2.12%	558.9278	0.55%	738	738.0000	0.00%	738.0000	0.00%
548	537.7475	1.87%	547.0163	0.18%	741	736.2820	0.64%	737.5128	0.47%
552	546.1393	1.06%	550.6893	0.24%	750	734.5884	2.05%	721.9751	3.74%
565	553.1885	2.09%	563.1379	0.33%	767	764.4997	0.33%	763.7783	0.42%
565	555.5726	1.67%	563.2845	0.30%	736	716.7393	2.62%	727.3115	1.18%
738	708.2119	4.04%	734.9669	0.41%	756	723.4224	4.31%	754.7114	0.17%
738	721.9806	2.17%	717.7974	2.74%	559	553.6224	0.96%	553.4257	1.00%
768	738.1007	3.89%	761.2526	0.88%	560	550.4380	1.71%	557.3371	0.48%
768	737.9937	3.91%	754.4035	1.77%	564	559.6480	0.77%	562.5770	0.25%
827	799.5710	3.32%	825.8372	0.14%	548	541.1877	1.24%	542.9184	0.93%
846	829.5004	1.95%	842.6713	0.39%	548	546.9363	0.19%	544.2247	0.69%
846	837.3865	1.02%	843.7442	0.27%	565	562.5255	0.44%	561.8438	0.56%
846	826.1638	2.34%	838.9900	0.83%	565	554.3046	1.89%	562.5858	0.43%
-	-	-	-	-	738	710.0593	3.79%	735.4884	0.34%
-	-	-	-	-	738	722.0668	2.16%	736.4473	0.21%
-	-	-	-	-	768	734.9762	4.30%	761.2766	0.88%
-	-	-	-	-	768	741.8414	3.41%	754.4035	1.77%
-	-	-	-	-	827	801.4182	3.09%	825.6868	0.16%
-	-	-	-	-	846	840.8321	0.61%	843.5580	0.29%
-	-	-	-	-	846	822.6394	2.76%	843.7442	0.27%
-	-	-	-	-	846	831.4000	1.73%	840.2255	0.68%
Grand average % gap		**2.89%**		**1.90%**			**2.30%**		**1.99%**

Table 6.10: Detailed comparison results of the CSMH vs *Two-Level VNS* for the Data *set-1* (T_1)

Name	T_1	Tnb	*Two-Level VNS*			CSMH Algorithm		
			Sol.	No. R.	Time (s)	Sol.	No. R.	Time (s)
eil22_50	390	1	371	3	2	371 *	3	2
	195	2	378	3	3	378 *	3	3
	130	3	**390** +	4	3	392 +	3	3
eil22_66	385	1	366	3	5	366 *	3	3
	193	2	396 +	4	4	**382** *	4	7
	129	3	**370** +	3	3	375 +	3	5
eil22_80	394	1	375	3	4	375 *	3	4
	197	2	378	4	5	378 *	4	10
	132	3	381	3	3	381 *	3	105
eil23_50	711	1	677	3	3	677 *	3	3
	355	2	711 +	3	2	**698** *	3	11
	237	3	725 +	3	5	725 +	3	8
eil23_66	672	1	640	3	4	640 *	3	3
	336	2	640	3	4	640 *	3	3
	224	3	702 +	3	3	702 +	3	4
eil23_80	654	1	623	2	4	623 *	2	2
	327	2	634	2	4	634 *	2	3
eil30_50	526	1	501	2	4	501 *	2	6
	264	2	507 +	2	3	507 +	2	8
eil30_66	564	1	537	3	6	537 *	3	7
	282	2	565 +	3	6	**552** *	3	2302
	188	3	541 +	3	5	541 +	3	7200
eil30_80	540	1	514	3	6	514 *	3	6
	270	2	540 +	3	7	**535** *	3	6172
	180	3	518	3	6	518 *	3	8
eil33_50	775	1	738	3	5	738 *	3	5
	388	2	766 +	3	6	766 +	3	7200
	258	3	822 +	3	4	822 +	3	7200
eil33_66	788	1	750	3	9	750 *	3	4
	394	2	772	3	8	772 *	3	93
	263	3	792 +	3	5	792 +	3	7200
eil33_80	773	1	736	3	6	736 *	3	9
	387	2	756	3	9	756 *	3	1087
	258	3	766 +	3	5	766 +	3	7200
eil51_50	587	1	559	3	9	559 *	3	14
	294	2	568	3	11	**562** *	4	108
	196	3	**574** +	3	10	605 +	4	7200
eil51_66	576	1	548	4	10	548 *	4	41
	288	2	552	4	11	552 *	4	171
	192	3	577 +	4	11	577 +	4	7200
	144	4	583 +	4	10	583 +	4	7200
eil51_80	594	1	565	4	13	565 *	4	159
	297	2	565	4	12	565 *	4	1352
	198	3	582	5	11	**578** ^	5	7200

Name	T_1	Tnb	*Two-Level VNS*			CSMH Algorithm		
			Sol.	No. R.	Time (s)	Sol.	No. R.	Time (s)
	149	4	**592** +	5	11	606 +	4	7200
eilA76_50	775	1	738	6	21	738 *	6	237
	388	2	738	6	23	738 *	6	458
	259	3	741	6	22	741 ^	6	7200
	194	4	787 +	6	23	787 +	6	7200
	155	5	**783** +	6	22	784 +	6	7200
	130	6	**779** +	6	22	780 +	6	7200
eilA76_66	807	1	768	7	23	768 *	7	2450
	404	2	768	7	21	768 *	7	6178
	269	3	**772**	7	23	775 ^	7	7200
	202	4	784	8	21	784 ^	8	7200
	162	5	**817** +	8	23	821 +	8	7200
	135	6	**788** +	8	23	800 +	7	7200
	116	7	793 +	8	22	793 +	8	7200
eilA76_80	821	1	781	8	23	781 ^	8	7200
	411	2	781	8	23	781 ^	8	7200
	274	3	784	8	22	784 ^	8	7200
	206	4	787	8	23	787 ^	8	7200
	165	5	**788** +	8	23	792 +	9	7200
	137	6	**807** +	9	24	811 +	9	7200
	118	7	816 +	8	23	816 +	8	7200
	103	8	834 +	8	23	834 +	8	7200
eilA101_50	869	1	827	5	39	827 *	5	6143
	435	2	**835**	5	42	842 ^	5	7200
	290	3	**849** +	5	42	864 +	5	7200
	218	4	**855** +	5	42	870 +	5	7200
	174	5	863 +	5	41	863 +	5	7200
eilA101_66	889	1	846	6	43	846 *	6	230
	445	2	846	6	41	846 *	6	6213
	297	3	846	6	42	846 *	6	6544
	223	4	**875** +	6	43	881 +	6	7200
	178	5	874 +	6	43	874 +	6	7200
	149	6	**906** +	7	42	907 +	7	7200
eilA101_80	902	1	859	7	42	859 ^	7	7200
	451	2	859	7	45	**858** ^	7	7200
	301	3	**859**	7	45	864 ^	7	7200
	226	4	**775** +	7	42	903 +	7	7200
	181	5	886 +	7	43	886 +	7	7200
	151	6	**886** +	7	42	891 +	7	7200
	129	7	**905** +	7	44	905 +	7	7200

Table 6.11: Detailed comparison results of the CSMH vs *Two-Level VNS* for the Data *set-1* (T_2)

Name	T_2	Tnb	*Two-Level VNS*			CSMH Algorithm		
			Sol.	No. R.	Time (s)	Sol.	No. R.	Time (s)
eil22_50	408	1	371	3	3	371 *	3	2
	204	2	375	3	4	375 *	3	4
	137	3	382 +	3	3	**378** *	3	4
eil22_66	403	1	366	3	2	366 *	3	3
	201	2	385 +	4	3	**382** *	4	7
	134	3	367 +	3	2	**366** *	3	3
eil22_80	413	1	375	3	3	375 *	3	4
	206	2	378	4	3	378 *	4	23
	138	3	381	3	4	381 *	3	24
eil23_50	745	1	677	3	4	677 *	3	2
	372	2	693 +	3	5	**689** *	3	3
	248	3	716	3	4	716 *	3	4
eil23_66	704	1	640	3	4	640 *	3	2
	352	2	640	3	4	640 *	3	4
	235	3	696	3	5	**694** *	3	3671
eil23_80	685	1	623	2	4	623 *	2	2
	343	2	631	2	4	631 *	2	3
eil30_50	551	1	501	2	4	501 *	2	3
	276	2	501	2	3	501 *	2	4
eil30_66	591	1	537	3	6	537 *	3	6
	296	2	552 +	3	7	**552** *	3	20
	197	3	538	3	5	538 *	3	25
eil30_80	565	1	514	3	6	514 *	3	8
	283	2	535	3	7	535 *	3	6452
	188	3	518	3	5	518 *	3	152
eil33_50	812	1	738	3	4	738 *	3	4
	406	2	769	3	8	**741** *	3	7
	271	3	799 +	3	4	799 +	3	7200
eil33_66	825	1	750	3	5	750 *	3	4
	413	2	767	3	9	767 *	3	44
	275	3	**775** +	3	5	786 +	3	7200
eil33_80	810	1	736	3	8	736 *	3	7
	405	2	756	3	6	756 *	3	1144
	270	3	**754** +	3	6	768 +	3	7200
eil51_50	615	1	559	3	10	559 *	3	12
	308	2	560	4	9	560 *	4	90
	205	3	568	3	11	**564** *	4	595
eil51_66	603	1	548	4	10	548 *	4	14
	302	2	548	4	11	548 *	4	35
	201	3	774	4	10	**772** ^	5	7200
	151	4	**570** +	4	11	585 +	5	7200
eil51_80	622	1	565	4	11	565 *	4	72
	311	2	565	4	10	565 *	4	208
	208	3	587	4	10	**578** ^	5	7200
	156	4	579	5	10	579 ^	5	7200

Name	T_2	Tnb	*Two-Level VNS*			CSMH Algorithm		
			Sol.	**No. R.**	**Time (s)**	**Sol.**	**No. R.**	**Time (s)**
eilA76_50	812	1	738	6	21	738 *	6	278
	406	2	738	6	22	738 *	6	940
	271	3	**738**	6	22	741 ^	6	7200
	203	4	**767** +	6	22	790 +	6	7200
	163	5	**775** +	6	24	778 +	6	7200
	136	6	**762** +	6	21	766 +	6	7200
eilA76_66	845	1	768	7	22	768 *	7	2412
	423	2	768	7	21	768 *	7	5345
	282	3	772	7	22	772 ^	7	7200
	212	4	769	7	22	769 ^	7	7200
	169	5	790 +	8	23	790 +	8	7200
	141	6	**783** +	8	22	795 +	7	7200
	121	7	**777** +	8	22	779 +	8	7200
eilA76_80	860	1	781	8	23	781 ^	8	7200
	430	2	781	8	22	781 ^	8	7200
	287	3	783	8	23	783 ^	8	7200
	215	4	783	8	22	783 ^	8	7200
	172	5	783	8	22	783 ^	8	7200
	144	6	**796** +	8	23	802 +	9	7200
	123	7	**805** +	8	23	817 +	9	7200
	108	8	**841** +	8	22	843 +	8	7200
eilA101_50	910	1	827	5	41	827 *	5	2209
	455	2	**827**	5	41	833 ^	5	7200
	304	3	855	5	43	**848** ^	5	7200
	228	4	847 +	5	42	847 +	5	7200
	182	5	851 +	5	42	851 +	5	7200
eilA101_66	931	1	846	6	43	846 *	6	300
	466	2	846	6	42	846 *	6	5364
	311	3	846	6	43	846 *	6	7200
	233	4	868	6	42	868 ^	6	7200
	187	5	862 +	6	43	862 +	6	7200
	156	6	904 +	6	44	904 +	6	7200
eilA101_80	945	1	859	7	42	**858** ^	7	7200
	473	2	859	7	43	**858** ^	7	7200
	315	3	**859**	7	46	865 ^	7	7200
	237	4	**859**	7	43	863 ^	7	7200
	189	5	**878** +	7	44	889 +	7	7200
	158	6	**883** +	7	45	903 +	7	7200
	135	7	**883** +	7	42	903 +	7	7200

6.4. Summary

In this chapter we have studied a new class of hybrid methodologies called *mat-heuristics* that combines mathematical programming techniques with heuristic methods to solve CO problems. We have developed a hybrid collaborative sequential *mat-heuristic* approach called the CSMH to solve the MT-VRPB. The exact method approach presented in Chapter 4 is hybridised with the *Two-Level VNS* algorithm of Chapter 5. The *Two-Level VNS* used three phases, i.e., initial solution by a modified *sweep-first-assignment-second* approach, improved solution by VNS, and packed solution by the BPP. Here a fourth phase, i.e., a mathematical model is incorporated in the *Two-Level VNS* algorithm to find optimal/better solution for the MT-VRPB. The overall performance of the CSMH remained very encouraging in terms of the solution quality and the average time taken. Comparing with the methodologies developed in the previous chapters (i.e., CPLEX and the *Two-Level VNS* meta-heuristic), the CSMH produced much better results on almost all fronts. As compared to CPLEX, it produced a higher number of optimal solutions and tighter lower bounds while spending a relatively much lower computation time. Comparing with the *Two-Level VNS* it also produced better quality solutions with a higher number of optimal/incumbent solutions, at the expense of requiring a larger computing time as one may expect.

Chapter 7

Adaptation of the *Two-Level VNS* and Mat-heuristic to the VRPB and the MT-VRP

In this chapter we investigate two special cases of the MT-VRPB namely, the Vehicle Routing Problem with Backhauls (VRPB) and the Multiple Trip Vehicle Routing Problem (MT-VRP). The *Two-Level VNS* and the CSMH algorithms developed for the MT-VRPB in Chapter 5 and Chapter 6 are adapted to solve the VRPB and the MT-VRP separately. The results produced by the *Two-Level VNS* and the CSMH algorithms are compared with the best published solutions of the benchmark instances of these problems from the literature. Our implementations show that the *Two-Level VNS* algorithm is easy to adapt to other variants of the VRP and the mat-heuristic is a powerful algorithm for solving a variety of VRPs.

7.1. The case of the VRPB

The VRPB is already explained along with the literature review in Chapters 2 and 3. In this section we adapt our approaches to solve the VRPB efficiently and to test the methodologies developed in Chapters 4, 5 and 6.

7.1.1. VRPB Formulation

The VRPB formulation is adapted from our MT-VRPB formulation presented in Chapter 4. This is a three-indexed commodity flow formulation. Before this Toth and Vigo (1997) and Mingozzi *at el*. (1999) provided two-indexed ILP formulations for their proposed exact methodologies for the VRPB.

Notations:

Sets

$\{0\}$ the depot (single depot)

L the set of linehaul customers

B the set of backhaul customers

K the set of vehicles

Input Variables

d_{ij} the distance between customers i and j ($i \in \{0\} \cup L \cup B, j \in \{0\} \cup L \cup B$)

q_i the demand of customer i (such that $i \in L$ for a delivery demand and $i \in B$ for a pickup demand)

Other Parameters

C vehicle capacity

Decision Variables

$$x_{ijk} = \begin{cases} 1, \text{if vehicle } k \text{ travels from locaton } i \text{ directly to location } j; \\ 0, \text{otherwise} \end{cases}$$

R_{ij} = is the amount of delivery or pickup on board on arc ij

Minimise $Z = \sum_{i\in\{0\}\cup L\cup B} \sum_{j\in\{0\}\cup L\cup B} \sum_{k\in K} d_{ij} x_{ijk}$ (7.1)

Subject to

$$\sum_{j\in\{0\}\cup L\cup B} \sum_{k\in K} x_{jik} = 1 \qquad i \in L \cup B \qquad (7.2)$$

$$\sum_{j\in\{0\}\cup L\cup B} \sum_{k\in K} x_{ijk} = 1 \qquad i \in L \cup B \qquad (7.3)$$

$$\sum_{j\in\{0\}\cup L\cup B} x_{jik} = \sum_{j\in\{0\}\cup L\cup B} x_{ijk} \qquad i \in L \cup B,\ \forall k \in K \qquad (7.4)$$

$$\sum_{i\in\{0\}\cup L} R_{ij} - q_j = \sum_{i\in\{0\}\cup L\cup B} R_{ji} \qquad j \in L \qquad (7.5)$$

$$\sum_{i\in L\cup B} R_{ij} + q_j = \sum_{i\in\{0\}\cup B} R_{ji} \qquad j \in B \qquad (7.6)$$

$$R_{ij} \le C \sum_{k\in K} x_{ijk} \qquad i \in L \cup B, j \in L \cup B;\ \forall k \in K \qquad (7.7)$$

$$\sum_{j\in L} x_{0jk} = 1 \qquad \forall (k \in K) \qquad (7.8)$$

$$R_{ij} = 0 \qquad i \in L,\ j \in B \cup \{0\} \qquad (7.9)$$

$$x_{ijk} = 0 \qquad i \in B,\ j \in L, k \in K \qquad (7.10)$$

$$x_{0jk} = 0 \qquad j \in B, k \in K \qquad (7.11)$$

$$R_{ij} \ge 0 \qquad i \in \{0\} \cup L \cup B,\ j \in L \cup B \qquad (7.12)$$

$$x_{ijk} = 0,1 \qquad i \in \{0\} \cup L \cup B,\ j \in \{0\} \cup L \cup B,\ k \in K \qquad (7.13)$$

Equation (7.1) illustrates the objective function representing the total distance travelled. Constraints (7.2) and (7.3) ensure that every customer is served exactly once (every customer has an incoming arc and every customer has an outgoing arc). Constraint (7.4)

states that the number of times vehicle k enters into customer i is the same as the number of times it leaves customer i. The vehicle load variation on a route is ensured by Constraints (7.5) and (7.6) for linehaul and backhaul customers respectively. Inequality (7.7) imposes the maximum vehicle capacity constraint. Inequality (7.8) imposes restrictions on every vehicle to be used once. Constraints (7.9) restricts that a load cannot be carried from a linehaul customer to a backhaul customer or to the depot. Constraints (7.10) and (7.11) impose a restriction that a vehicle cannot travel from a backhaul to a linehaul customer and neither can it travel directly from depot to a backhaul customer. Inequality (7.12) sets R_{ij} as a non-negative variable. Finally, in (7.13) the decision variable x_{ijk} is set as zero-one variable.

The validity of the mathematical formulation of the VRPB is checked using the IBM ILOG CPLEX 12.5. Hence it was implemented in CPLEX and it proved valid when tested on some VRPB benchmark instances.

Model variants:

Moreover, the above VRPB formulation can be relaxed from fixed fleet restriction by changing the precedence Constraint (7.8) by replacing $=$ to $\leq$ instead, see (7.14).

$$\sum_{j\in L} x_{0jk} \leq 1 \qquad \forall\ (k \in K) \qquad (7.14)$$

Furthermore, we can allow backhaul only routes by removing Constraint (7.11) and replacing Constraint (7.8) to Constraint (7.15).

$$\sum_{j\in L\cup B} x_{0jk} \leq 1 \qquad \forall\ (k \in K) \qquad (7.15)$$

7.1.2. The *Two-Level VNS* Algorithm for the VRPB

In Chapter 5, the VRPB was solved with free fleet (without imposing fixed fleet constraint) for the MT-VRPB. As explained in Chapter 3 the classical VRPB is studied in the literature mainly with the fixed fleet constraint. Since the MT-VRPB required multi-trip aspect this constraint was relaxed. However, here we would like to test if the *Two-Level VNS* algorithm is viable for the VRPB with fixed fleet utilization. Details of the implementation are provided in the following subsections.

The *Two-Level VNS* algorithm is already elaborated in Section 5.2 in detail; for ease, here we present the algorithm and its components with any implementation differences as shown in Figure 7.1.

Phase I: **Initial solution - *sweep-first-assignment-second* approach**

1. Generate LH and BH open-ended routes using the *sweep*
2. Create a distance matrix of end nodes from open-ended routes
3. Solve the assignment problem by calling CPLEX (see Section 5.2 for 1, 2 & 3)
4. Impose fixed fleet utilization steps if required to obtain an initial feasible solution x.

Phase II: Two-Level VNS **Algorithm**

Set: $iter$ = 1 and $iter_{max} = 400$

Repeat the process while $iter \leq iter_{max}$

Start *outer-level*

Let: $LS_k^O =< R_3, R_4, R_5 >$ set of refinement routines for the *outer-level*

Set: $k = 1$

Repeat the process while $k \leq N_{k_{max}}^I$

***a.1*:** Generate a neighbouring solution $x' \in N_k^O(x)$ at random;

***a.2*:** Apply LS_k^O on neighbouring solution x' to improve it

***a.3*:** Assign the resulting solution x' to x'_{best} $[x'_{best} = x']$

a.4*: Start *inner-level uses x'_{best}

Let: $LS_l^I = < \{R_1 \& R_6\}, \{R_2 \& R_6\}, \{R_3 \& R_6\}, \{R_4 \& R_6\}, \{R_5 \& R_6\} >$

Set: $l = 1$

Repeat the process while $l \leq N_{l_{max}}^I$

***a.4*(1):** Generate a neighbouring solution $x'' \in N_l^I(x'_{best})$ at random

***a.4*(2):** Apply LS_l^I [Multi-Layer local search optimiser framework] on the neighbouring solution x''

***a.4*(3):** Assign the resulting solution x'' to x''_{best} $[x''_{best} = x'']$

***a.4*(4):** If $x''_{best} < x'_{best}$ then $x'_{best} = x''_{best}$, set $l = 1$ and got to ***a.4*(1)**

Else set $l = l + 1$ and got to ***a.4*(1)**

a.5: If $x'_{best} < x$ then $x = x'_{best}$; set $k = 1$ and go to ***a.1***

Else set $k = k + 1$ and go to ***a.1***

Figure 7.1: Algorithmic steps of the *Two-Level VNS* for VRPB

Initial solution: (***Phase I***)

The initial solution for the VRPB is obtained by using the *sweep-first-assignment-second* developed for the MT-VRPB in Section 5.2. The sweep phase of the *sweep-first-assignment-second* procedure builds sets of open ended LH and BH routes. The LH/BH route matrix is then balanced by adding dummy LH/BH routes, if needed, containing the depot only, before solving the assignment problem to obtain the optimal matching of the combined LH/BH routes. Note that if the solution is not feasible in terms of the precedence backhauling constraints (explained in Section 3.1) then it can be amended by moving customers among routes before passing it on to the VNS stage of the algorithm. However this situation did not arise in solving the instances of the VRPB

data sets in this thesis. The initial solution obtained at this stage is feasible for the 'free fleet VRPB' but not necessarily feasible for the 'constrained VRPB' with the condition of using a given fixed fleet that must be utilised. Hence, the 'free fleet VRPB' initial solution is scrutinized for the 'constrained VRPB' condition; if the solution is found not to be complying with the fixed fleet condition, then a procedure is used to overcome this difficulty as follows.

***Fixed Fleet utilization procedure*:**

In the cases where, in a problem instance, the number of routes in the matched *sweep-first-assignment-second* solution is less than the given number of vehicles then empty dummy vehicle routes equivalent to the unassigned vehicles are added to the solution with no extra cost at this stage. And in cases where the number of routes in a problem instance solution is greater than the given fleet size then the additional routes (with least number of customers) are eliminated by moving customers from those routes and feasibly best inserted to other routes of the solution. Note that the process of re-locating customers may not be smooth in some cases due to large demands of some customers and the vehicle capacity constraints. In the case where such large customers could not be feasibly inserted into any of the other routes, then a route with the largest unused capacity is selected and some of its customers are moved to other routes before inserting the large customer to this route.

***Neighbourhoods*: (*Phase II*)**

The *Two-Level VNS* algorithm for the VRPB also uses all the six neighbourhoods, described in Section 5.3, in the same order.

***Multi-Layer local search optimiser framework*:**

The multi-layer local search optimiser framework including the local search refinements, described in Section 5.4, are also unchanged for the VRPB in the *Two-Level VNS* algorithm.

BPP: The Bin Packing aspect is not needed here and therefore that phase is made void.

7.1.2.1. Details of the VRPB Computations and the Data sets

The *Two-Level VNS* algorithm and the initial solution design are implemented in C++ programming within the Microsoft Visual Studio Environment and the experiments were executed on a PC with Intel(R) Core(TM) i7-2600 processor, CPU speed 3.40 GHz.

VRPB Data sets:

The computational results are reported for the two commonly used VRPB benchmark data sets. The first data set (referred to as data *set-2* in this study) was initiated in Toth and Vigo (1996, 1997). The second data set (referred to as the data *set-3* in this study) was introduced in Goetschalckx and Jacobs-Blecha (1989) and Toth and Vigo (1996, 1997).

The date *set-2* consisting of 33 instances was generated from 11 classical VRP test problems from the literature. These instances range in size between 21 and 100 customers. Each VRP problem instance was used to generate three VRPB instances, each with a linehaul percentage of 50%, 66% and 80%. For further details about the instances in data *set-2*, see Toth and Vigo (1996, 1997 and 1999) and Wassan (2007).

The data *set-3* consists of total 62 instances ranging in size between 25 and 150 customers with different backhauls percentages of 20%, 33% and 50%. In this data set, a uniform distribution of the vertex coordinates is done; where for the x values [0, 24000] interval is used and interval [0, 32000] is used for the y values. The coordinates [12000, 16000] are used for the depot which is located centrally. For further details about the instances in data *set-3*, see Goetschalckx and Jacobs-Blecha (1989), Toth and Vigo (1996, 1997 and 1999) and Wassan (2007). Note that all these data sets can be downloaded from CLHO.

The *Two-Level VNS* algorithm is run for a fixed number of iterations (i.e., 400 iterations) to test each VRPB problem instance of the data *set-2* and the *set-3*, which was empirically deemed acceptable in terms of solution quality and computational time. The algorithm was tested with different number of iterations on data sets and 400 iterations proved best in terms of solutions quality and computational time.

Glossary for tables:

n= Number of total customers in an instances,

L = Number of linehaul customers,

B = Number of backhaul customers,

v = Fixed fleet,

C = Vehicle capacity.

The RPD (Relative Percentage Deviation) is obtained as follows. RPD = (Heuristic solution – Best known/Best known)*100.

7.1.2.2. *Two-Level VNS* VRPB Results and Analysis

The *Two-Level VNS* algorithm produced very competitive results for both data sets when compared to the best known solutions from the literature, with an overall average relative percentage deviation ARPD (Average Relative Percentage Deviation) of 0.00 and 0.06 from the best known solution for the *set-2* and the *set-3*, respectively.

Comparison of the Two-Level VNS with some recent algorithms:

The performance of our *Two-level VNS* algorithm is compared with the best algorithms from the literature which include RTS-AMP (reactive tabu adaptive memory programming search of Wassan, 2007), MACS (multi-ant colony system of Gajpal and Abad, 2009), RPA (route promise methodology of Zachariadis and Kiranoudis, 2012) and ILS (iterated local search algorithm of Cuervo *et al*., 2014). Table 7.1 shows the information about some of the recent algorithms including the *Two-Level VNS* and their corresponding number of runs used in our comparisons.

It may not be possible to conduct a fair comparison of the algorithms with different number of runs, as the execution times for each run, different machines, etc. may differ. Nevertheless, we shall present results with basic explanation of the variations.

Table 7.1: Processor used and the number of runs for the published algorithms and the proposed *Two-Level VNS*

Algorithm	Processor	Runs
RTS-AMP: Wassan (2007)	50 MHz. Sun Sprac1000	5
MACS: Gajpal and Abad (2009)	2.40 GHz. Intel Xeon	8
RPA: Zachariadis and Kiranoudis (2012)	1.66 GHz. Intel Core 2 duo	10
ILS: Cuervo *et al*. (2014)	2.93 GHz. Intel Core i7	10
Two-Level VNS/CSMH	3.40 GHz. Intel Core i7	5/1

The performance analysis summaries of these algorithms for the data *set-2* and the data *set-3* are provided in Table 7.2 and Table 7.3 respectively. The columns in both tables show the number of best known matched solutions, the average solution cost, the overall average of the relative percentage deviations (RPD) from the best known solutions and the average execution time taken by each algorithm. It can be observed from the average results that the *Two-Level VNS* algorithm is very competitive when compared to the best existing algorithms. For data *set-2*, the *Two-Level VNS* outperformed two of the algorithms in terms of the number of best known solutions found and matched with the ILS (2014). For data *set-3*, it finds very good solutions as compared to RTS-AMP (2007) and MACS (2009); however RPA (2012) and ILS (2014) find the maximum number of best known solutions. The detailed results of the *Two-Level VNS* vs the best known solutions are provided in Table 7.4 and Table 7.5 for the *set-2* and the *set-3*, respectively. We can fairly claim that the proposed *Two-Level VNS* is an efficient and flexible enough performer that competes favourably against the powerful meta-heuristics that were specifically proposed for such particular problem.

Table 7.2: Comparison of the best VRPB algorithms with *Two-Level VNS* (data *set-2*)

Algorithm	Runs	# Best sol. (out of 33)	Avg. best sol. Cost	Avg. RPD	Avg. time (s)
RTS-AMP (2007)	5	21	706.49	0.80	608.11
MACS (2009)	8	28	701.49	0.09	25.65
RPA (2012)	-	-	-	-	-
ILS (2014)	10	33	700.63	0.00	7.35
Two-Level VNS	5	33	700.63	0.00	29.29

Table 7.3: Comparison of the *Two-Level VNS* with the best algorithms (data *set-3*)

Algorithm	Runs	# Best sol. (out of 62)	Avg. best sol. Cost	Avg. RPD	Avg. time (s)
RTS-AMP (2007)	5	40	290981.84	0.11	1835.98
MACS (2009)	8	46	290838.73	0.07	67.57
RPA (2012)	10	62	290576.06	0.00	246.89
ILS (2014)	10	62	290576.22	0.00	22.89
Two-Level VNS	5	51	290796.24	0.06	43.24

Table 7.4: Detailed results of the *Two-Level VNS* vs the Best-known (data *set-2*)

Name	***n***	***L***	***B***	***v***	***C***	**Best Known**	***Two-Level VNS***	**RPD**
eil22_50	21	11	10	3	6000	371	371	0.00
eil22_66	21	14	7	3	6000	366	366	0.00
eil22_80	21	17	4	3	6000	375	375	0.00
eil23_50	22	11	11	2	4500	682	682	0.00
eil23_66	22	15	7	2	4500	649	649	0.00
eil23_80	22	18	4	2	4500	623	623	0.00
eil30_50	29	15	14	2	4500	501	501	0.00
eil30_66	29	20	9	3	4500	537	537	0.00
eil30_80	29	24	5	3	4500	514	514	0.00
eil33_50	32	16	16	3	8000	738	738	0.00
eil33_66	32	22	10	3	8000	750	750	0.00
eil33_80	32	26	6	3	8000	736	736	0.00
eil51_50	50	25	25	3	160	559	559	0.00
eil51_66	50	34	16	4	160	548	548	0.00
eil51_80	50	40	10	4	160	565	565	0.00
eilA76_50	75	37	38	6	140	739	739	0.00
eilA76_60	75	50	25	7	140	768	768	0.00
eilA76_80	75	60	15	8	140	781	781	0.00
eilB76_50	75	37	38	8	100	801	801	0.00
eilB76_66	75	50	25	10	100	873	873	0.00
eilB76_80	75	60	15	12	100	919	919	0.00
eilC76_50	75	37	38	5	180	713	713	0.00
eilC76_66	75	50	25	6	180	734	734	0.00
eilC76_80	75	60	15	7	180	733	733	0.00
eilD76_50	75	37	38	4	220	690	690	0.00
eilD76_66	75	50	25	5	220	715	715	0.00
eilD76_80	75	60	15	6	220	694	694	0.00
eilA101_50	100	50	50	4	200	831	831	0.00
eilA101_66	100	67	33	6	200	846	846	0.00
eilA101_80	100	80	20	6	200	856	856	0.00
eilB101_50	100	50	50	7	112	923	923	0.00
eilB101_66	100	67	33	9	112	983	983	0.00
eilB101_80	100	80	20	11	112	1008	1008	0.00

Table 7.5: Detailed results of the *Two-Level VNS* vs the Best-known (data *set-3*)

Name	*n*	*L*	*B*	*C*	*v*	Best known Solution	*Two-level VNS* Solution	RPD
A1	25	20	5	1550	8	229885.65	229885.65	0.00
A2	25	20	5	2550	5	180119.21	180119.21	0.00
A3	25	20	5	4050	4	163405.38	163405.38	0.00
A4	25	20	5	4050	3	155796.41	155796.41	0.00
B1	30	20	10	1600	7	239080.16	239080.16	0.00
B2	30	20	10	2600	5	198047.77	198047.77	0.00
B3	30	20	10	4000	3	169372.29	169372.29	0.00
C1	40	20	20	1800	7	250556.77	250556.77	0.00
C2	40	20	20	2600	5	215020.23	215020.23	0.00
C3	40	20	20	4150	5	199345.96	199345.96	0.00
C4	40	20	20	4150	4	195366.63	195366.63	0.00
D1	38	30	8	1700	12	322530.13	322530.13	0.00
D2	38	30	8	1700	11	316708.86	316708.86	0.00
D3	38	30	8	2750	7	239478.63	239478.63	0.00
D4	38	30	8	4075	5	205831.94	205831.94	0.00
E1	45	30	15	2650	7	238879.58	238879.58	0.00
E2	45	30	15	4300	4	212263.11	212263.11	0.00
E3	45	30	15	5225	4	206659.17	206659.17	0.00
F1	60	30	30	3000	6	263173.96	263173.96	0.00
F2	60	30	30	3000	7	265214.16	265214.16	0.00
F3	60	30	30	4400	5	241120.78	241120.78	0.00
F4	60	30	30	5500	4	233861.85	233861.85	0.00
G1	57	45	12	2700	10	306305.40	306305.40	0.00
G2	57	45	12	4300	6	245440.99	245440.99	0.00
G3	57	45	12	5300	5	229507.48	229507.48	0.00
G4	57	45	12	5300	6	232521.25	232521.25	0.00
G5	57	45	12	6400	5	221730.35	221730.35	0.00
G6	57	45	12	8000	4	213457.45	213457.45	0.00
H1	68	45	23	4000	6	268933.06	268933.06	0.00
H2	68	45	23	5100	5	253365.50	253365.50	0.00
H3	68	45	23	6100	4	247449.04	247449.04	0.00
H4	68	45	23	6100	5	250220.77	250220.77	0.00
H5	68	45	23	7100	4	246121.31	246121.31	0.00
H6	68	45	23	7100	5	249135.32	249135.32	0.00
I1	90	45	45	3000	10	350245.28	350245.28	0.00
I2	90	45	45	4000	7	309943.84	309943.84	0.00
I3	90	45	45	5700	5	294507.38	294507.38	0.00
I4	90	45	45	5700	6	295988.45	295988.45	0.00
I5	90	45	45	5700	7	301236.01	301236.01	0.00
J1	94	75	19	4400	10	335006.68	335006.68	0.00
J2	94	75	19	5600	8	310417.21	310417.21	0.00
J3	94	75	19	8200	6	279219.21	279219.21	0.00
J4	94	75	19	6600	7	296533.16	296533.16	0.00
K1	113	75	38	4100	10	394071.17	394375.63	0.08
K2	113	75	38	5200	8	362130.00	362130.00	0.00
K3	113	75	38	5200	9	365694.08	365694.08	0.00
K4	113	75	38	6200	7	348949.39	348949.39	0.00
L1	150	75	75	4400	10	417896.72	417943.82	0.01
L2	150	75	75	5000	8	401228.80	401228.80	0.00
L3	150	75	75	5000	9	402677.72	403639.75	0.24
L4	150	75	75	6000	7	384636.33	384636.33	0.00
L5	150	75	75	6000	8	387564.55	387564.55	0.00
M1	125	100	25	5200	11	398593.19	398869.79	0.07
M2	125	100	25	5200	10	396916.97	397786.41	0.22
M3	125	100	25	6200	9	375695.42	377315.94	0.43
M4	125	100	25	8000	7	348140.16	348140.16	0.00
N1	150	100	50	5700	11	408100.62	408100.62	0.00
N2	150	100	50	5700	10	408065.44	408111.91	0.01
N3	150	100	50	6600	9	394337.86	397621.99	0.83
N4	150	100	50	6600	10	394788.36	398330.35	0.90
N5	150	100	50	8500	7	373476.30	373723.37	0.07
N6	150	100	50	8500	8	373758.65	376200.31	0.65

7.1.3. Solving the VRPB with Mat-heuristic (CSMH algorithm)

The CSMH algorithm methodology proposed in Chapter 6 is adapted for the VRPB. Here, as explained in Section 7.1.2, the initial solution is generated and the fixed fleet constraint is imposed in *Phase I* of the algorithm. In *Phase II*, the *Two-Level VNS* is used first to obtain the best solution (note that *Phase II* is run single time) followed by *Phase III* where VRPB mathematical formulation model that uses CPLEX optimiser is used (replacing the MT-VRPB formulation implemented in Chapter 6) to obtain the optimal or improved incumbent solution.

Computational experience

The CSMH methodology is implemented with the same programming language and the computer specifications as in Chapter 6.

The computational experiments are reported for two VRPB data sets (i.e., see *set-2* and the *set*-3, see Section 7.1.2.1). For each instance the CSMH algorithm is run for a maximum CPU time of 2 hours (7200 seconds) for all the three phases. Since the *Two-Level VNS* is fairly quick, within this time, the *Phase II* is run for 200 iterations. Note that the *Two-Level VNS* is run one time before the CPLEX optimiser is called.

Glossary for tables:

VNS Sol. = Solution obtained by *Two-Level VNS*,

Opt. Sol. = Optimal solution,

Incum. Sol. = Incumbent solution,

UB = Upper Bound,

LB = Lower Bound,

%Gap = % gap between optimal/incumbent solution and lower bound,

Time (s) = CPU time in seconds taken to reach the solution.

7.1.3.1. CSMH VRPB Results and Analysis

The CSMH algorithm performed well and produced very competitive results for both data sets. When compared with the best known solutions from the literature, it produced results with overall ARPB of 0.06 and 0.09 for the *set-2* and the *set-3*, respectively.

Comparison of the CSMH with the Two-Level VNS and some recent algorithms:

The performance of the CSMH algorithm is compared with the *Two-Level VNS* algorithm as well as some best published algorithms described earlier. Please see Table 7.1 for the Processor information for the algorithms compared below; for CSMH algorithm same machine is used as of *Two-Level VNS*. The performance analysis summaries of these algorithms for the data *set-2* and the data *set-3* are provided in Table 7.6 and Table 7.7 respectively. The columns in both tables show the number of runs the respective algorithms were executed, the number of best known matched solutions, the average solution cost, the overall average of the relative percentage deviations (RPD) from the best known solutions and the average execution time taken by each algorithm.

Table 7.6: Comparison of the CSMH with the *Two-Level VNS* and the best VRPB algorithms in the literature (data *set-2*)

Algorithm	Runs	# Best sol. (out of 33)	Avg. best sol. Cost	Avg. RPD	Avg. time (s)
RTS-AMP (2007)	5	21	706.49	0.80	608.11
MACS (2009)	8	28	701.49	0.09	25.65
RPA (2012)	-	-	-	-	-
ILS (2014)	10	33	700.63	0.00	7.35
Two-Level VNS	5	33	700.63	0.00	29.29
CSMH	1	29	701.18	0.06	3728.52

Table 7.7: Comparison of the CSMH with the *Two-Level VNS* and the best VRPB algorithms in the literature (data *set-3*)

Algorithm	Runs	# Best sol. (out of 62)	Avg. best sol. Cost	Avg. RPD	Avg. time (s)
RTS-AMP (2007)	5	40	290981.84	0.11	1835.98
MACS (2009)	8	46	290838.73	0.07	67.57
RPA (2012)	10	62	290576.06	0.00	246.89
ILS (2014)	10	62	290576.22	0.00	22.89
Two-Level VNS	5	51	290796.24	0.06	43.24
CSMH	1	48	290908.31	0.09	5735.73

Despite the fact that the CSMH algorithm was designed for MT-VRPB, it has produced encouraging results in terms of solution quality when implemented on the VRPB. Since the results of the best published algorithms are reported from several different runs, it is not straight forward to compare the solution quality results. The CSMH however spends comparatively more time which is due to the fact that the algorithm incorporates both heuristic and MP aspects. For data *set-2*, the CSMH algorithm outperformed two of the algorithms in terms of the number of best known solutions found; however it did not perform better than *Two-Level VNS* and ILS (2014). For data *set-3*, it finds better solutions when compared with the RTS-AMP (2007) and MACS (2009); however RPA (2012), ILS (2014) and *Two-Level VNS* produced the maximum number of best known solutions. In our opinion relatively inferior performance of the CSMH is due to the fixed fleet imposition constraint of the classical VRPB. The detailed results of the CSMH algorithm are provided in Table 7.8 and Table 7.9 for the *set-2* and the *set-3*, respectively.

Table 7.8: Detailed results of the CSMH algorithm (data *set-2*)

Name	CSMH Algorithm						
	VNS Sol.	Opt. Sol.	Incum. Sol.	UB	LB	%Gap	Time (s)
eil22_50	371	371	-	371.0000	371.0000	0.00%	1
eil22_66	366	366	-	366.0000	356.3833	2.63%	1
eil22_80	375	375	-	375.0000	356.0640	5.05%	12
eil23_50	682	682	-	682.0000	665.3576	2.44%	2
eil23_66	649	649	-	649.0000	622.8153	4.03%	2
eil23_80	623	623	-	623.0000	590.8078	5.17%	4
eil30_50	501	501	-	501.0000	501.0000	0.00%	4
eil30_66	537	537	-	537.0000	511.3064	4.78%	122
eil30_80	514	514	-	514.0000	492.2562	4.23%	43
eil33_50	738	738	-	738.0000	732.4866	0.75%	5
eil33_66	750	750	-	750.0000	734.0343	2.13%	8
eil33_80	736	736	-	736.0000	719.3315	2.26%	95
eil51_50	560	559	-	559.0000	548.4229	1.89%	119
eil51_66	551	548	-	548.0000	540.4508	1.38%	531
eil51_80	574	565	-	565.0000	553.9328	1.96%	5820
eilA76_50	741	739	-	739.0000	725.4580	1.83%	6733
eilA76_60	773	768	-	768.0000	755.4523	1.63%	5643
eilA76_80	781	-	781	781.0000	738.9720	5.38%	7200
eilB76_50	811	-	801	801.0000	764.4242	4.57%	7200
eilB76_66	873	-	873	873.0000	808.7466	7.36%	7200
eilB76_80	919	-	919	919.0000	869.3950	5.40%	7200
eilC76_50	713	-	713	713.0000	684.8103	3.95%	7200
eilC76_66	734	-	734	734.0000	708.3127	3.50%	7200
eilC76_80	733	-	733	733.0000	703.4388	4.03%	7200
eilD76_50	690	690	-	690.0000	687.1383	0.41%	210
eilD76_66	717	715	-	715.0000	713.8488	0.16%	7200
eilD76_80	696	-	696	696.0000	684.4152	1.66%	7200
eilA101_50	832	-	832	832.0000	808.6934	2.80%	7200
eilA101_66	846	846	-	846.0000	843.7409	0.27%	2886
eilA101_80	868	-	868	868.0000	828.4039	4.56%	7200
eilB101_50	923	-	923	923.0000	870.2547	5.71%	7200
eilB101_66	983	-	983	983.0000	920.6677	6.34%	7200
eilB101_80	1011	-	1011	1011.0000	959.1865	5.12%	7200

Table 7.9: Detailed results of the CSMH algorithm (data *set-3*)

Name	CSMH Algorithm						
	VNS Sol.	**Opt. Sol.**	**Incum. Sol.**	**UB**	**LB**	**%Gap**	**Time (s)**
A1	229885.65	229885.65	-	229885.65	228283.70	70%	102
A2	180119.21	180119.21	-	180119.21	176777.94	1.86%	14
A3	163405.38	163405.38	-	163405.38	158778.00	2.83%	6
A4	155796.41	155796.41	-	155796.41	151444.26	2.79%	4
B1	239080.16	-	239080.16	239080.16	231751.67	3.07%	7200
B2	198048.77	198048.77	-	198048.77	193238.26	2.43%	14
B3	169372.29	169372.29	-	169372.29	163685.22	3.36%	4
C1	350556.77	-	350556.77	350556.77	228042.37	8.99%	7200
C2	215020.23	-	215020.23	215020.23	209339.42	2.64%	7200
C3	199345.96	199345.96	-	199345.96	192.398.7666	3.48%	38
C4	195366.63	195366.63	-	195366.63	188064.16	3.74%	36
D1	322530.13	-	322530.13	322530.13	303851.93	5.79%	7200
D2	316708.86	-	316708.86	316708.86	290715.54	8.21%	7200
D3	239478.63	-	239478.63	239478.63	223393.60	6.72%	7200
D4	205831.94	-	205831.94	205831.94	192434.94	6.51%	7200
E1	238879.58	-	238879.58	238879.58	134351.75	1.88%	7200
E2	212263.11	-	212263.11	212263.11	206621.09	2.66%	7200
E3	206659.17	206659.17	-	206659.17	206054.83	0.29%	1105
F1	263173.96	-	263173.96	263173.96	245312.54	6.79%	7200
F2	265214.16	-	265214.16	265214.16	254872.20	3.90%	7200
F3	241120.78	241120.78	-	241120.78	240850.28	0.11%	541
F4	233861.85	233861.85	-	233861.85	233425.32	0.19%	466
G1	306305.40		306305.40	306305.40	282699.98	7.71%	7200
G2	245440.99	-	245440.99	245440.99	235855.88	3.90%	7200
G3	229507.48	-	229507.48	229507.48	218424.98	4.83%	7200
G4	232521.25	-	232521.25	232521.25	220434.50	5.20%	7200
G5	221730.35	-	221730.35	221730.35	213122.77	3.88%	7200
G6	213457.45	-	213457.45	213457.45	208343.07	2.40%	7200
H1	268933.06	-	268933.06	268933.06	260229.68	3.24%	7200
H2	253365.50	253365.50	-	253365.50	252543.17	0.32%	318
H3	247449.04	-	247449.04	247449.04	241010.73	2.60%	7200
H4	250220.77	-	250220.77	250220.77	239860.25	4.14%	7200
H5	264121.31	264121.31	-	264121.31	264121.31	0.00%	167

Name	CSMH Algorithm						
	VNS Sol.	Opt. Sol.	Incum. Sol.	UB	LB	%Gap	Time (s)
H6	249135.32	-	249135.32	249135.32	241378.92	3.11%	7200
I1	350567.90	-	350567.90	350567.90	333096.41	4.98%	7200
I2	309943.84	-	309943.84	309943.84	295645.55	4.61%	7200
I3	294833.96	-	294833.96	294833.96	285108.39	3.30%	7200
I4	295988.45	-	295988.45	295988.45	292129.46	1.30%	7200
I5	301236.01	301236.01	-	301236.01	300857.61	0.13%	7200
J1	335006.68	-	335006.68	335006.68	315165.06	5.93%	7200
J2	310417.21	-	310417.21	310417.21	288343.42	7.11%	7200
J3	279219.21	-	279219.21	279219.21	271760.64	2.67%	7200
J4	296533.16	-	296533.16	296533.16	277765.14	6.33%	7200
K1	294071.17	-	294071.17	294071.17	376498.69	4.46%	7200
K2	362360.27	-	362360.27	362360.27	329756.80	9.00%	7200
K3	365694.08	-	365694.08	365694.08	328565.66	10.15%	7200
K4	348949.39	-	348949.39	348949.39	323025.60	7.43%	7200
L1	417896.71	-	417896.71	417896.71	375739.91	10.09%	7200
L2	401228.80	-	401228.80	401228.80	360280.26	10.21%	7200
L3	406873.02	-	406873.02	406873.02	358571.33	11.87%	7200
L4	385615.90		385615.90	385615.90	347057.31	10.00%	7200
L5	387564.55		387564.55	387564.55	345160.55	10.94%	7200
M1	399070.20	-	399070.20	399070.20	372952.55	6.54%	7200
M2	400293.41		400293.41	400293.41	373797.02	6.62%	7200
M3	378921.05	-	378921.05	378921.05	337888.52	10.83%	7200
M4	348437.62	-	348437.62	348437.62	315738.45	9.38%	7200
N1	408100.62		408100.62	408100.62	369544.69	9.45%	7200
N2	409255.06	-	409255.06	409255.06	391009.37	4.46%	7200
N3	394337.86		394337.86	394337.86	353570.42	10.34%	7200
N4	394788.36		394788.36	394788.36	357078.74	9.55%	7200
N5	375100.10		375100.10	375100.10	329286.79	12.21%	7200
N6	378103.04		378103.04	378103.04	335022.98	11.39%	7200

7.2. The case of the MT-VRP

The MT-VRP is already explained along with the literature review in Chapters 2 and 3. In this section we adapt our approaches developed in Chapters 4, 5 & 6 to solve the MT-VRP.

7.2.1. Formulation of the Basic Case

The MT-VRP formulation is adapted from our MT-VRPB formulation presented in Chapter 4. This is a three-indexed commodity flow formulation. Before this Mingozzi *at el*. (2013) provided two set partitioning based formulations for their proposed exact methodologies for the MT-VRP.

Notations:

Sets

$\{0\}$ the depot (single depot)

L the set of customers

K the set of vehicles

Input Variables

d_{ij} the distance between customers i and j ($i \in \{0\} \cup L, j \in \{0\} \cup L$)

q_i the demand of customer i

Other Parameters

C vehicle capacity

T planning period (maximum driving time)

Decision Variables

$x_{ijk} = \begin{cases} 1, \text{if vehicle } k \text{ travels from location } i \text{ directly to location } j; \\ 0, \text{otherwise} \end{cases}$

R_{ij} = is the amount of goods on board on arc ij

Minimise $Z = \sum_{i\in\{0\}\cup L} \sum_{j\in\{0\}\cup L} \sum_{k\in K} d_{ij} x_{ijk}$ (7.16)

Subject to

$$\sum_{j\in\{0\}\cup L} \sum_{k\in K} x_{jik} = 1 \qquad i \in L \tag{7.17}$$

$$\sum_{j\in\{0\}\cup L} \sum_{k\in K} x_{ijk} = 1 \qquad i \in L \tag{7.18}$$

$$\sum_{j\in\{0\}\cup L} x_{jik} = \sum_{j\in\{0\}\cup L} x_{ijk} \qquad i \in L,\ \forall\, k \in K \tag{7.19}$$

$$\sum_{i\in\{0\}\cup L} R_{ij} - q_j = \sum_{i\in\{0\}\cup L} R_{ji} \qquad j \in L \tag{7.20}$$

$$R_{ij} \le C \sum_{k\in K} x_{ijk} \qquad i \in L, j \in L;\ \forall\, k \in K \tag{7.21}$$

$$\sum_{i\in\{0\}\cup L} \sum_{j\in\{0\}\cup L} d_{ij} x_{ijk} \le T \qquad \forall\ k \in K \tag{7.22}$$

$$R_{ij} \ge 0 \qquad i \in \{0\} \cup L,\ j \in L \tag{7.23}$$

$$x_{ijk} = 0,1 \qquad \begin{matrix} i \in \{0\} \cup L,\ j \in \{0\} \cup L \\ k \in K \end{matrix} \tag{7.24}$$

Equation (7.16) illustrates the objective function representing the total distance travelled. Constraints (7.17) and (7.18) ensure that every customer is served exactly once (every customer has an incoming arc and every customer has an outgoing arc). Constraint (7.19) states that the number of times vehicle k enters into customer i is the same as the number of times it leaves customer i. The vehicle load variation on a route is ensured by Constraints (7.20). Inequalities (7.21) and (7.22) impose the maximum

vehicle capacity constraint and the maximum working day period constraints in which a vehicle is allowed to serve the routes respectively. Inequality (7.23) sets R_{ij} as a non-negative variable. Finally, in (7.24) the decision variable x_{ijk} is set as zero-one variable.

The validity of the mathematical formulation of the MT-VRP is checked using the IBM ILOG CPLEX 12.5. Hence it was implemented in CPLEX and it proved valid when tested on some MT-VRP benchmark instances from the literature.

7.2.2. The *Two-Level VNS* methodology for the MT-VRP

In Chapter 5, the MT-VRP was extended and solved with backhauling aspect. However, as explained in the review of Chapter 4 the classical MT-VRP is studied independently in the literature. Here, we would like to test if the *Two-Level VNS* algorithm is viable for the MT-VRP. Details of the implementation are provided in the following subsections.

The *Two-Level VNS* algorithm is already elaborated in Section 5.1.1 in detail but for completeness here we present its steps while emphasising on any implementation differences.

Phase I: **Initial solution – *sweep* approach**

- Generate an initial solution x using the *sweep* method (see Section 5.2)
- Apply the following refinement routines in a sequential order to improve the initial solution x and then go to ***Phase II***

    - *1-Insertion_intra_route* (x)
    - *1-Insertion_inter_route* (x)
    - *Swap_1_1* (x)
    - *Swap_2_2* (x)
    - *Shift_2_0* (x)

- *Swap_2_1* (x)

(see Section 5.3 for all refinement routines in *Phase I*)

***Phase II: Two-Level VNS* Algorithm**

Initialize the solution pool data structure S_p and add the initial solution x to S_p,

Set: $iter$ = 1 and $iter_{max} = 200$

Repeat the process while $iter \leq iter_{max}$

Start *outer-level*

Let: $LS_k^O =< R_3, R_4, R_5 >$ set of refinement routines for the *outer-level*

Set: $k = 1$

Repeat the process while $k \leq N_{k_{max}}^I$

***a.1*:** Generate a neighbouring solution $x' \in N_k^O(x)$ at random;

***a.2*:** Apply LS_k^O on the neighbouring solution x'

***a.3*:** Assign the resulting solution x' to x'_{best} $[x'_{best} = x']$

a.4*: Start *inner-level using x'_{best}

Let: $LS_l^I =< \{R_1 \& R_6\}, \{R_2 \& R_6\}, \{R_3 \& R_6\}, \{R_4 \& R_6\}, \{R_5 \& R_6\} >$

Set: $l = 1$

Repeat the process while $l \leq N_{l_{max}}^I$

***a.4*(1):** Generate a neighbouring solution $x'' \in N_l^I(x'_{best})$ at random

***a.4*(2):** Apply LS_l^I [Multi-Layer local search optimiser framework] on the neighbouring solution x''

***a.4*(3):** Assign the resulting solution x'' to x''_{best} $[x''_{best} = x'']$

***a.4*(4):** If $x''_{best} < x'_{best}$ then $x'_{best} = x''_{best}$, set $l = 1$ got to ***a.4*(1)**

Else set $l = l + 1$ and got to ***a.4*(1)**

a.5: If $x'_{best} < x$ then $x = x'_{best}$; $S_p = x$, set $k = 1$ and go to ***a.1***

Else set $k = k + 1$ and go to ***a.1***

***Phase III:* Solving the Multiple Trips aspect using the BPP**

Initialize an special 3-dimentional data structure Sol_k and let Sol_{max} number of solutions stored in S_p and Let $iterBM_{max} = 5$.

Set: $iter_{Sol} = 1$

Repeat the process while $iter_{Sol} \leq Sol_{max}$

Step1. Solve the BPP for solution p using CPLEX optimiser ($p = 1, \dots, Sol_{max}$)

Step2. If solution *p* is feasibly packed then go to ***Step4***

Else, go to ***Step3***

Step3. Apply the *Bisection Method* to optimise the bin capacity

Set: $iterBM = 1$

Repeat the process while $iterBM \leq iterBM_{max}$

***Step3.*(1):** Use the *Bisection Method*

***Step3.*(2):** Solve the BPP for solution *p* using CPLEX optimiser

Step4. Store the solution in the special data structure Sol_k according to what routes are served by which bins (vehicles)

Figure 7.2: Algorithmic steps of the *Two-Level VNS* for the MT-VRP

Initial solution: (***Phase I***)

The initial solution for the MT-VRP is obtained by using the *sweep* procedure of Gillet and Miller (1974) clockwise as explained in Section 5.2 of Chapter 5. An illustrative example of our sweep implementation is shown in Figure 7.3. Moreover, we have used all six local search refinement routines is sequence in order to improve the initial solution before passing it to *Phase II*.

***Neighbourhoods*: (*Phase II*)**

We have used in total six neighbourhoods in order to generate the neighbouring solutions for the MT-VRP. The neighbourhoods are implemented in the same manner and are kept in the same order without any significant changes as explained in Chapter

5. To show how the MT-VRP neighbourhood moves are conducted without backhauls, we provide their respective graphs in Figure 7.4 as illustrations. When using these neighbourhoods for the MT-VRP, they are allowed to move customer/customers from one route to another route and end-up emptying the route. This is allowed if it leads to a feasible solution since there is no fixed fleet constraint or feasibility issues in terms of the types of customers as in the VRPB.

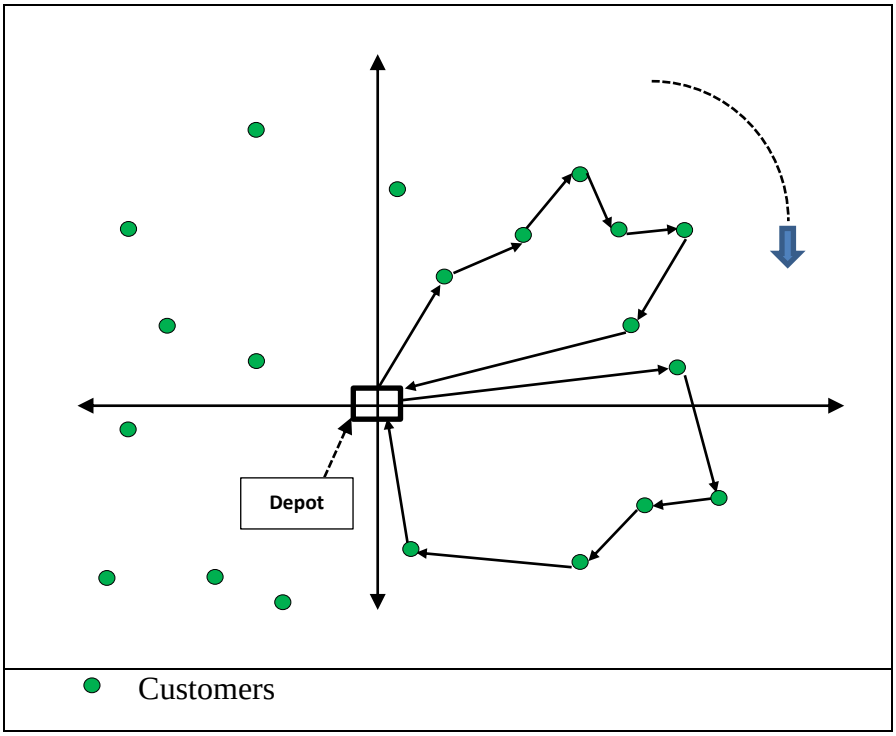

Figure 7.3: An illustrative example of the sweep procedure for the MT-VRP

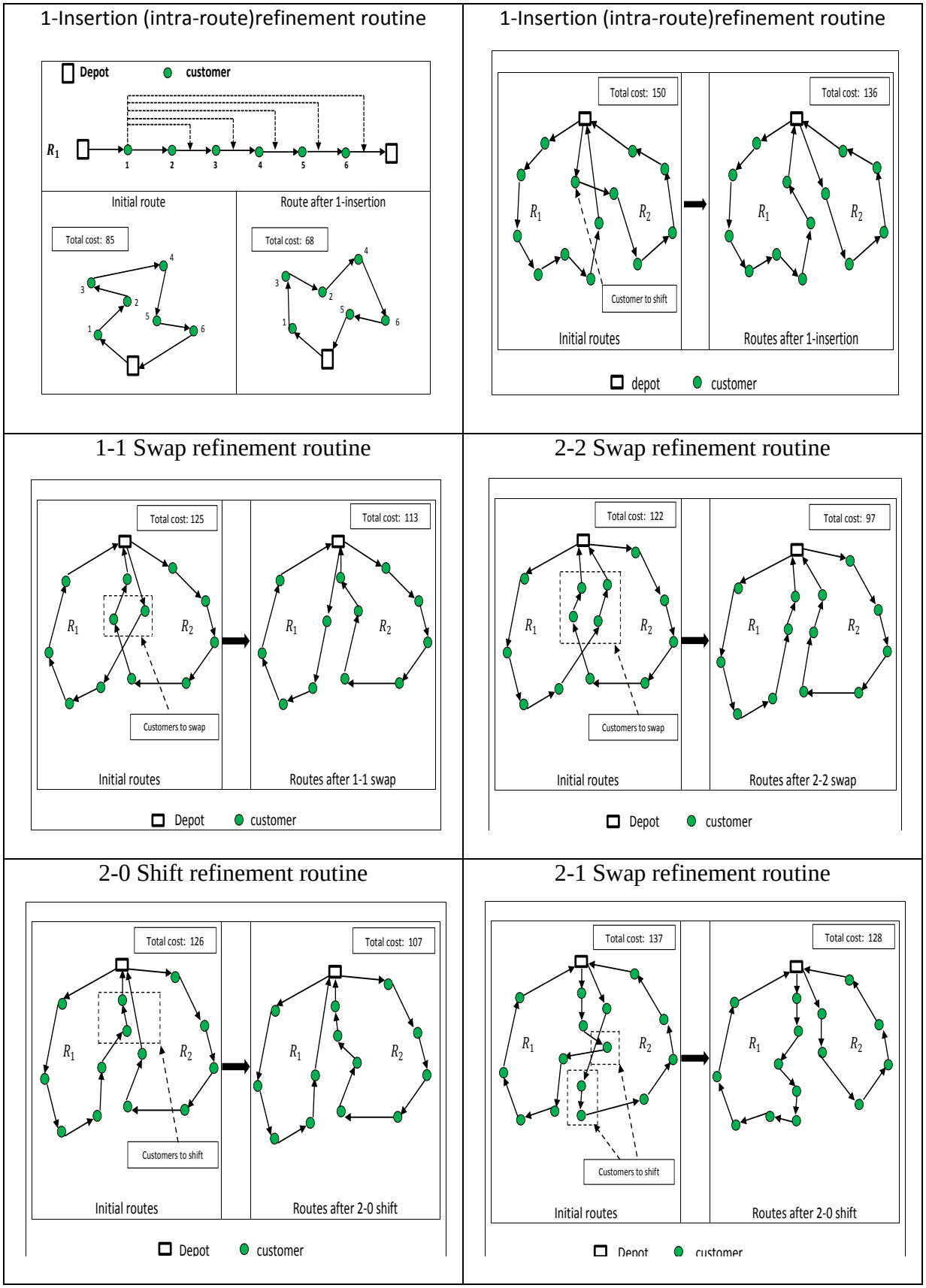

Figure 7.4: Illustration of all the refinement routines implemented for the MT-VRP

***The Multi-Layer local search optimiser framework*:**

The framework structure used in the MT-VRPB in Chapter 5 and in the VRPB in this chapter remained unchanged for the MT-VRP. However, the neighbourhood moves are more freely conducted here due to the fact that the MT-VRP is much less constrained than the MT-VRPB and the VRPB. Since the MT-VRP has no backhauling and/or fixed fleet utilization constraints, for example, while conducting these neighbourhood moves during the search process a vehicle can be emptied due to result of shifting all customers from one route to other routes given that the resulting solution is better.

***BPP implementation:* (*Phase III*)**

The Bin Packing process is implemented exactly as in the MT-VRPB (see Section 5.5) and the BPP is solved first followed by the *Bisection Method* to optimise the bin (vehicle) capacity.

7.2.2.1. Details of MT-VRP Computations and the Data sets

The *Two-Level VNS* algorithm for the MT-VRP is implemented in the same programming language and the computer specifications as in Chapter 5.

The *Two-Level VNS* algorithm is tested on data *set-4* using a fixed number of 200 iterations, which was experimentally found to be acceptable in terms of solution quality and the computational time affordability.

MT-VRP Data set:

The computational experiments are reported for the most studied MT-VRP benchmark data set proposed in Taillard *et al*. (1996). This data set is referred to as data *set-4* in this

study. The data *set-4* was generated from nine VRP problems 1-5 and 11-12 of Christofides, Mingozzi and Toth (1979) and 11-12 VRP problems of Fisher (1994). For the data *set-4* Taillard *et al*. (1996) used the same graphs, demands and vehicle capacities given in the VRP nine base problems. The authors have generated 104 sub-problems in total by applying different values of m (where m is the number of vehicles, (i.e. 1,…,4), starting with an integer between one and the maximum number of vehicles) and T (where T is a maximum driving time). Moreover two values of T are used as, T_1 and T_2 for each value of m (vehicles). The values of T_1 and T_2 are calculated as follows. $T_1 = [1.05\ z^*/m]$ and $T_2 = [1.1\ z^*/m]$ rounded to nearest integer, where z^* represents the VRP solution with unlimited number of vehicles used in Rochat and Taillard (1995). Moreover, a penalty factor $\theta = 2$ is associated with all routes whose length violates the maximum driver time T. This specifies that the driver overtime is penalized in this data set.

Moreover, the *set-4* is divided in three groups (G1, G2 and G3) in the literature. The G1 consists of 42 instances for which optimal solutions are known, whereas G2 consists of 56 instances for which feasible (solutions where no overtime is used) solutions are reported and finally G3 consist of 5 non-feasible (instances for which overtime is used) solutions are reported.

7.2.2.2. *Two-Level VNS* MT-VRP Results and Analysis

The *Two-Level VNS* algorithm produced very competitive results for the MT-VRP (in terms of the solution quality and the computational speed) for data *set 4* when compared with the best known solutions from the literature despite the fact that the proposed algorithm was originally designed for the MT-VRPB. The detailed results are provided

in Table 7.10, Table 7.11 and Table 7.12 for G1, G2 and G3, respectively; and the time comparison as shown in Table 7.13.

Comparison of the Two-Level VNS with some best metaheuristic algorithms:

The performance of the Two-Level VNS algorithm is compared with some well-known algorithms published in literature which provide detailed solutions for all the groups. The studies that are included in our comparisons are MRT (exact algorithm of Mingozzi et al., 2013); GA (genetic algorithm based heuristic of Salhi and Petch, 2007) and MA and MA+CLS (memetic algorithms of Cattaruzza et al., 2014a).

For G1 group of instances (Table 7.10), the MRT produced all 42 optimal solutions. The Two-Level VNS algorithm remained very competitive, in terms of the solution quality (i.e., optimality and feasibility with no use of overtime), producing 26 optimal (12 feasible) solutions comparing with none optimal (33 feasible) of the GA, 33 optimal (9 feasible) of the MA and 37 optimal (5 feasible) of MA+CLS. Regarding the number of optimal/feasible solutions on this group of instances, although the *Two-Level VNS* appears to be the third best heuristic algorithm in the literature, it has produced 1 new best solution (CMT2-75 (4)) in G1. In terms of ARPD (average relative percentage deviation), the *Two-Level VNS* solutions are less than 1% away from the best known.

For the G2 group of 56 instances (Table 7.11), the *Two-Level VNS* again performed quite well, in terms of the solution quality (i.e., no overtime used), producing 45 compared to all 56 of the MA and MA+CLS and 29 of the GA solutions. Moreover, the *Two-Level VNS* also performed competitively on the basis of the ARPD of instances where the solutions were obtained.

For the G3 group of 5 instances (Table 7.12), the *Two-Level VNS* algorithm overall remained competitive and produced better quality results than the GA but inferior as compared to MA and MA+CLS.

As for the computation time a fair comparison may not be possible since the algorithms compared here used different machines with different configurations. While PC machine specifications are provided in the beginning of this section, the other algorithms GA, MA and MA+CLS were run on Ultra Enterprise 450 dual processor 300 megahertz and Intel Xeon 2.80 GHz processor, respectively. Table 7.13 shows the average computational times (in seconds) for the individual classes in the data *set 4* as well as the overall average times in the last row of the table. The *Two- Level VNS* algorithm proved faster in all the classes of the data *set-4* with the exception of one where with a marginal difference was recorded.

Table 7.10: Detailed results for 42 instances in G1 (Data *set-4*)

Name (size)	*m*	*T*	MRT	SP	MA	MA + CLS	*Two-Level VNS*
			Optimal	**Best**	**Best**	**Best**	**Best**
CMT1 (50)	1	551	**524.61**	546.28	**524.61**	**524.61**	**524.61**
	2	275	**533.00**	x	**533.00**	**533.00**	**533.00**
	1	577	**524.61**	547.14	**524.61**	**524.61**	**524.61**
	2	289	**529.85**	549.42	**529.85**	**529.85**	**529.85**
	4	144	**546.29**	566.86	**546.29**	**546.29**	x
CMT2 (75)	1	877	**835.26**	869.06	**835.26**	**835.26**	**835.26**
	2	439	**835.26**	865.48	835.77	**835.26**	**835.26**
	3	292	**835.26**	x	**835.26**	**835.26**	**835.26**
	4	219	**835.26**	856.77	835.77	835.77	835.32
	5	175	**835.80**	x	836.18	836.18	837.40
	1	919	**835.26**	869.73	**835.26**	**835.26**	**835.26**
	2	459	**835.26**	881.50	**835.26**	**835.26**	**835.26**
	3	306	**835.26**	869.11	835.77	**835.26**	**835.26**
	4	230	**835.26**	880.90	838.17	**835.26**	**835.26**
	5	184	**835.26**	883.29	835.77	835.77	835.77
	6	153	**839.22**	x	843.09	**839.22**	x
CMT3 (100)	1	867	**826.14**	845.33	**826.14**	**826.14**	**826.14**
	2	434	**826.14**	850.65	**826.14**	**826.14**	**826.14**
	3	289	**826.14**	x	828.08	**826.14**	**826.14**
	1	909	**826.14**	845.33	829.45	829.45	**828.26**
	2	454	**826.14**	872.10	**826.14**	**826.14**	**826.14**
	3	303	**826.14**	869.48	**826.14**	827.39	**826.14**
	4	227	**826.14**	878.00	**826.14**	**826.14**	**826.14**
CMT11 (120)	1	1094	**1042.11**	1088.26	**1042.11**	**1042.11**	1072.95
	2	547	**1042.11**	x	**1042.11**	**1042.11**	1073.96
	3	365	**1042.11**	x	**1042.11**	**1042.11**	x
	5	219	**1042.11**	x	**1042.11**	**1042.11**	x
	1	1146	**1042.11**	1088.26	**1042.11**	**1042.11**	1075.83
	2	573	**1042.11**	1110.10	**1042.11**	**1042.11**	1073.44
	3	382	**1042.11**	1088.56	**1042.11**	**1042.11**	1085.28
	4	287	**1042.11**	x	**1042.11**	**1042.11**	1062.30
	5	229	**1042.11**	1092.95	**1042.11**	**1042.11**	1088.46
CMT12 (100)	1	861	**819.56**	819.97	**819.56**	**819.56**	**819.56**
	2	430	**819.56**	821.33	**819.56**	**819.56**	**819.56**
	3	287	**819.56**	826.98	**819.56**	**819.56**	**819.56**
	4	215	**819.56**	824.57	**819.56**	**819.56**	**819.56**
	1	902	**819.56**	819.97	**819.56**	**819.56**	**819.56**
	2	451	**819.56**	829.54	**819.56**	**819.56**	**819.56**
	3	301	**819.56**	851.16	**819.56**	**819.56**	**819.56**
	4	225	**819.56**	821.53	**819.56**	**819.56**	**819.56**
	5	180	**824.78**	833.85	**824.78**	**824.78**	826.90
	6	150	**823.14**	855.36	**823.14**	**823.14**	827.14
# of solutions found (out of 42)			**42**	**33**	**42**	**42**	**38**
# of optimal solutions found			**42**	**0**	**33**	**37**	**26**
ARPD				**3.41**	**0.04**	**0.02**	**0.63**

Table 7.11: Detailed feasible results for 56 instances in G2 (Data *set-4*)

Name (size)	*m*	*T*	Best known	SP	MA	MA + CLS	*Two-Level VNS*
				Best	Best	Best	Best
CMT1 (50)	3	192	552.68	560.26	552.68	552.68	558.16
CMT2 (75)	6	146	858.58	x	858.58	859.16	x
	7	131	844.70	x	853.88	844.70	x
CMT3 (100)	4	217	829.45	x	829.45	829.45	829.63
	5	173	832.89	x	832.89	832.89	x
	6	145	836.22	x	836.22	836.22	x
	5	182	832.34	901.30	833.02	832.34	x
	6	151	834.35	861.76	834.35	834.35	x
CMT4 (150)	1	1080	1031.00	1064.06	1031.00	1031.00	1032.96
	2	540	1031.07	1065.86	1032.65	1031.07	1032.55
	3	360	1028.42	x	1029.56	1028.42	1037.29
	4	270	1031.10	x	1036.25	1031.10	1039.13
	5	216	1031.07	x	1032.69	1031.07	1039.33
	6	180	1034.61	x	1043.42	1034.61	1061.32
	8	135	1056.54	x	1056.93	1056.54	x
	1	1131	1031.07	1088.93	1031.07	1031.07	1031.51
	2	566	1030.45	1070.50	1030.45	1034.08	1032.55
	3	377	1031.59	1077.24	1031.63	1031.59	1032.13
	4	283	1031.07	1119.05	1031.07	1031.96	1032.83
	5	226	1030.86	1085.38	1033.05	1030.86	1036.34
	6	189	1030.45	1112.03	1032.16	1030.45	1037.26
	7	162	1036.08	x	1043.92	1036.08	1043.94
	8	141	1044.32	x	1044.71	1044.32	x
CMT5 (199)	1	1356	1302.43	1347.34	1302.43	1302.43	1319.54
	2	678	1302.15	1346.63	1302.15	1306.26	1325.92
	3	452	1301.29	x	1301.41	1301.29	1325.18
	4	339	1304.78	x	1308.93	1304.78	1324.96
	5	271	1300.02	x	1307.78	1300.02	1319.86
	6	226	1303.37	x	1303.37	1308.40	1324.01
	7	194	1309.40	x	1315.41	1309.40	1329.24
	8	170	1303.91	x	1310.48	1303.91	1321.41
	9	151	1307.93	x	1329.86	1307.93	1325.66
	10	136	1323.01	x	1326.54	1323.01	1332.68
	1	1421	1299.86	1340.44	1299.86	1299.86	1317.01
	2	710	1305.35	1399.65	1305.35	1307.70	1324.34
	3	474	1301.03	1409.37	1301.03	1308.76	1323.57
	4	355	1303.65	1397.60	1303.65	1310.97	1324.72
	5	284	1300.62	1411.19	1308.04	1300.62	1326.44
	6	237	1306.17	1377.07	1306.17	1306.25	1328.94
	7	203	1301.54	1394.73	1311.35	1301.54	1324.64
	8	178	1308.78	x	1311.93	1308.78	1322.14
	9	158	1307.25	x	1312.28	1307.25	1330.12

	10	142	1308.81	x	1312.04	1308.81	1320.9
CMT11 (120)	4	274	1078.64	x	1080.12	1078.64	x
CMT12 (100)	5	172	845.56	x	849.89	845.56	x
F11 (71)	1	254	241.97	x	241.97	241.97	241.97
	2	127	250.85	x	250.85	250.85	x
	1	266	241.97	254.07	241.97	241.97	241.97
	2	133	241.97	254.07	241.97	241.97	241.97
	3	89	254.07	256.53	254.07	254.07	254.07
F12 (134)	1	1221	1162.96	1190.21	1162.96	1162.96	1174.98
	2	611	1162.96	1194.24	1162.96	1162.96	1176.17
	3	407	1162.96	1199.86	1162.96	1162.96	1175.36
	1	1279	1162.96	1183.00	1162.96	1162.96	1166.86
	2	640	1162.96	1199.64	1162.96	1162.96	1174.71
	3	426	1162.96	1215.43	1162.96	1162.96	1187.57
# of feasible solutions found (out of 56)				**29**	**56**	**56**	**45**
ARPD				**4.86**	**0.19**	**0.05**	**1.01**

Table 7.12: Detailed non-feasible results for 5 instances in G3 (Data *set-4*)

Name	***m***	***T***	**Best known**	**SP**	**MA**	**MA+CLS**	***Two-Level VNS***
				Best	**Best**	**Best**	**Best**
CMT1	3	184	569.54	586.32	**569.54**	**569.54**	588.51
CMT1	4	138	564.07	632.54	**564.07**	**564.07**	603.34
CMT2	7	125	866.58	1056.34	876.77	**866.58**	916.36
CMT12	6	143	845.48	898.88	**845.48**	**845.48**	**845.48**
F11	3	85	256.93	266.85	**256.93**	**256.93**	261.57
ARPD				**9.43**	**0.24**	**0.00**	**3.57**

Table 7.13: Average time (in seconds) for the problem classes of *set-4*

Instance Name	**#**	**GA**	**MA**	**MA+CLS**	***Two-Level VNS***
CMT1	8	16	10	30	**8**
CMT2	14	30	25	118	**18**
CMT3	12	70	52	173	**46**
CMT4	16	206	169	493	**155**
CMT5	20	484	354	1284	**312**
CMT11	10	1132	99	302	**74**
CMT12	12	45	**37**	138	45
F11	6	93	21	40	**17**
F12	6	584	87	87	**81**
Average		**295.56**	**94.89**	**296.11**	**84.00**

7.2.3. Solving the MT-VRP with the Mat-heuristic (CSMH algorithm)

The CSMH algorithm methodology proposed in Chapter 6 is adapted to solve the MT-VRP. Hence, at *Phase I* the initial solution is generated as explained in Section 7.2.2. In *Phase II*, the *Two-Level VNS* is used to obtain a pool of solutions and in *Phase III*, the bin packing problem is solved for the solutions in the pool, and finally in Phase *IV*, the MT-VRP mathematical formulation model that uses CPLEX optimiser is used to obtain the optimal or improved incumbent solution.

Computational experience

The CSMH methodology is implemented with the same programming language and the computer specifications as in Chapter 6.

Glossary for tables:

+ : Solution obtained with overtime

Rest is same as in Section 7.1.3

7.2.3.1. CSMH MT-VRP Results and Analysis

The CSMH algorithm is run for a maximum CPU time of 2 hours (7200 seconds) for all the four phases (in which the *Phase II* is set to 300 iterations).

The CSMH algorithm is tested on G1 group of instances from data *set-4*. The algorithm performed very well and produced very high quality results. The detailed results are provided in Table 7.14. Note that even though most of results in this table are put under the “Incum. Sol.” (Incumbent solution) column, most of them are optimal. The reason

behind it is that within 2-hours computational time, the desired gap between upper and lower bounds was not achieved.

Comparison of the CSMH with the Two-Level VNS and some recent algorithms:

The performance of the CSMH algorithm is compared with the *Two-Level VNS* implementation and some best published algorithms (see Section 7.2.2.2). The detailed performance analysis of these algorithms for G1 (data *set-4*) are provided in Table 7.15. As it can be observed that the MRT exact algorithm produced optimal solutions for all 42 instances in this group. Compared to the MRT, the CSMH algorithm remained extremely competitive in terms of solution quality, producing 39 optimal (3 incumbent/feasible) solutions. Whereas comparing to the heuristic algorithms, it clearly performed better when comparing to none optimal (33 feasible) of GA, 33 optimal (9 feasible) of the MA, 37 optimal (5 feasible) of the MA+CLS and 26 optimal (12 feasible) of the *Two-Level VNS*.

Table 7.14: Detailed results of the CSMH for 42 instances in G1 (Data *set-4*)

Name (size)	*m*	*T*	CSMH Algorithm						
			VNS Sol.	Opt. Sol.	Incum. Sol.	UB	LB	%Gap	Time (s)
CMT1 (50)	1	551	524.61	524.61	-	524.61	520.62	0.76%	6215
	2	275	533.00	-	533.00	533.00	511.71	3.99%	7200
	1	577	524.61	524.61	-	524.61	515.03	1.83%	6123
	2	289	529.85	-	529.85	529.85	510.73	3.61%	7200
	4	144	574.84+	-	546.29	546.29	511.77	6.32%	7200
CMT2 (75)	1	877	835.26	-	835.26	835.26	775.49	7.16%	7200
	2	439	835.77	-	835.26	835.26	763.67	8.57%	7200
	3	292	835.77	-	835.26	835.26	746.94	10.63%	7200
	4	219	735.28	-	835.26	835.26	745.33	10.77%	7200
	5	175	848.44	-	835.80	835.80	742.46	11.17%	7200
	1	919	836.18	-	835.26	835.26	782.20	6.35%	7200
	2	459	835.26	-	835.26	835.26	788.46	5.60%	7200
	3	306	835.77	-	835.26	835.26	783.77	6.16%	7200
	4	230	835.77	-	835.26	835.26	778.74	6.77%	7200
	5	184	838.60	-	835.26	835.26	778.20	6.83%	7200
	6	153	853.17+	-	839.22	839.22	773.55	7.82%	7200
CMT3 (100)	1	867	828.42	-	826.14	826.14	778.86	5.72%	7200
	2	434	828.42	-	826.14	826.14	799.63	3.21%	7200
	3	289	829.63	-	826.14	826.14	791.23	4.23%	7200
	1	909	828.56	-	826.14	826.14	778.70	5.74%	7200
	2	454	829.63	-	826.14	826.14	790.45	4.32%	7200
	3	303	829.51	-	826.14	826.14	766.53	7.22%	7200
	4	227	829.65	-	826.14	826.14	779.58	5.64%	7200
CMT11 (120)	1	1094	1077.14	-	1042.11	1075.03	967.02	10.36%	7200
	2	547	1072.90	-	1044.09	1072.90	943.18	12.42%	7200
	3	365	1042.11	-	1042.11	1042.12	935.47	10.23%	7200
	5	219	1045.32	-	1042.11	1042.12	933.35	10.44%	7200
	1	1146	1071.96	-	1042.11	1048.74	965.71	7.97%	7200
	2	573	1063.47	-	1042.11	1063.47	936.81	12.15%	7200
	3	382	1048.26	-	1048.26	1048.26	922.27	12.02%	7200
	4	287	1062.30	-	1044.09	1044.09	935.79	10.37%	7200
	5	229	1088.46	-	1042.11	1042.12	933.47	10.43%	7200
CMT12 (100)	1	861	819.56	-	819.56	819.56	767.53	6.35%	7200
	2	430	819.56	-	819.56	819.56	778.36	5.03%	7200
	3	287	819.56	-	819.56	819.56	782.91	4.47%	7200
	4	215	819.56	-	819.56	819.56	774.95	5.44%	7200
	1	902	819.56	-	819.56	819.56	766.25	6.50%	7200
	2	451	819.56	-	819.56	819.56	780.96	4.71%	7200
	3	301	819.56	-	819.56	819.56	785.94	4.10%	7200
	4	225	819.56	-	819.56	819.56	789.83	3.63%	7200
	5	180	825.38	-	824.78	824.78	776.24	4.24%	7200
	6	150	824.46+	-	823.14	823.14	785.56	4.57%	7200

Table 7.15: Comparison of the CSMH with some best algorithms for 42 instances in G1 (Data *set-4*)

Name (size)	*m*	*T*	MRT	SP	MA	MA + CLS	*Two-Level VNS*	CSMH
			Optimal	**Best**	**Best**	**Best**	**Best**	**Best**
CMT1 (50)	1	551	524.61	546.28	524.61	524.61	524.61	**524.61**
	2	275	533.00	x	533.00	533.00	533.00	**533.00**
	1	577	524.61	547.14	524.61	524.61	524.61	**524.61**
	2	289	529.85	549.42	529.85	529.85	529.85	**529.85**
	4	144	546.29	566.86	546.29	546.29	x	**546.29**
CMT2 (75)	1	877	835.26	869.06	835.26	835.26	835.26	**835.26**
	2	439	835.26	865.48	835.77	835.26	835.26	**835.26**
	3	292	835.26	x	835.26	835.26	835.26	**835.26**
	4	219	835.26	856.77	835.77	835.77	835.32	**835.26**
	5	175	835.80	x	836.18	836.18	837.40	**835.80**
	1	919	835.26	869.73	835.26	835.26	835.26	**835.26**
	2	459	835.26	881.50	835.26	835.26	835.26	**835.26**
	3	306	835.26	869.11	835.77	835.26	835.26	**835.26**
	4	230	835.26	880.90	838.17	835.26	835.26	**835.26**
	5	184	835.26	883.29	835.77	835.77	835.77	**835.26**
	6	153	839.22	x	843.09	839.22	x	**839.22**
CMT3 (100)	1	867	826.14	845.33	826.14	826.14	826.14	**826.14**
	2	434	826.14	850.65	826.14	826.14	826.14	**826.14**
	3	289	826.14	x	828.08	826.14	826.14	**826.14**
	1	909	826.14	845.33	829.45	829.45	828.26	**826.14**
	2	454	826.14	872.10	826.14	826.14	826.14	**826.14**
	3	303	826.14	869.48	826.14	827.39	826.14	**826.14**
	4	227	826.14	878.00	826.14	826.14	826.14	**826.14**
CMT11 (120)	1	1094	1042.11	1088.26	1042.11	1042.11	1072.95	**1042.11**
	2	547	1042.11	x	1042.11	1042.11	1073.96	1044.09
	3	365	1042.11	x	1042.11	1042.11	x	**1042.11**
	5	219	1042.11	x	1042.11	1042.11	x	**1042.11**
	1	1146	1042.11	1088.26	1042.11	1042.11	1075.83	**1042.11**
	2	573	1042.11	1110.10	1042.11	1042.11	1073.44	**1042.11**
	3	382	1042.11	1088.56	1042.11	1042.11	1085.28	1048.26
	4	287	1042.11	x	1042.11	1042.11	1062.30	1044.09
	5	229	1042.11	1092.95	1042.11	1042.11	1088.46	**1042.11**
CMT12 (100)	1	861	819.56	819.97	819.56	819.56	819.56	**819.56**
	2	430	819.56	821.33	819.56	819.56	819.56	**819.56**
	3	287	819.56	826.98	819.56	819.56	819.56	**819.56**
	4	215	819.56	824.57	819.56	819.56	819.56	**819.56**
	1	902	819.56	819.97	819.56	819.56	819.56	**819.56**
	2	451	819.56	829.54	819.56	819.56	819.56	**819.56**
	3	301	819.56	851.16	819.56	819.56	819.56	**819.56**
	4	225	819.56	821.53	819.56	819.56	819.56	**819.56**
	5	180	824.78	833.85	824.78	824.78	826.90	**824.78**
	6	150	823.14	855.36	823.14	823.14	827.14	**823.14**
# of solutions found (out of 42)				**33**	**42**	**42**	**38**	**42**
# of optimal solutions found				**0**	**33**	**37**	**26**	**39**
ARPD				**3.41**	**0.04**	**0.02**	**0.63**	**0.02**

MRT = Mingozzi *et al*. (2013); SP = Salhi and Petch (2007); MA and MA+CLS = Cattaruzza *et al.* (2014a)

7.3. Summary

This chapter presents the details of the implementations of our developed approaches in Chapters 4, 5 and 6 to solve two very important variants of the VRP, known as the vehicle routing problem with backhauls (VRPB) and the multiple trip vehicle routing problem (MT-VRP). One of the main objectives of this thesis is to design and implement an efficient and flexible algorithm that is able to solve the instances of a range of VRP variants. The *Two-Level VNS* methodology and its combination with mathematical programming the CSMH algorithms proved very successful implementation. We summarise here our findings and the analysis for both the VRPB and the MT-VRP, respectively as follows.

The VRPB:

In this chapter firstly the VRPB is formulated and its validity is checked using CPLEX.

The *Two-Level VNS* methodology developed for the MT-VRPB is then adapted to solve the VRPB. The VNS algorithm proved robust in its implementation since it was designed in such way that could be implemented on the instances of a range of VRP variants. The neighbourhood moves are conducted in the same conventions and the refinement schemes used in the same order remain unchanged. However the algorithm needed some minor changes at its initial solution stage due to different typical VRPB constraints of utilizing given number of vehicles; and the BPP implementation was not required for the VRPB.

The algorithm produced highly competitive results for both benchmark data sets when compared to the best known solutions from the literature, with an overall average

relative percentage deviation ARPD of 0.00 and 0.06 for the *set-2* and the *set-3*, respectively, while spending relatively lower computer times.

We then adapted the CSMH algorithm to solve the VRPB. The algorithm proved quite flexible in its implementation. The neighbourhood moved are implemented in the same conventions and the refinement routines used in the same order remain unchanged. However, the algorithm was slightly changed to accommodate the typical VRPB constraints of using the given fixed fleet and the MT-VRPB mathematical model used at *Phase IV* was replaced with the VRPB mathematical model. Hence, BPP is also removed from the algorithm as it is not needed for the VRPB. The algorithm produced very competitive results for the benchmark data sets when compared with the best algorithms in the literature.

The MT-VRP:

The MT-VRP is also formulated and the validity is checked using CPLEX.

The *Two-Level VNS* methodology is also adapted to solve the MT-VRP without any significant changes to the original algorithm developed to solve the MT-VRPB in Chapter 5. The conventions of the neighbourhood moves and the order of the refinement schemes remain unchanged. However the algorithm needed some changes at its initial solution stage due to no backhauling aspect in the MT-VRP. The initial solution is generated with sweeping for complete routes instead of open routes. The BPP and Bisection models are used in the same manner.

The algorithm produced quite competitive results (especially in terms of the computational speed) for the benchmark data *set 4* when compared with the best known

solutions from the literature. Also the *Two-Level VNS* algorithm produced one new best heuristic solution.

Moreover, the CSMH algorithm is also adapted and tested on a group of MT-VRP instances for which the optimal solutions are known. Minor changes are done such as sweep is used to generate complete routes instead of open routes at initial solution stage and a series of refinement routes are used to improve the initial solution before passing it to the second stage. All neighbourhood moves and refinement routines are used in the same order. The BPP and Bisection models are used in the same manner. Finally, MT-VRPB mathematical model used at stage four is replaced with the MT-VRP mathematical model.

The solutions produced by the CSMH algorithm are of a high quality. It outperformed all the heuristic/meta-heuristic algorithms and proved extremely competitive when compared with the exact algorithm of Mingozzi *et al*. (2013).

The successful implementation of the *Two-Level VNS* and its combined version with mat-heuristic the CSMH algorithm on the three VRP models proves the generalizability and robustness of this methodology.

Chapter 8

Conclusions

In this chapter we summarize the main findings and the contributions of the research along with some future research directions

8.1. Research Summary

With the growing and more accessible computational power, the demand for robust and sophisticated computerised optimisation has increased for logistical problems. By making a good use of computational technologies, the research in this thesis has mainly concentrated on efficient fleet management by studying a class of vehicle routing problems and developing software embedded efficient solution algorithms.

The research in this thesis starts by looking at the existing literature of the VRPs from various development angles. From the problem modelling side, clear efforts can be seen to bring the classical VRP models closer to the reality by developing their variants. However, apart from the real VRP applications (termed as 'rich' VRPs), it is also noticeable that the most of these classical VRP based variants address one or two additional characteristics from the real routing problem issues, concentrating on either operational or tactical aspects. Although the research in this thesis may not be

considered as comprehensive either but it is certainly one of those good efforts that bring the VRPs closer to the reality by addressing both the operational as well as tactical aspects.

On the solution methodologies development side, there are enormous and impressive developments. Having established that the VRPs are *NP* hard combinatorial class of problems, there is an ample effort on the development of exact methods. The literature covers a variety of heuristics methodologies including the classical and the most modern ones. The literature also points out towards some works being developed in hybridisation of heuristics approaches including the most recent *mat-heuristics* that combine heuristics and exact methods. The *mat-heuristics* appears to be comparatively in its infant age at this point in time. Hence, a part of the research in this thesis is devoted in the development of a hybrid approach that combines heuristics and mathematical programming techniques.

When reviewing the specific literature on the VRP problems focused in this thesis, the VRPB and the MT-VRP, there is not sufficient development on the problem modelling side in terms of bringing these problems closer to the reality. As for the methodological development to solve the VRPB and the MT-VRP there are some very successful efforts. For the VRPB, the literature records some early attempts in late 90s to solve the problem optimally though with a modest success. However, there are quite a few promising methodologies developed to solve this problem, divided in early traditional heuristic studies able to solve bigger instances of the problem with good enough solutions at the cost of reasonable computational efforts; and the more recent modern heuristics based algorithms able to perform much better in terms of solution quality but at noticeably higher computational costs. For the MT-VRP, there are some good studies

published in the literature, however as compared to the VRPB it has not drawn sufficient attention. The literature reports only one attempt on exact approach side; and several but comparatively less efficient heuristics works. One reason for this that could be deduced is that the MT-VRP is more closely related to the classical VRP which has been studied extensively in the literature. Hence there are more relevant works rather direct comparison studies of the MT-VRP. To fill the gap, the research in this thesis adds to the literature by investigating this problem directly and jointly with the VRPB.

To investigate these versions of the VRP jointly we introduced a new variant called the Multiple Trip Vehicle Routing Problem with Backhauls (MT-VRPB) which remain the main focus of the thesis. The problem is thoroughly described and an ILP mathematical formulation of the MT-VRPB along with its possible variations presented. The MT-VRPB is then solved optimally by using CPLEX along with providing an illustrative example showing validation of the formulation. A large set of MT-VRPB data instances is created which can be used for future benchmarking.

The CPLEX implementation produced optimal solutions for small and medium size data instances of the MT-VRPB and generated lower bounds for all instances. Although CPLEX found a good number of optimal solutions and lower bounds for all the instances, this success may be considered merely as modest. However, the results produced by CPLEX proved very important for validation of the results produced by the heuristic methodologies later in the thesis.

The MT-VRPB results show some big overall cost savings could be obtained by deciding the right fleet size and better vehicle utilisations with multiple trips and

backhauling. Hence, even at this point in thesis the results already prove the justification of studying the multiple trips and the backhauling aspects combined.

Hence the research results reveal some vital information and implications from the managerial point of view in terms of making the tactical (acquisition) and fleet management (operational) decisions.

As observed earlier the optimisation techniques could not cope with the larger instances of such hard complex problem, and relying on heuristics is an obvious choice. Hence we developed a two level VNS algorithm, called '*Two-Level VNS'* to solve the MT-VRPB. The choice of using VNS for the VRPs has increased in recent literature due its simplicity and speed. The *Two-Level VNS* algorithm uses skeletons of the classical VNS and VND methodologies. A number of neighbourhoods and local searches are employed in an innovative way to achieve diversification at the outer level (basic VNS) of the algorithm and intensification at the inner-level (VND with multi-layer local search framework). The *Two-Level VNS* algorithm found very encouraging solutions when compared with the solutions found by CPLEX. It matched the majority (87%) of the optimal solutions ranging in size 21-50. The *Two-Level VNS* solved all the 168 instances (105 feasibly with no overtime used); and the rest with a very small average overtime of only 5 and 10 units each for T_2 and T_1 data classes, respectively. Moreover, the speed of the algorithm remained outstanding spending less than 20 seconds on average per problem instance. These findings demonstrate the power of VNS yet again in terms of its speed, simplicity and efficiency.

The *Two-Level VNS* algorithm found a very high number of feasible solutions costing low computational time proving itself for what it is known in the literature. Nonetheless

we wanted to investigate it further with the new class of the hybrid methodologies called *mat-heuristics* that combines mathematical programming techniques with heuristic methods to solve CO problems. Hence, in Chapter 6, a hybrid collaborative sequential *mat-heuristic* approach called the CSMH to solve the MT-VRPB is developed. The exact method approach developed in Chapter 4 is hybridised with the *Two-Level VNS* algorithm developed in Chapter 5. The *Two-Level VNS* used three phases, i.e., initial solution by a modified *sweep-first-assignment-second* approach, improved solution by VNS, and packed solution by the BPP. Here the fourth phase, i.e., mathematical model is incorporated in the *Two-Level VNS* algorithm to find optimal/better solution for the MT-VRPB. The overall performance of the CSMH remained very inspiring in terms of the solution quality and the time taken on average. Comparing with the methodologies developed in the previous chapters (i.e., CPLEX and the *Two-Level VNS* meta-heuristic), the CSMH produced much better results on almost all fronts. As compared to CPLEX it produced a higher number of optimal solutions with bigger size instances and tighter lower bounds while spending lower computation time on average. Comparing with the *Two-Level VNS* it also produced better quality solutions with a higher number of optimal/incumbent on the expense of spending understandably larger average computing time.

Towards the end of the thesis, we tested our developed methodologies on the two versions of the VRP (VRPB and MT-VRP) mentioned in the beginning of this section. The reason of conducting these experiments was to see how far we have been successful in achieving one of the main objectives of the thesis which is to design and implement new efficient hybrid meta-heuristic/mat-heuristics algorithms that is able to solve a range of VRP variants.

In Chapter 7 a three-indexed mathematical formulation of the VRPB adapted from our MT-VRPB formulation is presented; and its validity is checked using CPLEX. Note that the complexity of the two-indexed VRPB formulations presented in the literature is not provided, however it is considered to be less complex as compared to our three-indexed ILP formulation. Moreover, the two-indexed ILP formulations were not directly tested with CPLEX, hence we did not compare the efficiency of these two types of formulations. We implemented the *Two-Level VNS* algorithm, developed in Chapter 5, to solve the VRPB. The VNS algorithm proved robust in its implementation since it was designed in such way that could be implemented on the instances of a range of VRP variants. The neighbourhood moves are conducted with the same conventions and the order of the refinements remain unchanged. However the algorithm needed some minor changes at its initial solution stage due to some different typical VRPB constraints such as 'must utilisation' of the given number of vehicles and disabling the use of the BPP implementation that is not required for the VRPB. The algorithm produced very competitive results for both benchmark data sets when compared to the best known solutions from the literature, with an overall average relative percentage deviation ARPD of 0.00 and 0.06 for the *set-2* and the *set-3*, respectively.

The CSMH algorithm of Chapter 6 is also tested for the VRPB. The implementation remained fairly straight forward by replacing the formulation and VNS parts of the MT-VRPB with the VRPB ones. The algorithm produced competitive results for the benchmark data sets when compared with the *Two-Level VNS* and the best algorithms in the literature. However, it was noted that the performance of the CSMH remain relatively inferior due the reason that this version of the VRPB uses a typical constraint

of fixed number of vehicles that must utilised which did not go well with the exact method part of the algorithm.

Moreover, in Chapter 7 we solved the classical MT-VRP. First, a three-indexed mathematical formulation of the MT-VRP adapted from our MT-VRPB formulation is presented; and its validity is checked using CPLEX. The *Two-level VNS* methodology is then implemented to solve the MT-VRP again without any significant changes to the original algorithm developed to solve the MT-VRPB in Chapter 5. Apart from the backhauling conventions that needed changes at the initial stage, the neighbourhood moves and the order of the refinement schemes remain unchanged. Here the initial solution is generated by sweeping for complete routes instead of open routes. The BPP and Bisection models are used in the same manner. The algorithm produced very competitive results (in terms of the solution quality and the computational speed) for the benchmark data *set 4* when compared with the best known solutions from the literature.

Lastly, the CSMH algorithm is tested for the MT-VRP. The implementation remained once again fairly straight forward by replacing the formulation and VNS parts of the MT-VRPB with the VRPB ones. The CSMH algorithm is tested on a group of MT-VRP instances for which the optimal solutions are known. The solutions produced by the CSMH algorithm are of very high quality. It outperformed all the previously published heuristic/meta-heuristic algorithms and proved extremely competitive matching most solutions when compared with the exact algorithm of Mingozzi *et al*. (2012). Matching most solutions with the only existing exact algorithm for this problem in the literature can be considered as significant development.

It can be observed that the successful implementation of the *Two-Level VNS* and the CSMH algorithms on the three VRP models with some trivial amendments prove their generalizability and the robustness.

8.2. Future Research

There are a number of ways in which the research in this thesis could be taken further.

Model extensions:

We hope to bring the MT-VRPB model even closer to reality by incorporating further "rich" aspects, such as time windows, multiple depots or heterogeneous fleet. We believe that the most promising aspect is to extend the backhauling part to other delivery and pickup models, as the "deliveries first, backhauls second" constraint is in our opinion very restrictive. The VRPB is a specific case of VRP with Deliveries and Pickups (VRPDP) models. If we remove the "deliveries first, backhauls second" restriction, we arrive at another model known as VRP with Mixed Deliveries and Pickups (VRPMDP). It is relatively easy to adapt the methods in this thesis for the VRPMDP, as in the main part of the algorithm we merely need to skip the steps of checking that no backhauls precede any deliveries. (The "fixed fleet utilisation" constraint is also removed in all VRPDP models apart from the VRPB.) However, we instead need to check the feasibility of routes for every arc on the route. This is due to the issue of fluctuating arc loads; see Wassan *et al*., (2008a, 2008b). Moreover, the initial solution is based on matching linehauls and backhauls, so we need to experiment whether this is still a sufficiently good initial solution. Another relevant model is the VRP with Simultaneous Deliveries and Pickups (VRPSDP). In this model each

customer may send and receive goods, so they are linehauls and backhauls in one. Conceptually, the VRPSDP and the VRPMDP differ little.

The above models, and the VRPB itself, have been criticised for the assumptions they require. It is considered excessively restrictive not to allow any backhauls before linehauls. Yet, the VRPMDP where this assumption is removed lead to the "load shuffling problem" (backhaul goods block access to linehaul goods on board in the vehicles), see Wassan and Nagy (2014) for a more detailed explanation. This led to a new model known as VRP with Restricted Mixing of Deliveries and Pickups, see Nagy *et al*., (2013). In this model some free space is required to maintain access to goods, unless the vehicle has only linehaul or only backhaul goods. Likewise, the VRPSDP makes the assumption that the linehaul and backhaul needs of a customer must be served in a single visit. Relaxing this leads to the model of VRP with Divisible Deliveries and Pickups (VRPDDP), see Nagy *et al*., (2015). Perhaps the most realistic version would be the VRP with Restricted Mixing of Divisible Deliveries and Pickups, as suggested in Wassan and Nagy (2014). All these models could be enhanced to include the multi-trip aspect. For further information on the various VRPDP models mentioned here, please refer to the overview provided by Wassan and Nagy (2014).

Methodological extensions:

We believe that the performance of our developed methods can be enhanced by hybridisation of tabu search or some other learning based meta-heuristics such as adaptive memory programming, reactive search mechanisms with VNS can enhance the quality of results though possibly at some extra computational cost.

In the near future we hope to continue the existing research work to investigate the further power of mat-heuristics. For instance, currently our CSMH algorithm uses only single pass between the *Two-Level VNS* and the mathematical programming technique. Nonetheless, it produced very interesting results for variety of VRP problems. However, we believe the efficiency of our CSMH algorithm can be increased by incorporating tabu search/learning aspects in the heuristics side of the algorithm. The performance of the developed mat-heuristic can be enhanced by designing a cyclic algorithm that exchanges information and interplays between the heuristic and the exact techniques.

We believe that further investigation of some of the key aspects highlighted above on both the modelling and the solutions methodologies side would achieve even better overall fleet management efficiency that is required in modern day business and environmental needs.

Bibliography

Ahlem, C., Racem, M. and Habib, C. (2011). Profitable vehicle routing problem with multiple trips: modelling and constructive heuristics. *IEEE*, 500-507.

Ahmadi, S. and Osman, I. (2005). Greedy random adaptive memory programming search for the capacitated clustering problem. *European Journal of Operational Research*, 162(1), 30-44.

Ahuja, R.K., Ergun, Ö., Orlin J.B. and Punnen, A.P. (2002). A survey of very large scale neighbourhood search techniques. *Discrete Applied Mathematics*, 123, 75-102.

Aleman, R. E. (2009). *A guided Neighbourhood Search Applied to the Split Delivery Vehicle Routing Problem*. Wright State University. PhD Thesis.

Alonso, F., Alvarez, M.J. and Beasley, J.E. (2008). A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. *Journal of the Operational Research Society*, 59(7), 963-976.

Altınel I K and Öncan T (2005). A new enhancement of the Clarke and Wright savings heuristic for the capacitated vehicle routing problem. *Journal of the Operational Research Society*, 56, 954-961.

Anily, S. (1996). The vehicle routing problems with delivery and back-haul options. *Naval Research Logistics*, 43, 415-434.

Archetti, C., Speranza, M.G. and Savelsbergh, M.W.P. (2008). An optimisation-based heuristic for the split delivery vehicle routing problem. *Transportation Science*, 42, 22-31.

Azi, N., Gendreau, M. and Potvin, J. (2010a). An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research*, 202, 756-763.

Azi, N., Gendreau, M. and Potvin, J. (2010b) An adaptive large neighbourhood search for a vehicle routing problem with multiple trips. *Computers and Operations Research*, 41, 167-173.

Bai, Y., Zhang, W. and Jin, Z. (2005) A new self-organizing maps strategy for the solving the travelling salesman problem. *Chaos, Solution and Fractals*, 28 (1), 1082-1089.

Baker, B. M. (1992). Further improvements to vehicle routing heuristics. *Journal of the Operational Research Society*, 43, 1009–1012.

Baldacci, R., Hadjiconstantinou, E. and Mingozzi, A. (2004). An exact algorithm for the capacitated vehicle routing problem based on a two-commodity network flow formulation. *Operations Research*, 52(5), 723-738.

Baptista, S., Oliveira, R. and Zuquete, E. (2002). A period vehicle routing case study. *European Journal of Operational Research*, 139, 220-229.

Battara, M., Monaci, M. and Vigo, D. (2009). An adaptive guidance approach for the heuristic solution of a minimum multiple trip vehicle routing problem. *Computers and Operations Research*, 36, 3041-3050.

Bramel, J. and Simchi-Levi, D. (1995). A location based heuristic for general routing problems. *Operations Research*, 43:649-660.

Beasley, J. (1983). Route-first cluster-second methods for vehicle routing. *OMEGA*, 11(4) 403-408.

Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, 34, 209-219.

Belenguer, J., Martinez, M. and Mota E. (2002). A lower bound for the split delivery vehicle routing problem. *Operational Research*, 48, 801-810.

Bin, Y., Zhen, Y.Z. and Baozhen, Y. (2009). An improved ant colony optimisation for vehicle routing problem. *European Journal of Operational Research*, 196, 171-176.

Blakely, F., Bozkaya, B., Cao, B., Hall, W. and Knolmajer, J. (2003). Optimizing periodic maintenance operations for Schindler Elevator Corporation. *Interfaces*, 33 (1), 67-79.

Blum, C., Puchinger, J., Raidl, G.R. and Roli, A. (2011): Hybrid metaheuristics in combinatorial optimisation: A survey. *Soft Computing*, 11(6), 4135-4151.

Bock, F. (1958). An Algorithm for Solving "Traveling-Salesman" and Related Network Optimisation Problems. *14th ORSA National Meeting*, St. Louis, MO.

Bodin, L., Golden, B., Assad, A. and Ball, M. (1983). Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, 10 (2), 63-211.

Bolduc, M. C., Laporte, G., Renaud, J. and Boctor, F. F. (2010). A tabu search heuristic for the split delivery vehicle routing problem with production and demand calendars. *European Journal of Operational Research*, 202 (1), 122-130.

Boussïd I, Lepagnot, J, and Siarry P. (2013). A survey on Optimisation metaheuristics. *Information Sciences*, 237, 82-117.

Brandao, J. (2006). A new tabu search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 173(2), 540-555.

Brandao, J. and Mercer, A. (1997). A tabu search algorithm for the multi-trip vehicle routing and scheduling problem. *European Journal of Operational Research*, 100, 180-191.

Brandao, J. and Mercer, A. (1998). The multi-trip vehicle routing problem. *Journal of the Operational Research Society*, 49, 799-805.

Braysy, O. and Gendreau, M. (2005). Vehicle Routing Problem with Time Windows, Part I: Route Construction and Local Search Algorithms. *Transportation Science*, 39(1), 104-118.

Braysy, O. and Gendreau, M. (2005). Vehicle Routing Problem with Time Windows, Part II: Metaheuristics. *Transportation Science*, 39(1), 119-139.

Ball, M. (2011). Heuristics based on mathematical programming. Survey in Operations Research and Management Science, 16, 21-38.

Carpaneto, G. and Toth, P. (1980). Some New Branching and Bounding Criteria for the Asymmetric Travelling Salesman Problem. *Management Science,* 26(7), 736-743.

Casco, D., Golden, B.L., and Wasil, E. (1988). Vehicle routing with backhauls: Models, algorithms and case studies. In: Golden, B.L. and Assad, A.A. eds. *Vehicle Routing: Methods and Studies,* Elsevier: Amsterdam, pp. 127-147.

Caserta, M. and Voß, S. (2010). Metaheuristics: Intelligent problem solving. In: Maniezzo, V., et al., eds. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Volume 10 of Annals of Information Systems, Springer: pp. 1-38.

Cattaruzza, D., Absi, N., Feillet, D. and Vidal, T. (2014a). A memetic algorithm for the multi trip vehicle routing problem. *European Journal of Operational Research*, 236(3), 833-848.

Cattaruzza, D., Absi, N., Feillet, D. and Vigo, D. (2014b). An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. *Computers and Operations Research*, 51, 257-267.

Černý, V. (1985). A Thermodynamical Approach to the Travelling Salesman Problem: And Efficient Simulation Algorithm. *Journal of Optimisation Theory and Applications*, 45, 41-51.

Chao, I, Golden, B. and Wasil, E. (1995). An improved heuristic for the period vehicle routing problem. *Networks*, 26, 25-44.

Chao, I-M., Golden, B. and Wasil, E. (1995). A computational study of a new heuristic for the site-dependant vehicle routing problem. *INFOR*, 37, 319-336.

Chen, J.F and Wu, T.H. (2006). Vehicle routing problem with simultaneous delivery and pickups. *Journal of the Operational Research Society,* 57, 579-587.

CLHO: Centre for Logistics and heuristic optimisation, (2015) Available from: http://www.kent.ac.uk/kbs/research/research-centres/clho/ [Accessed 23 March 2015].

Christofides, N. and Beasley, J. (1984) The period routing problem. *Network*, 14, 237-256.

Christofides, N. and Eilon, S. (1969). An algorithm for the vehicle dispatching problems. *Operations Research Quarterly*, 20 (3), 309-318.

Christofides, N., Mingozzi, A. and Toth, P. (1979) The vehicle routing problem. In: Christofides et al., eds. *Combinatorial Optimisation.* Chichester: Wiley, pp. 315-338.

Christofides, N., Mingozzi, A. and Toth, P. (1981a). Exact algorithms for the vehicle routing problem based on spanning tree and shortest path relaxations. *Mathematical Programming*, 20, 255-282.

Christofides, N., Mingozzi, A. and Toth, P. (1981b). Space state relaxation procedures for the computation of bounds to routing problems. *Networks*, 11, 145-164.

Clarke, G. and Wright, J. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, 12, 568-581.

Colorni, A., Dorigo, M. And Maniezzo, V. (1991). Distributed Optimisation by Ant Colonies. *Appeared in Proceedings of ECAL91: European Conference in Artificial Life*, Paris, France, pp. 134-142.

Cook, S.A (1971). On the complexity of theorem-proving procedures. In: *Third Annual ACM Symposium on Theory of Computing*, New York, pp. 151-158.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational Research Society*, 52, 928-936.

Cordeau, J.-F., Laporte, G., and Mercier, A. (2004). Improved tabu search algorithm for the handling of route duration constraints in vehicle routing problems with time windows. *Journal of the Operational Research Society*, 55, 542–546.

Cordeau, J.-F. and Laporte, G. (2001). A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR*, 39, 292-298.

Cordeau, J.-F., Gendreau, M. and Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problem. *Networks*, 30, 105-119.

Croes, G. (1958). A Method for Solving Traveling Salesman Problems. *Operations Research*, 6, 791-812.

Cuervo, D.P., Goos, P., Sorensen, K. and Arraiz, E. (2014). An iterated local search algorithm for the vehicle routing problem with backhauls. *European Journal of Operational Research*, 237(2), 454-464.

Cvijovic, D. and Klinowski, J. (1995). Taboo Search - An Approach to the Multiple Minima Problem. *Science*, 267, 664-666.

Dantzig, G.B. and Ramser, J.H. (1959). The truck dispatching problem. *Management Science*, 6, 80-91.

Deif, I. and Bodin, L. (1984). Extension of the Clarke and Wright algorithm for solving the vehicle routing problem with backhauling. In: Kidder, A. ed. *Proceedings of the Babson conference on Software uses in Transportation and Logistics Management*. Babson Park, pp. 75-96.

Derigs, U., Kurowsky, R. and Vogel, U. (2011) Solving a real-world vehicle routing problem with multiple use of tractors and trailers and EU-regulations for drivers arising in air cargo road feeder services. *European Journal of Operational Research*, 213, 309-319.

Derigs, U., Li, B. and Vogel, U. (2010). Local search-based metaheuristics for the split delivery vehicle routing problem. *Journal of the Operational Research Society*, 61, 1356-1364.

Desrochers, M., Desrosiers, J. and Solomon, M. (1992). A New Optimisation Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, 40(2), 342-354.

De Franceschi, R., Fischetti, M., and Toth, P. (2006). A new ILP-based refinement heuristic for vehicle routing problems. *Mathematical Programming*, 105:471-499.

DFT: Department of Transport, UK, Quarterly Road Traffic Estimates (n.d.). [Online]. Available from: https://www.gov.uk/government/statistics/road-traffic-estimates-for-great-britain-january-to-march-2015) [Accessed on 28/05/2015].

Dondo R, and Cerdá, J. (2006). A cluster-based optimisation approach for the multi-depot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research*, 176, 1478-1505.

Dorigo, M. (1992). *Optimisation, Learning and Natural Algorithms*. Milano: Politecnico di Milano.

Dorigo, M. and Stützle, T. (2004). *Ant Colony Optimisation*. MIT Press.

Dror, M. and Trudeau, P. (1989). Savings by split delivery routing. *Transportation Science*, 23, 141-145.

Dror, M. and Trudeau, P. (1990) Split delivery routing. *Naval Research Logistics*, 37, 383-402.

Dueck, G. (1993). New Optimisation Heuristics: The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, 104(1), 86-92.

Dueck, G. and Scheuer, T. (1990). Threshold Accepting: A General Purpose Optimisation Algorithm Appearing Superior to Simulated Annealing. *Journal of Computational Physics*, 90(1), 161-175.

Durbin, R. and Willshaw, D. (1987). An Analogue Approach to the Travelling Salesman Problem Using an Elastic Net Method. *Nature*, 326, 689-691.

Eilon S and Christofides N (1971). The Loading Problem. *Management Science*, 17(5), 259-268.

Eiselt, H. A., Gendreau, M. and Laporte, G. (1995). Arc Routing Problems, Part I: The Chinese Postman Problem. *Operations Research*, 43(2), 231-242.

Ergun, Ö., Orlin, J.B. and Steele-Feldman, A. (2006) Creating very large scale Neighbourhoods out of smaller ones by compounding moves. *Journal of Heuristics*, 12, 115-140.

Fischetti, M., Toth, P. and Vigo, D. (1994). A branch-and-bound algorithm for the capacitated vehicle routing problem on directed graphs. *Operations Research*, 42, 846-859.

Foster, B. and Ryan, D. (1976). An integer programming approach to the vehicle scheduling problem. *Operational Research Quarterly*, 27:367-384.

Fisher, M.L. (1994). Optimal solutions of vehicle routing problems using minimum k-trees. *Operations Research*, 42, 626-642.

Fisher, M.L. and Jaikumar, R. (1981). A generalized assignment heuristic for vehicle routing. *Networks*, 11, 109-124.

Fleischmann, B. (1990). The vehicle routing problem with multiple use of vehicles. Working paper, Fachbereich Wirtschaftswissenschafte, Universidad Hamburg, Hamburg, Germany.

Fleszar, K., Osman, I. H. and Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195, 803-809.

Flood, M. M. (1956). The Traveling-Salesman Problem. *Operations Research*, 4(1), 61-75.

Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966) *Artificial Intelligence Through Simulated Evolution*. Wiley, New York.

Fukasawa, R. et al. (2003). Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Technical Report RPEP Vol.3. no.8, Uiversidade Federal Fluminense, Engenharia de Producao, Niteroi, Brazil.

Gajpal, Y. and Abad, P.L. (2009). Multi-ant colony system (MACS) for a vehicle routing problem with backhauls. *European Journal of Operational Research*, 196, 102-117.

Gamboa, D., Rego, C. and Glover, F. (2006). Implementation analysis of efficient heuristic algorithms for the traveling salesman problem. *Computers and Operations Research*, 33, 1154-1172.

Gribkovskaia I., Gullberg B.O., Hovden K.J., Wallace S.W. (2006) Optimization model for a livestock collection problem, *International Journal of Physical Distribution & Logistics Management*, 36, 136-52.

Goel A., Gruhn V. (2008) A General Vehicle Routing Problem, *European Journal of Operational Research*, 191, 650-660.

Ganesh, K. and Nallathambi, A.S. (2007). Variants, solution approaches and applications for vehicle routing problems in supply chain: agile framework and comprehensive review. *International Journal of Agile Systems and Management*, 2 (1), 50-72.

Ganesh, K. and Narendran, T. T. (2007) ClOVES: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. *European Journal of Operational Research*, 178, 699-717.

Gavish, B. and Graves, S.C. (1982). Scheduling and routing in transportation and distribution systems: formulations and new relaxations. Grad. Sch. Management, Univ. Rochester, Rochester, NY, Working Paper.

Gendreau, M., Hertz, A. and Laporte, G. (1992). New insertion and post-optimisation procedures for travelling salesman problem. *Operations Research*, 40 (6), 1086-1094.

Gendreau, M., Hertz, A. and Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem, *Management Science*, 40 (10), 1276-1290.

Ghaziri, H. (1991). Solving Routing Problems by a Self-Organizing Map. *In*: Kohonen, T., Makisara, K., Simula, O. and Kangas, J. eds. *Artificial Neural Networks*. North-Holland, Amsterdam, pp. 829-834.

Ghaziri, H. (1996). Supervision in the Self-Organizing Feature Map: Application to the Vehicle Routing Problem. *In*: Osman, I. H. and Kelly, J. P. eds. *Meta-Heuristics: Theory and Applications*. Boston: Kluwer, pp. 651-660.

Ghaziri, H. and Osman, I.H. (2003). A neural network algorithm for travelling salesman problem with backhauls. *Computers and Industrial Engineering*, 44, 267-281.

Ghaziri, H. and Osman, I.H. (2006). Self-organizing feature maps for the vehicle routing problem with backhauls. *Journal of Scheduling*, 9, 97-114.

Gillet, B.E. and Miller, L.R. (1974). A heuristics algorithm for the vehicle dispatch problem. *Operations Research*, 22, 340-349.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers and Operations Research*, 13(5), 533-549.

Glover, F. (1989). Tabu search-Part I. *Journal of Computing (ORSA)*, 1, 190-206.

Glover, F. (1989). Tabu search-Part II. *Journal of Computing (ORSA)*, 2, 4-32.

Glover, F. (1990). Tabu search: A tutorial. *Interfaces*, 20, 74-94.

Glover, F. (1991). *Multilevel Tabu Search and Embedded Search Neighbourhoods for the Traveling Salesman Problem*. Graduate School of Business and Administration, University of Colorado at Boulder, pp. 1-80.

Glover, F. (1992). New ejection chain and alternating path methods for travelling salesman problems. *Computer Science and Operations Research*, 449-509.

Glover, F. and Laguna, M. (1993). Tabu Search. *In*: Reeves, C. R. ed. *Modern Heuristic Techniques for Combinatorial Problems*. Oxford: Blackwell, pp. 70-150.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Norwell, MA: Springer.

Goetschalckx, M. and Jacobs-Blecha, C. (1989). The vehicle routing problem with backhauls. *European Journal of Operational Research*, 42, 39-51.

Golden, B. L., Bodin, L., Doyle, T. and Stewart Jr., W. (1980). Approximation Traveling Salesman Algorithms. *Operations Research*, 28(3, Part II), 694-711.

Golden, B., Assad, A., Levy, L. and Gheysens, F. (1984). The fleet size and mix vehicle routing problem. *Computers and Operations Research*, 11, 49-66.

Golden, B., Magnanti, T.L. and Nguyen, H.Q. (1977). Implementing vehicle routing algorithms. *Networks*, 7, 113-148.

Golden, B.L., Baker, E.K., Alfaro, J.L., and Schaffer, J.R. (1985). The Vehicle Routing Problem with Backhauling: Two Approaches, working paper MS/S 85-017, University of Maryland, College Park.

Garey, M.R. and Johnson, D.S. (1979). Computers and Intractability, A guide to the Theory of NP-completeness. Freeman, W.H., San Francisco, CA.

Hahsler, M. and Hornik, K. (2006). TSP - Infrastructure for the Traveling Salesperson Problem, Research Report Series, 45. Department of statistics and mathematics, WU Vienna University of Economics and Business, Vienna.

Halse, K. (1992) *Modelling and Solving complex Vehicle Routing Problems*. Published thesis (PhD), Technical University of Denmark.

Hemmelmayr, V.C., Doerner, K.F. and Hartl, R.F. (2009). A variable neighbourhood search heuristic for periodic routing problem. *European Journal of Operational Research*, 195 (3), 791-802.

Hertz, A. and de Werra, D. (1990). The Tabu Search Metaheuristic: How We Used It. *Annals of Mathematics and Artificial Intelligence,* 1(1-4), 111-121.

Ho, S. and Haugland, D. (2004). A Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows and Split Deliveries. *Computers and Operations Research*, 31, 1947-1964.

Hoffman, K. L. (2000). Combinatorial optimisation: Current successes and direction for future. *Journal of Computational and Applied Mathematics*, 124 (1-2), 341-360.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press.

Hopfield, J. J. and Tank, D. W. (1985). Neural Computation of Decisions in Optimisation Problems. *Biological Cybernetics*, 52, 141-152.

IBM Knowledge Center (2015). Concert Technology tutorial for C++ users [Online]. Available from: http://www-01.ibm.com/support/knowledgecenter/#!/SSSA5P_12.5.1/ilog.odms.cplex.help/CPLEX/GettingStarted/topics/tutorials/Cplusplus/cpp_synopsis.html [Accessed 3 March 2015].

Imran, A., Salhi, S. and Wassan, N.A (2009). A variable neighbourhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, 197(2), 509-518.

Jarpa, G.G., Desaulniers, G., Laporte, G. and Marianov,V. (2010). A branch-and-cut algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, 206, 341-349.

Jin, M., Liu, K. and Bowden, R. (2007). A two-stage algorithm with valid inequalities for the split delivery vehicle routing problem. *International Journal of Production Economics*, 105, 228-242.

Jin, M., Liu, K. and Eksioglua, B. (2008). A column generation approach for the split delivery vehicle routing problem. *Operations Research Letters*, 36 (2), 265-270.

Jourdan. L, Basseur. M, and Talbi E-G. (2009). Hybridizing exact methods and metaheuristics: A survey. *European Journal of Operational Research*, 199, 620-629.

Karp, R.M. (1978). *A Patching Algorithm for the Nonsymmetric Traveling-Salesman Problem*. Electronics Research Laboratory, College of Engineering. Berkeley: University of California Berkeley.

Kinderwater, G.A.P., and Savelsbergh, M.W.P. (1997). Vehicle routing: Handling edge exchanges. In: Aarts, E.H.L., and Lenstra, J.K. eds. *Local Search in Combinatorial Optimisation*, Wiley, Chichester, pp. 337-360.

Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P. (1983). Optimisation by simulated annealing. *Science*, 220, 671-680.

Kleene, S.C. (1991). *Introduction to Metamathematics*. 10th ed.: North-Holland Publishing Company.

Kelly, J. and Xu, J. (1999). A set-partitioning-based heuristic for the vehicle routing problem. *INFORMS Journal on Computing*, 11:161-172.

Knuth, D.E. (1973). *The Art of Computer Programming*. Addison-Wesley.

Kohonen, T. (1988). *Self-Organization and Associative Memory*. Berlin: Springer.

Kowalski, R. (1979). Algorithm = Logic + Control. *Communications of the ACM*, 22(7), 424-436.

Koza, J. (1992). *Genetic Programming: On Programming Computers by Means of Natural Selection and Genetics*. MIT Press, Cambridge, MA.

Kennedy, J. Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks. pp.* 1942–1948.

Land, A.H. and Doig, A.G. (1960). An Automatic Method for Solving Discrete Programming Problems. *Econometrica*, 28, 497-520.

Laporte, G. (1992). The Vehicle Routing Problem: an overview of exact and approximate algorithms. *European Journal of Operational Research*, 59, 345-358.

Laporte, G., Gendreau, M., Potvin, J.Y. and Semet, F. (2000). Classical and modern heuristic for the vehicle routing problem. *International Transactions in Operational Research*, 7, 285-300.

Laporte, G., Nobert, Y. and Desrochers M. (1985). Optimal routing under capacity and distance restrictions. *Operations Research*, 33, 1050-1073.

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

Lawler, E.L., Lenstra, J.K. and Rinnooy Kan, A.H.G. (1985). *The Travelling Salesman Problem: a guided tour of combinatorial optimisation.* New York: Wiley.

Lewis, R., Song, X., Dowsland, K. and Thompson, J. (2011). An investigation into two bin packing problems with ordering and orientation implications. *European Journal of Operational Research*, 213(1), 52-65.

Lin, L. and Tao, L. (2011). Solving mixed vehicle routing problem with backhauls by adaptive memory programming methodology. In: *Third International Conference on Measuring Technology and Mechatronics Automation, 2011*, 310-313.

Liu, S-C. and Chung, C-H. (2009). A heuristic method for the vehicle routing problem with backhauls and inventory. *Journal of Intelligent Manufacturing*, 20(1), 29-42.

Lahyani R., Khemakhem M., Semet F. (2015) Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241, 1–14.

Markov, A.A. (1954). *Teoriya Algerifmov*. Moscow, Russia: [Translated by Jacques J. Schorr-Kon and PST staff], Imprint Moscow, Academy of Sciences of the USSR.

Matsuyama, Y. (1991). Self-Organization via Competition, Cooperation and Categorization Applied to Extended Vehicle Routing Problems. *Proceedings of the International Joint Conference on Neural Networks*, Seattle, WA, pp. 385-390.

Mester, D. and Braysy, O. (2007). Active-guided evolution strategies for large-scale capacitated vehicle routing problem. *Computers and Operations Research*, 34, 2964-2975.

Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pickup. *Transportation Research*, 23, 377-386.

Mingozzi, A., and Baldacci, R. (1999). An exact method for the vehicle routing with backhauls. *Transportation Science*, 33(3), 315-329.

Mingozzi, A., Baldacci, R. and Giorgi, S. (1996). An exact method for the vehicle routing problem with backhauls. *Department of Mathematics,* University of Bologna, Bologna, Italy.

Mingozzi, A., Roberti, R. and Toth, P. (2013). An Exact algorithm for the multi-trip vehicle routing problem. *INFORMS Journal on Computing*, 25(2), 193-207.

Mladenović, N. (1995). A Variable Neighbourhood Algorithm - A New Metaheuristic for Combinatorial Optimisation. *Presented at Optimisation Days, Montréal*.

Mladenović, N. and Hansen, P. (1997). Variable neighbourhood search. *Computers and Operations Research*, 24, 1097-1100.

Mladenović, N., Todosijević, R. and Urosević, D. (2014). Two level general variable neighbourhood search for attractive travelling salesman problem. *Computers and Operations Research*, 52, 341-348.

Mohamed, N.H.B. (2012) *Hybridisation of Heuristics and Exact Methods for the Split Delivery Vehicle Routing Problem*. Published thesis (PhD), University of Kent.

Moscato, P.A., and Cotta, C. (2003). A gentle introduction to memetic algorithms. In: Glover, F. and Kochenberger, G. eds. *Handbook of metaheuristics* Dordrecht, The Netherlands: Kluwer Academic Publishers, pp. 105-144.

Moyson, F., and Manderick, B. (1988). The collective behaviour of ants: An example of selforganization in massive parallelization. *Proceedings of the AAAI Spring Symposium on Parallel Models of Intelligence,* Stanford, CA.

Nag, B., Golden, B. and Assad, A. (1988). Vehicle routing with site dependencies. In: Golden, B. and Assad, A. eds. *Vehicle Routing: Methods and Studies*, Studies in Management Science and Systems, Volume 16. North-Holland: Amsterdam, The Netherlands, pp. 149-159.

Nagy, G. and Salhi, S. (1996). Nested heuristic methods for the location-routing problem. *Journal of the Operational Research Society,* 1166-1174.

Nagy, G. and Salhi, S. (2005). Heuristic algorithms for single and multiple depot vehicle routing problems with pickups and deliveries. *European Journal of Operational Research*, 162(1), 126-141.

Nagy, G. and Wassan, N.A. and Salhi, S. (2013). The Vehicle Routing Problem with Restricted Mixing of Deliveries and Pickups. *Journal of Scheduling*, 16(2), 199-213. ISSN 1094-6136.

Nagy, G., Wassan, N.A., Speranza, M.G. and Archetti, C. (2015). The Vehicle Routing Problem with Divisible Deliveries and Pickups. *Transportation Science*, 49(2), 271-294.

Olivera, A. and Viera, O. (2007). Adaptive memory programming for the vehicle routing problem with multiple trips. *Computers and Operations Research*, 34, 28-47.

Ong, J.O. and Suprayogi (2011). Vehicle Routing Problem with Backhauls, Multiple Trips and Time Windows. *Journal Teknik Industri*, 13(1), 1-10.

Or, I. (1976) *Travelling salesman-type combinatorial problems and their relation to the logistics of regional blood banking.* Northwestern University. Ph.D. Thesis.

Osman, I.H. (1991). *Metastrategy Simulated Annealing and tabu Search Algorithms for Combinatorial Optimisation Problems*. The Management School. London: Imperial College. Ph.D. Thesis.

Osman, I.H. (1993). Metastrategy simulated annealing and tabu search algorithms for the vehicle routing problem. *Annals of Operations Research*, 41, 421-451.

Osman, I.H. and Wassan, N.A. (2002). A reactive tabu meta-heuristic for the vehicle routing problem with back-hauls. *Journal of Scheduling*, 5, 263-285.

Petch, R.J. and Salhi, S. (2004). A multi-phase constructive heuristic for the vehicle routing problem with multiple trips. *Discrete Applied Mathematics*, 133, 69-92.

Piniganti, L. (2014). *A Survey of Tabu Search in Combinatorial Optimisation* [online]. Available from:

http://digitalscholarship.unlv.edu/cgi/viewcontent.cgi?article=3133&context=thesesdissertations [Access date 09/06/2015].

Pólya, G. (1945). *How to Solve It. Princeton University Press*. ISBN 0-691-08097-6.

Potvin, J., Kervahut, T., Garcia, B. and Rouseau, J. (1996). The vehicle routing problem with time windows - Part II: Genetic search. *INFORMS Journal on Computing*, 8, 165-172.

Potvin, J. (1993). The Traveling Salesman Problem: A Neural Network Perspective. *ORSA Journal of Computing*, 5, 328-348.

Prins, C. (2004). A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers and Operations Research*, 31, 1985-2002.

Puchinger, J. and Raidl, G.R. (2005). Combining metaheuristics and exact algorithms in combinatorial optimisation: A survey and classification. In: *Proceedings of the First International Work-Conference on the Interplay between Natural and Artificial Computation*, Part II. Volume 3536 of LNCS, Springer (2005), pp. 32-41.

Raidl, G.R. (2006). A unified view on hybrid metaheuristics. In: Almeida, F., Blesa, M.J., Blum, C., Moreno-Vega, J.M., Perez, M.M., Roli, A. and Sampels, M., eds. *Hybrid Metaheuristics*. Volume 4030 of Lecture Notes in Computer Science. Springer (2006), pp. 1-12.

Ralphs, T.K. (2003). Parallel Branch and Cut for Capacitated Vehicle Routing. *Parallel Computing*, 29, 607-629.

Rayward-Smith, V., Osman, I., Reeves, I. and Smith, G. (1996). *Modern Heuristic Search Methods*. Wiley, England.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog Verlag, Stuttgart

Reeves, C.R. eds. (1993). *Modern Heuristic Techniques for Combinatorial Problems.* Oxford: Blackwell Publications Ltd.

Rego, C. and Roucairol, C. (1996). Tabu search algorithm using ejection chains for the vehicle routing problem. In: *Metaheuristics: Theory & Applications,* pp. 661-675.

Reimann, M. (2004). D-Ants: Savings Based Ants Divide and Conquer the Vehicle Routing Problem. *Computers and Operations Research,* 31(4), 563-591.

Rei,W., Gendreau, M., and Soriano, P. (2010). A hybrid monte carlo local branching algorithm for the single vehicle routing problem with stochastic demands. *Transportation Science,* 44:136-146.

Renaud, J. and Boctor, F.F. (2002). A sweep-based algorithm for the fleet size and mix vehicle routing problem. *European Journal of Operational Research,* 140, 618-628.

Renaud, J., Boctor, F.F. and Laporte, G. (1996). An Improved Petal Heuristic for the Vehicle Routeing Problem. *Journal of the Operational Research Society,* 47(2), 329-336.

Renaud, J., Laporte, G. and Boctor, F.F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers and Operations Research*, 23(3), 229-235.

Rochat, Y. and Taillard, E. (1995). Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 1, 147-167.

Ropke, S and Pisinger, D. (2006). A unified heuristic for a large class of Vehicle Routing Problems with Backhauls. *European Journal of Operational Research*, 171(3), 750-775.

Ropke, S and Pisinger, D. (2006). *An adaptive large neighbourhood search heuristic for the pickup and delivery problem with time windows*. University of Copenhagen. Technical Report 04/14, DIKU.

Rosenkrantz, D., Stearns, R. and Lewis, P. (1974). (1977). An Analysis of Several Heuristics for the Traveling Salesman Problem. *SIAM Journal on Computing,* 6(5), 563–581.

Russell, R. and Igo, W. (1979). An assignment routing problem. *Networks*, 9, 1-17

Ryan, D.M., Hjorring, C. and Glover, F. (1993). Extensions of the Petal Method for Vehicle Routeing. *Journal of Operational Research Society*, 44(3), 289-296.

Salhi, S. (1987). *The integration of routing into the location-allocation and vehicle composition problem*. University of Lancaster. Ph.D. Thesis.

Salhi, S. and Nagy, G. (1999). A cluster insertion heuristic for the single and multiple depot vehicle routing problems with backhauling. *Journal of Operational Research Society*, 50, 1034-1042.

Salhi, S. and Petch, R.J. (2007). A GA based heuristic for the vehicle routing problem with multiple trips. *Journal of Mathematical Modelling and Algorithms*, 6-4, 591-316.

Salhi, S. and Rand, G.K. (1987). Improvements to vehicle routing heuristics. *Journal of Operational Research Society*, 38(3), 293-295.

Salhi, S. and Sari, M. (1997). A Multi-Level Composite Heuristic for the Multi Depot Vehicle Fleet Mix Problem. *European Journal of Operational Research,* 103, 95-112.

Salhi, S. and Wade, A. (2001). An ant system algorithm for the vehicle routing problem with backhauls. In: *4th International Conference on Metaheuristics, 16-20 July, 2001*. Porto: Portugal.

Salhi, S. and Wassan, N.A. and Hajarat, M., (2013). The Fleet Size and Mix vehicle Routing Problem with backhauls: Formulation and Set partitioning-based heuristics. *Transportation Research Part E*, 56, 22-35. ISSN 1366-5545.

Schmid, V., Doerner, K., Hartl, R., and Salazar-González, J. (2010). Hybridization of very large neighbourhood search for ready-mixed concrete delivery problems. *Computers and Operations Research*, 37:559-574.

Schmid, V., Doerner, K., Hartl, R., Savelsbergh, M., and Stoecher, W. (2009). A hybrid solution approach for ready-mixed concrete delivery. *Transportation Science*, 43:70–85.

Schrijver, A. (2003) *Combinatorial Optimisation: Polyhedra and Efficiency*. Berlin: Springer.

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H. and Dueck, G. (2000). Record breaking optimisation results-using the ruin and recreate principle. *Journal of Computational Physics,* 159, 139-171.

Schumann, M. and Retzko, R. (1995). Self-Organizing Maps for Vehicle Routing Problems - Minimizing an Explicit Cost Function. Fogelman-Soulie, F. ed. *Paris, Proceedings of the International Conference on Artificial Neural Networks,* pp. 401-406.

Sen, A. and Bulbul, K. (2008). A survey on multi trip vehicle routing problem. In: *VI. International Logistic and Supply Chain Congress, 6-7 November, 2008*. Istanbul, Turkey.

Sesiano, J. (1982). *Books IV to VII of Diophantus' Arithmetica: In the Arabic Translation Attributed to Qusta Ibn Luqa (Sources and Studies in the History of Mathematics and Physical Sciences)* Sept 1982. Springer .

Shaw, P. (1997). *A New Local Search Algorithm providing High Quality Solutions to Vehicle Routing Problems*. Department of Computer Science, University of Strathclyde, Scotland.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In: *Proceedings CP-98. Fourth International Conference on Principles and Practice of Constraint Programming*.

Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 47, 254-265.

Song, X., Lewis, R., Thompson, J. and Wu, Y. (2012). An incomplete m-exchange algorithm for solving the large-scale multi-scenario knapsack problem. *Computers and Operations Research*, 39(9), 1988-2000.

Taillard, E. (1993). Parallel iterative search method for vehicle routing problem. *Networks*, 23, 661-676.

Taillard, E. (1999). A heuristic column generation method for the heterogeneous fleet VRP. *Operations Research*, 33, 1-14.

Taillard, E., Badeau, P., Gendreau, M., Guertin, F. and Potvin, J.Y. (1997). A tabu search heuristic for the vehicle routing problem with soft time windows. *Transportation Science*, 31 (2), 170-186.

Taillard, E., Laporte, G. and Gendreau, M. (1996). Vehicle routing with multiple use of vehicles. *Journal of the Operational Research Society*, 47(8), 1065-1070.

Tarantilis, C., Kiranoudis, C. and Vassiliadis (2004). A threshold accepting metaheuristics for heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, 152, 148-158.

Tarantilis, C.D, Kiranoudis, C.T. (2002). BoneRoute: an adaptive memory-based method for effective fleet management. *Annals of Operations Research*, 115(1), 227-41.

Thompson, P.M. and Orlin, J.B. (1989). *The Theory of Cyclic Transfers*. Vol. OR 200-89. Sloan School of Management, MIT.

Thompson, P.M. and Psaraftis, H.N. (1993). Cyclic Transfer Algorithms for Multi-Vehicle Routing and Scheduling Problems. *Operations Research*, 41(5), 935-946.

Toth, P and Vigo, D. eds. (2002). *The Vehicle Routing Problem.* USA: Siam.

Toth, P. and Vigo, D. (1996). A heuristic algorithm for the vehicle routing problem with backhauls. In: Bianco, L. and Toth, P. eds. *Advanced Models in Transportation Analysis*. Springer: Berlin, 585-608.

Toth, P. and Vigo, D. (1997). An Exact Algorithm for the vehicle routing problem with backhauls. *Transportation Science*, 31, 372-385.

Toth, P. and Vigo, D. (1999). A heuristic algorithm for the symmetric and asymmetric vehicle routing problems with backhauls. *European Journal of Operational Research*, 113, 528-543.

Toth, P., Vigo, D. (2003). The granular tabu search and its application to the vehicle routing problems. *INFORMS Journal of Computing*, 15(4), 333-346.

Tutuncu, G.Y., Carreto, C.A.C. and Baker, B.M. (2009). A visual interactive approach to classical and mixed vehicle routing problems with backhauls. *Omega*, 37, 138-154.

Wade, A.C. and Salhi, S. (2002). An investigation into a new class of vehicle routing problem with backhauls. *The International Journal of Management Science*, 30, 479-487.

Wassan, N. (2007). Reactive tabu adaptive memory programming search for the vehicle routing problem with backhauls. *Journal of Operational Research Society*, 58, 1630-1641.

Wassan, N.A., Wassan, A.H. and Nagy, G. (2008a). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimization*, 15, 368-386.

Wassan, N.A., Nagy, G. and Ahmadi, S. (2008b). A heuristic method for the vehicle routing problem with mixed deliveries and pickups. *Journal of Scheduling*, 11, 149-161.

Wassan, N.A. and Nagy, G. (2014). Vehicle routing problem with deliveries and pickups: Modelling issues and meta-heuristics solution approaches. *International Journal of Transportation*, 2(1), 95-110.

Wassan, N.A. and Osman, I.H. (2002). Tabu Search variants for the mix fleet vehicle routing problem. *Journal of the Operational Research Society*, 53, 768-782.

Wassan, N.A., Salhi, S., Nagy, G., Wassan, N. and Wade, A. (2013). Solving the Mixed Backhauling Vehicle Routing: Problem with Ants. *International Journal of Energy Optimisation and Engineering*, 2(2), 62-77.

Wassan, N.A., Nagy, G. and Ahmadi, S. (2008b). A heuristic method for the vehicle routing problem with mixed deliveries and pickups. *Journal of Scheduling*, 11, 149-161.

Wassan, N.A., Salhi, S., Nagy, G., Naveed-Wassan and Wade, A.C. (2013). Solving the mixed backhauling vehicle routing: Problem with Ants. *International Journal of Energy Optimisation and Engineering,* 2(2), 62-77.

Wassan, N.A., Wassan, A.H. and Nagy, G. (2008a). A reactive tabu search algorithm for the vehicle routing problem with simultaneous pickups and deliveries. *Journal of Combinatorial Optimisation*, 15, 368-386.

Waters, C.D.J. (1987). A solution procedure for the vehicle scheduling problem based on iterative route improvement. *Journal of the Operational Research Society*, 38(9), 833-839.

Wilbaut, C., Salhi, S. and Hanafi, S. (2009). An iterative variable-based fixation heuristic for the 0-1 multidimensional knapsack problem. *European Journal of Operational Research,* 199, 339-348.

Yaman, H. (2006). Formulations and Valid Inequalities for the Heterogeneous Vehicle routing Problem. *Mathematical Programming,* 106(2), 365-390.

Yano, C.A., Chan, T.J., Richter, L., Cutler, T., Murty, K.J. and McGettigan, D. (1987). Vehicle routing at Quality Stores. *Interfaces*, 17(2), 52-63.

Yellow, P.C. (1970). A computational modification to the savings method of vehicle scheduling. *Operational Research Quarterly*, 21(2), 281-283.

Yeun, L.C., Ismail, W.R., Omar, K. and Zirour, M. (2008). Vehicle Routing Problems: Models and Solutions. *Journal of Quality Measurement and Analysis*, 4 (1), 205-218.

Zachariadis, E. and Kiranoudis, C. (2012). An effective local search approach for the vehicle routing problem with backhauls. *Expert Systems with Applications*, 39(3), 3174-3184.

Zachariadis, E., Tarantilis, C.D. and Kiranoudis, C.T. (2010). An Adaptive memory methodology for the vehicle routing problem with simultaneous pick-ups and deliveries. *European Journal of Operational Research,* 202(2), 401-411.

Zhong, Y. and Cole, M.H. (2005). A vehicle routing problem with backhauls and time windows: a guided local search solution. *Transportation Research part E: Logistics and Transportation Review*, 41(2), 131-144.

Appendix A:

Connecting CPLEX with Microsoft Visual Studio

One of the efficient features of CPLEX is that it comes with a set of different libraries through which its optimisers can be embedded in different programming languages on different operating platforms. CPLEX provides two ways known as *Concert Technology* and *Callable Library* through which it facilitates the programs coded in different programming languages to successfully use CPLEX optimisers. Brief descriptions of these two features are given below.

Concert Technology: The Concert Technology comes with set of Java, C++ and .Net class libraries. The primary job of these libraries is to facilitate the Application Programming Interface (API) that also consists of modelling facilities. Hence, this interface permits programmers to embed CPLEX optimisers in Java, C++ or .Net applications. Figure a.1 shows the set of libraries Concert Technology consists of different programming languages used on different operating system platforms.

The CPLEX Callable Library: The Callable Library is also a set of C libraries through which programmers can embed CPLEX optimisers in many applications developed in various programming languages such as C, C++, Visual Basic, FORTRAN or any other language that is capable of calling C functions. Therefore, Callable Library consists of cplexXXX.lib and cplexXXX.dll libraries for Windows platforms and libcplex.a, libcplex.so, and libcplex.sl for UNIX platforms.

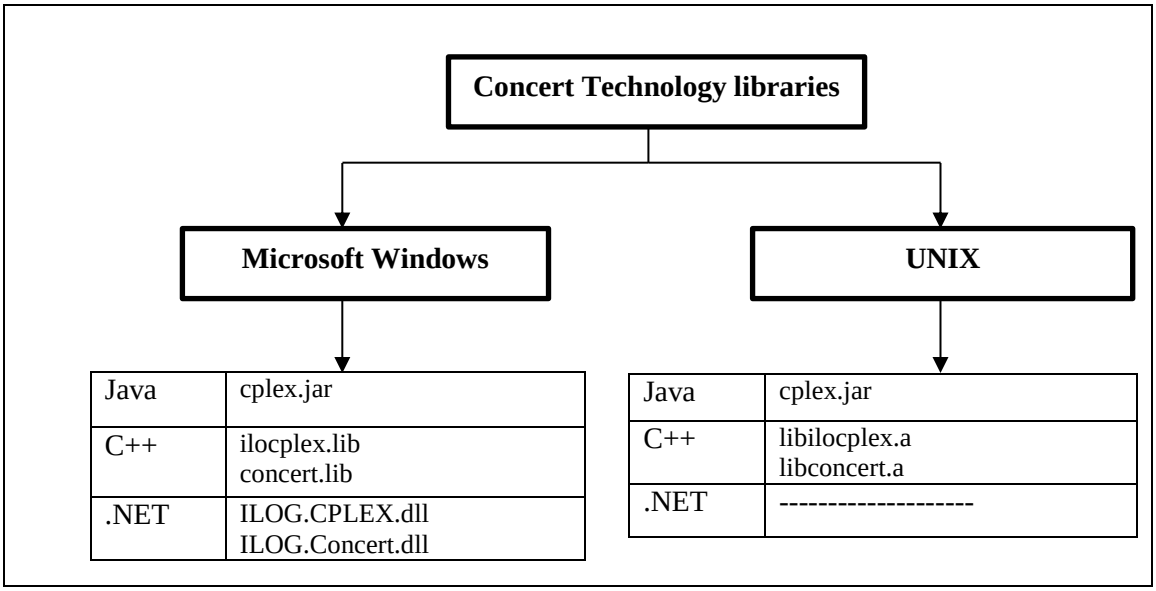

Figure a.1: Concert technology libraries for different operating systems Source: User's Manual for CPLEX V12.5

C++ code of the *MIPstart*

```
IloNumVarArray startVar(env);
IloNumArray startVal(env);

for(i=0; i<nbTotCust; i++){
     for(j=0; j<nbTotCust; j++){
          startVar.add(R[i][j]);
          startVal.add(R_VNS[i][j]);

          for(k=0; k<nbTotBins; k++){
               startVar.add(X[i][j][k]);
               startVal.add(X_VNS[i][j][k]);
          }
     }
}
cplex.addMIPStart(startVar, startVal);
startVar.end();
startVal.end();
```

Figure a.2: C++ code for the *MIPstart*

Contributions to the subject knowledge

A list of the contributions made by the research in this thesis to the subject knowledge and understanding is as follows.

- The research in thesis reviews the VRP literature extensively, both the modelling and methodological developments, and reproduces it in a different format for better understanding of the readers.
- A new variant of the VRP, multiple trip vehicle routing with backhauls (MT-VRPB) is introduced with a graph theoretical description.
- A mathematical formulation of the MT-VRPB is presented and a large set data instances generated which could serve as future benchmarks in the subject area research.
- Optimal solution is obtained for small and medium size instances by implementing CPLEX.
- For instances of large size, a VNS algorithm based on two levels (*Two-Level VNS*) is designed to obtain a continuous balance between intensification and diversification which produced very competitive results for a range of VRP variants.
- A new hybrid collaborative sequential mat-heuristic algorithm (CSMH) is developed which combines our two level VNS meta-heuristic and the exact methodology used in CPLEX through the *MIPstart* mechanism provided by the IBM ILOG CPLEX Optimisation Studio. The CSMH proved very high quality results on all three variants of the VRP tested in this thesis.

- Two further variants the VRPB and the MT-VRP are studied, mathematical formulations presented, and the *Two-Level VNS* and the CSHM algorithm are successfully implemented and tested on those problems with some trivial changes which demonstrate the generalizability and the robustness of the developed approaches.
- The better fleet management modelling and the results of this thesis may not only be utilised for commercial advantage to the relevant businesses but also have a positive impact on environment issues such as reduction in CO2 emissions due to less vehicle working hours, fuel savings, etc.

Conference papers:

1. Wassan Naveed, Nagy G, Salhi S. 'Solving the Vehicle Routing Problem with Backhauls using Variable Neighbourhood Search', OR55 Annual Conference, University of Exeter, Sep 2013.

2. Wassan Naveed, Nagy G, Salhi S. 'A Two-Level Variable Neighbourhood Search Algorithm for the Vehicle Routing Problem with Backhauls and Multiple-Trips', IFORS, Barcelona, June 2014.

Journal papers:

1. Wassan, N., Wassan, N., Nagy, G. and Salhi, S. (2016). The Multiple Trip Vehicle Routing Problem with Backhauls: Formulation and a Two-Level Variable Neighbourhood Search. *Computers and Operations Research,* (In Press).

2. Another paper based on the research work in Chapters 6 and 7, titled “A Collaborative Sequential Mat-heuristic approach for a range of VRP variants” will be submitted to a suitable ¾* journal.