5-2017

Service Consistency in Vehicle Routing

Kunlei Lian
University of Arkansas, Fayetteville

Service Consistency in Vehicle Routing

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Engineering

by

Kunlei Lian
Huazhong University of Science and Technology
Bachelor of Science in Industrial Engineering, 2009
Huazhong University of Science and Technology
Master of Science in Industrial Engineering, 2012

May 2017
University of Arkansas

This dissertation is approved for recommendation to the Graduate Council.

Dr. Ashlea B. Milburn
Dissertation Director

Dr. Kelly M. Sullivan
Committee Member

Dr. Ronald L. Rardin
Committee Member

Dr. Scott J. Mason
Committee Member

Abstract

This thesis studies service consistency in the context of multi-period vehicle routing problems (VRP) in which customers require repeatable services over a planning horizon of multiple days. Two types of service consistency are considered, namely, driver consistency and time consistency. Driver consistency refers to using the fewest number of different drivers to perform all of the visits required by a customer over a planning horizon and time consistency refers to visiting a customer at roughly the same time on each day he/she needs service. First, the multi-objective consistent VRP is defined to explore the trade-offs between the objectives of travel cost minimization and service consistency maximization. An improved multi-objective optimization algorithm is proposed and the impact of improving service consistency on travel cost is evaluated on various benchmark instances taken from the literature to facilitate managerial decision making. Second, service consistency is introduced for the first time in the literature to the periodic vehicle routing problem (PVRP). In the PVRP, customers may require multiple visits over a planning horizon, and these visits must occur according to an allowable service pattern. A service pattern specifies the days on which the visits required by a customer are allowed to occur. A feasible service pattern must be determined for each customer before vehicle routes can be optimized on each day. Various multi-objective optimization approaches are implemented to evaluate their comparative competitiveness in solving this problem and to evaluate the impact of improving service consistency on the total travel cost. Third, a branch-and-price algorithm is developed to solve the consistent vehicle routing problem in which service consistency is enforced as a hard constraint. In this problem, the objective is to minimize the total travel cost. New constraints are devised to enhance the original mixed integer formulation of the problem. The improved formulation outperforms the original formulation regarding CPLEX solution times on all benchmark instances taken from the literature. The proposed branch-and-price algorithm is shown to be able to solve instances with more than fourteen customers more

efficiently than either the existing mixed integer formulation or the one we propose in this paper.

Acknowledgments

I would like to express special thanks to my advisor, Dr. Ashlea B. Milburn, for her support and guidance on the long journey to finishing my Ph.D thesis. I feel I can never thank her enough for dedicating countless hours of patience and expertise to help me advance my research. I would also like to thank Dr. Ronald L. Rardin for his constant guidance and encouragement, without which this work would not have been possible. Additionally, I would like to thank Dr. Kelly M. Sullivan for serving on my committee and also providing numerous advice for my research. I thank Dr. Scott J. Mason for serving on my committee.

I also thank Dr. Shengfan Zhang and Dr. Haitao Liao for their help. I also want to express sincere thanks to many of my friends from the department: Jingying Zhang, Fan Wang, Bin Li, Emre Kirac and Jingming Liu.

Finally, I want to express my deepest gratitude to my parents. They have always been my strongest support in the past thirty years. I also want to give special thanks to my wife, Liya Wang, and my daughter, Xinyuan Lian, without whom I would not have the courage to finish my Ph.D program.

Dedication

I dedicate this work to my parents, my wife and my daughter.

Contents

1 Introduction **1**

2 An Improved Multi-directional Local Search Algorithm for the Multi-objective Consistent Vehicle Routing Problem **7**

2.1 Introduction 7

2.2 Related literature 10

2.2.1 Service consistency in the small package delivery industry 10

2.2.2 Service consistency in the home health care industry 13

2.2.3 Summary and contribution to the literature 15

2.3 Problem description 16

2.4 Improved MDLS for the MoConVRP 18

2.4.1 Multi-objective optimization 19

2.4.2 The original multi-directional local search algorithm 19

2.4.3 Improved multi-directional local search (IMDLS) 21

2.4.4 Initial solution generation 22

2.4.5 Large neighborhood search 24

2.4.5.1 Algorithm framework 25

2.4.5.2 Removal and reinsertion operators 25

2.5 Computational experiments and analysis 31

2.5.1 Benchmark instances and experiment setup 31

2.5.2 Metrics for comparison algorithms 33

2.5.3 Parameter tuning 34

2.5.4 Algorithm performance comparison 35

2.5.5 Trade-off analysis 41

2.6 Conclusion 45

3 A Multi-objective Approach for the Consistent Periodic Vehicle Routing Problem **51**

3.1 Introduction . . . 51

3.2 Related literature . . . 53

3.3 Problem description . . . 57

3.4 Solution approach . . . 61

3.4.1 Multi-objective optimization algorithms . . . 61

3.4.2 Starting solution generation . . . 64

3.4.3 Local search for the travel cost objective . . . 65

3.4.4 Local search for the consistency objectives . . . 69

3.5 Computational experiments and analysis . . . 70

3.5.1 Benchmark instances and experiment design . . . 70

3.5.2 Algorithm performance comparison . . . 72

3.5.3 Trade-off analysis . . . 75

3.6 Conclusion . . . 77

4 A Branch-and-Price Algorithm for the Consistent Vehicle Routing Problem **82**

4.1 Introduction . . . 82

4.2 Related literature . . . 84

4.3 Problem formulation . . . 87

4.4 Branch and price . . . 89

4.4.1 Problem reformulation . . . 90

4.4.2 Branch-and-price framework . . . 93

4.4.3 Generating starting column set Ω_{P_0} for root node P_0 . . . 96

4.4.3.1 Initial solution generation . . . 97

4.4.3.2 New population generation . . . 97

4.4.3.3 Crossover . . . 100

4.4.4 Column generation framework 100
4.4.5 Column generation heuristic 101
4.4.5.1 Algorithm framework 101
4.4.5.2 Starting column generation and evaluation 102
4.4.5.3 Large neighborhood search operators 103
4.4.5.4 Operator selection 108
4.4.6 Branching rule 109
4.5 Computational experiments 110
4.5.1 Existing instances 110
4.5.2 New instances 112
4.6 Conclusion 115

5 Conclusion and Future Research Directions **119**

List of Figures

2.1 Confliction between travel cost and service consistency 8
2.2 Key steps of MDLS 20
2.3 Key steps of IMDLS 23
2.4 Level diagram of instance 1 42

List of Tables

2.1 A summary of the literature on routing problems with service consistency . . 15

2.2 LNS operator pair used for each objective 31

2.3 Parameter α tuning results on $Group_{0.7}$ 35

2.4 Hypervolume results for $Group_{0.5}$ 36

2.5 Hypervolume results for $Group_{0.7}$ 37

2.6 Hypervolume results for $Group_{0.9}$ 37

2.7 Coverage comparison results 38

2.8 Unary multiplicative indicator results for $Group_{0.5}$ 39

2.9 Unary multiplicative indicator results for $Group_{0.7}$ 39

2.10 Unary multiplicative indicator results for $Group_{0.9}$ 40

2.11 Frequency with which various operator pairs are invoked in IMDLS 41

2.12 Four solutions identified in $\mathcal{R}$ 44

2.13 Comparisons of four solutions 45

2.14 Four solutions identified in $\mathcal{R}$ with flexible vehicle departure time 46

2.15 Comparisons of four solutions with flexible vehicle departure time 47

3.1 Summary of PVRP instances 71

3.2 Hypervolume comparison 73

3.3 Coverage comparison 74

3.4 Unary multiplicative epsilon comparison 75

3.5 Four solutions 76

3.6 Comparisons of four solutions 77

4.1 Computational results on literature instances 112

4.2 Computational results on new instances 116

4.3 Computational results on new instances 117

List of Published Papers

Chapter 2. Lian, K., Milburn, A. B., and Rardin, R. L. (2016). An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. IIE Transactions, 48(10), 975-992.

1. Introduction

This thesis studies service consistency in the context of multi-period Vehicle Routing Problems (VRPs). Two types of service consistency are considered, namely, driver consistency and time consistency. Driver consistency refers to using the fewest number of different drivers to perform all of the visits required by a customer over a planning horizon and time consistency refers to visiting a customer at roughly the same time on each day he/she needs service. The traditional objective of VRP is to minimize total travel cost, which is often conflicting with the objective of service consistency maximization due to the variations in customer demands and constraints on vehicle capacity and maximum route duration. Minimizing total travel distance could result in a customer being visited by multiple drivers at highly varying times over the planning horizon. On the other hand, optimizing service consistency usually comes at the cost of increased travel cost. Therefore, this thesis aims to study the impact of improving service consistency in multi-period VRPs on the total travel cost.

Consistent service is valued in a number of service industries where repeatable services are required. In the small package delivery industry, assigning the same driver to visit the same set of customers over time helps enhance the customer-driver relationship and improve service quality as well as efficiency as the driver becomes familiar with a set of customers. Also, improved driver consistency usually results in improved region familiarity which facilitates the driver serving future customers in the same area and thus gaining more business (Smilowitz et al., 2013). In the retail industry, using a consistent service provider has a big impact on the customer-retailer relationship. As an example, in the case of scheduling 39 sales representatives to visit over 5000 ticket retailers for the Missouri lottery, service provider consistency must be considered since many representatives have established long standing relationships with their retailers and inconsistent service will jeopardize sales (Jang et al., 2006). In the education sector of Belgium, consistent teaching assistants (TA) are employed to help students with unique educational needs. It is required that a disabled

pupil can only be assigned to one TA across a certain period (Maya et al., 2012).

Home health care is another industry in which consistent service is valued. Home health care is an important part of the healthcare system in the U.S. and has seen rapid growth in the past few years due to (i) the ever-increasing demand of aging population, (ii) its affordability when compared to hospitals or nursing homes, and (iii) higher level of comfort and dignity when patients receive personal care at their own homes. According to the National Association for Home Care and Hospice, in 2008 alone, there were 12 million patients that received 428 million visits and the home health workforce drove more than 5 billion miles (NAHC, 2010). A key feature of home health care is its prolonged episode of care requirement and multiple visits are usually demanded from patients. Therefore, service consistency has been recognized as key to improving patient satisfaction.

Two types of service consistency are considered in home healthcare practice, namely, nurse consistency and time consistency. Nurse consistency refers to minimizing the number of different caregivers visiting a patient across the planning horizon. Time consistency measures the maximum arrival time differential at a patient throughout the planning horizon. Employing inconsistent nurses to visit a patient compromises development of rapport, increases communication complexity and diminishes a caregiver's ability to make accurate observations across time (Woodward et al., 2004). It has been shown that improved nurse consistency results in better health outcomes, including decreased rate of episodes ending in hospitalization and increased likelihood of improving functioning in activities of daily living between admission and discharge from home health care (Russell et al., 2011). With consistent timing of care, patients can plan their day more readily without too many disturbances (Woodward et al., 2004).

Chapter 2 explores the trade-offs between the objectives of travel cost minimization and service consistency maximization in the context of a multi-period vehicle routing problem using a multi-objective approach. This chapter is motivated by the driver scheduling problem in the small package delivery industry in which customers request deliveries on

predetermined days over a planning horizon. Three objectives are considered, including minimization of total travel cost, maximization of driver consistency and maximization of time consistency. The resultant problem is referred to as the Multi-objective Consistent Vehicle Routing Problem (MoConVRP). Three primary contributions of Chapter 2 are summarized below.

- The trade-offs between travel cost and consistency objectives in the context of the traditional Consistent Vehicle Routing Problem (ConVRP) are studied for the first time in the literature to facilitate managerial decision making.

- An Improved Multi-directional Local Search (IMDLS) is proposed for general multi-objective optimization problems. The performance of IMDLS is compared with the original Multi-directional Local Search (MDLS) algorithm from Tricoire (2012) and five other algorithms. Three are traditional multi-objective algorithms: the Nondominated Sorting Genetic Algorithm II (NSGAII) from Deb et al. (2002), the Nondominated Neighbor Immune Algorithm (NNIA) from Gong et al. (2008), and the Strength Pareto Evolutionary Algorithm 2 (SPEA2) from Zitzler et al. (2002). The remaining two are more recent multi-objective algorithms: the Nondominated Sorting Genetic Algorithm III (NSGAIII) from Deb and Jain (2014) and the Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) from Qingfu and Hui (2007). The performance of IMDLS is validated on a variety of benchmark instances taken from the ConVRP literature.

- Large Neighborhood Search (LNS) operators are developed to improve each of three objectives studied in this chapter.

Chapter 3 studies the impact of improving service consistency on travel cost in the context of the Periodic Vehicle Routing Problem (PVRP). In the PVRP, customers may require multiple visits over a planning horizon, and these visits must occur according to an allowable service pattern. A service pattern specifies the days on which the visits required

by a customer are allowed to occur. An allowable service pattern must be determined for each customer before vehicle routes can be optimized on each day. Also, the number of available vehicles is limited in PVRP. Three objectives are considered in this chapter, namely, minimization of total travel cost, maximization of driver consistency and maximization of time consistency. The resultant problem is named the Multi-objective Consistent Periodic Vehicle Routing Problem (MoConPVRP). The primary results of this chapter are below.

- Service consistency is considered for the first time in the literature in the context of the PVRP. A mathematical formulation of MoConPVRP is given and the Pareto frontier is approximated using multi-objective algorithms to study the trade-offs between the objectives of travel cost minimization and service consistency maximization.

- Various multi-objective algorithms are implemented to solve the MoConPVRP, including MDLS, IMDLS, MOEA/D, NNIA, SPEA2, NSGAII and NSGAIII. Their comparative competitiveness is verified on the problem using benchmark instances taken from the literature.

- Local search operators are developed to improve each of the three objectives.

Chapters 2 and 3 study service consistency using a multi-objective approach in which service consistency is treated as an objective to strive for rather than as an absolute requirement. Chapter 4 aims to study the impact of enforced service consistency on the total travel cost. To this end, a branch-and-price algorithm is proposed to solve the ConVRP in which each customer can only be visited by a single driver on each day he/she needs service across the planning horizon, and the maximum arrival time differential at a customer cannot exceed a given limit throughout the planning horizon. The objective of ConVRP is to minimize the total distance traveled by all vehicles across the planning

horizon while enforcing driver and time consistency as hard constraints. The primary contributions of this chapter to the vehicle routing literature are listed as below.

- A branch-and-price algorithm is developed for the first time in the literature for the ConVRP. A set-covering reformulation of ConVRP is given along with the definition of the pricing subproblem. An effective heuristic is designed to identify new columns with negative reduced cost during the column generation process. The performance of the proposed algorithm is validated on benchmark instances taken from the literature and instances that are randomly generated following the same procedure used in the literature. The proposed algorithm usually generates tighter optimality gaps on larger instances (i.e., those with more than 14 customers) than those provided by the integer programming formulation in the literature.

- A set of new constraints are identified and added to the original Mixed Integer Program (MIP) of ConVRP. It is shown through computational experiments that the resulting MIP with this new set of constraints outperforms the original MIP on all instances, but is generally outperformed by the branch-and-price algorithm on larger instances.

Chapter 5 summarizes the conclusions of this dissertation and proposes a number of future research directions.

Reference

K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. Evolutionary Computation, IEEE Transactions on, 18(4):577–601, 2014.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on, 6(2):182–197, 2002.

Maoguo Gong, Licheng Jiao, Haifeng Du, and Liefeng Bo. Multiobjective immune algorithm with nondominated neighbor-based selection. Evolutionary Computation, 16 (2):225–255, 2008.

Wooseung Jang, Huay H. Lim, Thomas J. Crowe, Gail Raskin, and Thomas E. Perkins. The missouri lottery optimizes its scheduling and routing to improve efficiency and balance. Interfaces, 36(4):302–313, 2006.

Pablo Maya, Kenneth Srensen, and Peter Goos. A metaheuristic for a teaching assistant assignment-routing problem. Computers and Operations Research, 39(2):249–258, 2012.

NAHC. Basic statistics about home care. Report, National Association for Home Care and Hospice, 2010.

Zhang Qingfu and Li Hui. Moea/d: A multiobjective evolutionary algorithm based on decomposition. Evolutionary Computation, IEEE Transactions on, 11(6):712–731, 2007.

David Russell, Robert J. Rosati, Peri Rosenfeld, and Joan M. Marren. Continuity in home health care: Is consistency in nursing personnel associated with better patient outcomes? Journal for Healthcare Quality, 33(6):33–39, 2011.

Karen Smilowitz, Maciek Nowak, and Tingting Jiang. Workforce management in periodic delivery operations. Transportation Science, 47(2):214–230, 2013.

Fabien Tricoire. Multi-directional local search. Computers & Operations Research, 39(12): 3089–3101, 2012.

Christel A. Woodward, Julia Abelson, Sara Tedford, and Brian Hutchison. What is important to continuity in home care?: Perspectives of key stakeholders. Social Science and Medicine, 58(1):177–192, 2004.

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. CIMNE, Spain, 2002.

2. An Improved Multi-directional Local Search Algorithm for the Multi-objective Consistent Vehicle Routing Problem

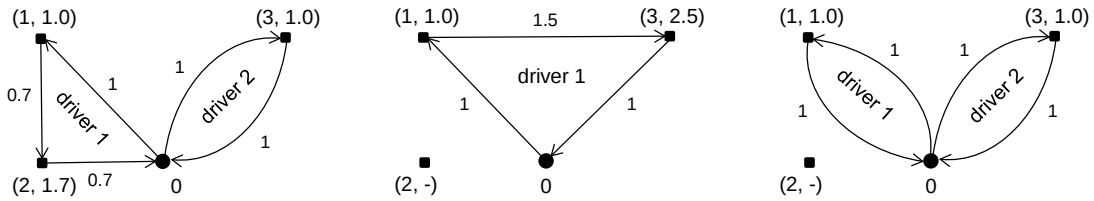
2.1 Introduction

Service consistency has attracted attention in recent years in a number of service industries where customers receive repeatable deliveries throughout a planning horizon, such as small package delivery (Groër et al., 2009), vendor-managed inventory systems (Coelho et al., 2012) and home health care. Two primary elements of service consistency include driver consistency and time consistency. Driver consistency refers to using the same driver for a given customer as often as possible, while time consistency refers to making the deliveries to the customers at roughly the same time each visit. These concepts were first discussed in Groër et al. (2009) with the introduction of a routing problem variant named the Consistent Vehicle Routing Problem (ConVRP). A more recent review of routing problems with consistency considerations is provided in Kovacs et al. (2014a).

Reducing transportation costs has historically been the primary objective in managerial decision making in logistics firms. However, due to the recognized importance of service consistency, it is desirable for some firms to obtain solutions that balance transportation costs and service consistency objectives. These objectives are often conflicting because of the variation in customer demands, vehicle capacity and driver work time limits.

Optimizing routing plans with regards to travel cost may result in a customer being visited by multiple drivers at highly varying times over the planning horizon. However, pursuing optimal service consistency with no consideration given to transportation costs will increase travel costs.

This dilemma is illustrated in the small example given in Figure 2.1, in which three customers require service across a two day horizon. Node 0 denotes the depot, and the first element of the node label at other points denotes the customer number. Arc labels provide the euclidean distance between pairs of locations. Note that customers 1 and 3 require



(a) Day 1 (TD = 4.4) (b) Day 2 (TD = 3.5) (c) Day 2 (TD = 4.0)

Figure 2.1: Confliction between travel cost and service consistency

service on both days, while customer 2 only requires a visit on day 1. Suppose a vehicle has the capacity to visit two customers, and there is no limit on route length. Then on day 1, two drivers will be required. The solution for day 1 that minimizes total distance uses driver 1 to visit customers 1 and 2, in that order, and uses driver 2 to visit customer 3. The resulting tours are illustrated in Figure 2.1a and the associated arrival times at each customer are provided in the second element of the node labels in parentheses. Now on day 2, if minimizing travel cost is the only objective, the route depicted in Figure 2.1b can be used, requiring one driver and a distance of 3.5. However, customer 3 is visited by two different drivers across the planning horizon. If instead, minimizing driver consistency were the only objective, the route depicted in Figure 2.1c could be used. In this case the distance is higher (4.0 instead of 3.5) but now, both customers 1 and 3 are only visited by a single driver across the planning horizon. Note that the day 2 route illustrated in Figure 2.1c is also better than the route in Figure 2.1b with respect to time consistency, as the arrival times to both customers 1 and 3 do not change between days 1 and 2. If the route in Figure 2.1b were used on day 2, customer 3 would be visited at a later time on day 2 than day 1 (time 2.5 compared with time 1.0). Note also that vehicles are not allowed to wait at customer locations to improve time consistency.

The traditional approach used in the literature to study the relationship between travel cost and consistency objectives has been to enforce service consistency by incorporating side constraints in vehicle routing problem (VRP) models while minimizing travel cost using a single-objective (Groër et al., 2009; Tarantilis et al., 2012; Kovacs et al., 2014b,

2015a). This approach approximates the increase in travel costs that can be expected when service consistency measures are strictly enforced. However, it may be desirable in some applications to treat service consistency measures as something to strive for rather than to enforce. That is, service consistency measures can be treated as objectives along with travel costs instead of constraints, and compromise (trade-off) solutions can be sought. Modeling service consistency using this new approach requires multi-objective VRP models. Kovacs et al. (2015a) make a stride in this direction by using a single weighted objective function to simultaneously consider travel cost minimization and time consistency maximization. Milburn and Spicer (2013) also introduce a multi-objective VRP variant to consider service consistency. They develop an approach to approximate the Pareto frontier for a routing problem variant with travel cost, driver consistency and workload balance objectives. Kovacs et al. (2015b) analyze the trade-offs between the objectives of minimizing travel cost and improving driver consistency and time consistency in the context of a generalized consistent vehicle routing problem. Customers in their study are associated with AM/PM time windows. However, no paper of which we are aware has studied a multi-objective variant of the traditional ConVRP without time windows where the Pareto frontier is developed explicitly. The primary objective of this paper is to fill this gap.

In this paper, we use a multi-objective approach to study the relationship between travel cost and service consistency objectives. Specifically, the three objectives considered include minimization of traveling cost, maximization of driver consistency and maximization of time consistency. There are three primary contributions in this paper. First, to the best of our knowledge, this is the first paper to approach the traditional Consistent VRP with both driver and time consistency from the perspective of developing the Pareto frontier. This enables us to study the explicit trade-offs between travel cost and consistency objectives, and use the results of our computational study to make observations that can facilitate managerial decision making. Second, we propose an improved variant of the multi-directional local search (MDLS) framework for general multi-objective optimization

problems (Tricoire, 2012). We denote this IMDLS (Improved MDLS). The effectiveness of IMDLS is compared with the original MDLS and three traditional multi-objective algorithms: the nondominated sorting genetic algorithm II (NSGAII), the nondominated neighbor immune algorithm (NNIA) and the strength pareto evolutionary algorithm 2 (SPEA2) (Deb et al., 2002; Gong et al., 2008; Zitzler et al., 2002). Additionally, IMDLS is compared with two more recent multi-objective algorithms: the non-dominated sorting genetic algorithm III (NSGAIII) and the multi-objective evolutionary algorithm based on decomposition (MOEA/D) (Deb and Jain, 2014; Qingfu and Hui, 2007). The competitive performance of IMDLS is observed on a variety of benchmark instances taken from the Consistent VRP literature. Finally, large neighborhood search (LNS) operators are designed to explicitly improve each of the three objectives studied in this paper. These are different from the LNS operators in the literature, which are designed to improve travel cost while enforcing service consistency measures as constraints.
The remainder of this paper is organized as follows. Section 2.2 provides a review of the Consistent VRP literature as it pertains to the application areas of small package delivery and home healthcare. The multi-objective problem we study is formally introduced in Section 2.3. Our proposed solution methodology is detailed in Section 2.4. Then, the results of our computational study are provided in Section 2.5 with final remarks given in Section 3.6.

2.2 Related literature

This section provides a brief review of existing research regarding service consistency in routing problems. The papers are classified according to two primary application areas.

2.2.1 Service consistency in the small package delivery industry

Groër et al. (2009) are the first to introduce the consistent vehicle routing problem (ConVRP). The problem is presented in the context of the small package delivery industry.

A formulation of ConVRP as a mixed-integer program is provided. The objective is to minimize total transportation costs and there are side constraints to enforce consistency requirements. Specifically, each customer is required to be visited by the same driver at roughly the same time on each day that he/she needs service. The time consistency aspect is modeled by imposing an upper limit on the arrival time differential for each customer. For a single customer, this is measured as the difference between the earliest and latest arrival time to the customer. The authors employ a record-to-record travel heuristic (ConRTR) to obtain near optimal solutions quickly. Optimal driver consistency is enforced and time consistency is encouraged throughout the search process by utilizing the concept of template routes. *Template routes* are a set of artificial routes that only include visits to customers that require service on more than one day of the planning horizon. To construct actual routes for a specific day d, customers that do not need service on day d are removed from the template and customers who require service *only* on day d are added.

Two additional papers employ the concept of template routes for the Consistent VRP. Namely, Tarantilis et al. (2012) and Kovacs et al. (2014b) develop a template-based tabu search heuristic (TTS) and a template-based adaptive large neighborhood search heuristic (TALNS) for ConVRP, respectively. Both of these algorithms outperform ConRTR with respect to solution quality, with TALNS offering the best performance among these three approaches. A relaxed variant of the Consistent VRP is also considered in Kovacs et al. (2014b), in which vehicles are allowed to wait at the depot before beginning their routes. Improved performance with respect to time consistency is observed in this problem variant.

Kovacs et al. (2015a) generalize the ConVRP in a number of ways. First, instead of requiring that exactly one driver visit each customer throughout the planning horizon, they include a constraint that limits the maximum number of different drivers that visit a customer. Second, the maximum arrival time differential used to model time consistency is penalized in the objective function instead of being included as a constraint. Additionally, customers are each associated with AM/PM time windows and vehicles are allowed to wait

at the depot before beginning their routes, as in Kovacs et al. (2014b). The minimization objective considered in this ConVRP generalization is a weighted sum of travel cost and maximal arrival time differential. A large neighborhood search that does not make use of template routes is proposed. Note that the ConVRP is a special case of the new problem these authors propose. Therefore, they use LNS to solve ConVRP and compare results with ConRTR, TTS and TALNS. The superior performance of the new approach is validated. Based on this research, Kovacs et al. (2015b) recently study the multi-objective generalized consistent vehicle routing problem in which improving driver consistency and arrival time consistency and minimizing travel cost are treated as independent objectives. Two exact solution approaches based on the ϵ-constraint framework are used to solve small instances with 10-12 customers and a planning horizon of 3 days to optimality. A heuristic multi-directional large neighborhood search algorithm is proposed to solve large instances. Trade-off analysis shows that 70% better arrival time consistency can be achieved by increasing travel cost by not more than 3.84%.

Smilowitz et al. (2013) design a modified tabu search heuristic to study the effects of three workforce management principles on routing costs in periodic vehicle routing problems. The workforce management principles include driver consistency, customer familiarity and region familiarity. Three objectives are proposed to model these principles and combined into a single weighted objective. Experimental results show that trade-off solutions having relatively good performance with respect to the workforce management principles can be obtained at the cost of increasing traveling cost by at most 5.3%.

Zhong et al. (2007) develop a two-stage model to improve drivers' familiarity with their service territories in a vehicle dispatching problem for local package deliveries. In the first stage of their model, a set of core areas to serve as service territories for each driver are designed. Then, the optimal sequences of visits to customers in each core area are determined in the second stage. The value of the two-stage model is compared with a single stage 'no-core area" model which simply reoptimizes routes on a daily basis.

Computational results show the two-stage model is able to provide more consistent service than the no-core model.

Luo et al. (2015) study a multi-period vehicle routing problem in which customers require visits within specific time windows over a time horizon and customers can be served by at most a certain number of different vehicles over the planning horizon. The primary objective is to minimize the number of vehicles required, and in case of ties, solutions are evaluated using a secondary objective of minimizing the total travel distance. A mixed integer programming model is proposed and the limited visiting quota is enforced using hard constraints. A three-stage heuristic approach is developed to solve the problem. Computational results show that when vehicle capacity is not the primary limiting constraint, consistent customer service can be achieved with slight increases in operational costs. However, when only a small number of customers can be serviced by a vehicle because of its capacity limit, enforcing service consistency leads to more significant increases in operational cost.

2.2.2 Service consistency in the home health care industry

Service consistency is especially important when the customer must be present to receive the delivery or service, as in home health care. Home health care involves using skilled licensed professionals to provide prescribed medical services to homebound patients over a prescribed episode of care. Typically, the duration of an episode of care is seven to ten weeks (Dey et al., 2011). In this industry, service consistency has been recognized as key in improving customer satisfaction and loyalty (Woodward et al., 2004). Driver (nurse) consistency can increase the familiarity between patients and caregivers, and thus reduce the complexity of communication. It also improves the ability of caregivers to make accurate observations, thereby benefiting health outcomes (Woodward et al., 2004). Time consistency allows patients to plan their days more readily, without causing too many disturbances to their daily routines (Woodward et al., 2004). A number of papers in the

operations research literature have studied home health nurse routing and scheduling problems with consistency requirements.

Bennett and Erera (2011) present a rolling horizon myopic planning approach for the single nurse routing and scheduling problem in which patients are revealed dynamically over a time horizon. In their approach, both driver and time consistency are strictly enforced.

Eveborn et al. (2006) consider nurse consistency in the development of LAPS CARE, a decision support system for determining nurse routes in Sweden. The minimization of the total number of different nurses visiting a patient is incorporated into a weighted objective function in their model. The objective function also includes a number of additional elements, such as travel time, travel cost and patient preferences for particular staff members. They provide a set partitioning formulation of their problem and describe a repeated matching algorithm for its solution. Macdonald et al. (2009) also study a nurse scheduling and routing problem where the objective includes a weighted sum of travel cost and the number of different nurses visiting a patient. Nurse routes with good consistency and routing costs are reported.

Nickel et al. (2012) study both short-term and mid-term home health nurse routing problems using an approach that aims to minimize the weighted sum of four objectives: the number of unscheduled tasks, overtime costs, patient-nurse loyalty and travel distance. Patient-nurse loyalty is analogous to driver consistency in the ConVRP literature, defined as the number of different nurses visiting each patient during the planning horizon. Constraint programming and tabu search methods are developed.

Another paper that uses a multi-objective approach to incorporate consistency considerations in home health nurse routing and scheduling problems is Milburn and Spicer (2013). They consider a problem with nurse consistency, travel cost and balanced nurse workload in the objective function. They approximate the Pareto frontier for these three objectives using a tabu search based approach.

Reference	Time Consistency	Driver Consistency	Constraints
Eveborn et al. (2006)		✓	Soft
Nickel et al. (2012)	✓	✓	Soft
Macdonald et al. (2009)		✓	Soft
Bennett and Erera (2011)	✓		Hard
Milburn and Spicer (2013)		✓	Soft
Groër et al. (2009)	✓	✓	Hard
Tarantilis et al. (2012)	✓	✓	Hard
Kovacs et al. (2014b)	✓	✓	Hard
Kovacs et al. (2015a)	✓	✓	Hard
Kovacs et al. (2015b)	✓	✓	Soft
Smilowitz et al. (2013)		✓	Soft
Zhong et al. (2007)		✓	Soft
Feillet et al. (2014)	✓		Hard
Luo et al. (2015)		✓	Hard

Table 2.1: A summary of the literature on routing problems with service consistency

2.2.3 Summary and contribution to the literature

Table 2.1 summarizes the reviewed research on routing problems with service consistency. Columns are included for time consistency and driver consistency, and a checkmark indicates whether each type of consistency is considered in each paper. A third column indicates whether the consistency elements are treated as hard constraints or soft. A review of the literature indicates there is currently a gap with respect to simultaneously considering the three objectives of travel cost, driver consistency and time consistency within an approach aimed at approximating the Pareto frontier in the context of traditional ConVRP. We aim to fill this gap. Most other multi-objective approaches use models that combine the objectives into a single weighted objective function. Such approaches require choosing a set of weights to represent the relative importance among objectives. Our model is based on the premise that a diverse set of decision makers may have different opinions regarding the relative importance of the three objectives. Instead of using such a set of weights to recommend a single best solution, our approach obtains a set of promising compromise solutions effectively.

2.3 Problem description

The multi-objective consistent vehicle routing problem (MoConVRP) studied in this paper is defined on a complete directed graph $\mathcal{G} = (\mathcal{N}^0 = \mathcal{N} \cup \{0\}, \mathcal{A})$, where $\mathcal{N} = \{1, \ldots, n\}$ is the set of customers and $\{0\}$ indicates the depot, and $\mathcal{A} = \{(i, j) \mid i, j \in \mathcal{N}^0, i \neq j\}$. A time horizon of D days is considered and each customer $i \in \mathcal{N}$ has a predetermined non-negative demand q_{id} and service time s_{id} on day $d \in D$. An auxiliary parameter w_{id} is defined such that w_{id} equals 1 if customer i requires service on day d (i.e., $q_{id} > 0$), and equals 0 otherwise. There are K homogenous vehicles available at the depot, where they start and end their daily operations. Each vehicle is restricted by its physical capacity Q, and a limit on route duration T. The number of vehicles is unlimited; practically, we can set K to n. In this paper, the terms *driver* and *vehicle* are used interchangeably. The problem is to determine a set of vehicle routes for each day of the planning horizon that are feasible with respect to capacity and route duration constraints. Each route must begin and end at the depot and each customer must be visited by exactly one vehicle on each day that he/she needs service. The objectives are to minimize travel cost and maximize driver and time consistency.

To model this problem, the following decision variables are defined:

- a_{id}: continuous variable describing vehicle arrival time at customer i on day d,
- x^d_{ijk}: binary variable indicating whether arc (i, j) is visited by driver k on day d,
- y^d_{ik}: binary variable indicating whether customer i is visited by driver k on day d.

Additionally, to facilitate the linearization of consistency objectives, the following auxiliary variables are defined:

- z_{ik}: binary variable indicating whether customer i is visited by driver k at any point during the planning horizon,
- a^e_i: continuous variable describing the arrival time of the earliest visit to customer i over the planning horizon,
- a^l_i: continuous variable describing the arrival time of the latest visit to customer i over the planning horizon.

Then, the multi-objective problem is formulated as follows, extended from the MIP given in Groër et al. (2009):

$$\min \quad f_{TD} = \sum_{d=1}^{D}\sum_{k=1}^{K}\sum_{i=0}^{n}\sum_{j=0}^{n} t_{ij}x_{ijk}^{d}, \tag{2.1}$$

$$\min \quad f_{DC} = z^{max}, \tag{2.2}$$

$$\min \quad f_{TC} = a^{max}, \tag{2.3}$$

$$\text{s.t.} \quad y_{0k}^{d} = 1,\ \forall\ k \in K, d \in D \tag{2.4}$$

$$a_{0d} = 0,\ \forall\ d \in D \tag{2.5}$$

$$\sum_{k=1}^{K} y_{ik}^{d} = w_{id},\ \forall\ i \in \mathcal{N}, d \in D \tag{2.6}$$

$$\sum_{i=1}^{n} q_{id}y_{ik}^{d} \leq Q,\ \forall\ k \in K,\ d \in D \tag{2.7}$$

$$0 \leq a_{id} + w_{id}(s_{id} + t_{i0}) \leq Tw_{id},\ \forall\ i \in \mathcal{N},\ d \in D \tag{2.8}$$

$$\sum_{i=0}^{n} x_{ijk}^{d} = \sum_{i=0}^{n} x_{jik}^{d} = y_{jkd},\ \forall\ j \in \mathcal{N}^{0},\ k \in K,\ d \in D \tag{2.9}$$

$$a_{id} + x_{ijk}^{d}(s_{id} + t_{ij}) - T(1 - x_{ijk}^{d}) \leq a_{jd},\ \forall\ d \in D,\ k \in K,\ i \in \mathcal{N}^{0},\ j \in \mathcal{N} \tag{2.10}$$

$$a_{id} + x_{ijk}^{d}(s_{id} + t_{ij}) + T(1 - x_{ijk}^{d}) \geq a_{jd},\ \forall\ d \in D,\ k \in K,\ i \in \mathcal{N}^{0},\ j \in \mathcal{N} \tag{2.11}$$

$$z_{ik} \geq y_{ik}^{d},\ \forall\ i \in \mathcal{N},\ k \in K,\ d \in D \tag{2.12}$$

$$\sum_{k \in K} z_{ik} \leq z^{max},\ \forall i \in \mathcal{N} \tag{2.13}$$

$$a_{i}^{l} \geq a_{id} \geq a_{i}^{e},\ \forall\ i \in \mathcal{N},\ d \in D \tag{2.14}$$

$$a_{i}^{l} - a_{i}^{e} \leq a^{max},\ \forall\ i \in \mathcal{N} \tag{2.15}$$

$$x_{ijk}^{d} \in \{0, 1\},\ \forall\ i \in \mathcal{N},\ j \in \mathcal{N},\ k \in K,\ d \in D \tag{2.16}$$

$$y_{ik}^{d} \in \{0, 1\},\ \forall\ i \in \mathcal{N},\ k \in K,\ d \in D \tag{2.17}$$

$$a_{id} \geq 0,\ \forall\ i \in \mathcal{N},\ d \in D \tag{2.18}$$

$$z_{ik} \geq 0,\ \forall\ i \in \mathcal{N},\ k \in K \tag{2.19}$$

$$a_i^e, a_i^l \geq 0, \ \forall \ i \in \mathcal{N} \tag{2.20}$$

$$a^* \geq 0. \tag{2.21}$$

The first objective (2.1) aims to minimize the total traveling distance of all vehicles on all days of the planning horizon. Objective (2.2) seeks to minimize the maximum number of different drivers that visit a customer. Objective (2.3) tries to minimize the maximal arrival time differential over all customers. Constraint sets (2.4) and (2.5) require that the depot be departed at time 0 by all vehicles on all days. Note that constraint (2.5) may be relaxed for the case of flexible vehicle departure time. Constraint set (2.6) ensures that customers are visited by exactly one vehicle on each day they need service. The vehicle capacity and route duration constraints are given in sets (2.7) and (2.8), respectively. Constraint set (2.9) makes sure that each customer has only one predecessor and successor. Constraint sets (2.10) and (2.11) determine the customer arrival times and also serve to eliminate subtours. Constraint set (2.12) and (2.13) compute the maximum number of different drivers that visit a customer over the time horizon. Time consistency is computed in constraint sets (2.14) and (2.15). The remaining constraints give the variable types. Note that objectives (2.2, 2.3) and constraint sets (2.12, 2.13, 2.14, 2.15) are newly introduced compared to the original model given in Groër et al. (2009). In addition, the consistency constraints in Groër et al. (2009) are removed.

2.4 Improved MDLS for the MoConVRP

In this section, we first introduce the basic concepts of Pareto-based multi-objective optimization. Next, we describe the original MDLS algorithm and comment on its performance. We then present an improved MDLS framework for multi-objective optimization problems, followed by the description of large neighborhood search which is used as a subroutine of our proposed framework.

2.4.1 Multi-objective optimization

In the context of multi-objective optimization, solutions are evaluated according to an objective function vector $\vec{f} = (f_1, f_2, \ldots, f_M)$ with M objectives. Pareto dominance is then employed to compare the strength of different solutions. For a minimization problem, an objective vector $\vec{f}(x)$ is said to dominate another objective vector $\vec{f}(x')$, denoted by $\vec{f}(x) \prec \vec{f}(x')$, if and only if:

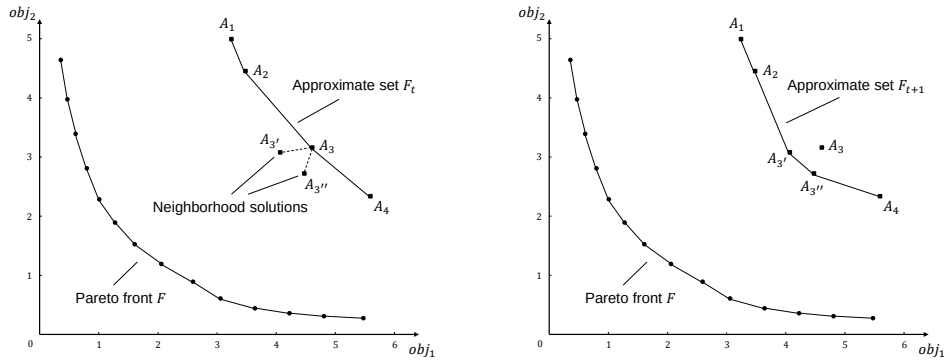
$$f_i(x) \leq f_i(x'),\ \forall\ i \in \{1, 2, \ldots, M\} \text{ and } \exists\ j \in \{1, 2, \ldots, M\} : f_j(x) < f_j(x'). \tag{2.22}$$

Accordingly, a solution x in the decision space X is said to dominate another solution x' if and only if $\vec{f}(x) \prec \vec{f}(x')$. If there exists no solution $x' \in X$ that dominates x, x is said to be *Pareto optimal*. The set of Pareto optimal solutions in X is referred to as the *Pareto set* and its image in the objective space is called the *Pareto frontier*. Multi-objective algorithms aim to attain the Pareto frontier if possible. In many cases, heuristic approaches are utilized to approximate the Pareto frontier as closely as possible. In addition, the final non-dominated solutions obtained by a heuristic algorithm should be distributed along the Pareto frontier as evenly as possible in order to represent a diverse approximation to the true Pareto frontier.

2.4.2 The original multi-directional local search algorithm

MDLS is proposed by Tricoire (2012) for general multi-objective optimization problems. It is motivated by the concept of Pareto dominance, which implies that a neighbor solution x' of x is either (i) dominating x or (ii) non-comparable with x if x' is better than x on at least one objective. Therefore, it is sufficient to improve upon one objective at a time to find desirable neighbor solutions of x. This facilitates the utilization of single-objective local search algorithms within the overall MDLS framework.

The input to MDLS is an initial set $\mathcal{F}$ of non-dominated solutions. The set $\mathcal{F}$ can be



(a) Local search on chosen solution (b) Approximate set update

Figure 2.2: Key steps of MDLS

obtained by randomly generating a population of initial solutions, assigning dominance rank using the sorting algorithm proposed in Deb et al. (2002) and deleting all dominated solutions. The solution generation scheme used to generate a population of initial solutions for the MoConVRP in this paper will be detailed in a later section. At each iteration, MDLS selects a solution x from $\mathcal{F}$ and initiates an empty set $\mathcal{G}$ for keeping neighbor solutions of x. Then for each of M objectives, a corresponding local search method $LS_m(x)$ is applied on x and the resulting neighbor solution x' is saved in $\mathcal{G}$. After that, the non-dominated set $\mathcal{F}$ is updated by merging solutions in $\mathcal{F}$ and $\mathcal{G}$, and deleting all dominated solutions. Algorithm 1 in the online supplement summarizes the MDLS framework.

Figure 2.2 illustrates two key steps of MDLS. In Figure 2.2a, the non-dominated solution set at iteration t, $\mathcal{F}_t$, has four solutions. Solution A_3 is the chosen solution and its two neighbor solutions are obtained by applying local search on each of its two objectives. Because the two neighbor solutions both dominate A_3 and are not dominated by any other solutions in $\mathcal{F}_t$, both of them enter $\mathcal{F}_{t+1}$, as shown in Figure 2.2b.

Our preliminary experiments reveal two limitations of MDLS. First, MDLS does not limit the size of $\mathcal{F}$ during its search process, which may become very large for complex problems like MoConVRP. Because $\mathcal{F}$ has to be updated at each iteration, maintaining a large

number of non-dominated solutions in $\mathcal{F}$ induces computational burden. In addition, MDLS focuses on only one randomly selected solution to improve at each iteration, which may lead to slow convergence speed and poor diversity of the final non-dominated set. Based on these observations, we propose an improved MDLS framework and validate its effectiveness on the MoConVRP studied in this paper.

2.4.3 Improved multi-directional local search (IMDLS)

Algorithm 1 shows the improved MDLS framework. In addition to an initial non-dominated solution set $\mathcal{F}$, the input to IMDLS includes a size limit on $\mathcal{F}$, denoted F_{max}, which specifies the maximum number of solutions that can be maintained in $\mathcal{F}$ throughout the search process. At each iteration, IMDLS begins by exploring neighbor solutions of every solution $x \in \mathcal{F}$ with respect to each of the M objectives. As with MDLS, a variety of local search methods can be used in IMDLS to find neighbor solutions with respect to each objective. In this paper, a large neighborhood search heuristic is employed. Every new neighbor solution enters a set $\mathcal{G}$ which is later used to update $\mathcal{F}$. Performing local search on all solutions in $\mathcal{F}$ instead of only one solution helps IMDLS converge more quickly to the Pareto frontier. The additional computation time introduced by this expanded search is remedied to some extent by the time saved in updating the set $\mathcal{F}$ at the end of each iteration because a reduced $\mathcal{F}$ is maintained. Next, if the size of the updated $\mathcal{F}$ exceeds F_{max}, the crowding distance is computed for solutions in $\mathcal{F}$ (Deb et al., 2002). The purpose is to guide the selection of specific solutions in $\mathcal{F}$ to delete.

The *crowding distance* of a solution on the Pareto frontier measures the density of solutions surrounding it. It is defined as the average distance of two neighboring solutions on either side of the solution along each of the objectives. For example, suppose a problem having only two objectives is being considered. To compute the crowing distance of a solution on the Pareto frontier, one must find the distances between the solutions on the Pareto frontier just better than and just worse than it for both of the objectives. Lower values of crowding

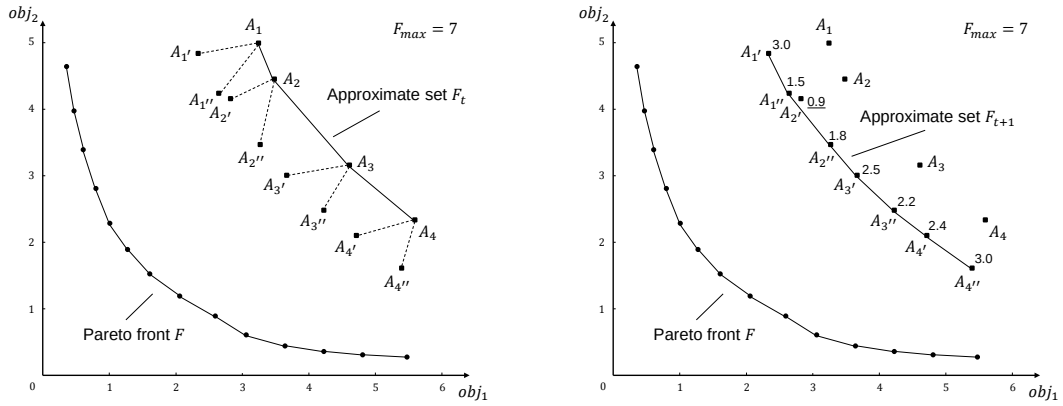
Algorithm 1 Improved multi-directional local search

Input: A set of non-dominated solutions $\mathcal{F}$ and its size limit F_{max}
repeat
  $\mathcal{G} \leftarrow \emptyset$
  for all $x \in \mathcal{F}$ **do**
    for $m \leftarrow 1$ to M **do**
      $\mathcal{G} \leftarrow \mathcal{G} \cup \{LS_m(x)\}$
    end for
  end for
  $update(\mathcal{F}, \mathcal{G})$
  if $|\mathcal{F}| > F_{max}$ **then**
    compute crowding distance for $x \in \mathcal{F}$
    $truncate(\mathcal{F})$
  end if
until stopping criterion is met
Output: $\mathcal{F}$

distance refer to more crowded solutions. In IMDLS, once the crowding distance has been computed for every solution in $\mathcal{F}$, a function $truncate(\mathcal{F})$ iteratively removes solutions from $\mathcal{F}$ in order of non-decreasing crowding distance until the number of solutions that remain is no greater than F_{max}. With this modification, IMDLS is able to focus attention on less-crowded areas of the non-dominated set and hence improves the diversity of $\mathcal{F}$. Figure 2.3 highlights the two primary differences between IMDLS and MDLS. Comparing Figure 2.3a to Figure 2.2a, it can be seen that the neighborhoods of all solutions in $\mathcal{F}_t$ are explored in IMDLS, while only a single neighborhood is explored in MDLS. Then, Figure 2.3b illustrates the process in IMDLS for updating $\mathcal{F}$ when it becomes too large. Note that after the local search step, a total of 8 non-dominated solutions are contained in $\mathcal{F}_{t+1}$ while the size limit F_{max} is set to 7. Therefore, $\mathcal{F}_{t+1}$ must be truncated by one. The crowding distances of all solutions in $\mathcal{F}_{t+1}$ are computed and $A_{2''}$ is identified as the solution with minimum crowding distance (with a value of 0.9). Therefore, $A_{2''}$ is removed from set $\mathcal{F}_{t+1}$.

2.4.4 Initial solution generation

The proposed IMDLS framework requires an initial set of non-dominated solutions as input. Although widely used in the literature to generate solutions with good service consistency, template routes, as described in Section 2.2, are not utilized in this paper to



(a) Local search on all solutions (b) Update/truncation of approximation set

Figure 2.3: Key steps of IMDLS

generate initial solutions. This is because the resulting daily routes derived from template routes always have optimal driver consistency. This limits the possibility to explore trade-offs between the objectives of cost minimization and consistency maximization. Therefore, instead of employing the concept of template routes, our IMDLS framework operates on the entire routing plan. For each day of the planning horizon, the routing plan contains a route for each vehicle in service that day. The set of routes corresponding to a particular day contains visits to all customers requiring service on that day.

An overview of the initial solution generation scheme we use is provided here, with details in Algorithm 2 in the online supplement. The algorithm begins generating an initial solution (i.e., routing plan) s by constructing an empty route for each day of the planning horizon. Any time an empty route is added to the routing plan s, a driver is assigned to it. Then, customers are considered for insertion into s in random order. The first step in inserting a customer c to a routing plan is to identify a "good" driver for the customer. In general, a driver r is a good choice for customer c if they are already operating routes in close proximity to customer c on the days customer c requires service. Selecting such a driver r for customer c should encourage good driver consistency and low travel costs. To this end, a parameter α is used to identify the set of routes that are adjacent to c on each

day customer c requires service. A route r is said to be *adjacent* to customer c if the angle made between the location of customer c and the centroid of all locations in route r, taking the depot as the origin, is no greater than α. Also, all empty routes are considered to be adjacent to all customers. Then, the driver r^* appearing most frequently in the set of routes adjacent to c is determined. Next, for each day d on which customer c requires service, the visit to customer c is placed into its cheapest insertion location in route r^*. If no feasible insertion locations exist in route r^* on day d for the visit to customer c, the visit will be inserted into its cheapest feasible insertion location among all routes. Doing so may require creating a new empty route. Note that while this method of defining adjacency using the angle between two locations with respect to the depot is appropriate for problem instances having Euclidean distances, such as those studied in this paper, it may need further investigation for problem instances having distances along real street networks. For example, two locations separated by a natural barrier such as a lake or river may be considered as adjacent using the method employed here, but the road distance between them could be quite large. In this case, the real distance between two customers could be used to decide whether they are adjacent. To decide the closeness of a customer and a route, the average distance of the customer to all the customers on the route could be used. The objective of this generation scheme is to create an initial set of non-dominated solutions that are relatively good. This scheme encourages the creation of solutions with low travel cost and good driver consistency, thus the initial set of solutions will represent closer approximation to the Pareto frontier. Moreover, randomness is introduced in the scheme to ensure the diversity of generated solutions.

2.4.5 Large neighborhood search

In this section, we first explain the large neighborhood search (LNS) framework used in IMDLS to find neighbor solutions. Then we describe the removal and reinsertion operators designed for LNS to improve each of the three objectives considered in this paper.

2.4.5.1 Algorithm framework

Both MDLS and IMDLS are algorithm frameworks designed for general multi-objective optimization problems; they rely on problem-specific local search operators to obtain neighbor solutions. Any local search algorithm can be used for this purpose and we employ large neighborhood search (LNS) in this research for its simplicity and effectiveness (Shaw, 1998). LNS searches for a neighbor of a current solution using two steps: *removal* and *reinsertion*. Essentially, *removal* involves choosing a number of customer visits to remove from the solution according to some specific criteria and *reinsertion* aims to reinsert removed customer visits back into the solution according to some other criteria. The neighbor solution is the routing plan obtained after one iteration of removal and reinsertion is executed on the current solution.

Algorithm 2 describes the workflow of LNS within the overall algorithm framework of IMDLS. Every time LNS is called with an input solution s, a pair of removal and reinsertion operators, denoted by τ, is selected and applied to s to obtain its neighbor solution s'. If s' dominates or is non-comparable with s, then s' replaces s. The LNS subroutine will return the new neighbor solution to the IMDLS procedure. In this paper, operator pairs are randomly selected in each iteration of LNS according to probabilities that depend upon their previous performance. That is, for each pair, we keep record of the ratio of the number of times the pair has succeeded in improving a solution to the total number of times it has been invoked. This ratio is initialized to 1 for all pairs of operators at algorithm onset. Note that the removal and reinsertion operator performance is updated at every iteration in this research, while in Ropke and Pisinger (2006) and Kovacs et al. (2014b) it is updated only at certain predefined iterations.

2.4.5.2 Removal and reinsertion operators

This section describes the removal and reinsertion operators used in LNS. First, there are four types of removal operators and two reinsertion operators. One reinsertion operator

Algorithm 2 Large neighborhood search

1: **Input:** solution s
2: select a removal/reinsertion pair τ based on their previous performance
3: apply τ on s to get neighbor solution s'
4: **if** s' dominates or is non-comparable with s **then**
5: let $s = s'$
6: **end if**
7: update performance of τ
8: **Output:** solution s

aims to achieve good travel cost, and the other to achieve good driver consistency. Any of these removal operators can be paired with the appropriate reinsertion operator if the objective of interest during the current iteration of local search is travel cost or driver consistency. However, these removal and reinsertion pairs are not used to improve the time consistency objective. This is because the removal operators tend to remove many customer visits from a solution at once, as will be described. Customer arrival times are highly interrelated, so it is difficult to reinsert several visits at once if the objective is to improve time consistency. Therefore, a designated pair of removal and reinsertion operators for improving the time consistency objective is described last.

Random removal

We employ two random removal operators to remove randomly chosen customer visits from a solution. This is intended to diversify the search process. Both begin by generating a removal probability $p_c \sim U[0, 1]$ for each customer c. The first random removal operator, denoted $Rand_1$, then randomly generates a removal threshold $\rho \sim U[0.2, 0.4]$. For every customer c for which $p_c < \rho$, the visits to the customer on every day they need service are removed from the routing plan s. The second random removal operator, $Rand_2$, randomly generates a removal threshold $\rho^d \sim U[0.2, 0.4]$ for each day d of the planning horizon. Then, for every customer and day pair (c,d), the visit to customer c on day d is removed from routing plan s if $p_c < \rho^d$. Thus, $Rand_1$ removes all visits for select customers from s, while $Rand_2$ removes only a subset of visits for select customers.

Adjacent removal

There are two removal operators that aim to deconstruct routes in a particular geographic area. These adjacent removal operators define any two customer b and c as adjacent if the angle made between them, taking the depot as the origin, is no greater than α. The first, Adj_1, begins by randomly choosing a seed customer c. Then, the set of customers adjacent to c are identified. Finally, visits to c and all its adjacent customers are removed from the routing plan s on every day the customers require service. The second removal operator, Adj_2, identifies d seed customers, one for each day of the planning horizon. Denote the seed customer on a given day d as c_d. The customers adjacent to c_d are identified, and then the visits on day d to c_d and all its adjacent customers on day d are removed from the routing plan s. This process is repeated for each day d. For both operators Adj_1 and Adj_2, α is set to $360/K_{active}$, where K_{active} is the maximum number of drivers working in a given day in routing plan s.

Worst driver consistency removal

The worst driver consistency removal operators attempt to remove visits to customers who are seen by the largest number of different drivers over the planning horizon. The first operator, $WorstDC_1$, sets a removal threshold $K_{active}/2$, where K_{active} is as previously defined. All visits to every customer seen by more than $K_{active}/2$ drivers throughout the planning horizon are removed. The second operator, $WorstDC_2$, compares the number of days a particular customer c requires visits (denoted m_c) to the number of different drivers that visit c in routing plan s (denoted n_c). If these two numbers are equal (that is, $m_c = n_c$, implying a different driver visits customer c each time they require service), then all visits to customer c are removed from routing plan s. Otherwise, if $n_c < m_c$, then the the driver d_c that visits customer c most frequently is identified. Visits to customer c are removed from s for every day c is visited by a driver different from d_c. This process is repeated for every customer c.

Worst time consistency removal

The worst time consistency removal operators try to remove visits for customers whose arrival times differentials across the planning horizon are high. Both operators set a removal threshold $f_{TC}/2$, where f_{TC} is the value of the time consistency objective value of routing plan s. Denote the arrival time differential for customer c as a_c. According to the first operator, $WorstTC_1$, all visits to every customer c for which $a_c > f_{TC}/2$ are removed from s. According to the second operator, $WorstTC_2$, if $a_c > f_{TC}/2$ for a particular customer c, then the arrival times for all visits to c are placed into a list $\mathcal{L}_c$ and sorted in either increasing or decreasing order. Next, list $\mathcal{L}_c$ is partitioned into two lists $\mathcal{L}_c^1$ and $\mathcal{L}_c^2$ by splitting the list $\mathcal{L}_c$ at the largest difference between two consecutive arrival times. Finally, for the list with smaller cardinality, the visits to customer c associated with each of the arrival times in the list are removed from routing plan s. This process is repeated for every customer c. This operator is inspired by a similar one in Kovacs et al. (2015a).

Reinsertion

Note that after removal has been carried out according to one of the above operators, a partial solution (i.e., partial routing plan) exists. That is, the *partial routing plan* is missing some of the required customer visits. In what follows, we let σ denote the partial routing plan that remains after removal is carried out on routing plan s.

Reinsertion to reduce travel cost ($Reinsert_1$)

This operator focuses on travel cost when reinserting customer visits. For every day d of the planning horizon, a list $\mathcal{L}_d$ of customer visits on day d that need to be reinserted into σ is maintained. Then, every customer visit $i \in \mathcal{L}_d$ that requires reinsertion is placed into the cheapest feasible insertion location among all routes in σ on day d, updating the partial routing plan after each insertion. If no feasible insertion locations exist for a particular customer visit, a new empty route is added to σ. Each list $\mathcal{L}_d$ is shuffled to introduce diversity into the order in which customer visits are considered for reinsertion.

Reinsertion to improve driver consistency ($Reinsert_2$)

This operator focuses on driver consistency when reinserting customer visits. A list $\mathcal{L}$ of customers who need one or more visits reinserted into σ is maintained. For each customer $c \in \mathcal{L}$, two possible scenarios exist: either customer c requires reinsertion for *all* of their visits, or *a subset* of their visits. These scenarios are discussed separately below.

Customer c requires reinsertion for a subset of their visits: For each driver, a count n_d of the number of times they visit customer c in σ is maintained. For a particular day d on which c requires reinsertion, the driver with maximum n_d is identified, where the driver has a route on day d and a feasible insertion location for the visit to c exists in that route. If multiple drivers meet these conditions, one is selected arbitrarily. A new empty route is added on day d if no drivers meet these conditions. The visit to customer c is inserted into the selected route on day d in the cheapest feasible insertion location. The driver count (n_d) for the selected driver is updated and the process is repeated for each day that customer c requires reinsertion. The order in which each day is considered is randomized to promote diversity.

Customer c requires reinsertion for all of their visits: For every day customer c requires service, the cheapest insertion locations within each route for which feasible insertion locations exist for customer c are noted. Each of these locations are recorded using (d, r, j, δ) where d is the day, r is the driver/route, j is the index of the customer that c will immediately precede if inserted and δ is the insertion cost. Next, for every driver r, a count n_r of the number of such locations in which they appear is tabulated. Furthermore, for every driver r, the sum of insertion costs over all such locations in which they appear as driver is computed and denoted Δ. Note that n_r can be at most m, the number of visits customer c requires. Note also that Δ represents the total travel cost associated with using driver r to perform all of the identified visits to customer c. Next, the driver r^* with maximal visit count n_{r^*} is identified. Visits to customer c are inserted into the previously

recorded locations in driver r^*'s routes (preceding customer j). If there are multiple such drivers, ties are broken using Δ (the minimum total insertion cost will be selected). Finally, if the maximum visit count n_{r^*} were equal to m, then the process of reinserting customer c is complete. Otherwise, for every visit for which customer c still requires reinsertion, the location with minimum δ is selected. If at any point in this process there are no feasible insertion locations on a required day, an empty route is created and a visit for customer c is inserted into it.

Note that when a new route is created in any of the reinsertion operators, the driver numbered $r + 1$ will be assigned to it if there already exist r drivers on the same day. In removal operators, it is possible that all customer visits on a route are removed; in this case, the route simply exists as an empty route and is not deleted from the routing plan. Note also that empty routes are defined to be a neighbor of any customer to be reinserted.

Removal and reinsertion to improve time consistency (RR)

This operator pair identifies the customer c^* with the biggest arrival time differential. Then, the visit to customer c^* which will be removed is identified as follows. First, the average of the earliest and latest arrival time to customer c is denoted $\bar{a}$ and the median arrival time to customer c is denoted $med(a)$. Then, if $med(a) > \bar{a}$, the visit with the earliest arrival time will be removed, and otherwise, the visit with the latest arrival time will be removed. Denote the day of this visit d^*. The process for reinserting the visit on day d^* for customer c^* begins by examining each route on day d^*. These routes are sorted into a list $\mathcal{L}$ in non-decreasing order of the angle between the location of customer c^* and the centroid of the route, taking the depot as the origin. Beginning with the first route in $\mathcal{L}$, the operator examines whether there is a feasible insertion location in the route that will improve the time consistency objective. If there are multiple such locations within the route, the one that improves time consistency the most is selected. If there are no feasible locations in the current route, the search proceeds through the list $\mathcal{L}$. If the end of list $\mathcal{L}$ is reached, the visit to customer c^* on day d^* is reinserted back into its original location.

Although the two worst time consistency removal operators, $WorstTC_1$ and $WorstTC_2$, are not used explicitly in this removal and reinsertion operator pair, they do aid in improving this objective. This is because they remove the customer with the largest arrival time differential. The removal and reinsertion operators used for each objective are shown in Table 2.2.

Obj.	Removal operator	Reinsertion operator
f_{TD}	$Rand_1$, $Rand_2$, Adj_1, Adj_2, $WorstDC_1$, $WorstDC_2$, $WorstTC_1$, $WorstTC_2$	$Reinsert_1$
f_{DC}	$Rand_1$, $Rand_2$, Adj_1, Adj_2, $WorstDC_1$, $WorstDC_2$, $WorstTC_1$, $WorstTC_2$	$Reinsert_2$
f_{TC}	RR	

Table 2.2: LNS operator pair used for each objective

2.5 Computational experiments and analysis

In this section, we first introduce the benchmark instances used in this paper to validate the performance of our proposed IMDLS for solving MoConVRP. Then, metrics used to compare the performance of the various algorithms are described. Finally, the results of the computational study are presented and a trade-off analysis between the objectives of cost minimization and consistency maximization is provided.

2.5.1 Benchmark instances and experiment setup

A total of 36 instances are taken from the literature to verify the performance of our proposed algorithm. They appeared first in Groër et al. (2009) and Kovacs et al. (2014b) based on the Christofides benchmark instances for VRP (Christofides and Eilon, 1969). All instances are generated in such a way that each customer requires service with a certain probability, namely, 0.5, 0.7 and 0.9, on each day of the planning horizon. These instances are classified into three groups, namely, $Group_{0.5}$, $Group_{0.7}$ and $Group_{0.9}$, according to their service probability.

The performance of our proposed IMDLS is compared against the original MDLS, three classical multi-objective algorithms including NSGAII, NNIA and SPEA2, and two more recent multi-objective algorithms, namely, NSGAIII and MOEA/D. For MOEA/D, there exist three decomposition approaches that convert a multi-objective optimization problem into a set of scalar optimization subproblems, which results in three algorithms, namely, the MOEA/D with the weighted sum approach (MOEA/D-WS), the MOEA/D with the Tchebycheff approach (MOEA/D-TCH) and the MOEA/D with the penalty-based boundary intersection arpproach (MOEA/D-PBI). Neither MDLS nor IMDLS require the recombination of two solutions during execution. However, crossover is an essential element of the comparison algorithms NSGAII, NNIA, SPEA2, NSGAIII and MOEA/D. The crossover operator used in this paper works in a removal and reinsertion fashion as in LNS: given two parent solutions A and B, a child solution is initialized as a copy of parent A. Then all the customers in the child solution whose driver consistency objectives are worse than that of parent solution B are removed from the child. After removal, $Reinsert_2$ is employed to reinsert the removed customers back to the partial child solution.
All of the comparison algorithms are implemented following their descriptions in the literature. The same crossover operator and large neighborhood search described in the literature are used whenever applicable. Both crossover and LNS operators are applied to parent solutions with probability 1 in the comparison algorithms. The population size in NSGAII is set to 100. The population size and archive size in SPEA2 are both set to 100. The sizes of the dominant population, active population and clone population in NNIA are set to 100, 20 and 100, respectively. The population size of NSGAIII is determined by the size of the user-defined reference point set. In this paper, we use the reference point set generated by Deb and Jain (2014) for three-objective optimization problems. It contains 91 reference points and the NSGAIII population size is set to 91. Similarly, the MOEA/D population size is determined by the number of weight vectors and we use the weight vectors provided by Qingfu and Hui (2007) for three-objective optimization problems. The

MOEA/D population size is therefore set to 351. The number of initial non-dominated solutions is set to 100 for both MDLS and IMDLS. The F_{max} in IMDLS is set to 100 in all experiments. A time limit of 30 minutes is set on all the algorithms and 10 replications are run for each instance. All algorithms are implemented in C++ and experiments are conducted on a computer with 2.70GHz CPU.

2.5.2 Metrics for comparison algorithms

Three metrics are employed to compare the performance of different multi-objective algorithms. First, *hypervolume* (I_H) is a unary operator that is able to indicate the convergence and diversity of a non-dominated approximation set for the Pareto frontier (Coello et al., 2007). It measures the size of the objective space covered by a set of non-dominated solutions $\mathcal{F}$. A reference point z is necessary to compute the hypervolume of $\mathcal{F}$. For maximization problems, it is common to set z as the origin (0, 0, 0). For minimization problems, z is usually set to a point with the worst values for each objective. Either way, larger hypervolumes indicate better performance. For our problem with three objectives, each solution $x \in \mathcal{F}$ in the objective space covers a cuboid defined by its coordinates $(f_1(x), f_2(x), f_3(x))$ and the reference point z. The hypervolume is computed as the size of the union of all such cuboids covered by solutions in $\mathcal{F}$. The WFG algorithm descried in While et al. (2012) is used to compute the hypervolume metric. Because the three objectives considered in this paper do not have the same scale, it is necessary to normalize the objective values before computing the hypervolume metric. To this end, for each instance, we record the best and worst values for each of the three objectives obtained by all algorithms over all 10 replications. Then all objective values are normalized into the range [0, 1] and the reference point is set to (1.1, 1.1, 1.1) for all instances.
Second, *coverage* (I_C) is a binary operator that compares the convergence of different non-dominated solution sets (Zitzler et al., 2000) . It measures the extent to which one solution set B is covered by another solution set A by comparing the number of solutions

in B that are dominated by solutions in A to the cardinality of B:

$$I_C(A, B) = \frac{|\{b \in B : \exists\ a \in A, a \prec b\}|}{|B|}. \tag{2.23}$$

The value $I_C(A, B) = 1$ means that all solutions in B are dominated by solutions in A, while $I_C(A, B) = 0$ means that none of the solutions in B are dominated by those in A. Note that $I_C(A, B)$ does not necessarily equal $1 - I_C(B, A)$; both values need to be computed. Larger values of coverage indicate better performance.

Third, the *unary multiplicative epsilon indicator* (I_ϵ) computes the minimum factor ϵ by which each point in the reference set R can be multiplied such that the resulting set is weakly dominated by set A (Zitzler et al., 2003):

$$I_\epsilon(A, R) = \inf_\epsilon \{\forall r \in R\ \exists a \in A : a \preceq_\epsilon r\} \tag{2.24}$$

This indicator is based on the ϵ-dominance relation, $\preceq_\epsilon$, which is defined as:

$$a \preceq_\epsilon r \Leftrightarrow \forall i = 1, \ldots M, a_i \leq \epsilon \cdot r_i \tag{2.25}$$

for a minimization prolem with M objectives and assuming that all points are positive in all objectives. A smaller I_ϵ value indicates better performance. The reference set R is constructed for each instance by taking the union of all replications of all algorithms and removing dominated solutions. Note that a I_ϵ value smaller than 1 indicates that A strictly dominates the reference set R.

2.5.3 Parameter tuning

In the proposed initial solution generation scheme, the parameter α is defined to decide whether a route r is considered as the neighbor of a customer c that is to be inserted. The performances of different α values are evaluated using the hypervolume metric on 12 instances from $Group_{0.7}$. Results from five replications are obtained for each instance using

the proposed IMDLS algorithm. A total of 18 α values are chosen from the range $(0, 360]$. Table 2.3 shows the comparison results. The first column in the table indicates the instance number. The first row shows the 18 α values. Each hypervolume value in the table is the averaged value of five replications. The last row gives the average hypervolume values across 12 instances. It can be seen from Table 2.3 that the best performance (the biggest hypervolume value, 0.87) is observed with $\alpha = 60$; therefore, this value is used in the final experiments.

α	20	40	60	80	100	120	140	160	180	200	220	240	260	280	300	320	340	360
1	1.24	1.23	1.24	1.23	1.22	1.15	1.16	1.17	1.04	1.11	1.14	1.22	1.23	1.18	1.22	1.22	1.15	1.16
2	1.00	0.93	1.01	0.92	0.77	0.89	0.72	0.82	0.77	0.72	0.77	0.77	0.77	0.77	0.76	0.77	0.78	0.77
3	1.00	1.03	1.02	1.00	1.00	0.97	1.02	0.79	0.81	0.96	0.88	1.01	0.92	0.86	0.94	0.96	0.86	0.91
4	0.71	0.78	0.79	0.75	0.62	0.62	0.59	0.61	0.60	0.59	0.57	0.58	0.58	0.59	0.56	0.60	0.58	0.58
5	0.58	0.53	0.62	0.52	0.50	0.49	0.44	0.41	0.44	0.40	0.42	0.37	0.43	0.42	0.44	0.43	0.43	0.42
6	1.15	1.13	1.13	1.07	1.08	1.08	1.13	1.14	1.13	1.13	1.14	1.14	1.15	1.14	1.14	1.14	1.14	1.14
7	0.84	0.85	0.88	0.80	0.85	0.78	0.81	0.76	0.88	0.85	0.77	0.83	0.89	0.89	0.86	0.89	0.89	0.90
8	0.87	0.87	0.85	0.76	0.77	0.82	0.78	0.80	0.78	0.84	0.88	0.87	0.87	0.88	0.88	0.86	0.88	0.87
9	0.44	0.42	0.66	0.52	0.50	0.61	0.52	0.60	0.62	0.65	0.64	0.65	0.65	0.65	0.66	0.65	0.65	0.66
10	0.42	0.46	0.44	0.42	0.33	0.34	0.44	0.33	0.43	0.44	0.45	0.47	0.48	0.49	0.47	0.49	0.48	0.48
11	0.73	0.79	0.83	0.81	0.83	0.84	0.82	0.77	0.84	0.81	0.78	0.78	0.82	0.69	0.74	0.80	0.73	0.71
12	1.00	1.02	1.02	0.79	0.90	0.97	0.85	1.00	0.87	0.85	0.91	0.99	1.01	1.03	1.02	1.02	1.02	1.01
$\bar{I_H}$	0.83	0.84	**0.87**	0.80	0.78	0.80	0.77	0.77	0.77	0.78	0.78	0.81	0.82	0.80	0.81	0.82	0.80	0.80

Table 2.3: Parameter α tuning results on $Group_{0.7}$

2.5.4 Algorithm performance comparison

This section presents a comparison of the multi-objective algorithms according to the metrics described in the previous section. Then, the frequencies with which various LNS operator pairs in IMDLS are invoked are presented.

Table 2.4 provides the hypervolumes for the instances in $Group_{0.5}$. The first column indicates the instance number. Each of the next nine columns corresponds to one of the comparison algorithms. Each entry in the table gives the average hypervolume across 10 replications for the instance number and algorithm indicated by the row and column labels. The last two columns give the hypervolumes for the reference set R and the $I_H^{\%}$ value defined by $I_H(\text{IMDLS})/I_H(R)$, respectively. The last row provides the averages across all 12 instances for each of the comparison algorithms. The best result for each instance is

highlighted in bold.

It can be seen from Table 2.4 that IMDLS outperforms MDLS for all instances in $Group_{0.5}$. However, IMDLS is outperformed by other comparison algorithms on 6 out of the 12 instances. The reference set R can be considered as a near-optimal approximation to the real Pareto frontier and IMDLS is able to cover 90.48 percent of the objective space covered by R on average. Tables 2.5 and 2.6 provide hypervolumes for instances in $Group_{0.7}$ and $Group_{0.9}$, respectively. For both groups of instances, IMDLS outperforms MDLS. On the other hand, IMDLS is outperformed by other comparison algorithms on most instances. However, IMDLS is able to cover about 85 percent of the objective space covered by R in both instance groups.

Instance	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS	R	$I_H^{\%}$
			WS	TCH	PBI						
1	1.163	1.175	1.151	1.176	1.120	1.162	1.156	1.016	**1.179**	1.239	95.17%
2	1.174	1.167	1.152	1.161	1.038	1.136	1.165	0.854	**1.176**	1.239	94.89%
3	1.170	1.172	1.139	1.165	1.018	1.159	1.186	0.883	**1.194**	1.252	95.38%
4	1.047	1.054	0.990	1.023	0.732	0.991	1.026	0.533	**1.103**	1.212	90.95%
5	1.121	1.123	1.034	1.131	0.912	1.099	1.098	0.717	**1.144**	1.262	90.68%
6	1.113	1.126	1.121	**1.147**	1.098	1.093	1.114	0.962	1.139	1.220	93.34%
7	1.101	1.115	1.082	**1.125**	1.048	0.979	1.113	0.609	1.044	1.223	85.32%
8	**1.116**	1.100	1.066	1.096	0.969	1.077	1.083	0.610	1.104	1.228	89.84%
9	0.965	1.038	0.879	**1.054**	0.916	0.891	1.011	0.355	0.987	1.208	81.70%
10	1.026	1.011	0.941	**1.070**	0.893	0.893	1.054	0.391	0.995	1.235	80.56%
11	1.143	**1.184**	1.040	1.120	0.931	1.157	1.158	0.607	1.155	1.283	90.01%
12	1.239	1.226	1.213	1.230	1.010	1.227	1.225	0.854	**1.248**	1.274	97.92%
Average	1.115	1.124	1.067	**1.125**	0.974	1.072	1.116	0.699	1.122	1.240	90.48%

Table 2.4: Hypervolume results for $Group_{0.5}$

Table 2.7 provides the coverage comparisons of IMDLS with other algorithms for different instance groups. The $M_C(*, \text{IMDLS})$ value indicates the percentage of solutions produced by IMDLS that are dominated by at least one solution generated by the comparison algorithm *. Take the third entry in the second row for example. The value 22.60% means 22.60% of the non-dominated solutions produced by IMDLS are dominated by at least one of the solutions in the set produced by NSGAII. On the other hand, the 58.37% in the third row indicates that 58.37% of the solutions obtained by NSGAII are dominated by at least one of the solutions in the set generated by IMDLS. It is worth noting that 98.65% of

Instance	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS	R	$I_H^{\%}$
			WS	TCH	PBI						
1	1.163	1.150	1.141	1.149	1.022	1.159	1.155	0.946	**1.173**	1.216	96.42%
2	1.144	**1.149**	1.116	1.135	0.937	1.109	1.124	0.762	1.144	1.228	93.15%
3	**1.170**	1.167	1.111	1.164	0.957	1.147	1.162	0.708	1.130	1.243	90.88%
4	1.117	**1.127**	1.101	1.066	0.801	1.079	1.021	0.570	1.035	1.247	82.99%
5	**0.977**	0.845	0.828	0.826	0.586	0.770	0.819	0.316	0.877	1.197	73.21%
6	1.165	1.166	1.138	1.138	1.042	1.150	1.149	0.767	**1.173**	1.238	94.70%
7	**1.082**	1.071	0.981	1.071	0.926	0.929	1.042	0.412	0.995	1.220	81.51%
8	**1.085**	1.077	1.042	1.079	0.956	1.052	1.045	0.460	1.039	1.230	84.48%
9	1.035	1.059	0.970	**1.077**	0.800	0.856	1.036	0.269	0.798	1.220	65.43%
10	**1.010**	0.972	0.896	1.005	0.829	0.796	0.884	0.236	0.829	1.215	68.24%
11	**1.243**	1.232	1.180	1.210	1.060	1.187	1.172	0.406	1.173	1.283	91.42%
12	**1.160**	1.146	1.061	1.118	0.834	1.095	1.131	0.561	1.120	1.249	89.70%
Average	**1.113**	1.097	1.047	1.086	0.896	1.028	1.062	0.534	1.040	1.232	84.34%

Table 2.5: Hypervolume results for $Group_{0.7}$

Instance	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS	R	$I_H^{\%}$
			WS	TCH	PBI						
1	**1.152**	1.142	1.097	1.131	1.018	1.144	1.137	0.942	1.143	1.219	93.78%
2	1.111	1.096	1.031	1.046	0.861	1.082	1.081	0.455	**1.122**	1.217	92.23%
3	**1.159**	1.150	1.103	1.133	1.069	1.137	1.130	0.727	1.086	1.241	87.54%
4	**1.148**	1.135	1.082	1.016	0.940	1.085	1.121	0.466	1.067	1.263	84.49%
5	**1.119**	1.065	0.974	0.936	0.775	0.938	1.116	0.473	1.024	1.278	80.14%
6	**1.138**	1.129	1.058	1.121	0.965	1.109	1.097	0.933	1.106	1.213	91.16%
7	**1.169**	1.165	1.120	1.132	0.951	1.111	1.137	0.505	1.131	1.252	90.34%
8	**1.108**	1.104	0.990	1.052	0.849	1.082	1.074	0.569	1.097	1.214	90.39%
9	**1.189**	1.166	1.119	1.115	1.027	1.043	1.148	0.305	0.949	1.265	75.02%
10	1.059	**1.062**	1.048	1.012	0.965	0.940	0.967	0.231	0.881	1.255	70.22%
11	1.077	**1.130**	1.019	1.030	0.940	1.054	1.088	0.422	0.962	1.284	74.92%
12	**1.231**	1.210	1.105	1.140	0.961	1.201	1.166	0.574	1.169	1.278	91.49%
Average	**1.138**	1.130	1.062	1.072	0.943	1.077	1.105	0.550	1.061	1.248	85.14%

Table 2.6: Hypervolume results for $Group_{0.9}$

the solutions produced by the original MDLS are dominated by at least one of the solutions obtained by IMDLS, while only 0.26% solutions generated by IMDLS are dominated by solutions produced by MDLS. Overall, our proposed IMDLS produces the best non-dominated solution set among the nine comparison algorithms for all instance groups except NSGAII and NSGAIII on $Group_{0.9}$. These two occurrences are indicated in bold in Table 2.7. Detailed pairwise comparison results are given in Tables 1, 2 and 3 in the online supplement.

Table 2.8 shows the unary multiplicative epsilon indicator comparisons of IMDLS with other algorithms for instances in $Group_{0.5}$. The first columns gives the instance number

Instance group	Coverage metric	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS
				WS	TCH	PBI			
$Group_{0.5}$	$M_C(*, \text{IMDLS})$	22.60%	24.11%	12.00%	22.92%	5.06%	15.56%	20.92%	0.26%
	$M_C(\text{IMDLS}, *)$	58.37%	56.25%	53.08%	47.40%	71.27%	71.80%	61.78%	98.65%
$Group_{0.7}$	$M_C(*, \text{IMDLS})$	27.95%	29.10%	11.73%	21.11%	4.60%	16.00%	20.67%	0.14%
	$M_C(\text{IMDLS}, *)$	49.25%	49.03%	48.70%	41.27%	45.99%	65.08%	58.23%	98.22%
$Group_{0.9}$	$M_C(*, \text{IMDLS})$	**44.80**%	**41.21**%	18.44%	24.61%	12.38%	28.04%	35.63%	0.34%
	$M_C(\text{IMDLS}, *)$	34.82%	38.11%	45.09%	45.93%	39.26%	55.03%	49.31%	97.25%

Table 2.7: Coverage comparison results

and the next nine columns correspond to one of the comparison algorithms. Each entry in the table gives the average I_ϵ across 10 replications for the instance number and algorithm indicated by the row and column labels. The last row provides the averages across all 12 instances for each of the comparison algorithms. The best value for each instance is shown in bold.

It can be seen from Table 2.8 that IMDLS outperforms the comparison algorithms for all instances in $Group_{0.5}$ except instance 11. These observations are also supported by the comparisons given in Tables 2.9 and 2.10, corresponding to instances in $Group_{0.7}$ and $Group_{0.9}$, respectively.

Note that there exist some large unary epsilon indicator values, sometimes over 300, in Tables 2.8, 2.9 and 2.10. They are due to very small normalized travel cost objective values. Such a scenario occurs any time the final non-dominated solution set of an algorithm includes a solution with very good travel distance while another comparison algorithm fails to do so.

It can be seen from the above comparisons that IMDLS outperforms other comparison algorithms with respect to the coverage and multiplicative epsilon indicators. On the other hand, IMDLS is outperformed by other comparison algorithms with respect to hypervolume. These conflicting observations with hypervolume and unary epsilon indicators are due to the fact that these two indicators work on different principles and therefore opposite preference orderings for two approximation sets A ans B may result (Knowles et al., 2006). In the next section, we conduct our trade-off analysis based on the reference set instead of the approximation set produced by IMDLS alone.

Instance	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS
			WS	TCH	PBI				
1	14.58	14.41	13.50	16.41	29.41	14.19	17.36	63.61	**12.18**
2	43.74	47.15	27.58	87.39	160.13	50.21	59.74	175.82	**15.32**
3	54.92	41.66	36.07	60.14	198.62	46.72	47.70	180.20	**18.22**
4	120.22	96.23	69.39	153.27	420.58	112.76	111.09	341.22	**40.39**
5	186.80	200.65	116.61	251.49	462.45	163.71	200.43	356.19	**60.93**
6	25.95	28.10	24.89	29.61	61.34	25.03	25.12	41.46	**16.21**
7	37.51	47.37	41.68	59.99	60.10	25.64	37.65	138.45	**14.02**
8	45.11	59.83	45.50	78.29	261.88	31.72	44.80	165.59	**12.90**
9	154.30	164.52	108.72	227.74	283.14	143.98	158.88	379.83	**49.48**
10	187.53	210.49	191.86	293.09	490.39	213.90	194.74	548.13	**50.05**
11	103.74	75.24	155.71	160.95	286.47	**57.86**	103.82	374.36	62.71
12	21.09	24.96	19.39	31.02	248.79	18.41	22.72	219.54	**9.33**
Average	82.958	84.219	70.907	120.783	246.942	75.343	85.338	248.702	**30.144**

Table 2.8: Unary multiplicative indicator results for $Group_{0.5}$

Instance	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS
			WS	TCH	PBI				
1	22.98	28.38	18.97	44.10	189.15	19.92	31.47	93.01	**14.48**
2	88.99	103.25	71.89	126.13	432.68	84.49	108.74	267.52	**39.11**
3	65.18	44.71	68.59	75.65	303.68	45.19	42.07	236.01	**19.32**
4	188.02	161.23	202.92	297.15	535.37	168.06	186.83	425.45	**66.34**
5	230.24	251.09	293.34	321.38	642.59	202.48	214.12	520.50	**57.79**
6	32.27	41.87	56.14	71.84	154.47	28.44	63.61	121.73	**11.84**
7	106.07	86.54	113.35	175.32	371.77	78.14	101.63	338.01	**24.20**
8	123.45	123.26	141.66	184.85	409.71	102.28	124.57	288.91	**41.62**
9	251.46	197.96	270.47	390.78	749.32	197.71	265.33	588.60	**67.68**
10	257.56	281.08	361.82	387.46	771.87	311.03	255.07	680.68	**69.54**
11	58.82	60.91	118.70	122.02	482.20	120.37	84.38	870.78	**57.74**
12	92.91	99.36	83.23	178.82	585.47	95.34	98.19	408.62	**40.57**
Average	126.496	123.304	150.091	197.959	469.024	121.121	131.335	403.320	**42.519**

Table 2.9: Unary multiplicative indicator results for $Group_{0.7}$

Next the frequencies with which each removal/reinsertion operator pairs are invoked in IMDLS is reported. Table 2.11 provides these results for the instances in $Group_{0.7}$. The first row gives the name of the removal operator and the second row gives the name of the reinsertion operator that is paired with it. Each of the eight removal operators is paired with both of the reinsertion operator designed for travel distance (TD) and driver consistency (DC), which results in a total of 16 operator pairs. The operator pair designed for TC is not shown in this table because it is invoked with frequency 1.0 in IMDLS by definition. Each element in the table is an average frequency over 10 replications for each

Instance	NSGAII	NSGAIII	MOEA/D			NNIA	SPEA2	MDLS	IMDLS
			WS	TCH	PBI				
1	27.45	23.10	31.48	38.01	135.96	22.81	27.37	100.41	**17.61**
2	84.19	98.13	144.11	177.26	377.95	99.99	127.23	363.62	**50.96**
3	51.72	43.00	82.94	76.78	120.82	38.93	58.80	222.16	**23.84**
4	141.23	129.35	176.59	245.84	328.03	168.09	144.64	476.90	**93.79**
5	152.19	133.88	248.76	314.99	579.71	182.68	135.19	587.47	**110.91**
6	**40.31**	48.00	85.98	59.23	224.61	56.61	71.57	171.77	48.22
7	72.53	54.31	139.42	145.68	452.53	71.32	76.41	427.57	**38.86**
8	107.14	106.46	169.22	150.00	361.14	106.86	112.53	291.76	**71.68**
9	89.14	102.69	267.40	279.12	495.92	165.84	130.55	662.54	**67.49**
10	281.37	246.13	443.97	521.05	588.19	343.82	269.01	956.20	**186.51**
11	600.71	**409.76**	739.77	724.90	1005.15	608.35	529.74	1323.39	620.42
12	**59.38**	77.63	211.80	169.89	391.30	82.38	113.85	521.52	81.96
Average	142.281	122.704	228.453	241.896	421.775	162.306	149.741	508.776	**117.687**

Table 2.10: Unary multiplicative indicator results for $Group_{0.9}$

instance in $Group_{0.7}$. In each row, the entries in the eight columns corresponding to either TD or DC sum to 1.0. The average application frequency for each operator pair over all instances is shown in the last row of the table.

It can be seen from the last row of Table 2.11 that with regard to the TD objective, removal operator $WorstDC$ is chosen with frequency 22.72% (18.92% + 3.80%) on average, more than any other operator. The next most frequently invoked removal operator is $WorstTC$, with frequency 14.26% (5.57% + 8.69%) on average. Moreover, random removal seems to be more effective in reducing travel cost than adjacent removal (Adj_1 and Adj_2). With regard to the driver consistency objective, it is clear that $WorstDC$ is the most effective in improving the objective with a frequency of 34.59% on average. Next, $WorstTC$ removal is the next most frequently invoked operator to improve driver consistency. Additionally, random removal is more effective than adjacent removal in improving driver consistency. Table 2.11 also shows that random removal and adjacent removal are more effective in improving travel cost than driver consistency.

Removal:	$Rand_1$		$Rand_2$		Adj_1		Adj_2		$WorstDC_1$		$WorstDC_2$		$WorstTC_1$		$WorstTC_2$	
Reinsertion:	TD	DC	TD	DC	TD	DC	TD	DC	TD	DC	TD	DC	TD	DC	TD	DC
1	4.28%	2.55%	3.46%	3.79%	4.36%	2.26%	4.11%	2.62%	14.13%	20.07%	5.38%	9.78%	5.15%	4.49%	9.13%	4.44%
2	2.98%	1.11%	3.48%	2.01%	2.39%	2.35%	2.07%	2.28%	20.58%	25.32%	2.89%	10.12%	5.66%	2.83%	9.96%	3.98%
3	3.86%	1.47%	3.51%	2.47%	3.34%	1.23%	2.46%	1.28%	18.96%	26.73%	5.02%	10.31%	4.73%	3.21%	8.13%	3.31%
4	2.87%	1.38%	3.48%	1.01%	3.35%	1.62%	3.49%	2.58%	19.01%	29.57%	4.57%	10.23%	4.69%	0.97%	8.53%	2.64%
5	4.96%	1.69%	4.76%	3.91%	4.45%	1.41%	3.88%	2.67%	16.58%	22.77%	1.48%	11.96%	6.41%	3.00%	7.48%	2.60%
6	4.41%	2.90%	3.73%	2.59%	4.07%	2.83%	3.76%	2.69%	17.66%	16.48%	3.21%	10.45%	5.28%	4.90%	7.87%	7.15%
7	3.24%	1.93%	1.52%	1.04%	3.81%	1.59%	2.37%	2.35%	22.89%	27.31%	4.65%	9.28%	4.62%	3.87%	6.90%	2.64%
8	4.64%	0.96%	4.15%	3.05%	2.08%	1.89%	1.43%	1.60%	19.21%	23.04%	4.22%	12.49%	4.76%	2.56%	9.51%	4.40%
9	3.69%	2.17%	2.93%	2.91%	3.97%	1.99%	3.04%	2.48%	19.07%	23.38%	3.04%	10.02%	6.03%	4.00%	8.24%	3.05%
10	4.79%	1.15%	4.48%	3.47%	3.24%	2.74%	3.88%	1.11%	17.44%	23.42%	1.08%	9.82%	6.72%	4.21%	8.37%	4.08%
11	2.69%	1.21%	2.62%	0.87%	0.70%	1.14%	0.35%	2.35%	18.98%	21.03%	6.82%	12.53%	6.89%	5.03%	10.95%	5.83%
12	1.72%	1.22%	2.16%	1.22%	3.47%	1.12%	1.82%	1.22%	22.49%	27.49%	3.17%	11.50%	5.95%	2.70%	9.23%	3.52%
Average	3.68%	1.65%	3.36%	2.36%	3.27%	1.85%	2.72%	2.10%	18.92%	23.88%	3.80%	10.71%	5.57%	3.48%	8.69%	3.97%

Table 2.11: Frequency with which various operator pairs are invoked in IMDLS

2.5.5 Trade-off analysis

In this section, we aim to analyze the non-dominated reference set $\mathcal{R}$ obtained by all algorithms in all runs to facilitate managerial decision making, for there usually exist many solutions in $\mathcal{R}$ and it is desirable to identify the best compromise solution to implement in practice. To this end, we employ the level diagram technique proposed by Blasco et al. (2008). First, the objective values of non-dominated solutions in $\mathcal{R}$ are normalized on a [0,1] scale. Next, the euclidean norm is computed for each solution $s \in \mathcal{R}$, where $\bar{f}_i$ represents the normalized value of objective i: $E(s) = \sqrt{\sum_{i=1}^{3} \bar{f}_i^{\,2}(s)}$. The original objective values of solutions in $\mathcal{R}$ are then plotted against their euclidean distance in separate graphs corresponding to each of the three objectives.

Figure 2.4 depicts the level diagram of non-dominated reference set $\mathcal{R}$ for instance 1 in $Group_{0.9}$. The x axis in each graph provides the original objective values for TD, DC and TC, respectively. The y axis provides the euclidean distances for solutions in $\mathcal{R}$. Taken together, the three points at the same coordinate in each of the three graphs correspond to a particular solution in $\mathcal{R}$. For example, the point denoted by a triangle in black color corresponds to a particular solution in $\mathcal{R}$ having a travel distance of 2412.96, driver consistency of 2 and time consistency of 107.87. This point corresponds to the solution in $\mathcal{R}$ having lowest travel distance. In the graph, a smaller value of euclidean distance indicates a better compromise solution. Therefore, the best compromise solution in $\mathcal{R}$ is

denoted by a red square in Figure 2.4. It has a travel distance of 2507.75, driver consistency 1 and time consistency 14.05.

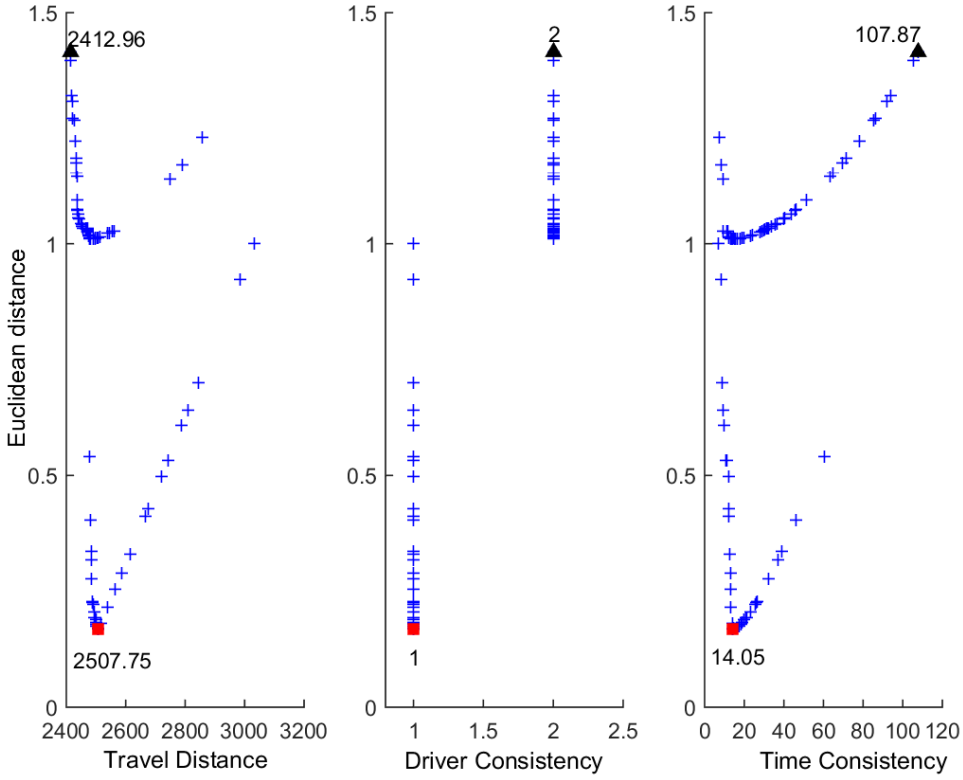


Figure 2.4: Level diagram of instance 1

Let s^{BC} denote the best compromise solution in the reference set $\mathcal{R}$. It is identified as described above. Furthermore, let s^{TD}, s^{DC} and s^{TC} denote the solutions with best travel distance, driver consistency and time consistency values. This results in twelve values recorded for each test instance. Table 2.12 provides these twelve values for each test instance. Consider for example the best compromise solution s^{BC} for instance 1 of $Group_{0.5}$ in Table 2.12. The travel distance of the best travel distance solution for this instance is 1528.72, while the driver and time consistency values are 3 and 91.00, respectively. For comparison purposes, the travel distance of the best compromise solution (s^{BC}) is 1642.34, approximately 7.4% higher than the travel distance in the best travel distance solution. However, the increase in travel distance results in better consistency values, with driver

and time consistencies of 1 (a 66.67% decrease) and 38.13 (a 58.10% decrease), respectively. From the table, it can be seen that improvement in time consistency typically comes at a higher cost, in terms of travel distance, than does improvement in driver consistency.

The next table provides a summary of the trade-offs between the traditional efficiency objective of travel cost and the consistency considerations. Specifically, Table 2.13 compares the three objective values for three pairs of solutions: the best travel distance solution compared with the best compromise solution, the best travel distance compared with the best driver consistency solution, and the best travel distance compared with the best time consistency solution. The three rows of the table present the three comparisons in that order. The first row indicates that if a manager chooses the best compromise solution, they can expect to increase travel distance by 5.02%, decrease driver consistency objective value by 61.57%, and decrease time consistency objective value by 76.47%, on average, when compared with the compromise solution having best travel distance. That is, an approximate 60% improvement in driver consistency and 75% improvement in time consistency comes at a cost of a 5% increase in travel distance. Now suppose instead of seeking the best compromise among all three objectives, the manager is only interested in improving driver consistency. Then, they can expect travel distance to increase by 5.49%, driver consistency objective value to decrease by 64.81%, and time consistency objective value to decrease by 76.30%, when compared with the compromise solution with best travel distance, on average. This table also supports the conclusion that improvements in time consistency come at a greater increase in travel costs than do improvements in driver consistency.

In previous sections, it is assumed that all vehicles leave the depot at time 0. This rigidness limits the potential to further improve time consistency. In this paper, we also consider the case where vehicles are allowed to wait at the depot to improve time consistency. Note that vehicles are not permitted to wait at customer locations after finishing service. In order to determine the vehicle departure time, we employ the post-optimization algorithm proposed

Group	Inst.	s^{BC}			s^{TD}			s^{DC}			s^{TC}		
		f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}
0.5	1	1642.34	1	38.13	1528.72	3	91.00	1642.34	1	38.13	2044.93	1	10.44
	2	2407.38	2	28.34	2313.96	3	82.55	2576.84	1	21.43	2815.56	2	10.16
	3	2661.75	1	32.92	2500.36	3	141.42	2661.75	1	32.92	2994.89	2	15.63
	4	3364.70	1	28.04	3091.29	3	106.36	3364.70	1	28.04	3997.73	2	15.87
	5	4067.11	1	36.02	3831.96	3	99.20	4067.11	1	36.02	4364.44	1	16.98
	6	1749.24	1	40.59	1593.66	3	130.66	1749.24	1	40.59	2285.23	2	16.83
	7	2852.82	1	41.35	2633.98	3	134.61	2852.82	1	41.35	3154.58	2	20.10
	8	2908.39	1	58.28	2705.74	3	195.55	2908.39	1	58.28	3194.53	2	31.83
	9	3811.95	1	54.09	3490.22	4	170.80	3811.95	1	54.09	4549.87	2	30.55
	10	4593.66	1	50.14	4308.64	3	169.30	4593.66	1	50.14	5310.85	1	33.97
	11	3265.39	1	20.44	3222.56	2	221.78	3265.39	1	20.44	3657.10	2	10.71
	12	2866.80	1	15.51	2713.87	3	75.95	2866.80	1	15.51	3066.74	1	7.70
0.7	1	2110.59	1	28.01	1963.07	3	122.24	2110.59	1	28.01	2272.77	2	12.38
	2	3459.85	2	18.79	3220.35	3	74.22	3586.04	1	23.65	4167.13	2	9.72
	3	3274.16	1	27.81	3152.65	3	128.49	3274.16	1	27.81	3468.50	1	14.47
	4	4512.07	1	18.14	4253.73	3	101.52	4512.07	1	18.14	4943.33	2	11.09
	5	5726.25	1	22.54	5416.51	3	101.52	5726.25	1	22.54	6175.66	2	10.93
	6	2340.72	1	42.28	2226.34	4	157.76	2340.72	1	42.28	2717.13	1	22.13
	7	3905.77	1	43.52	3650.57	3	137.37	3905.77	1	43.52	4541.16	2	21.81
	8	3619.01	1	54.24	3444.17	3	184.26	3619.01	1	54.24	3926.42	1	31.24
	9	4989.20	2	54.74	4820.86	4	160.47	5209.78	1	52.82	5744.70	1	31.58
	10	6122.81	2	60.76	5825.86	4	159.41	6182.47	1	75.49	6459.38	2	34.14
	11	4487.90	1	16.47	4398.88	3	185.00	4487.90	1	16.47	5986.82	2	8.29
	12	3509.10	1	12.45	3244.45	3	85.78	3509.10	1	12.45	4005.38	1	6.92
0.9	1	2507.75	1	14.05	2412.96	2	107.87	2507.75	1	14.05	3031.65	1	7.06
	2	4026.83	1	13.02	3854.00	3	70.75	4026.83	1	13.02	4953.14	1	2.09
	3	4029.06	1	13.72	3956.20	2	102.97	4029.06	1	13.72	4523.35	1	4.15
	4	5021.37	1	16.56	4930.39	2	93.63	5021.37	1	16.56	5513.28	1	5.12
	5	6612.57	1	6.38	6467.25	2	76.96	6612.57	1	6.38	7794.14	1	2.83
	6	2618.31	1	38.60	2468.51	3	167.49	2618.31	1	38.60	3255.85	2	17.22
	7	4404.69	1	30.76	4287.76	3	121.26	4404.69	1	30.76	5460.18	1	11.62
	8	4204.22	1	43.24	4094.70	3	185.45	4204.22	1	43.24	4621.41	1	18.43
	9	5793.50	1	33.68	5684.89	3	171.27	5793.50	1	33.68	6714.30	1	21.24
	10	7151.66	1	35.88	6951.80	3	158.25	7151.66	1	35.88	8451.13	1	20.21
	11	5073.55	1	13.51	5018.10	2	118.06	5073.55	1	13.51	6861.26	1	2.98
	12	4018.81	1	7.44	3967.80	3	58.44	4018.81	1	7.44	4885.07	1	2.17

Table 2.12: Four solutions identified in $\mathcal{R}$

	f_{TD}	f_{DC}	f_{TC}
s^{TD} vs. s^{BC}	5.02% ↗	61.57% ↘	76.47% ↘
s^{TD} vs. s^{DC}	5.49% ↗	64.81% ↘	76.30% ↘
s^{TD} vs. s^{TC}	21.88% ↗	50.00% ↘	88.77% ↘

Table 2.13: Comparisons of four solutions

by Kovacs et al. (2015a). This algorithm is called every time before the time consistency objective value is computed. Table 2.14 shows the four solutions identified from the reference set $\mathcal{R}$ generated by all the mentioned multi-objective algorithms considering flexible vehicle departure time. Table 2.15 shows the trade-off analyais in the case of flexible vehicle departure time. Compared to Table 2.13 in which 76.47% improvement of time consistency is obtained at the cost of 5.02% increase in travel cost, the first row in Table 2.15 indicates that 73.33% improvement of time consistency could be achieved at the cost of 4.40% increase in travel cost. This comparison shows that better time consistency could be achieved by allowing flexible vehicle departure time.

2.6 Conclusion

This paper presents a multi-objective variant of the Consistent Vehicle Routing Problem (MoConVRP). Instead of modeling consistency considerations such as driver consistency and time consistency as constraints as in the majority of the ConVRP literature, they are included as objectives. Furthermore, instead of formulating a single weighted objective that relies on specifying relative priorities among objectives, an approach to approximate the Pareto frontier is developed. Specifically, an improved version of multi-directional local search is developed. The updated algorithm, IMDLS, makes use of large neighborhood search (LNS) to find solutions which are improved according to at least one objective to add to the set of non-dominated solutions at each iteration. The performance of IMDLS is compared with MDLS, three classical multi-objective algorithms and two recent state-of-the-art multi-objective algorithms on a set of ConVRP test instances from the literature. The computational study validates the superior performance of IMDLS.

Group	Inst.	s^{BC}			s^{TD}			s^{DC}			s^{TC}		
		f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}
0.5	1	1586.99	2	30.6853	1529.39	3	91.0022	1633.75	1	40.721	1753.46	2	15.1533
	2	2434.79	2	19.5592	2321.68	3	80.7748	2582.27	1	17.7844	2957.48	1	10.4705
	3	2664.21	1	28.358	2502.41	3	92.7609	2664.21	1	28.358	2959.68	1	13.8148
	4	3389.79	1	29.2694	3103.61	3	93.8438	3389.79	1	29.2694	3713.89	1	14.7006
	5	4089.95	1	30.1747	3878.07	3	74.4973	4089.95	1	30.1747	4393.96	1	16.2428
	6	1642.92	2	45.7791	1593.66	3	139.511	1713.12	1	61.0159	1907.17	1	21.9954
	7	2852.59	1	38.4889	2631.1	3	128.383	2852.59	1	38.4889	3346.09	2	24.241
	8	2761.84	2	68.9921	2688.73	3	194.013	2900.04	1	73.2841	3064.59	2	37.3059
	9	3814.39	1	66.6259	3492.12	3	155.025	3814.39	1	66.6259	4089.27	2	38.2727
	10	4546.31	2	60.3602	4314.94	4	169.721	4660.58	1	63.0563	5010.58	2	41.0381
	11	3288.04	1	26.3474	3221.5	3	223.044	3288.04	1	26.3474	3705.56	1	10.2576
	12	2768.09	2	10.8564	2715.22	3	64.5884	2869.35	1	17.7951	3161.04	1	7.4221
0.7	1	2013.28	2	32.8865	1963.33	3	148.845	2110.59	1	28.0129	2349.85	1	11.2718
	2	3377.64	2	30.0914	3226.73	3	55.2659	3544.2	1	19.7436	3798.37	1	12.6151
	3	3205.74	2	30.8166	3161.99	4	97.2011	3272.45	1	28.836	3333.61	1	15.1584
	4	4548.69	1	22.7975	4297.84	3	71.0542	4548.69	1	22.7975	4719.92	2	13.1476
	5	5741.13	1	38.9752	5499.39	3	74.3167	5741.13	1	38.9752	6009.83	2	10.5656
	6	2332.69	1	41.4395	2234.74	3	170.489	2332.69	1	41.4395	2428.15	2	22.2407
	7	3924.78	1	36.5621	3641.56	4	121.943	3924.78	1	36.5621	4552.54	1	24.3287
	8	3684.04	1	54.1701	3446.63	3	194.33	3684.04	1	54.1701	4078.4	2	35.7646
	9	5167.4	1	54.034	4815.78	3	156.357	5167.4	1	54.034	5828.51	2	34.9573
	10	6068.1	1	63.7833	5781.7	4	169.293	6068.1	1	63.7833	6660.54	2	37.3423
	11	4502.34	1	13.9685	4411.71	2	119.241	4502.34	1	13.9685	6156.64	1	6.92
	12	3513.9	1	13.1922	3239.13	3	80.5811	3513.9	1	13.1922	4272.47	1	6.3995
0.9	1	2507.75	1	14.0522	2412.96	2	88.544	2507.75	1	14.0522	2883.21	1	6.3807
	2	4072.92	1	8.4836	3904.45	3	52.0331	4072.92	1	8.4836	4675.58	2	1.8941
	3	4013.72	1	16.6845	3953.32	2	113.316	4013.72	1	16.6845	4392.39	1	6.9418
	4	5007.31	1	16.5556	4911.24	3	82.2048	5007.31	1	16.5556	5701.5	1	5.8038
	5	6588.17	1	14.9041	6499.9	2	64.4741	6588.17	1	14.9041	6904.98	1	3.4049
	6	2618.31	1	38.5992	2476.73	3	164.468	2618.31	1	38.5992	3041.66	1	16.0138
	7	4402.07	1	30.7596	4259.99	3	105.628	4402.07	1	30.7596	5231.76	1	11.9757
	8	4250.79	1	33.5165	4151.71	2	167.186	4250.79	1	33.5165	5011.52	1	16.9418
	9	5752.77	1	34.406	5655.93	3	170.887	5752.77	1	34.406	6218.04	1	22.7931
	10	7143.1	1	31.956	6948.86	3	154.354	7143.1	1	31.956	7880.53	1	20.6984
	11	5325.04	1	12.8723	5091.6	2	127.292	5325.04	1	12.8723	6952.12	1	4.1422
	12	4024.25	1	5.662	3968.65	2	103.227	4024.25	1	5.662	4996.38	1	2.169

Table 2.14: Four solutions identified in $\mathcal{R}$ with flexible vehicle departure time

	f_{TD}	f_{DC}	f_{TC}
s^{TD} vs. s^{BC}	4.40% ↗	56.48% ↘	73.33% ↘
s^{TD} vs. s^{DC}	5.44% ↗	64.35% ↘	73.04% ↘
s^{TD} vs. s^{TC}	18.48% ↗	53.70% ↘	86.62% ↘

Table 2.15: Comparisons of four solutions with flexible vehicle departure time

Traditionally, travel distance has been the most common objective in routing problems in the literature. While customer service considerations such as driver and time consistency are becoming increasingly important in certain industries such as small package delivery and home healthcare, their associated costs are not well known. There are studies that quantify increases in travel distance when consistency considerations are implemented using hard constraints. However, depending on the application, strict enforcement of consistency considerations may not be required. The relative costs of "good but not perfect" consistency was not previously known. The results of our computational study suggest that pursuing the best compromise solution among all three objectives may increase travel costs by about 5% while improving driver and time consistency by approximately 60% and over 75% on average, when compared with a compromise solution having lowest overall travel distance.

Directions for future work can include studying consistency concerns in the context of other routing problem variants. For example, in the general ConVRP, customers require service on various days throughout a planning horizon, and the specific days each customer requires service are treated as problem input. The days each customer requires service could be modeled using decision variables instead, as in the Periodic VRP. Additionally, the problem studied in this paper is static and deterministic. Real problems are often dynamic and/or stochastic in nature. Therefore, dynamic and/or stochastic variants of the Consistent VRP can be explored.

Reference

Ashlea R. Bennett and Alan L. Erera. Dynamic periodic fixed appointment scheduling for home health. IIE Transactions on Healthcare Systems Engineering, 1(1):6–19, 2011.

X. Blasco, J. M. Herrero, J. Sanchis, and M. Martnez. A new graphical visualization of n-dimensional pareto front for decision-making in multiobjective optimization. Information Sciences, 178(20):3908–3924, 2008.

N. Christofides and S. Eilon. An algorithm for the vehicle-dispatching problem. OR, 20(3): 309–318, 1969.

Leandro C. Coelho, Jean-Franois Cordeau, and Gilbert Laporte. Consistency in multi-vehicle inventory-routing. Transportation Research Part C: Emerging Technologies, 24(0):270–287, 2012.

Carlos Coello Coello, Gary B Lamont, and David A Van Veldhuizen. Evolutionary algorithms for solving multi-objective problems. Springer, 2007. ISBN 0387367977.

K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. Evolutionary Computation, IEEE Transactions on, 18(4):577–601, 2014.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on, 6(2):182–197, 2002.

Judy Goldberg Dey, Margaret Johnson, MBA William Pajerowski, Myra Tanamor, and MPP Alyson Ward. Home health study report. See http://www.cms.gov/Medicare/Medicare-Fee-for-Service-Payment/HomeHealthPPS/downloads/HHPPS_LiteratureReview.pdf (last checked 18 October 2013), 2011.

Patrik Eveborn, Patrik Flisberg, and Mikael Rnnqvist. Laps carean operational system for staff planning of home care. European Journal of Operational Research, 171(3):962–976, 2006.

Dominique Feillet, Thierry Garaix, Fabien Lehud, Olivier Pton, and Dominique Quadri. A new consistent vehicle routing problem for the transportation of people with disabilities. Networks, 63(3):211–224, 2014.

Maoguo Gong, Licheng Jiao, Haifeng Du, and Liefeng Bo. Multiobjective immune algorithm with nondominated neighbor-based selection. Evolutionary Computation, 16 (2):225–255, 2008.

Chris Groër, Bruce Golden, and Edward Wasil. The consistent vehicle routing problem. Manufacturing & Service Operations Management, 11(4):630–643, 2009.

Joshua D. Knowles, Lothar Thiele, and Eckart Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. Report 214, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Switzerland, 2006.

Attila A. Kovacs, Bruce L. Golden, Richard F. Hartl, and Sophie N. Parragh. Vehicle routing problems in which consistency considerations are important: A survey. Networks, 64(3):192–213, 2014a.

Attila A. Kovacs, Sophie N. Parragh, Richard F. Hartl, and Sophie N. Parragh. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. Networks, 63(1):60–81, 2014b.

Attila A. Kovacs, Bruce L. Golden, Richard F. Hartl, and Sophie N. Parragh. The generalized consistent vehicle routing problem. Transportation Science, 49(4):796–816, 2015a.

Attila A. Kovacs, Sophie N. Parragh, and Richard F. Hartl. The multi-objective generalized consistent vehicle routing problem. European Journal of Operational Research, 247(2):441–458, 2015b.

Zhixing Luo, Hu Qin, ChanHou Che, and Andrew Lim. On service consistency in multi-period vehicle routing. European Journal of Operational Research, 243(3):731–744, 2015.

Thomas Macdonald, Karl Dorner, and Xavier Gandibleux. Metaheuristics for the consistent nurse scheduling and routing problem, 2009.

Ashlea Bennett Milburn and Jessica Spicer. Multi-objective home health nurse routing with remote monitoring devices. International Journal of Planning and Scheduling, 1(4): 242–263, 2013.

Stefan Nickel, Michael Schrder, and Jrg Steeg. Mid-term and short-term planning support for home health care services. European Journal of Operational Research, 219(3): 574–587, 2012.

Zhang Qingfu and Li Hui. Moea/d: A multiobjective evolutionary algorithm based on decomposition. Evolutionary Computation, IEEE Transactions on, 11(6):712–731, 2007.

Stefan Ropke and David Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40(4):455–472, 2006.

Paul Shaw. Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems, volume 1520 of Lecture Notes in Computer Science, chapter 30, pages 417–431. Springer Berlin Heidelberg, 1998.

Karen Smilowitz, Maciek Nowak, and Tingting Jiang. Workforce management in periodic delivery operations. Transportation Science, 47(2):214–230, 2013.

C. D. Tarantilis, F. Stavropoulou, and P. P. Repoussis. A template-based tabu search algorithm for the consistent vehicle routing problem. Expert Systems with Applications, 39(4):4233–4239, 2012.

Fabien Tricoire. Multi-directional local search. Computers & Operations Research, 39(12): 3089–3101, 2012.

L. While, L. Bradstreet, and L. Barone. A fast way of calculating exact hypervolumes. Evolutionary Computation, IEEE Transactions on, 16(1):86–95, 2012.

Christel A. Woodward, Julia Abelson, Sara Tedford, and Brian Hutchison. What is important to continuity in home care?: Perspectives of key stakeholders. Social Science and Medicine, 58(1):177–192, 2004.

Hongsheng Zhong, Randolph W. Hall, and Maged Dessouky. Territory planning and vehicle dispatching with driver learning. Transportation Science, 41(1):74–89, 2007.

E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. Evolutionary Computation, IEEE Transactions on, 7(2):117–132, 2003.

Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, 8(2):173–195, 2000.

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. CIMNE, Spain, 2002.

3. A Multi-objective Approach for the Consistent Periodic Vehicle Routing Problem

3.1 Introduction

In a number of industries in which customers require multiple visits throughout a planning period of multiple days, service consistency has been recognized as a key to improving customer satisfaction and customer loyalty (Russell et al., 2012; Woodward et al., 2004). Two types of service consistency, namely, driver consistency and time consistency, have been studied extensively in recent years in the context of vehicle routing problems (VRPs) spanning multiple periods. Driver consistency refers to using the fewest number of different drivers to perform all of the visits required by a customer over a planning horizon and time consistency refers to visiting a customer at roughly the same time on each day he/she needs service. The Consistent Vehicle Routing Problem (ConVRP), proposed in Groër et al. (2009), is a pioneering work in this research area. The ConVRP is inspired from routing problems occurring in the small package delivery industry where a set of vehicle routes must be designed to provide consistent services to customers requiring visits over a planning horizon of multiple days. Specifically, each customer needing services on more than one day (*frequent* customer) is required in ConVRP to be visited by the same driver each time it requires service. Time consistency is achieved by limiting the difference between the earliest and latest arrival time at each frequent customer not to exceed a certain value. The objective is to minimize the total distance traveled by all vehicles on all days across the planning horizon.

Another example service industry in which companies strive to provide consistent services is the home healthcare industry. In this industry, home-bound patients are scheduled to be visited on a regular basis by home healthcare workers, including home health aides, nurses, social workers or rehabilitation therapists. Minimizing the number of different caregivers assigned to a patient helps provide an opportunity to build up rapport between patients

and caregivers. With increased familiarity and reduced communication complexity, caregivers will be able to make more accurate observations. It has been shown that improved nurse consistency often results in better health outcomes, such as lower rates of hospitalization and fewer visits to the emergency department (Russell et al., 2012). With consistent service starting time, patients will be able to plan their daily routines more readily without too many disturbances (Woodward et al., 2004).

Kovacs et al. (2014a) provide a comprehensive review on the importance of service consistency in service industries. The existing studies on VRPs with service consistency considerations have assumed the days on which a customer requires service over a planning horizon is a problem input rather than a decision (Groër et al., 2009; Subramanyam and Gounaris, 2016; Kovacs et al., 2015a,b; Lian et al., 2016). The travel cost increase is then examined when service consistency is either enforced using hard constraints in a single-objective optimization approach, or treated as objectives within a multi-objective optimization framework. However, there exist various applications in which customers require multiple visits over a planning horizon, where the days those visits occur must be selected according to a set of allowable service patterns. A service pattern specifies the days on which the visits required by a customer are allowed to occur. The selection of a service pattern for each customer must be determined before vehicle routes can be optimized on each day. This multi-period VRP with service pattern selection has been named the Periodic Vehicle Routing Problem (PVRP) in the literature and various studies have been conducted to address its solution methods and novel applications. These will be reviewed in Section 3.2. To the best of our knowledge, no existing paper examines service consistency in the context of PVRP. We intend to fill this gap with this paper.

It has been shown in the ConVRP literature that the service consistency objective is often conflicting with the traditional travel cost objective due to variation in customer demands, limited vehicle availability and constraints on vehicle capacity and route duration (Lian et al., 2016). Minimizing travel cost with no consideration given to service consistency may

result in the assignment of multiple drivers to a particular customer and highly varying vehicle arrival times. However, optimizing service consistency may result in a routing plan with increased travel cost. In this paper, we use a multi-objective approach to study the relationship between the objectives of minimizing total travel cost and maximizing service consistency. There are three primary contributions in this paper. First, we study service consistency for the first time in the literature in the context of PVRP. The Pareto frontier is approximated using multi-objective algorithms to explicitly study the trade-offs between travel cost and service consistency objectives, which can facilitate managerial decision making. Second, various multi-objective algorithms are employed to solve the studied problem. The performance of these algorithms is validated on benchmark instances taken from the literature. Last, local search operators are designed to improve each of the three objectives.

The remainder of this paper is organized as follows. Section 3.2 reviews related literature on the PVRP and service consistency. Section 3.3 gives the mathematical model of the multi-objective PVRP with service consistency. Section 3.4 discusses the various multi-objective algorithms used in this paper to address the problem. The local search operator designed for each objective is also presented. Computational experiments and results are shown in Section 3.5. Section 3.6 concludes the paper.

3.2 Related literature

The PVRP is first discussed by Beltrami and Bodin (1974) in their seminal paper in which they examine the vehicle routing problem for municipal waste collection in New York City. The garbage is collected from large industrial sites which may require services either three or six times a week and there exist two feasible service day combinations for customers needing services three times a week. The service pattern must be selected first for each customer before vehicle routes on each day can be determined. The objective is to minimize both the number of vehicles used and the total travel cost. Two other earlier

papers related to PVRP appear in Russell and Igo (1979) and Christofides and Beasley (1984). Russell and Igo (1979) consider an assignment routing problem in which each customer has a predetermined number of days to be serviced in a week. The problem is to assign each customer to distinct days of the week in order to minimize the total distance traveled by all vehicles across the week. Christofides and Beasley (1984) give the first exact formulation of the PVRP.

The PVRP has been applied in various service industries. Matos and Oliveira (2004) develop an ant colony system to solve the PVRP modeled from a solid waste collection system involving 8087 containers in Viseu, Portugal. Nuortio et al. (2006) study the municipal solid waste collection problem in Eastern Finland and model it as a Stochastic Periodic Vehicle Routing Problem with Time Windows and a limited number of vehicles (SPVRPTW). A guided variable neighborhood thresholding metaheuristic is developed to solve the real-life waste collection problem. PVRP is also used to model the problem of collecting recyclable materials (Bommisetty et al., 1998), recycling paper containers (Baptista et al., 2002), waste vegetable oil (Aksen et al., 2012), infectious waste (Shih and Lin, 1999; Shih and Chang, 2001), animal waste (Coene et al., 2010), and others.

Banerjea-Brodeur et al. (1998) examine the daily linen delivery problem in the Jewish General Hospital in Montreal and model it as a PVRP. A tabu search algorithm is used to solve the problem. Jang et al. (2006) study a sales representatives routing problem for a lottery company in Missouri. Thirty-nine representatives are scheduled to visit 5043 ticket retailers periodically to check on product inventory, replenish supplies, collect returned tickets, clean point-of-sale counters and inspect equipments. Maya et al. (2012) consider an assignment routing problem in which disabled pupils are visited by assigned teaching assistants at their own schools. Pupils with various level of disability have different assistance frequency. A solution approach based on an auction algorithm and a variable neighborhood search heuristic is proposed.

Many solution approaches have been proposed since the pioneering work of Beltrami and

Bodin (1974). Christofides and Beasley (1984) propose an exact formulation for PVRP but solve it via a heuristic. Cordeau et al. (1997) propose a tabu search algorithm to solve the PVRP. Other recent algorithms proposed for PVRP include variable neighborhood search (Hemmelmayr et al., 2009), ant colony optimization (Matos and Oliveira, 2004) and a genetic algorithm (Vidal et al., 2012). Campbell and Wilson (2014) provide an extensive review on PVRP applications and solution methods.

Service consistency in vehicle routing is first formulated in the Consistent Vehicle Routing Problem (ConVRP) (Groër et al., 2009). The authors consider a routing problem inspired from the small package delivery industry where customers require one or more visits over a planning horizon of multiple days. Consistent services are enforced during the construction of vehicle routes using hard constraints and a record-to-record travel heuristic (ConRTR) is proposed to obtain near optimal solutions quickly. The concept of template routes is used in the heuristic to ensure a customer is visited by the same driver throughout the planning horizon and also to encourage better time consistency. Template routes consider only frequent customers that require service on more than one day. A route for day d can be derived from the template route by removing all those customers who do not require service on day d and inserting customers who require service on only day d. A tabu search heuristic (TTS) and adaptive large neighborhood search algorithm (TALNS) are developed in Tarantilis et al. (2012) and Kovacs et al. (2014b) to solve the ConVRP using the idea of template routes.

In the Generalized Consistent Vehicle Routing Problem (GenConVRP) (Kovacs et al., 2015a), strict driver consistency is relaxed by allowing multiple drivers to visit a customer and time consistency is incorporated into the objective function using a weighted sum approach. A large neighborhood search heuristic is proposed to solve this problem efficiently. Luo et al. (2015) investigate a multi-period VRP in which the maximum number of different drivers visiting a customer is limited and each customer is associated with a time window. A three-stage solution approach is proposed to solve this problem.

The impact of imposing different levels of service consistency on operational cost is studied. Braekers and Kovacs (2016) introduce driver consistency to the classical Dial-a-ride Problem (DARP) by limiting the maximum number of different drivers assigned to service a user across the planning horizon. The authors suggest two mathematical formulations and a branch-and-cut solution method. A large neighborhood search heuristic is also developed to obtain near-optimal solutions quickly. Computational results show that enforcing only one driver visiting a customer may cause up to 27.98% increase in travel cost, while the routing cost increase is no more than 5.80% when at least two drivers are allowed per customer.

Subramanyam and Gounaris (2016) study the Consistent Traveling Salesman Problem (ConTSP) over a multi-day planning horizon and consistent arrival times are imposed for frequent customers. The authors examine different mixed-integer linear programming formulations and develop a branch-and-cut framework with a new class of valid inequalities.

Research on the trade-offs between consistency objectives and other objectives appear in Milburn and Spicer (2013), Kovacs et al. (2015b), and Lian et al. (2016). Milburn and Spicer (2013) study a home healthcare nurse routing and scheduling problem with the objectives of minimizing total travel cost, maximizing nurse consistency and balancing nurse workload. A tabu search based multi-objective algorithm is developed to approximate the Pareto frontier. Kovacs et al. (2015b) employ two ϵ-constraint-based exact multi-objective solution approaches to analyze the trade-offs between the objective of travel cost minimization and the objectives of driver and time consistency maximization. A multi-directional large neighborhood search heuristic is also proposed to solve large instances. It is shown that 70% better time consistency can be achieved for the tested benchmark instances by at most 3.84% increase in total travel cost. Most recently, Lian et al. (2016) investigate a multi-objective version of the ConVRP in which driver consistency and time consistency are optimized as individual objectives together with the minimization of travel cost. An improved multi-directional local search algorithm is

proposed for general multi-objective optimization problems. It is shown that approximately 60% better driver consistency and 75% better time consistency can be achieved at the cost of 5% increase in travel cost.

The review of the literature indicates that there is currently a gap with respect to considering service consistency in the context of the periodic vehicle routing problem. We aim to fill this gap by studying the trade-offs between the traditional objective of minimizing total travel cost and the consistency objectives of maximizing driver consistency and time consistency. By analyzing the Pareto frontier obtained using various multi-objective algorithms, observations of the impact of improving service consistency on travel cost in the PVRP will be made to facilitate managerial decision making.

3.3 Problem description

The multi-objective consistent periodic vehicle routing problem (MoConPVRP) studied in this paper can be defined on a complete directed graph $\mathcal{G} = (\mathcal{N}^0, \mathcal{A})$, where $\mathcal{N}^0 = \mathcal{N} \cup \{0\}$ with $\mathcal{N} = \{1, \ldots, n\}$ representing the set of customers and $\{0\}$ indicating the depot, and $\mathcal{A} = \{(i,j)|i,j \in \mathcal{N}^0, i \neq j\}$. Associated with each arc $(i,j) \in \mathcal{A}$ is a travel time t_{ij} and triangle inequality is satisfied. A time horizon $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$ is considered and there are $|\mathcal{K}|$ homogeneous vehicles available at the depot. Each vehicle starts and ends its daily operations at the depot, and can only service a limited number of customers due to the restriction on its physical capacity Q and maximum route duration T. In this paper, the terms *driver* and *vehicle* are used interchangeably.

Each customer $i \in \mathcal{N}$ is associated with a service frequency m_i and has a predetermined set of allowable service patterns C_i. A service pattern $r \in C_i$ specifies the days on which the customer i is allowed to receive service. A constant w_{dr} is defined such that $w_{dr} = 1$ if and only if day d belongs to pattern r, and 0 otherwise. A non-negative demand q_i and service duration s_i are known in advance. The problem is to choose a single allowable service pattern for each customer and determine a set of vehicle routes for each day of the

planning horizon that are feasible with respect to capacity and route duration constraints. Each route must begin and end at the depot and each customer must be visited by exactly one vehicle on each day that he/she needs service. The objectives include minimizing total travel cost, minimizing the number of different drivers visiting a customer across the planning horizon and minimizing the maximum arrival time differential at a customer throughout the planning horizon.

The problem can be modeled using the following decision variables:

- x_{ijkd}: binary variable indicating whether arc (i, j) is traversed by vehicle k on day d, $(i, j) \in \mathcal{A}$, $k \in \mathcal{K}$ and $d \in \mathcal{D}$
- y_{ikd}: binary variable indicating whether customer i is visited by vehicle k on day d, $i \in \mathcal{N}^0$, $k \in \mathcal{K}$ and $d \in \mathcal{D}$
- z_{ir}: binary variable indicating whether service pattern r is chosen for customer i, $r \in C_i$ and $i \in \mathcal{N}$
- a_{id}: continuous variable describing the arrival time at customer i on day d, $i \in \mathcal{N}^0$ and $d \in \mathcal{D}$

The following auxiliary variables are defined to facilitate the modeling and linearization of consistency objectives:

- u_{ik}: binary variable indicating whether customer i is visited by vehicle k over the planning horizon, $i \in \mathcal{N}^0$ and $k \in \mathcal{K}$
- a_i^e: continuous variable describing the earliest arrival time at customer i over the planning horizon, $i \in \mathcal{N}^0$
- a_i^l: continuous variable describing the latest arrival time at customer i over the planning horizon., $i \in \mathcal{N}^0$

- u^{max}: continuous variable representing the maximum number of different drivers visiting a customer
- a^{max}: continuous variable representing the maximum arrival time differential at a customer

Combining aspects of the PVRP formulation in Cordeau et al. (1997) and the ConVRP formulation in Groër et al. (2009), we formulate the MoConPVRP as follows:

$$\min \quad \sum_{d\in\mathcal{D}}\sum_{k\in\mathcal{K}}\sum_{i\in\mathcal{N}^0}\sum_{j\in\mathcal{N}^0} t_{ij}x_{ijkd}, \tag{3.1}$$

$$\min \quad u^{max}, \tag{3.2}$$

$$\min \quad a^{max}, \tag{3.3}$$

$$\text{s.t.} \quad \sum_{r\in C_i} z_{ir} = 1,\ \forall i \in \mathcal{N} \tag{3.4}$$

$$\sum_{j\in\mathcal{N}^0}\sum_{k\in\mathcal{K}} x_{ijkd} - \sum_{r\in C_i} w_{dr}z_{ir} = 0,\ \forall i \in \mathcal{N},\ d \in \mathcal{D} \tag{3.5}$$

$$\sum_{k\in\mathcal{K}} y_{ikd} - \sum_{r\in C_i} w_{dr}z_{ir} = 0,\ \forall i \in \mathcal{N},\ d \in \mathcal{D} \tag{3.6}$$

$$\sum_{i\in\mathcal{N}^0} x_{ijkd} = \sum_{i\in\mathcal{N}^0} x_{jikd} = y_{jkd},\ \forall\ j \in \mathcal{N}^0,\ k \in \mathcal{K},\ d \in \mathcal{D} \tag{3.7}$$

$$\sum_{j\in\mathcal{N}} x_{0jkd} \leq 1,\ \forall k \in \mathcal{K},\ d \in \mathcal{D} \tag{3.8}$$

$$y_{0kd} = 1,\ \forall k \in \mathcal{K},\ d \in \mathcal{D} \tag{3.9}$$

$$\sum_{i\in\mathcal{N}} q_i y_{ikd} \leq Q,\ \forall k \in \mathcal{K},\ d \in \mathcal{D} \tag{3.10}$$

$$0 \leq a_{id} + \sum_{r\in C_i} w_{dr}z_{ir}(s_i + t_{i0}) \leq T\sum_{r\in C_i} w_{dr}z_{ir},\ \forall\ i \in \mathcal{N},\ d \in \mathcal{D} \tag{3.11}$$

$$a_{0d} = 0,\ \forall d \in \mathcal{D} \tag{3.12}$$

$$a_{id} + x_{ijkd}(s_i + t_{ij}) - T(1 - x_{ijkd}) \leq a_{jd},\ \forall\ d \in \mathcal{D},\ k \in \mathcal{K},\ i \in \mathcal{N}^0,\ j \in \mathcal{N} \tag{3.13}$$

$$a_{id} + x_{ijkd}(s_i + t_{ij}) + T(1 - x_{ijkd}) \geq a_{jd},\ \forall\ d \in \mathcal{D},\ k \in \mathcal{K},\ i \in \mathcal{N}^0,\ j \in \mathcal{N} \tag{3.14}$$

$$u_{ik} \geq y_{ikd},\ \forall i \in \mathcal{N},\ k \in \mathcal{K},\ d \in \mathcal{D} \tag{3.15}$$

$$\sum_{k \in \mathcal{K}} u_{ik} \leq u^{max},\ \forall i \in \mathcal{N} \tag{3.16}$$

$$a_i^l \geq a_{id} \geq a_i^e,\ \forall i \in \mathcal{N},\ d \in \mathcal{D} \tag{3.17}$$

$$a_i^l - a_i^e \leq a^{max},\ \forall i \in \mathcal{N} \tag{3.18}$$

$$x_{ijk}^d \in \{0,1\},\ \forall\ i \in \mathcal{N}^0,\ j \in \mathcal{N}^0,\ k \in \mathcal{K},\ d \in \mathcal{D} \tag{3.19}$$

$$y_{ikd} \in \{0,1\},\ \forall\ i \in \mathcal{N},\ k \in \mathcal{K},\ d \in \mathcal{D} \tag{3.20}$$

$$z_{ir} \in \{0,1\},\ \forall i \in \mathcal{N},\ r \in C_i \tag{3.21}$$

$$a_{id} \geq 0,\ \forall\ i \in \mathcal{N}^0,\ d \in \mathcal{D} \tag{3.22}$$

$$u_{ik} \geq 0, u^{max} \geq 0,\ \forall\ i \in \mathcal{N},\ k \in \mathcal{K} \tag{3.23}$$

$$a_i^e, a_i^l \geq 0, a^{max} \geq 0\ \forall\ i \in \mathcal{N} \tag{3.24}$$

Objective (3.1) minimizes the total travel distance of all vehicles on all days of the planning horizon. Objective (3.2) minimizes the maximum number of different drivers that visit any one customer. Objective (3.3) minimizes the maximum arrival time differential experience by any one customer. Constraint set (3.4) ensures that each customer is assigned one and only one allowable service pattern. Constraint set (3.5) ensures that the days on which a customer receives visits are those days appearing in the service pattern selected for the customer. Constraint set (3.6) ensures that each required visit to a customer is performed by exactly one vehicle. Constraint set (3.7) makes sure that each customer has only one predecessor and successor whenever it requires service. Constraint sets (3.8) and (3.9) ensure that each vehicle departs the depot at most once per day. The vehicle capacity and route duration constraints are given in constraint sets (3.10) and (3.11), respectively. Constraint sets (3.12), (3.13) and (3.14) aim to compute vehicle arrival times at customers and also serve to eliminate subtours. Constraint sets (3.15) and (3.16) compute the maximal number of different drivers visiting a customer. Constraint sets (3.17) and (3.18) compute the maximum arrival time differential over all customers. The

variable types are given in remaining constraint sets. Note that objectives (3.2)–(3.3), constraint sets (3.6) and (3.15)–(3.18), and variables (3.19)–(3.24) are newly introduced in this paper. The objective (3.1) and constraint sets (3.4), (3.5) and (3.7)–(3.14) are taken from Cordeau et al. (1997) and Groër et al. (2009).

3.4 Solution approach

This section briefly describes the multi-objective algorithms employed in this paper to approximate the Pareto frontier of the MoConPVRP. The solution generation method and local search operators for the three objectives will be detailed in following sections. The term *solution* used from this point forward refers to a routing plan consisting of a set of vehicle routes for each day of the planning horizon such that each customer is assigned an allowable service pattern, all customer demands are satisfied and vehicle capacity and maximum route duration constraints are respected.

3.4.1 Multi-objective optimization algorithms

Using exact solution approaches to obtain the Pareto frontier of MoConPVRP is time-consuming since these methods rely on iteratively solving single-objective optimization problems. Given that PVRP reduces to classical VRP when the planning horizon is set to one, PVRP is also NP-hard and therefore the time required to obtain the Pareto frontier of MoConPVRP is prohibitive. In this paper, we use seven multi-objective algorithms to approximate the Pareto frontier of MoConPVRP. The rationale is that different algorithms have different strengths in approximating the true Pareto frontier and a better approximation of the true Pareto frontier can be obtained by utilizing a set of multi-objective algorithms together. Also, it provides us an opportunity to evaluate the performance of various multi-objective algorithms for the MoConPVRP. The seven multi-objective algorithms include the Multi-directional Local Search (MDLS), the Improved Multi-directional Local Search (IMDLS), the Multi-objective Evolutionary

Algorithm based on Decomposition (MOEA/D), the Nondominated Neighbor Immune Algorithm (NNIA), the Strength Pareto Evolutionary Algorithm 2 (SPEA2), the Nondominated Sorting Genetic Algorithm II (NSGAII) and the Nondominated Sorting Genetic Algorithm III (NSGAIII) (Tricoire, 2012; Lian et al., 2016; Qingfu and Hui, 2007; Gong et al., 2008; Zitzler et al., 2002; Deb et al., 2002; Deb and Jain, 2014).

MDLS is an algorithmic solution framework for general multi-objective optimization problems (Tricoire, 2012). It is inspired by the concept of Pareto dominance. Starting from any solution x in the solution space, a neighboring solution x' is desirable if it is not dominated by x, which means that x' is either dominating x or non-comparable with x. The concept of Pareto dominance simply requires x' be better than x on at least one objective in order for x' to dominate x. Therefore, it is enough to apply local search operators to each objective individually to find such a neighbor solution x'. MDLS starts with an initial set of non-dominated solutions $\mathcal{F}$ and goes through three steps in each subsequent iteration: (1) selecting a solution, (2) applying local search on this solution for each objective to obtain a new solution corresponding to each objective and (3) updating the set of non-dominated solutions $\mathcal{F}$ using the newly generated solutions. Lian et al. (2016) introduce three new features to the original MDLS and propose the IMDLS. Specifically, in IMDLS, the size of $\mathcal{F}$ is bounded by a fixed number and all the non-dominated solutions in $\mathcal{F}$ are selected for local search in each iteration. If the size of $\mathcal{F}$ exceeds the input bound, a truncation procedure is applied on $\mathcal{F}$ using a crowding distance based selection rule.

MOEA/D, proposed by Qingfu and Hui (2007), utilizes a decomposition strategy to transform a multi-objective optimization problem into a number of scalar optimization problems and optimize them simultaneously. In MOEA/D, an initial population of solutions are first generated and each of them is assigned to a predefined weight vector. All the neighbors of a solution in the population are then identified based on their corresponding weight vectors. In each iteration, each solution x in the current population is

examined and a new solution y is obtained using two of x's neighboring solutions. The new solution y is then improved using local search operators to get y' which is used to update all the neighboring solutions of x based on their corresponding weight vectors. The authors propose three decomposition approaches, including the weighted sum approach, the Tchebycheff approach and the penalty-based boundary intersection approach. Each of the decomposition strategies is used to convert the various objectives to a scalar value and the corresponding MOEA/D variants are named MOEA/D-WS, MOEA/D-TCH and MOEA/D-PBI, respectively.

NNIA is a multi-objective optimization algorithm inspired from the immune system's ability to adapt its B-cells to new types of antigens (Gong et al., 2008). The term *antibody* is used to represent a solution. The algorithm maintains a set of non-dominated antibodies, namely, the dominant population D. In each iteration t, a fixed number of antibodies are selected from the dominant population D_t to create an active population A_t. Each active antibody is then cloned a number of times based on its crowding distance value. The resulting clone population C_t is subject to recombination and hypermutation, and C'_t denotes the new clone population. The C'_t is then combined with D_t to identify the new dominant population D_{t+1}.

SPEA2 is proposed by Zitzler et al. (2002) to improve the original SPEA. It maintains a solution population P and an archive population $\bar{P}$ throughout its search process. SPEA2 starts with generating an initial population P_0 and empty archive $\bar{P}_0$, and in each following iteration t, all non-dominated solutions are first identified from P_t and $\bar{P}_t$ and saved in $\bar{P}_{t+1}$. Binary tournament selection is then used to fill the mating pool based on $\bar{P}_{t+1}$. The new population in iteration $t+1$, P_{t+1}, is obtained by applying recombination and mutation operators to the mating pool. This process repeats until a maximum number of iterations are reached.

NSGAII is proposed by Deb et al. (2002) and starts with creating a random parent population P_0 of size N. An offspring population Q_0 of the same size is generated by

applying binary tournament selection, recombination and mutation operators on P_0. In each subsequent iteration t, P_t and Q_t are first combined to form R_t which is sorted into different dominance levels $(F_1, F_2, \dots)$. The new population P_{t+1} is then generated by selecting these levels one at a time until the size of P_{t+1} reaches N. If $|P_{t+1}|$ exceeds N by including a dominance level F_l, only those solutions in F_l with best crowding distance values are accepted. The new offspring population Q_{t+1} is generated using crowded-comparison based binary tournament selection, recombination and mutation on solutions from P_{t+1}. NSGAIII (Deb and Jain, 2014) follows the same framework with NSGAII. The difference is how solutions from dominance level F_l are selected when creating P_{t+1}. Instead of using a crowding distance based selection operator as in NSGAII, NSGAIII employs a more complex rule to identify the solutions in F_l to enter P_{t+1}. Let $S_t = \sum_{i=1}^{l} F_i$, where the objective values of solutions in S_t are first normalized. A reference set Z^r is created and each solution $s \in S$ is associated with a reference point in Z^r. A niche count ρ_j is computed for each reference point $j \in Z^r$. Finally, solutions to enter P_{t+1} are determined based on ρ.

3.4.2 Starting solution generation

All of the multi-objective algorithms used in this paper require a starting set of randomly generated solutions for the MoConPVRP. The solution generation process starts by assigning an allowable service pattern to each customer sequentially and the order in which customers are considered for assignment is randomized. For a customer being considered for service pattern assignment, the service pattern that will balance the total demand on each day is chosen. To this end, for a service pattern, the average daily total demand across the planning horizon is first noted if this service pattern is chosen for this customer. Then the total daily demand difference from this average demand is computed. The service pattern that results in the smallest demand difference will be chosen for this customer. After the service pattern for each customer is determined, the resulting vehicle routing

problem is solved on each day using a cluster-first route-second approach. On each day of the planning horizon, the Sweep heuristic (Gillett and Leland, 1974) for VRP is used to order the customers with respect to the angles they make with the horizontal line and the depot. Then, the Next Fit bin packing heuristic (Coffman et al., 1984) is used with this ordering to assign customers to vehicles. A new vehicle will be used if the vehicle capacity constraint is violated. All of the remaining customers will be assigned to the last vehicle if no more vehicles are available. In this way, all but the last vehicle will satisfy the vehicle capacity constraint. The actual route operated by a vehicle is determined using Farthest Insertion (Rosenkrantz et al., 1974). To initialize the heuristic, the convex hull for all customers assigned to this vehicle and the depot serves as the initial route. Note that the resulting vehicle route may violate the maximum route duration constraint.
With any feasible solution, its total travel cost can be computed by summing up the distance traveled by all vehicles across the planning horizon. The driver consistency objective value can be determined by identifying the maximum number of different drivers visiting a customer across the planning horizon. The time consistency value can be computed by determining the maximum arrival time differential at a customer across the planning horizon. Objective values of infeasible solutions will be penalized using formula (3.25) in Section 3.4.3.

3.4.3 Local search for the travel cost objective

This section describes a tabu search heuristic used in this paper to minimize the total travel cost. Algorithm 3 shows the framework of the heuristic. Infeasible solutions are allowed during the search process and the vehicle capacity and route duration constraint violations are penalized using coefficients γ and δ, respectively. Specifically, the penalized travel distance of a vehicle route k on day d is computed as

$$\bar{f}_{TD}(k,d) = f_{TD}(k,d) + \gamma * \max(Q_{k,d} - Q, 0) + \delta * \max(T_{k,d} - T, 0), \qquad (3.25)$$

where $f_{TD}(k,d)$ is the travel distance of the route k on day d, and $Q_{k,d}$ and $T_{k,d}$ are the total load and travel time of vehicle k on day d, respectively. The penalty coefficient γ is applied to any vehicle whose total load exceeds its capacity ($Q_{k,d} > Q$). Similarly, the penalty coefficient δ is applied to any vehicle whose total travel time exceeds its limit ($T_{k,d} > T$).

Algorithm 3 Tabu search heuristic for minimizing total travel cost

Input: initial solution π_0
initialize coefficients γ, δ and λ
initialize tabu list and frequency list and let iteration $\rho = 0$
let current solution $\pi_\rho = \pi_0$ and let π^* denote the best solution, $\pi^* = \pi_\rho$
apply local search operator on vehicle routes on each day separately
repeat
    construct a list of candidate neighboring solutions of π_ρ
    choose the best admissible solution π' from the list
    let new current solution $\pi_\rho = \pi'$
    update π^* if $\bar{f}_{TD}(\pi_\rho) < \bar{f}_{TD}(\pi^*)$
    update tabu list and frequency list
    update penalty coefficients
    apply local search operator on daily vehicle routes every ξ iterations
    let $\rho = \rho + 1$
until stopping criteria is met

The proposed heuristic starts with a randomly generated solution π_0 and it is evaluated using equation (3.25) to get its total travel cost $\bar{f}_{TD}(\pi_0)$. In each subsequent iteration ρ, a list of candidate solutions can be obtained from the current solution π_ρ by selecting a single customer in π_ρ and changing its current service pattern to another randomly selected allowable service pattern. Thus the total number of candidate solutions generated in each iteration equals the total number of customers having more than one allowable service pattern. To change the service pattern of a customer, its old and new service patterns are compared on each day d sequentially: if the customer requires service on day d only in the old service pattern, the visit to this customer is removed from the solution; if the customer requires service on day d only in the new service pattern, the visit to this customer is inserted to the solution with least penalized travel distance increase; no change is made for

the days on which the customer requires visits in both the old and new service patterns. A tabu list is initialized such that each allowable service pattern of a customer is assigned a tabu status which is an integer value initialized as 0 at the beginning of the algorithm. In iteration ρ, to decide whether a candidate solution π' should be accepted as the new current solution $\pi_{\rho+1}$, the only customer c with changed service pattern in π' compared to π_ρ is identified and the tabu status of the service pattern for c in π' is checked. If the tabu status value is smaller than the current iteration number, the candidate solution π' is accepted as the new current solution $\pi_{\rho+1}$; otherwise, the π' is still accepted as the $\pi_{\rho+1}$ if the aspiration criterion is satisfied, that is, the π' is better than the best solution π^* encountered since the beginning of the algorithm. If π' is accepted as $\pi_{\rho+1}$, the tabu status of the service pattern of c used in π_ρ is updated as the summation of the current iteration number and the tabu length ζ. In this way, the old service pattern is forbidden to be selected for customer c for the next ζ number of iterations unless aspiration criteria is met.
A frequency list is also created to record the number of times an allowable service pattern is chosen for a customer throughout the search process. This is to decrease the likelihood that a candidate solution with worse objective value than the incumbent will be accepted if the service pattern in the candidate solution has been frequently used. Every time a candidate solution π' is accepted as the new current solution $\pi_{\rho+1}$, the frequency $\mu \in \mathbb{N}$ of the new service pattern r of the aforementioned customer c in π' is increased by 1. This service pattern selection frequency is used to penalize a candidate solution π' whose penalized objective value is bigger than that of the current solution π_ρ (Cordeau et al., 1997). In other words, if $\bar{f}_{TD}(\pi') > \bar{f}_{TD}(\pi_\rho)$, the frequency-based penalized objective value is $g(\pi') = \bar{f}_{TD}(\pi') + \lambda \bar{f}_{TD}(\pi')\sqrt{n}\mu$, where λ is the penalty coefficient and n the total number of customers; otherwise, $g(\pi') = \bar{f}_{TD}(\pi')$ if $\bar{f}_{TD}(\pi') \leq \bar{f}_{TD}(\pi_\rho)$. In each iteration, all the candidate solutions are sorted in non-decreasing order of $g(\pi')$ and they are then checked sequentially to decide whether to become the new current solution $\pi_{\rho+1}$.
The penalty coefficients γ and δ are updated at each iteration based on the feasibility of

the current solution. Specifically, after a new current solution π_ρ is obtained, its feasibility is checked regarding vehicle capacity and maximum route duration constraints. If π_ρ is feasible, the penalty coefficients γ and δ are updated by multiplying by factor σ; otherwise, they are updated by adding a factor τ. The parameter σ is a relatively small number in order to decrease the penalty applied on infeasible routes, while the τ represents a small increment to the penalty coefficients.

In Algorithm 3, a local search operator is applied on daily vehicle routes periodically. It is a tabu search heuristic that aims to optimize the vehicle routes on any given day. Algorithm 4 shows the framework of the tabu search heuristic. In this heuristic, a solution refers to the set of vehicle routes on day d. In each iteration ρ, a candidate solution π' is created from the current solution π_ρ by removing a customer c from its current vehicle and inserting it into another vehicle. The total number of candidate solutions generated in each iteration equals the product of the total number of customers requiring service on day d and the total number of vehicles minus 1.

A tabu status value is assigned to each customer-vehicle combination such that a customer c is forbidden to be inserted into a vehicle k for ζ_d iterations if c is removed from vehicle k. All tabu status values are initialized to 0 at the beginning of the algorithm. The same candidate acceptance rule and aspiration criterion as in Algorithm 3 are used here. Similarly, a frequency list is created to record the number of times a customer c is inserted into a vehicle k throughout the search process. The same frequency update rule as in Algorithm 3 is used here. Also, candidate solutions with worse penalized objective values are further penalized using the frequency-based approach explained previously in this section. Note that in Algorithm 4, the penalty coefficients γ_d and δ_d are set as constant values.

Algorithm 4 Tabu search heuristic for minimizing total travel cost on day d

Input: initial solution π_0^d
initialize coefficients γ_d, δ_d and λ_d
initialize tabu list and frequency list and let iteration $\rho = 0$
let current solution $\pi_\rho^d = \pi_0^d$ and let π_d^* denote the best solution, $\pi_d^* = \pi_\rho^d$
repeat
    construct a list of candidate neighboring solutions of π_ρ^d
    choose the best admissible solution π_d' from the list
    let new current solution $\pi_\rho^d = \pi_d'$
    update π_d^* if $\bar{f}_{TD}(\pi_\rho^d) < \bar{f}_{TD}(\pi_d^*)$
    update tabu list and frequency list
    let $\rho = \rho + 1$
until stopping criteria is met

3.4.4 Local search for the consistency objectives

For the time consistency objective, the same heuristic proposed in Lian et al. (2016) to achieve time consistency in the context of the MoConVRP is used. The heuristic is not adapted to explicitly address the service pattern decisions present in the MoConPVRP. This is due to the difficulty of identifying an alternative allowable service pattern that can improve the time consistency objective. The solution techniques tailored to addressing this challenge are saved for future work.

The local search operators used to optimize driver consistency do explicitly treat service pattern selection decisions. These operators are described in this section. This heuristic for improving driver consistency will first randomly choose a customer with the worst driver consistency objective value and all the required visits to this customer will be removed from the solution. If there is only one service pattern available for this customer, all the vehicle routes on all the days on which it requires service will be checked to determine whether this customer can be feasibly inserted without violating vehicle capacity and route duration constraints. Then all available vehicles will be checked sequentially to compute the maximum number of days they can service this customer and the total travel cost increase that would result. The vehicle v^* that can visit this customer most frequently with least travel cost increase will be selected to visit this customer whenever it is feasible to do

so. For the days on which the customer visit requests are not satisfied, all of the vehicles are checked again using a similar procedure and the best vehicle is selected. This process repeats until all of the required visits to this customer are satisfied.

If there are multiple service patterns available for this customer, all of the vehicle routes on all days across the planning horizon will be checked to determine whether this customer can be feasibly inserted without violating the vehicle capacity and route duration constraints. For each available service pattern, the algorithm determines the minimum number of different drivers that can satisfy all of the service requirements of this customer and the total travel cost increase associated with this service pattern. This determination process follows the same steps as in the previous one service pattern scenario. Then the service pattern with the best driver consistency objective value for this customer will be selected. The above removal and reinsertion process will be repeated until no improvement in the driver consistency objective can be made.

3.5 Computational experiments and analysis

In this section, the benchmark instances used in this paper to evaluate the performance of various multi-objective algorithms for solving MoConPVRP are first introduced. Next, the three metrics used to compare the performance of the various algorithms are described. Finally, the results of the computational study are presented and a trade-off analysis between the objectives of cost minimization and consistency maximization is provided.

3.5.1 Benchmark instances and experiment design

A total of 26 instances are taken from the literature (Cordeau et al., 1997) to validate the performance of the various multi-objective algorithms and to study the trade-offs between the objectives of cost minimization and consistency maximization. There exist some instances in the literature in which each customer requires service only once and they are excluded from the computational study in this paper. Table 3.1 summarizes the

Table 3.1: Summary of PVRP instances

Name	$\|\mathcal{N}\|$	$\|\mathcal{D}\|$	$\|\mathcal{K}\|$	Name	$\|\mathcal{N}\|$	$\|\mathcal{D}\|$	$\|\mathcal{K}\|$
p02	50	5	3	p20	184	4	4
p05	75	5	6	p21	60	4	4
p08	100	5	5	p22	114	4	6
p10	100	5	4	p23	168	4	6
p11	139	5	4	p24	51	6	3
p12	163	5	3	p25	51	6	3
p13	417	7	9	p26	51	6	3
p14	20	4	2	p27	102	6	6
p15	38	4	2	p28	102	6	6
p16	56	4	2	p29	102	6	6
p17	40	4	4	p30	153	6	9
p18	76	4	4	p31	153	6	9
p19	112	4	4	p32	153	6	9

characteristics of these instances. The total number of customers, the total number of days in the planning horizon and the number of vehicles available are given for each instance.

All of the multi-objective algorithms considered in this paper are implemented following their descriptions in the literature. The same local search operators described in the previous sections are used whenever possible. For algorithms other than MDLS and IMDLS, a recombination operator is required to generate offspring solution from two parent solutions. The crossover operator works as follows: an empty offspring solution π_o is first created and a random recombination point p_{cx} is selected from $U(1, n)$. Customers with identity less than p_{cx} will use the same service pattern of corresponding customers in parent solution π_m, and other customers will use the same service pattern of their counterparts in parent solution π_f. The daily vehicle routes of the offspring solution are determined using the same procedure given in Section 3.4.2.

The parameter values of the multi-objective algorithms are set the same as those in Lian et al. (2016). For MDLS and IMDLS, the number of initial solutions is set to 100 and the F_{max} in IMDLS is set to 100. For MOEA/D, we use the weight vectors generated by Qingfu and Hui (2007) for three-objective optimization problems. The population size is determined by the number of weight vectors and is therefore set to 351. For NNIA, the

sizes of the dominant population, active population, and clone population are set to 100, 20, and 100, respectively. For SPEA2, the population size and archive size are both set to 100. For NSAGII, the population size is set to 100. For NSGAIII, the population size is determined by the size of the user-defined reference point set. In this paper, we use the reference point set created by Deb and Jain (2014) for three-objective optimization problems. It contains 91 reference points and the NSGAIII population size is set to 91. The parameter values of local search operators are set based on experience. For the tabu search heuristic, both γ and δ are set to 500.0 at the beginning of the algorithm. The penalty updating factors σ and τ are set to 0.10 and 1, respectively. The frequency-based penalty coefficient λ is set to 0.015 and the local search on daily routes is applied every $\xi = 50$ iterations. In addition, the γ_d and δ_d values are set to corresponding γ and δ values when the local search operator on daily routes is invoked. The parameter λ_d is set to 0.015. All algorithms are implemented in C++ and five replications are solved for each instances with a time limit of one hour.

3.5.2 Algorithm performance comparison

Hypervolume (I_H), coverage (I_C) and unary multiplicative epsilon indicator (I_ϵ) metrics are used in this paper to compare the performance of the various multi-objective algorithms for MoConPVRP. Their definitions are detailed in Lian et al. (2016). To perform the comparisons, the superset is constructed for each instance-algorithm pair by taking the union of the non-dominated solution sets from all five replications.
Table 3.2 shows the hypervolume comparison results. The instance number is given in the first column. Each of the next nine columns represents one of the comparison multi-objective algorithms implemented in this paper. Each entry in the table gives the hypervolume value computed based on the superset for the instance number and algorithm indicated by the row and column labels. The last row provides the average hypervolume value across all 26 instances for each of the nine algorithms.

Table 3.2: Hypervolume comparison

Instance	MDLS	IMDLS	MOEA/D			NNIA	SPEA2	NSGAII	NSGAIII
			WS	TCH	PBI				
2	0.8575	0.9035	1.0114	1.0317	1.0388	1.0536	1.0071	1.1466	1.0721
5	0.9500	1.0352	0.9073	0.8984	0.9753	1.0372	1.0064	0.9963	1.0133
8	0.8626	0.9505	0.9254	0.9387	0.9428	1.0132	0.9546	1.0681	1.1569
10	0.9463	0.9489	0.9644	0.9863	0.9564	1.1412	1.0762	1.1051	1.0955
11	0.9745	0.9489	0.8705	0.8439	0.8702	1.0424	1.0340	0.9580	0.9925
12	1.0871	1.1178	0.9856	0.9558	0.9779	1.1516	1.1008	1.1452	1.0829
13	0.7321	0.4001	0.6530	0.7069	0.6501	0.7700	0.6307	0.6456	0.5525
14	1.2066	1.2063	1.2187	1.2190	1.2194	1.2209	1.2194	1.2199	1.2194
15	1.2162	1.2160	1.2109	1.2110	1.2064	1.2177	1.2125	1.2132	1.2132
16	1.2279	1.2220	1.1942	1.2049	1.1847	1.2158	1.1991	1.2136	1.1991
17	1.1917	1.1458	1.1587	1.0719	1.0405	1.1962	1.2035	1.2195	1.2071
18	1.1340	1.1209	1.0579	1.0416	1.1220	1.1120	1.0672	1.1182	1.1404
19	1.1379	1.2074	1.1318	1.1221	1.1716	1.1508	1.1519	1.1958	1.2190
20	1.0217	1.1929	0.9568	1.0922	0.9088	1.1094	1.0553	1.1367	1.0733
21	0.9198	0.9409	0.7840	0.8240	0.7633	0.8178	0.8582	0.8959	0.8969
22	1.0009	1.0571	1.0215	0.9953	0.9895	1.1168	1.1087	1.1590	1.1479
23	0.7972	0.7012	0.9336	0.9659	0.9055	1.0585	0.9694	1.1182	1.0452
24	0.9312	0.9471	0.9546	0.9371	0.9199	1.1009	1.0137	1.0222	1.0983
25	0.8988	0.8846	0.9363	0.9162	0.9765	0.9866	0.9892	1.0064	0.9544
26	0.7973	0.8487	0.8887	0.8390	0.9094	0.9288	1.0425	0.9528	1.0095
27	0.9306	0.9080	0.9434	0.8603	0.8657	0.9443	0.9707	0.9382	0.9177
28	0.9564	0.9541	0.8876	0.8859	0.9189	1.0027	1.0008	0.9846	0.9962
29	0.9964	0.9325	0.8995	0.8928	0.9411	0.9838	1.0071	0.9911	1.0001
30	0.7227	0.6514	0.7936	0.7472	0.7232	0.8387	0.8264	0.8284	0.9142
31	0.7511	0.7893	0.8067	0.7676	0.7504	0.8245	0.8413	0.8244	0.8229
32	0.8002	0.7553	0.7307	0.7928	0.7328	0.7998	0.8472	0.8842	0.7694
Average	0.9634	0.9610	0.9549	0.9519	0.9485	1.0321	1.0152	**1.0380**	1.0312

It can be seen from Table 3.2 that according to the hypervolume metric, NSGAII performs the best among all comparison algorithms. NSGAIII achieves the second best average hypervolume value across all instances. MDLS and IMDLS perform similarly, and all three versions of MOEA/D perform the worst.

Table 3.3 shows the coverage comparison of all considered multi-objective algorithms. Each entry in the table indicates the percentage of solutions produced by the algorithm corresponding to the column label that are dominated by at least one solution generated by the algorithm corresponding to the row label. The second entry in the first row indicates that 45.13% of the nondominated solutions generated by IMDLS are dominated by at least

Table 3.3: Coverage comparison

	MDLS	IMDLS	MOEA/D			NNIA	SPEA2	NSGAII	NSGAIII	Average
			WS	TCH	PBI					
MDLS	-	45.13%	52.99%	55.14%	53.12%	33.59%	37.17%	33.84%	33.93%	43.11%
IMDLS	37.87%	-	58.00%	59.75%	57.89%	33.67%	39.58%	38.78%	39.25%	45.60%
MOEA/D-WT	25.16%	28.40%	-	45.69%	47.17%	9.98%	12.55%	11.89%	14.15%	24.37%
MOEA/D-TCH	24.95%	26.68%	37.88%	-	43.29%	9.34%	12.22%	13.11%	9.33%	22.10%
MOEA/D-PBI	23.54%	27.80%	35.95%	39.18%	-	9.72%	12.75%	12.64%	13.98%	21.94%
NNIA	46.20%	54.03%	75.26%	79.90%	72.61%	-	51.66%	43.35%	49.36%	**59.04%**
SPEA	40.94%	48.81%	71.23%	77.85%	70.55%	35.14%	-	32.81%	36.66%	51.75%
NSGAII	48.09%	52.28%	70.28%	76.72%	68.93%	39.37%	45.06%	-	41.54%	55.29%
NSGAIII	44.43%	48.20%	70.42%	77.88%	72.16%	39.69%	42.01%	39.73%	-	54.32%

one solution in the nondominated solution set produced by MDLS. On the other hand, (as indicated by the first entry in the second row) 37.87% of the solutions obtained by MDLS are dominated by at least one of the solutions in the set generated by IMDLS. It can be seen from the table that NNIA and NSGAII perform the best and second best among all of the comparison algorithms according to the coverage metric. MDLS and IMDLS perform similarly with IMDLS working slightly better. MOEA/D generally performs the worst among all algorithms.

Table 3.4 shows the unary multiplicative epsilon indicator comparisons of all multi-objective algorithms implemented in this paper. The first column gives the instance number and the next nine columns correspond to one of the comparison algorithms. Each entry in the table gives the I_ϵ value computed based on the superset for the instance number and algorithm indicated by the row and column labels. The last row gives the average I_ϵ value across all 26 instances for each of the comparison algorithms. It can be seen from the table that MDLS outperforms all other multi-objective algorithms on average. IMDLS, NSGAII and NSGAIII have similar performance with respect to this metric.

It can be seen from the above comparisons that NNIA and NSGAII perform similarly among all comparison algorithms with respect to the hypervolume and coverage metrics. On the other hand, they are outperformed by MDLS with respect to the multiplicative epsilon indicator. In the next section, we conduct our trade-off analysis based on the super

Table 3.4: Unary multiplicative epsilon comparison

Instance	MDLS	IMDLS	MOEA/D			NNIA	SPEA2	NSGAII	NSGAIII
			WS	TCH	PBI				
2	22.32	21.00	21.00	14.54	14.89	15.89	21.00	1.33	14.87
5	6.00	2.31	66.48	96.99	103.82	19.28	16.73	13.69	18.43
8	52.75	43.40	85.10	88.09	83.40	32.33	40.25	17.87	5.92
10	19.85	24.79	138.23	126.16	139.00	8.16	5.00	5.63	4.49
11	16.51	24.86	57.35	63.98	61.28	5.33	11.90	22.51	18.65
12	8.92	3.77	111.11	133.78	115.25	7.73	10.91	9.17	10.55
13	58.17	11.19	301.74	385.18	331.44	165.63	180.71	190.42	166.02
14	4.05	4.05	2.51	2.51	2.51	2.51	1.61	2.51	1.61
15	5.37	5.37	5.37	5.37	8.29	1.54	5.37	5.37	5.37
16	1.00	4.52	29.13	32.05	33.75	13.72	22.60	11.00	15.07
17	2.90	11.00	57.39	45.54	50.64	4.48	2.90	1.63	2.90
18	5.58	4.75	55.99	56.42	39.87	25.37	43.38	14.05	12.62
19	51.69	33.98	55.58	89.81	56.21	32.51	34.16	10.26	13.80
20	11.00	4.58	134.02	135.97	324.12	135.66	155.59	30.54	129.52
21	1.91	30.36	68.83	64.57	72.54	28.50	29.42	28.36	28.50
22	30.69	29.09	242.58	210.53	213.93	26.52	28.39	11.10	3.18
23	74.29	62.59	520.68	473.73	496.78	188.83	71.12	140.49	160.22
24	50.28	42.74	190.49	167.37	190.49	1.91	18.33	24.21	5.88
25	19.68	4.93	5.23	5.23	46.77	3.29	4.92	4.92	4.42
26	20.82	7.26	1.98	4.30	5.51	3.29	3.74	3.29	3.29
27	89.61	158.47	846.10	785.30	700.40	40.60	47.60	93.10	158.92
28	18.35	14.09	569.60	677.40	383.20	46.10	42.00	56.80	48.30
29	42.70	73.51	791.20	1243.40	989.50	88.74	73.51	82.73	69.25
30	504.20	959.13	5182.80	4678.00	4809.10	623.32	457.10	30.70	134.44
31	217.46	147.20	3349.60	3930.31	3630.10	322.30	313.20	219.50	196.20
32	177.00	11.00	5127.19	4053.29	6188.19	610.00	458.20	498.20	443.50
average	**58.20**	66.92	692.97	675.76	734.27	94.37	80.76	58.82	64.46

set obtained by taking the union of all the nondominated solutions generated by all algorithms in all replications.

3.5.3 Trade-off analysis

In this section, we conduct the trade-off analysis using the super set $\mathcal{R}$ of all non-dominated solutions obtained by all multi-objective algorithms in all replications. There usually exist many nondominated solutions in $\mathcal{R}$ and it is desirable to identify the best compromise solution to facilitate managerial decision making. Specifically, the level diagram technique proposed by Blasco et al. (2008) is employed here to identify four special solutions in the super set of each instance.

Let s^{BC} denote the best compromise solution in the super set $\mathcal{R}$. Furthermore, let s^{TD},

Table 3.5: Four solutions

Inst.	s^{BC}			s^{TD}			s^{DC}			s^{TC}		
	f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}	f_{TD}	f_{DC}	f_{TC}
2	1493.69	1	47.0687	1322.87	3	81.1521	1493.69	1	47.0687	1623.52	1	28.3847
5	2045.28	3	49.2837	2039.84	4	83.0219	2653.06	1	72.32	2435.15	3	48.9537
8	2188.37	3	69.5397	2042.1	4	102.664	2410.21	1	67.4573	2523.02	1	40.5877
10	1948.12	1	62.001	1610.73	3	89.1024	1948.12	1	62.001	2068.86	2	44.0197
11	952.9	1	29.1384	781.15	3	51.3433	952.9	1	29.1384	1234.75	2	11.7552
12	1285.21	1	14.9957	1204.31	3	56.887	1285.21	1	14.9957	1641.23	2	4.0544
13	4304.54	1	46.9806	3646.68	2	48.7686	4304.54	1	46.9806	4587.07	2	29.124
14	971.323	1	2.9154	954.807	1	5.3523	971.323	1	2.9154	1517.37	1	2.3054
15	1862.63	1	15.9017	1862.63	1	15.9017	1862.63	1	15.9017	1879.15	1	12.9863
16	2875.24	1	23.4297	2875.24	1	23.4297	2875.24	1	23.4297	2875.24	1	23.4297
17	1637.53	3	6.3969	1597.75	2	94.3632	1714.91	1	7.5118	1637.53	3	6.3969
18	3248.1	2	181.055	3131.09	3	226.375	3576.36	1	182.666	4398.1	2	115.811
19	4893.14	1	282.699	4834.34	3	386.351	4893.14	1	282.699	5739.09	2	244.103
20	8376.21	1	521.491	8367.4	3	524.989	8376.21	1	521.491	12152.4	2	360.734
21	2720.53	1	74.4719	2172.29	4	133.351	2720.53	1	74.4719	3442.78	3	46.1994
22	4353.74	2	188.086	4195.74	4	181.828	4943.28	1	181.316	5380.42	2	162.707
23	6739.68	2	311.363	6441.23	3	307.049	8594.84	1	279.107	8807.6	2	223.713
24	3865.41	2	117.758	3687.46	2	175.492	4285.86	1	122.157	4633.7	1	61.5742
25	4138.17	2	84.1048	3780.56	3	174.349	4566.07	1	125.335	4342.77	3	55.4473
26	4224.89	2	61.1	3795.32	2	122.641	4178.84	1	117.791	3960.5	3	55.4483
27	22307.8	4	805.372	21965.9	5	1050.67	29579.6	1	824.496	30357.9	2	613.267
28	22480	4	608.573	22329.7	6	992.933	29878.5	1	888.111	22453.6	5	593.097
29	22742.8	4	608.277	22605.4	6	985.627	29491.2	1	1005.43	33118.3	2	520.531
30	79778.2	5	2131.04	74997.2	5	2921.05	105167	1	2463.56	116977	2	1810
31	103157	2	2444.25	77230.5	6	3345.39	103248	1	3148.27	78362.4	6	1812.38
32	103375	2	2197.44	78540.3	6	2903.9	109361	1	3397.3	82183.7	5	1794.45

s^{DC} and s^{TC} denote the solutions with best travel distance, driver consistency, and time consistency value, respectively. Table 3.5 shows these four solutions for each of the 26 instances. Take the best compromise solution s^{BC} and the best travel distance solution s^{TD} for instance 2 in Table 3.5 as an example. The travel distance of s^{TD} for this instance is 1322.87, with driver and time consistency values being 3 and 81.1521, respectively. The travel distance of the best compromise solution s^{BC} is 1493.69, approximately 12.91% higher than the travel distance in the best travel distance solution. However, the increase in travel distance results in better consistency values, with driver and time consistencies of 1 (a 66.67% decrease) and 47.0687 (a 41.99% decrease), respectively.

Table 3.6 summarizes the trade-offs between the objectives of travel cost minimization and service consistency maximization. Specifically, three pairs of solutions are compared: the best travel distance solution compared with the best compromise solution, the best travel

	f_{TD}	f_{DC}	f_{TC}
s^{TD} vs. s^{BC}	8.88% ↗	34.42% ↘	30.96% ↘
s^{TD} vs. s^{DC}	18.81% ↗	61.28% ↘	22.12% ↘
s^{TD} vs. s^{TC}	26.79% ↗	24.81 % ↘	48.26% ↘

Table 3.6: Comparisons of four solutions

distance compared with the best driver consistency solution, and the best travel distance compared with the best time consistency solution. The first row indicates that the best compromise solution increases travel distance by 8.88% on average and decreases driver and time consistency by 34.42% and 30.96% on average, respectively, when compared with the solution obtained by optimizing travel distance alone. That is, an approximate 34% improvement in driver consistency and 30% improvement in time consistency comes at a cost of a 9% increase in travel distance. The second row shows that, if only driver consistency is considered, the best compromise solution, on average, increases travel distance by 18.81%, decreases the driver consistency by 61.28%, and time consistency by 22.12%, when compared with the solution with best travel distance.

3.6 Conclusion

This paper studies service consistency in the context of periodic vehicle routing problems using a multi-objective optimization approach and defines, for the first time in the literature, the MoConPVRP. Two service consistency objectives (maximization of driver consistency and maximization of time consistency) are considered separately with the traditional objective of minimizing total travel distance. Seven multi-objective optimization algorithms are employed to solve the studied problem and their performance is validated on a total of 26 benchmark instances taken from the literature. Trade-off analysis on the nondominated solutions obtained by all algorithms suggest that pursuing the best compromise solution among all three objectives may increase travel cost by about 8% while improving driver and time consistency by approximately 34% and 30% on average, when compared with a compromise solution having lowest overall travel distance.

Directions for future work can include studying service consistency in PVRP using a single objective approach where travel distance is minimized while service consistency is enforced via hard constraints.

Reference

Deniz Aksen, Onur Kaya, F. Sibel Salman, and Yeliz Aka. Selective and periodic inventory routing problem for waste vegetable oil collection. Optimization Letters, 6(6):1063–1080, 2012.

M. Banerjea-Brodeur, J. F. Cordeau, G. Laporte, and A. Lasry. Scheduling linen deliveries in a large hospital. The Journal of the Operational Research Society, 49(8):777–780, 1998.

Susana Baptista, Rui Carvalho Oliveira, and Eduardo Zquete. A period vehicle routing case study. European Journal of Operational Research, 139(2):220–229, 2002.

E. J. Beltrami and L. D. Bodin. Networks and vehicle routing for municipal waste collection. Networks, 4(1):65–94, 1974.

X. Blasco, J. M. Herrero, J. Sanchis, and M. Martnez. A new graphical visualization of n-dimensional pareto front for decision-making in multiobjective optimization. Information Sciences, 178(20):3908–3924, 2008.

Devika Bommisetty, Mohamed Dessouky, and Larry Jacobs. Scheduling collection of recyclable material at northern illinois university campus using a two-phase algorithm. Computers and Industrial Engineering, 35(34):435–438, 1998.

Kris Braekers and Attila A. Kovacs. A multi-period dial-a-ride problem with driver consistency. Transportation Research Part B: Methodological, 94:355–377, 2016.

Ann Melissa Campbell and Jill Hardin Wilson. Forty years of periodic vehicle routing. Networks, 63(1):2–15, 2014.

N. Christofides and J. E. Beasley. The period routing problem. Networks, 14(2):237–256, 1984.

S. Coene, A. Arnout, and F. C. R. Spieksma. On a periodic vehicle routing problem. The Journal of the Operational Research Society, 61(12):1719–1728, 2010.

E. G. Coffman, M. R. Garey, and D. S. Johnson. Approximation Algorithms for Bin-Packing An Updated Survey, pages 49–106. Springer Vienna, Vienna, 1984.

Jean-Franois Cordeau, Michel Gendreau, and Gilbert Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks, 30(2):105–119, 1997.

G. A. Croes. A method for solving traveling-salesman problems. Operations Research, 6 (6):791–812, 1958.

K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. Evolutionary Computation, IEEE Transactions on, 18(4):577–601, 2014.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. Evolutionary Computation, IEEE Transactions on, 6(2):182–197, 2002.

Billy E. Gillett and R. Miller Leland. A heuristic algorithm for the vehicle-dispatch problem. Operations Research, 22(2):340–349, 1974.

Maoguo Gong, Licheng Jiao, Haifeng Du, and Liefeng Bo. Multiobjective immune algorithm with nondominated neighbor-based selection. Evolutionary Computation, 16 (2):225–255, 2008.

Chris Groër, Bruce Golden, and Edward Wasil. The consistent vehicle routing problem. Manufacturing & Service Operations Management, 11(4):630–643, 2009.

Vera C. Hemmelmayr, Karl F. Doerner, and Richard F. Hartl. A variable neighborhood search heuristic for periodic routing problems. European Journal of Operational Research, 195(3):791–802, 2009.

Wooseung Jang, Huay H. Lim, Thomas J. Crowe, Gail Raskin, and Thomas E. Perkins. The missouri lottery optimizes its scheduling and routing to improve efficiency and balance. Interfaces, 36(4):302–313, 2006.

Attila A. Kovacs, Bruce L. Golden, Richard F. Hartl, and Sophie N. Parragh. Vehicle routing problems in which consistency considerations are important: A survey. Networks, 64(3):192–213, 2014a.

Attila A. Kovacs, Sophie N. Parragh, Richard F. Hartl, and Sophie N. Parragh. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. Networks, 63(1):60–81, 2014b.

Attila A. Kovacs, Bruce L. Golden, Richard F. Hartl, and Sophie N. Parragh. The generalized consistent vehicle routing problem. Transportation Science, 49(4):796–816, 2015a.

Attila A. Kovacs, Sophie N. Parragh, and Richard F. Hartl. The multi-objective generalized consistent vehicle routing problem. European Journal of Operational Research, 247(2):441–458, 2015b.

Kunlei Lian, Ashlea Bennett Milburn, and Ronald L. Rardin. An improved multi-directional local search algorithm for the multi-objective consistent vehicle routing problem. IIE Transactions, 48(10):975–992, 2016.

Zhixing Luo, Hu Qin, ChanHou Che, and Andrew Lim. On service consistency in multi-period vehicle routing. European Journal of Operational Research, 243(3):731–744, 2015.

AnaCristina Matos and RuiCarvalho Oliveira. An Experimental Study of the Ant Colony System for the Period Vehicle Routing Problem, volume 3172 of Lecture Notes in Computer Science, book section 26, pages 286–293. Springer Berlin Heidelberg, 2004.

Pablo Maya, Kenneth Srensen, and Peter Goos. A metaheuristic for a teaching assistant assignment-routing problem. Computers and Operations Research, 39(2):249–258, 2012.

Ashlea Bennett Milburn and Jessica Spicer. Multi-objective home health nurse routing with remote monitoring devices. International Journal of Planning and Scheduling, 1(4): 242–263, 2013.

Teemu Nuortio, Jari Kytjoki, Harri Niska, and Olli Brysy. Improved route planning and scheduling of waste collection and transport. Expert Systems with Applications, 30(2): 223–232, 2006.

Zhang Qingfu and Li Hui. Moea/d: A multiobjective evolutionary algorithm based on decomposition. Evolutionary Computation, IEEE Transactions on, 11(6):712–731, 2007.

D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis. Approximate algorithms for the traveling salesperson problem. In 15th Annual Symposium on Switching and Automata Theory (swat 1974), pages 33–42, 1974.

David Russell, Robert J. Rosati, and Evie Andreopoulos. Continuity in the provider of home-based physical therapy services and its implications for outcomes of patients. Physical Therapy, 92(2):227–235, 2012.

R. Russell and W. Igo. An assignment routing problem. Networks, 9(1):1–17, 1979.

L. Shih and Y. Lin. Optimal routing for infectious waste collection. Journal of Environmental Engineering, 125(5):479–484, 1999.

Li-Hsing Shih and Hua-Chi Chang. A routing and scheduling system for infectious waste collection. Environmental Modeling and Assessment, 6(4):261–269, 2001.

Anirudh Subramanyam and Chrysanthos E. Gounaris. A branch-and-cut framework for the consistent traveling salesman problem. European Journal of Operational Research, 248 (2):384–395, 2016.

C. D. Tarantilis, F. Stavropoulou, and P. P. Repoussis. A template-based tabu search algorithm for the consistent vehicle routing problem. Expert Systems with Applications, 39(4):4233–4239, 2012.

Fabien Tricoire. Multi-directional local search. Computers & Operations Research, 39(12): 3089–3101, 2012.

Thibaut Vidal, Teodor Gabriel Crainic, Michel Gendreau, Nadia Lahrichi, and Walter Rei. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. Operations Research, 60(3):611–624, 2012.

Christel A. Woodward, Julia Abelson, Sara Tedford, and Brian Hutchison. What is important to continuity in home care?: Perspectives of key stakeholders. Social Science and Medicine, 58(1):177–192, 2004.

Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization. CIMNE, Spain, 2002.

4. A Branch-and-Price Algorithm for the Consistent Vehicle Routing Problem

4.1 Introduction

In today's competitive markets, consistent service is a key to improving customer satisfaction and customer loyalty (Kovacs et al., 2014a). Practitioners in a number of service industries, including home health care and small package delivery, strive to provide two important types of service consistency: driver consistency and time consistency. In the home health care industry, caregivers are scheduled to visit home-bound patients over an enrolled treatment horizon. Minimizing the number of different caregivers assigned to a patient helps increase familiarity between them and reduce communication complexity. It also improves caregivers' ability to make accurate observations and benefits health care outcomes, such as lower rates of hospitalization and fewer visits to the emergency department (Russell et al., 2012). Consistent visiting times allow patients to plan their days more readily without causing too many disturbances to their daily routines (Woodward et al., 2004). In the small package delivery industry, consistent driver and delivery time increases driver familiarity with a delivery region and also fosters customer-centered service that leads to possible additional business gains (Wong, 2008). Service consistency has also been recognized as a key characteristic of high customer loyalty in other service applications, including vendor-managed inventory systems (Day et al., 2009), beer and wine distribution (Erera et al., 2009) and aircraft fleet routing and scheduling (Ioachim et al., 1999).

The Consistent Vehicle Routing Problem (ConVRP), first defined in Groër et al. (2009), models an important class of transportation problems with service consistency considerations encountered in the aforementioned service industries. It is the first vehicle routing problem (VRP) variant in the literature that focuses on improving customer experience though consistent service. In the traditional VRP, a set of vehicle routes must be determined such that each customer is visited once and only once, and the total travel

distance of all vehicles is minimized. ConVRP generalizes the traditional VRP to consider an extended planning horizon over which service consistency is enforced. In the ConVRP, a customer may require visits on one or more predetermined days over a planning horizon of multiple days. A customer is said to be a *frequent* customer if it needs visits on more than one day. Service consistency constraints are defined to require that each frequent customer be visited by the same driver across the planning horizon (driver consistency), and the arrival time differential at a customer must not exceed a preset limit (time consistency). The objective of ConVRP is to minimize the total travel distance of all vehicles across the planning horizon such that all customer requirements are fulfilled without violating vehicle capacity, maximum route duration and service consistency constraints.

The ConVRP is NP-hard, given that it reduces to the traditional VRP when the planning horizon is set to one or to multiple separate VRPs when no customers are frequent customers. Noting the imposed service consistency constraints on frequent customers, the ConVRP does not separate into individual VRPs and hence poses a significant challenge from a solution perspective. There exist a number of heuristic algorithms in the literature that aim to identify near-optimal solutions quickly. These are reviewed in Section 4.2. However, it remains a challenge to solve the ConVRP exactly within a reasonable amount of time and very limited efforts have been made in the literature to address this. This paper intends to fill this gap. There are two primary contributions in this paper. First, we identify a new set of constraints based on the existing mixed-integer program (MIP) formulation for the ConVRP in the literature. The enhanced MIP is able to reduce computational times significantly compared to the existing MIP. Second, we propose a branch-and-price (B&P) solution method for the ConVRP for the first time in the literature. A heuristic algorithm is designed to solve the subproblem efficiently. The performance of B&P is validated on benchmark instances taken from the literature and newly generated instances of the same form. Computational results show its competitiveness in solving instances with more than 14 customers.

The remainder of this paper is organized as follows: Section 2 reviews existing solution methods in the literature that aim to solve consistency-related vehicle routing problems. Section 3 provides the MIP formulation of the ConVRP from Groër et al. (2009) and introduces a set of new constraints to sharpen its LP relaxation. Section 4 details the proposed B&P algorithm. Computational results are given in Section 5, followed by conclusions in Section 6.

4.2 Related literature

This section provides a brief review of research related to our study. We focus on the existing solution approaches related to ConVRP.

The ConVRP is first introduced by Groër et al. (2009) in the context of the small package delivery industry where customers require repeatable visits over a planning horizon of multiple days. The ConVRP is modeled by Groër et al. (2009) as a MIP and a record-to-record travel heuristic (ConRTR) is proposed to obtain near optimal solutions quickly. The concept of template routes is used in the algorithm to ensure optimal driver consistency and encourage better time consistency. Template routes consider only frequent customers that require service on more than one day. A route for day d can be derived from the template route by removing all those customers who do not require service on day d and inserting customers who require service on only day d. The MIP formulation takes up to several days to solve small-sized instances with 10 or 12 customers using CPLEX 11.0.

Utilizing the concept of template routes, Tarantilis et al. (2012) propose a two-stage tabu search heuristic (TTS) for the ConVRP. In the first stage, template routes involving only frequent customers are optimized using tabu search to minimize total travel time. During the search process, daily routes are derived from template routes in order to evaluate a neighborhood solution and check its feasibility. Based on the template routes identified in the first stage, tabu search is employed in another mode to optimize daily vehicle routes including both frequent customers and customers that only require service on that day.

Computational experiments on the benchmark instances created in Groër et al. (2009) show that TTS outperforms ConRTR both in terms of total travel times and total number of vehicles used.

Kovacs et al. (2014b) develop an adaptive large neighborhood search algorithm (TALNS) that employs a number of destroy and repair operators to improve template routes involving only frequent customers. Actual daily routes are derived from template routes during the search process in order to compute the objective values of template routes. The proposed algorithm shows better performance than both ConRTR and TTS with respect to solution quality on the instances created in Groër et al. (2009). The authors also consider a relaxed variant of the ConVRP in which vehicles are allowed to wait at the depot before beginning their routes and improved time consistency is observed in this problem variant.

Luo et al. (2015) examine a multi-period VRP variant in which each customer must receive service within a given time window and can only be serviced by a limited number of different vehicles over the planning horizon. A MIP model is formulated and a three-stage heuristic algorithm named Decomposition, Repair and Distance Reduction (DRDR) is devised. The first stage tries to construct a feasible initial solution using a decomposition strategy. An iterative tree-search-with-repair mechanism is then employed to reduce the number of vehicles used in the second stage. A unified tabu search is applied in the last stage to reduce total travel distance of the solution. The proposed heuristic is applied to solve the ConVRP instances created in Groër et al. (2009) and performs better than ConRTR in terms of total travel time, but worse than TTS and TALNS. It is shown that DRDR can achieve the smallest arrival time differential on average among all tested algorithms.

Kovacs et al. (2015) study a generalized variant of the ConVRP that relaxes the strict driver consistency in the ConVRP by allowing multiple different drivers to visit a customer. Time consistency considerations are included in the objective function using a weighted sum approach. In addition, customers are each associated with AM/PM time

windows which limit the earliest possible service starting time at customer locations. The authors propose a large neighborhood search heuristic that does not use the concept of template routes. The ConVRP instances created in Groër et al. (2009) are solved using the proposed algorithm and performance superior to ConRTR, TTS and TALNS is observed. The only exact solution methods proposed for consistency-related routing problems that we are aware of in the literature are contributed by Braekers and Kovacs (2016) and Subramanyam and Gounaris (2016). Braekers and Kovacs (2016) consider a dial-a-ride problem in the transportation of disabled and elderly people over a multi-period time horizon. Each customer is associated with a time window and maximum ride time. Driver consistency is imposed such that the maximum number of different drivers that transport a customer over the planning horizon does not exceed a preset bound. Two mathematical formulations are proposed and a branch-and-cut solution method is suggested. Seven types of valid inequalities are used to ensure feasibility and strengthen the model. A large neighborhood search heuristic is developed to find near optimal solutions efficiently. Computational results on a set of newly generated instances show that enforcing only one driver visiting a customer may cause up to 27.98% increase in travel cost, while the routing cost increase is no more than 5.80% when at least two drivers are allowed per customer. Subramanyam and Gounaris (2016) study the consistent traveling salesman problem (ConTSP) in which a minimum cost set of routes should be designed for a single vehicle in order to visit customers requiring service over a planning horizon of multiple days. Consistent arrival times constraints on frequent customers are imposed. A branch-and-cut solution framework is developed. Specifically, three MIP formulations for this problem are introduced, and a new class of inconsistent path elimination inequalities is defined. Computational experiments show that instances with up to 50 customers over a five-period horizon can be solved to optimality. These exact methods can not be generalized to solve the ConVRP directly due to the availability of multiple vehicles with constraints and different definitions of driver consistency. In addition, Braekers and Kovacs (2016) do not

consider time consistency in their formulation.

This review of the literature shows that there is no effective solution approach in the literature to solve the the ConVRP optimally. We intend to fill this gap by proposing a B&P algorithm for this problem. B&P algorithms have proven effective in solving other VRP variants, including capacitated VRP (Pecin et al., 2014), pickup and delivery problem with time windows (Cherkesly et al., 2016) and split-delivery VRP with time windows (Desaulniers, 2010).

4.3 Problem formulation

In the ConVRP, customers in set $\mathcal{N} = \{1, \ldots, |\mathcal{N}|\}$ require service over a planning horizon $\mathcal{D} = \{d_1, \ldots, d_{|\mathcal{D}|}\}$. Located at the depot $\{0\}$ there are a set of homogeneous vehicles $\mathcal{K} = \{1, \ldots, |\mathcal{K}|\}$ available to serve the customers. Each vehicle $k \in \mathcal{K}$ has the same physical capacity Q and maximum route duration T. The number of vehicles $|\mathcal{K}|$ can be as large as $|\mathcal{N}|$.

The ConVRP is defined on a complete directed graph $\mathcal{G} = (\mathcal{N}^0 = \mathcal{N} \cup \{0\}, \mathcal{A})$, where $\mathcal{A} = \{(i, j) \mid i, j \in \mathcal{N}^0, i \neq j\}$. Each arc (i, j) is associated with a travel time t_{ij}. On each day $d \in \mathcal{D}$, each customer i has a predetermined non-negative demand q_{id} and service duration s_{id}. An auxiliary parameter w_{id} is defined such that it equals 1 if $q_{id} > 0$ and 0 otherwise. It is assumed that a driver is always assigned to the same vehicle across the planning horizon and this paper uses the terms *driver* and *vehicle* interchangeably. The time consistency is imposed on all frequent customers by limiting the maximum arrival time differential not to exceed L.

Groër et al. (2009) define a MIP, which we denote as MIP1, for the ConVRP using the following decision variables:

- x_{ijkd}: equals 1 if vehicle k visits customer j immediately after customer i on day d and equals 0 otherwise, $d \in \mathcal{D}$, $k \in \mathcal{K}$, $i \in \mathcal{N}^0$, $j \in \mathcal{N}^0$

- a_{id}: the vehicle arrival time at customer i on day d, $i \in \mathcal{N}^0$, $d \in \mathcal{D}$
- y_{ikd}: equals 1 if customer i is visited by vehicle k on day d and equals 0 otherwise, $i \in \mathcal{N}^0$, $k \in \mathcal{K}$, $d \in \mathcal{D}$

Using this notation, MIP1 is given as follows:

$$\min \quad f = \sum_{d\in\mathcal{D}} \sum_{k\in\mathcal{K}} \sum_{(i,j)\in\mathcal{A}} t_{ij} x_{ijkd}, \tag{4.1}$$

$$\text{s.t.} \quad y_{0kd} = 1,\ \forall k \in \mathcal{K}, d \in \mathcal{D}, \tag{4.2}$$

$$a_{0d} = 0,\ \forall d \in \mathcal{D}, \tag{4.3}$$

$$\sum_{k\in\mathcal{K}} y_{ikd} = w_{id},\ \forall i \in \mathcal{N}, d \in \mathcal{D}, \tag{4.4}$$

$$\sum_{i\in\mathcal{N}} q_{id} y_{ikd} \le Q,\ \forall k \in \mathcal{K}, d \in \mathcal{D}, \tag{4.5}$$

$$\sum_{i\in\mathcal{N}^0} x_{ijkd} = \sum_{i\in\mathcal{N}^0} x_{jikd} = y_{jkd},\ \forall j \in \mathcal{N}^0, k \in \mathcal{K}, d \in \mathcal{D}, \tag{4.6}$$

$$y_{ikd_\alpha} - y_{ikd_\beta} \ge w_{id_\alpha} + w_{id_\beta} - 2,\ \forall d_\alpha, d_\beta \in \mathcal{D}, \alpha \ne \beta, i \in \mathcal{N}^0, k \in \mathcal{K}, \tag{4.7}$$

$$y_{ikd_\alpha} - y_{ikd_\beta} \le -(w_{id_\alpha} + w_{id_\beta} - 2),\ \forall d_\alpha, d_\beta \in \mathcal{D}, \alpha \ne \beta, i \in \mathcal{N}^0, k \in \mathcal{K}, \tag{4.8}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) - T(1 - x_{ijkd}) \le a_{jd},\ \forall d \in \mathcal{D}, k \in \mathcal{K}, i \in \mathcal{N}^0, j \in \mathcal{N}, \tag{4.9}$$

$$a_{id} + x_{ijkd}(s_{id} + t_{ij}) + T(1 - x_{ijkd}) \ge a_{jd},\ \forall d \in \mathcal{D}, k \in \mathcal{K}, i \in \mathcal{N}^0, j \in \mathcal{N}, \tag{4.10}$$

$$0 \le a_{id} + w_{id}(s_{id} + t_{i0}) \le T w_{id},\ \forall i \in \mathcal{N}, d \in \mathcal{D}, \tag{4.11}$$

$$a_{id_\alpha} - a_{id_\beta} \ge -L + T(w_{id_\alpha} + w_{id_\beta} - 2),\ \forall i \in \mathcal{N}^0, d_\alpha, d_\beta \in \mathcal{D}, \alpha \ne \beta, \tag{4.12}$$

$$a_{id_\alpha} - a_{id_\beta} \le L - T(w_{id_\alpha} + w_{id_\beta} - 2),\ \forall i \in \mathcal{N}^0, d_\alpha, d_\beta \in \mathcal{D}, \alpha \ne \beta, \tag{4.13}$$

$$x_{ijkd} \in \{0,1\},\ y_{ikd} \in \{0,1\},\ a_{id} \ge 0, \forall i, j \in \mathcal{N}^0, k \in \mathcal{K}, d \in \mathcal{D}. \tag{4.14}$$

The objective function (4.1) minimizes the total travel distance of all vehicles over all days in the planning horizon. Constraint sets (4.2) and (4.3) require that the depot be visited at time 0 by all vehicles on all days. Constraint set (4.4) ensures that customers are visited exactly once when they require service on any day. Constraint set (4.5) guarantees that

each vehicle carries no more than Q units on any given day. Constraint set (4.6) makes sure that each customer has only one predecessor and one successor. Constraint sets (4.7) and (4.8) ensure that each customer is served by the same driver whenever he/she needs service. Constraint sets (4.9) and (4.10) determine the arrival times at the individual customers. Constraint set (4.11) requires that the vehicle travel time limit is not violated. Constraint sets (4.12) and (4.13) enforce the maximum arrival time differential at customer i on any two days is no more than L units. Constraint set (4.14) specifies the decision variable types.

In this paper, we introduce a new MIP formulation, denoted MIP2, by adding new variables and constraints to the previous MIP1. Define a new binary variable n_{kd} to indicate whether there exists any customer on day d that is visited by vehicle k. The new constraints can then be defined as follows:

$$n_{kd} \geq y_{ikd}, \ \forall i \in \mathcal{N}, k \in \mathcal{K}, d \in \mathcal{D}, \tag{4.15}$$

$$n_{kd} \leq \sum_{i \in \mathcal{N}} y_{ikd}, \ \forall k \in \mathcal{K}, d \in \mathcal{D}, \tag{4.16}$$

$$\sum_{i \in \mathcal{N}^0} x_{i0kd} = \sum_{i \in \mathcal{N}^0} x_{0ikd} = n_{kd}, \ \forall k \in \mathcal{K}, d \in \mathcal{D}. \tag{4.17}$$

Constraint sets (4.15) and (4.16) compute the value of n_{kd}. Constraint set (4.17) requires the number of in-arcs and out-arcs at the depot equals n_{kd}.

4.4 Branch and price

This section describes the proposed B&P algorithm for the ConVRP. Section 4.4.1 introduces the master problem and subproblem associated with the proposed B&P algorithm. Section 4.4.2 describes the B&P framework. A starting set of columns is generated using the heuristic detailed in Section 4.4.3. The column generation process is described in Section 4.4.4. The proposed column generation heuristic is explained in

Section 4.4.5. Section 4.4.6 describes the branching approach.

4.4.1 Problem reformulation

The ConVRP can be reformulated as the following set covering model:

$$\min \quad \sum_{v\in\Omega} c_v\theta_v, \tag{4.18}$$

$$\text{s.t.} \quad \sum_{v\in\Omega} g_{iv}\theta_v \geq 1,\ \forall i \in \mathcal{N}, \tag{4.19}$$

$$\sum_{v\in\Omega} \theta_v \leq |\mathcal{K}|, \tag{4.20}$$

$$\theta_v \in \mathbb{N}^0,\ \forall v \in \Omega. \tag{4.21}$$

In this formulation, the set Ω includes all the feasible columns for the ConVRP and each column $v \in \Omega$ represents a routing plan visiting a set of customers $\mathcal{N}_v \subset \mathcal{N}$ across the planning horizon. A column consists of a set of single vehicle routes; one for each day $d \in \mathcal{D}$ that visits all customers in $\mathcal{N}_v$ requiring service on that day (i.e., such that $w_{id} = 1$). Furthermore, the maximum arrival time differential at any frequent customer in $\mathcal{N}_v$ across the planning horizon must not exceed the time limit L. By enforcing that the daily routes in v are operated by the same driver, customers in $\mathcal{N}_v$ also satisfy driver consistency as required in the formulation of the ConVRP. In addition, c_v represents the total travel distance of all the routes in column v.

Usually there exist a very large number of columns in Ω: we use a decision variable θ_v to indicate whether a column v is in the optimal solution. Variable θ_v is set to be a nonnegative integer (given by the set $\mathbb{N}^0$) instead of a binary variable in order to avoid constraints $\theta_v \leq 1$ in the LP relaxation of (4.18)–(4.21). It is clear that any solution with $\theta_v \geq 2$ for any v would not be optimal. Constant g_{iv} equals 1 if customer i is included in column v, and equals 0 otherwise. Constraint set (4.19) requires that every customer i must appear in at least one column. Constraint set (4.20) limits the total number of

vehicles used. Constraint set (4.21) sets the variable types.

Define the Master Problem ($MP(\Omega)$) as the LP relaxation of (4.18)–(4.21), i.e.,

$$\min \quad \sum_{v\in\Omega} c_v\theta_v, \tag{4.22}$$

$$\text{s.t.} \quad \sum_{v\in\Omega} g_{iv}\theta_v \geq 1,\ \forall i \in \mathcal{N}, \tag{4.23}$$

$$\sum_{v\in\Omega} \theta_v \leq |\mathcal{K}|, \tag{4.24}$$

$$\theta_v \geq 0,\ \forall v \in \Omega, \tag{4.25}$$

and the Restricted Master Problem ($RMP(\Omega_1)$) associated with a subset $\Omega_1 \subset \Omega$ as

$$\min \quad \sum_{v\in\Omega_1} c_v\theta_v, \tag{4.26}$$

$$\text{s.t.} \quad \sum_{v\in\Omega_1} g_{iv}\theta_v \geq 1,\ \forall i \in \mathcal{N}, \tag{4.27}$$

$$\sum_{v\in\Omega_1} \theta_v \leq |\mathcal{K}|, \tag{4.28}$$

$$\theta_v \geq 0,\ \forall v \in \Omega_1, \tag{4.29}$$

and the dual problem $D(\Omega_1)$ of $RMP(\Omega_1)$ as

$$\max \quad \sum_{i\in\mathcal{N}} \lambda_i + |\mathcal{K}|\lambda_0, \tag{4.30}$$

$$\text{s.t.} \quad \sum_{i\in\mathcal{N}} g_{iv}\lambda_i + \lambda_0 \leq c_v,\ \forall v \in \Omega_1, \tag{4.31}$$

$$\lambda_i \geq 0,\ \forall i \in \mathcal{N}, \tag{4.32}$$

$$\lambda_0 \leq 0, \tag{4.33}$$

where λ_i is the dual variable associated with the i^{th} constraint (4.27) and λ_0 is the dual variable associated with constraint (4.28).

Then the subproblem is to find a new column $v \in \Omega \backslash \Omega_1$ such that

$$r_v : \ c_v - \sum_{i \in \mathcal{N}} g_{iv} \lambda_i^* - \lambda_0^* < 0 \tag{4.34}$$

To formulate the subproblem, define the following variables:

- y_i: equals 1 if customer i is included in the new column, 0 otherwise, $i \in \mathcal{N}^0$
- n_d: equals 1 if there is at least one customer requiring service on day d, 0 otherwise, $d \in \mathcal{D}$
- x_{ijd}: equals 1 if arc (i,j) is used on day d, $(i,j) \in \mathcal{A},\ d \in \mathcal{D}$
- a_{id}: arrival time at customer i on day d, $i \in \mathcal{N}^0,\ d \in \mathcal{D}$

The subproblem can then be stated as follows:

$$\min \quad \sum_{d \in \mathcal{D}} \sum_{(i,j) \in \mathcal{A}} t_{ij} x_{ijd} - \sum_{i \in \mathcal{N}} y_i \lambda_i^* - \lambda_0^*, \tag{4.35}$$

$$\text{s.t.} \quad y_0 = 1, \tag{4.36}$$

$$x_{dii} = 0,\ \forall d \in \mathcal{D}, i \in \mathcal{N}, \tag{4.37}$$

$$\sum_{j \in \mathcal{N}^0} x_{ijd} = \sum_{j \in \mathcal{N}^0} x_{jid} = y_i w_{id},\ \forall i \in \mathcal{N}, d \in \mathcal{D}, \tag{4.38}$$

$$n_d \geq y_i,\ \forall d \in D, i \in \mathcal{N}, \tag{4.39}$$

$$n_d \leq \sum_{i \in \mathcal{N}} y_i,\ \forall d \in D, \tag{4.40}$$

$$\sum_{i \in \mathcal{N}} x_{i0d} = \sum_{i \in \mathcal{N}} x_{0id} = n_d,\ \forall d \in D, \tag{4.41}$$

$$a_{0d} = 0,\ \forall d \in \mathcal{D}, \tag{4.42}$$

$$a_{id} + x_{ijd}(s_{id} + t_{ij}) - T(1 - x_{ijd}) \leq a_{jd},\ \forall d \in \mathcal{D}, i \in \mathcal{N}^0, j \in \mathcal{N}, \tag{4.43}$$

$$a_{id} + x_{ijd}(s_{id} + t_{ij}) + T(1 - x_{ijd}) \geq a_{jd},\ \forall d \in \mathcal{D}, i \in \mathcal{N}^0, j \in \mathcal{N}, \tag{4.44}$$

$$\sum_{i \in \mathcal{N}} q_{id} y_i w_{id} \leq Q,\ \forall d \in \mathcal{D}, \tag{4.45}$$

$$0 \leq a_{id} + y_i w_{id}(s_{id} + t_{i0}) \leq T y_i w_{id},\ \forall i \in \mathcal{N}, d \in \mathcal{D}, \tag{4.46}$$

$$a_{id_\alpha} - a_{id_\beta} \geq -L + T(y_i w_{id_\alpha} + y_i w_{id_\beta} - 2),\ \forall i \in \mathcal{N}, d_\alpha, d_\beta \in \mathcal{D}, \alpha \neq \beta, \tag{4.47}$$

$$a_{id_\alpha} - a_{id_\beta} \leq L - T(y_i w_{id_\alpha} + y_i w_{id_\beta} - 2),\ \forall i \in \mathcal{N}, d_\alpha, d_\beta \in \mathcal{D}, \alpha \neq \beta, \tag{4.48}$$

$$y_i \in \{0,1\}, x_{ijd} \in \{0,1\}, a_{id} \geq 0,\ \forall i \in \mathcal{N}^0, j \in \mathcal{N}^0, d \in \mathcal{D}. \tag{4.49}$$

The objective function (4.35) minimizes the total travel distance across all days minus the reward due to including customer $i \in \mathcal{N}^0$ in the column. Constraint set (4.36) requires that the depot is always included in the column. Constraint set (4.37) forbids the use of arcs (i, i) on any day in the column. Constraint set (4.38) requires that each node has one outgoing arc and one incoming arc on each day if they require service. Constraint sets (4.39) and (4.40) determine whether there exists any customer requiring service on day d. Constraint set (4.41) sets the number of outgoing and incoming arcs at the depot on day d based on the value of n_d. Constraint set (4.42) sets the arrival time at the depot on each day to 0. Constraint sets (4.43) and (4.44) compute the arrival times at customers if they require visits. These constraints also serve to eliminate subtours. Constraint set (4.45) makes sure that the total load on the tour must not exceed the vehicle capacity. Constraint set (4.46) limit the total route duration on each day to be at most T. Constraint sets (4.47) and (4.48) enforce the arrival time differential at a customer on any two different days to be no more than L units. Constraint set (4.49) specifies the decision variable types. Compared with MIP1 given in (4.1)–(4.14), this subproblem can be viewed as a prize-collecting ConTSP where the selection of customers to be serviced by the single vehicle is affected by their associated prizes that are indicated by λ defined in (4.30)–(4.33).

4.4.2 Branch-and-price framework

The B&P algorithm combines the branch-and-bound (B&B) framework with the column generation process at each tree node of B&B. Algorithm 5 shows the framework of the proposed B&P algorithm. In this paper, we use P_x and P_0 to denote a general B&B tree

node and the root node, respectively. Each tree node P_x stores (i) a set of columns denoted by Ω_{P_x}; (ii) an optimal LP objective value $f(P_x)$; (iii) an optimal set of columns $V(P_x)$; and (iv) an underlying graph $\mathcal{G}(P_x)$. The proposed B&P algorithm starts with populating the empty Ω_{P_0} using columns obtained from the procedure `FindInitialCols(`$|M|$`,` I_{max}`)`. Since the number of vehicles is unlimited, a special set of columns Ω_0 is also created by using a separate vehicle to visit each individual customer and supplemented to Ω_{P_0}. This is to ensure the feasibility of the root node master problem.

The master problem associated with P_0 (i.e., $MP(\Omega_{P_0})$) is then solved using a column generation process. New columns are generated by solving the subproblem defined in (4.35)–(4.49) heuristically or/and exactly: a heuristic is employed first to identify new columns and the exact algorithm will be skipped if the heuristic succeeds; otherwise, the exact algorithm (i.e., applying CPLEX to model (4.35)–(4.49)) will be used after the heuristic algorithm fails to find new columns. The master problem is said to be optimal if no more new columns can be found using the exact method. If the optimal solution of the master problem is integer feasible, the ConVRP is solved; otherwise, branching is required to create two child nodes on the B&B tree. The column generation process is applied immediately upon branching for each child tree node to solve its associated master problem. After the master problem associated with a tree node P is solved using column generation, this tree node P is added into set Φ for further processing, including checking integer feasibility or branching. The processing of tree nodes in Φ takes a last-in-first-out approach.

This algorithm takes as input in Step 1 the time limit of the whole algorithm. The lower and upper bound are set in Step 2. Step 3 creates the root node P_0 with empty Ω_{P_0} and an empty set Φ. The starting set of columns to be used in the restricted master problem of P_0 are generated using procedure `FindInitialCols(`$|M|$`,` I_{max}`)` in Step 4, which is a genetic algorithm. This is described in detail in Section 4.4.3. The special set of single-customer columns Ω_0 is added to Ω_{P_0} too. The column generation process is denoted by `ColGen(`P_0`)`

Algorithm 5 Branch-and-Price workflow

1: **Input:** time limit
2: let incumbent column set $V^* = \emptyset$, lower bound $LB = -\infty$ and upper bound $UB = \infty$
3: create tree node P_0 with empty column set Ω_{P_0} and empty set Φ
4: populate Ω_{P_0} using procedure `FindInitialCols(`$|M|$`,` I_{max}`)` and Ω_0
5: solve $MP(\Omega_{P_0})$ using procedure `ColGen(`P_0`)` and let $\Phi = \Phi \cup P_0$
6: **repeat**
7: take the last node P_x from Φ and let $\Phi = \Phi \backslash P_x$
8: **if** $MP(\Omega_{P_x})$ is feasible **then**
9: **if** $V(P_x)$ of $MP(\Omega_{P_x})$ is integer feasible **then**
10: **if** $f(P_x) < UB$ **then** let $UB = f(s^*_{P_x})$ and update $V^* = V(P_x)$
11: **end if**
12: terminate this node by solving, no branching required
13: **else**
14: use procedure `Branch(`P_x`)` to create two child nodes P_l and P_r
15: apply `ColGen(`P_l`)` and `ColGen(`P_r`)` on P_l and P_r
16: let $\Phi = \Phi \cup \{P_l, P_r\}$
17: let $LB = \min f(P_x)$ where $P \in \Phi$
18: **end if**
19: **else**
20: discard this node by infeasibility
21: **end if**
22: **until** Φ is empty or time limit is reached
23: **Output:** LB, UB and V^*

as in Step 5. In Steps 7 to 21, a tree node P_x is taken from Φ and checked for feasibility: P_x is discarded if it is infeasible (Step 20); otherwise, the integer feasibility is checked in Step 9 to decide whether branching is required. If there exist fractional values in the optimal solution, procedure `Branch(`P_x`)` is called to generate two child tree nodes P_l and P_r (Step 14). The two new nodes are immediately solved using `ColGen` and added to set Φ for future processing. Every time new tree nodes enter the set Φ, the global lower bound is updated using all tree nodes in set Φ (Step 17). The algorithm stops when set Φ is empty, which indicates global optimality, or the given time limit is reached. The following sections describe the algorithm in details.

4.4.3 Generating starting column set Ω_{P_0} for root node P_0

This section describes the procedure `FindInitialCols(`$|M|$`,` I_{max}`)` in Step 4 of Algorithm 5 to generate the initial set of feasible columns for the Restricted Master Problem at root node P_0. A good set of initial columns at P_0 helps speed up the solution process. In this paper, a Genetic Algorithm (GA) is developed to find near optimal solutions to the ConVRP quickly. Algorithm 6 shows its workflow. Note that the term *solution* is used here to indicate a routing plan for the ConVRP and a solution may consist of multiple columns for (4.18)–(4.21).

Algorithm 6 describes the GA framework that takes as input a population size $|M|$ and maximal number of non-improving iterations I_{max}. The initial population M_0 in Step 2 is generated using the procedure in the following section. All feasible columns that exist in M_0 are saved to Ψ in Step 3. In each following iteration, a new population M'_η is created from M_η using a crossover operator (Step 6). Solutions from M_η and M'_η are combined to generate $M_{\eta+1}$ (Step 7). The best solution m^* is then updated and all feasible columns in $M_{\eta+1}$ are saved to Ψ.

Algorithm 6 Workflow of `FindInitialCols(`$|M|$`,` I_{max}`)`

1: **Input:** population size $|M|$ and maximal number of non-improving iterations I_{max}
2: let iteration $\eta = 0$ and create an initial population M_η of $|M|$ randomly generated solutions
3: save all feasible columns in each solution of the population to set Ψ
4: let m^* denote the best solution encountered so far
5: **repeat**
6: employ `GenChildPop()` to create a new population M'_η based on M_η
7: use `GenNextPop()` to create $M_{\eta+1}$ based on M_η and M'_η
8: update Ψ and m^*
9: let $\eta = \eta + 1$
10: **until** m^* not improved for I_{max} iterations
11: **Output** feasible column set Ψ

4.4.3.1 Initial solution generation

This section explains the algorithm used to generate the solutions in population M_0 in Algorithm 6. The algorithm starts with sorting all customers in the non-decreasing order of the clockwise angle they make with the depot and the vector pointing from the depot along the positive horizontal line. An empty solution m is then created and customers are considered for insertion into m following the determined order. The first customer to be inserted is randomly selected from the sorted customer list and all required visits to a customer are inserted into m at once when the customer is chosen for insertion. For a customer i being considered for insertion, the algorithm identifies all the vehicles in which all required visits to i can be appended to the tail of the vehicle route on corresponding days feasibly with respect to vehicle capacity, maximum route duration and the time consistency constraint for i. Note that there may exist multiple empty vehicles at this step and only the one with the smallest index is chosen. The total travel distance increase of inserting all required visits to i into any feasible vehicle k is noted. Finally, vehicle k^* with the smallest travel distance increase is chosen and all required visits to i are appended to the end of k^* on the corresponding days.

4.4.3.2 New population generation

In `GenChildPop()`, each solution in the child population M'_η is generated by applying a crossover operator on two randomly selected solutions from M_η. Each solution in M'_η is further improved by a Large Neighborhood Search (LNS) heuristic with probability p_l. The LNS is described in the following section. All feasible columns in any solution of M'_η are saved to Ψ. In `GenNextPop()`, M_η and M'_η are combined and sorted in non-decreasing order of total travel distance. The top $0.8|M|$ solutions in the combined population enter the new population M_{i+1}. The remaining solutions in $M_{\eta+1}$ are generated randomly to introduce diversity. With probability p_l, a newly generated solution is improved by the LNS. The percentage values of 0.8 and 0.2 are determined based on experience.

The LNS used in this paper follows the same LNS framework proposed in Kovacs et al. (2015). Four removal operators are taken from Kovacs et al. (2015), including random removal, related removal, cluster removal and worst time consistency removal. Two reinsertion operators are adapted from the repair operators proposed in Kovacs et al. (2015) to insert removed customer visits back to the partial solution after removal:
(1) `GreedyInsertion`: This operator starts with identifying the set of customers whose required visits are removed from the solution on at least one days and their total number of removed visits. The sequence in which these customers are considered for reinsertion is random and all removed visits to a customer are reinserted back to the solution at once if the customer is chosen for reinsertion. If the customer i being considered for reinsertion into m does not have all required visits removed, this operator will first identify the vehicle k that operates the un-removed visits and check whether it is feasible to use k to fulfill removed visits with respect to vehicle capacity and maximum route duration constraints. If feasible, all removed visits to i will be inserted at the best position in k with least travel distance increase on corresponding days. Otherwise, all un-removed visits to i are removed from m and all remaining vehicles are checked for feasibility to insert all required visits to i with respect to vehicle capacity and maximum route duration constraints. All required visits to i will be inserted into the feasible vehicle k^* with least travel distance increase on corresponding days. Note that among empty vehicles, only the one with the smallest index will be considered in this step. If the customer i being considered has all its required visits removed from the solution m, all vehicles are checked for feasibility to insert all required visits to i regarding vehicle capacity and maximum route duration constraints. The feasible vehicle k^* with smallest travel distance increase across all days will be selected for visiting i.
A variant of the above operator is obtained by reducing the vehicle capacity artificially during reinsertion (Kovacs et al., 2015). Specifically, the new vehicle capacity will be chosen uniformly in the interval $[\max\{\max q_{id}, Q/2\}, Q]$. The variant works the same with the original operator except that the new artificial vehicle capacity is used to determine

insertion feasibility.

(2) `RegretInsertion`: this operator starts with identifying the set of customers Γ_p whose required visits are partially removed from the solution m, and those customers Γ_a whose required visited are all removed from m. The operator will try to reinsert removed visits to customers in Γ_p first and the sequence in which customers in Γ_p are considered is random. For the customer i being considered for insertion, the vehicle k that operates the un-removed visits to i will be identified and checked for feasibility of inserting all removed visits with respect to vehicle capacity and maximum route duration constraints. If feasible, all removed visits to i will be inserted at a position in k with least travel distance increase on corresponding days. Otherwise, all un-removed visits to i are removed from m and i is inserted into set Γ_a. After all customers in Γ_p are processed, customers in Γ_a will be considered for insertion into m by taking into account the loss that might arise if reinsertion of visits to a customer is delayed into future iterations. For each customer, this operator checks all available vehicles in solution m for feasibility of inserting all the required visits to this customer on corresponding days. The total travel distance increase σ_i^k of inserting all required visits to customer i into a feasible vehicle k is noted. All the feasible vehicles are sorted in non-decreasing order of σ_i^k. That is, let $s(i,h)$ for $h = 1, \ldots, K$ denote the vehicle that appears in the h^{th} position in the sorted list associated with customer i, such that

$$\sigma_k^{s(i,1)} \leq \sigma_k^{s(i,2)} \leq \cdots \leq \sigma_k^{s(i,|K|)}. \tag{4.50}$$

Then customer i^* is selected according to

$$i^* \in \arg\max_{i \in N} \{ \sum_{h=2}^{\min\{b,o\}} (\sigma_i^{s(i,h)} - \sigma^{s(i,1)}) \}. \tag{4.51}$$

Parameter b defines the number of feasible vehicles that are considered in the regret operator and o denotes the total number of available vehicles. All the required visits to i^* will be inserted into the vehicle associated with σ_i^1 in the previously ordered list at the best

positions on corresponding days. This process repeats until all visits to customers in Γ_a are reinserted into m. Four possible b values are used in the algorithm, namely, 2, 3, 4, o.

4.4.3.3 Crossover

The crossover operator works in the way of removal and reinsertion described in the LNS. With two parent solutions m_1 and m_2, the child solution m_c is initialized as a copy of m_1 and all required visits to customers in m_c whose time consistency is worse than that of m_2 are removed. The first version of the previously described greedy reinsertion is used to reinsert removed visits back to the partial solution.

4.4.4 Column generation framework

This section describes the column generation process at each tree node of the B&P algorithm. Note that a set covering model given in (4.18)–(4.21) is defined at each tree node P based on its underlying graph $\mathcal{G}(P)$ and its LP relaxation given in (4.22)–(4.25), denoted by $MP(\Omega)$, is solved. Since only a subset Ω_1 of Ω is available, the restricted master problem given in (4.26)–(4.29) is solved and new columns are generated if possible. This column generation process repeats until no more columns can be identified, which indicates the optimality of $MP(\Omega)$. Algorithm 7 shows the framework of the column generation process. `ColGenExact(`λ`)` solves (4.35)–(4.49) exactly using CPLEX. Likewise, `ColGenHeurAlg()` generates new columns by solving (4.35)–(4.49) heuristically. Every time the RMP is solved, the `ColGenHeurAlg()` is applied first to identify new column(s) with negative reduced cost. The `ColGenExact(`λ`)` will be employed if the `ColGenHeurAlg()` fails to find new column(s).

Algorithm 7 Workflow of `ColGen`(P_0)

Input: Ω_1 the starting set of columns
solve $RMP(\Omega_1)$ and get the dual values λ associated with each customer
repeat
    use `ColGenHeurAlg()` to generate new columns Ω_h and set $\Omega_1 = \Omega_1 \cup \Omega_h$
    if Ω_h is empty, use `ColGenExact(`λ`)` to find new column Ω_e and set $\Omega_1 = \Omega_1 \cup \Omega_e$
    if either Ω_h or Ω_e is not empty **then**
        solve $RMP(\Omega_1)$ and get its dual values
    end if
until both Ω_h and Ω_e are empty

4.4.5 Column generation heuristic

This section describes the `ColGenHeurAlg()` that is used to generate new column(s) with negative reduced cost in the column generation process.

4.4.5.1 Algorithm framework

Algorithm 8 shows the framework of the LNS. It starts with an initial column v and generates new columns in subsequent iterations using a set of designed operators. The new column replaces the current column with certainty if it has better objective value. A worse column can also be accepted as current column with a given aspiration probability. The algorithm stops after a maximal number of iterations and all feasible columns with negative reduced cost encountered during the search process will be output.

Algorithm 8 Large neighborhood search

Require: optimal dual values of the RMP, maximal number of iterations η^{max}, aspiration probability p^a, operator performance update interval U
create an initial column v and empty set Ψ, let $\Psi = \Psi \cup v$ if v is feasible and $r_v < 0$
let $v^{curr} = v,\ v^{best} = v$, $\eta = 0$
repeat
    choose an operator δ
    apply operator δ on v^{curr} to create a new column v^{new} and compute its objective value $r_{v^{new}}$
    if $r_{v^{new}} < r_{v^{curr}}$ **then**
        $v^{curr} = v^{new}$
        let $v^{best} = v^{curr}$ if v^{new} is feasible and $r_{s^{new}} < r_{s^{best}}$
        let $\Psi = \Psi \cup v^{new}$ if v^{new} is feasible and $r_{s^{new}} < 0$
    else
        let $v^{curr} = v^{new}$ with probability p^a
    end if
    update operator performance every T iterations
    let $\eta = \eta + 1$
until $\eta > \eta^{max}$
return column set Ψ

4.4.5.2 Starting column generation and evaluation

To generate the initial column v, a customer is randomly selected and all its required visits are inserted into an empty column on corresponding days. The next customer to be included in the column is determined by computing for each remaining customer its reduced cost increase if its required visits are inserted at the end of the route on corresponding days and identifying the customer with the smallest reduced cost increase. The reduced cost increase of a customer is defined as the total travel cost increase across all days minus its optimal dual value obtained from RMP. All required visits of the identified customer will be inserted at the end of the route on corresponding days. This customer identification and visit insertion process continue until no more customer can be inserted into the column feasibly.

To evaluate a column, its total travel distance f_{TD} across all days on the planning horizon is first noted. The optimal dual values associated with all customers included in the

column are then subtracted from f_{TD}. In addition, three penalty factors, namely, p_c, p_d and p_{tc}, are defined to account for infeasibility caused by violations of vehicle capacity, maximal route duration and time consistency constraints. A column v is evaluated using the following formula

$$\begin{aligned} r_v = f_{TD}(v) - \sum_{i \in v} \lambda_i \\ + p_c \sum_{d \in D} \max\left(Load_d - Q, 0\right) \\ + p_d \sum_{d \in D} \max\left(Dura_d - T, 0\right) \\ + p_{tc} \max\left(TC - L, 0\right) \end{aligned}$$

where $Load_d$ and $Dura_d$ are the load and duration of the route on day d in the column, and TC is the maximum arrival time differential among all customers in the column.

4.4.5.3 Large neighborhood search operators

There are four types of operators that can be used to generate a new column based on an input column: (1) removal, (2) insertion, (3) swap and (4) removal and reinsertion.

(1) Removal: Removal operators remove visits to selected customers from a column without reinserting them. Six different removal operators are defined:

- `rm_rand`: this operator randomly selects a customer currently in the column and removes all its required visits from the column.
- `rm_rand_prob`: in this operator, every customer in the column has a certain probability ρ to be selected and all required visits to selected customers will be removed from the column. In case that no customer is selected, a customer is randomly chosen and all its required visits are removed from the column.

- `rm_reduced_cost`: this operator first computes for each customer in the column its reduced cost contribution $\bar{c}_i$ which is defined as the total travel cost increase for visiting this customer on all its required days minus its dual value. The customers in the column are then sorted in list L in decreasing order of $\bar{c}_i$ and the customer $L[\lfloor y^g|L|\rfloor]$ is selected. Note that y is a random number in $U(0,1)$ and g a parameter that controls the degree of randomization. Therefore, customer with greater $\bar{c}_i$ will have higher probability to be chosen. All required visits to the chosen customer are removed from the column.

- `rm_worst_tc`: in this operator, the customer in the column that has the biggest arrival time differential across the planning horizon is identified and all its required visits are removed from the column.

- `rm_isolated_cus`: this operator computes for each customer in the column the total travel cost increase for visiting this customer on all its required days. The customers in the column are then sorted in list L in decreasing order and the customer $L[y^g|L|]$ is selected. Note that y and g are calculated as in `rm_reduced_cost`. All required visits to the chosen customer are removed from the column.

- `rm_cluster`: this operator groups all the customers in the column into two clusters using Kruskal's algorithm to find the minimum spanning tree among them and deleting the longest edge in the tree. One of the two clusters is chosen randomly and all required visits to customers in the chosen cluster are removed from the column.

- `rm_taboo_arc`: this operator randomly selects a customer whose connecting arc is declared taboo after branching and removes all its required visits from the column.

Note that the `rm_cluster` only applies to columns with at least three customers while the other removal operators apply to columns with at least two customers. This is to avoid generating empty columns. The `rm_taboo_arc` is only used for Restricted Master Problems after branching.

(2) Insertion: These operators reinsert customers currently not in the column. In the operators described below, the best insertion position for a customer c into a route k is defined as the position on k such that the travel cost increase is minimized if c is inserted into the route. Three different insertion operators are defined:

- `is_rand`: this operator randomly selects a customer not in the column and inserts all its required visits into the column on corresponding days. A customer visit is always inserted into the route at the best insertion position.
- `is_reduced_cost`: this operator first computes for each customer currently not in the column its reduced cost contribution $\bar{c}_i$ if all the required visits to this customer are inserted into the column on corresponding days. Parameter $\bar{c}_i$ is computed as the total travel cost increase associated with the best insertion positions on corresponding days minus the dual value of the customer being inserted. The customers not in the column are then sorted in list L in increasing order of $\bar{c}_i$ and the customer $L[y^g|L|]$ is selected. y is a random number in $U(0,1)$ and g a parameter that controls the degree of randomization. All required visits to the chosen customer are inserted into the column at its best insertion positions on corresponding days.
- `is_min_dist`: this operator first chooses a customer randomly from the column and identifies its nearest customer that is not in the column. All the required visits to this nearest neighbor customer will be inserted into the column at the best insertion positions. on corresponding days.

(3) Swap: These operators replace one customer that is currently in the column with another customer that is not currently in the column. Three swap operators are defined:

- `sw_rand`: this operator first removes all required visits to a randomly chosen customer in the column and then reinserts all required visits to a randomly chosen customer

not in the column. That is, `sw_rand` consists of applying `rm_rand` and `is_rand` in succession.

- `sw_reduced_cost`: this operator first follows the same procedure defined in `rm_reduced_cost` to find a customer i to insert into the column. The customer j to be removed from the column is identified using the same procedure given in `rm_reduced_cost`. All visits to the identified customer j are first removed from the column, followed by insertion of all required visits to customer i into the column at the best insertion positions on corresponding days.

- `sw_min_dist`: in this operator, a customer i is randomly chosen from the column and its nearest neighbor j that is not in the column is identified. All of the customer's required visits are first removed from the column on corresponding days, followed by insertion of all required visits to customer j into the column at its best insertion positions on corresponding days.

- `sw_taboo_arc_rcost`: this operator first identifies all the customers in the column that have connecting arcs that are declared as taboo after branching. Their corresponding reduced costs are noted and the customer i with the biggest reduced cost is chosen. The customer j that is to enter the column is the one with the smallest reduced cost if all its required visits are inserted into the column. All required visits of i are removed, followed by insertion of visits to customer j into the column.

- `sw_taboo_arc_mdist`: this operator selects the customer i to be removed as defined in `sw_taboo_arc_rcost`. The nearest neighbor j of customer i that is not in the column is to enter the column. All required visits of i are removed, followed by insertion of visits to customer j into the column.

(4) Removal-reinsertion: These operators remove a number of customer visits from the

column and then reinsert them to the column. Seven removal operators are defined:

- `rr_rand_day`: in this operator, each vehicle route on each day is checked separately. Customer visits on a route are subject to removal with a given probability ρ.
- `rr_rand_cus`: in this operator, customers in the column are checked sequentially and marked as to be removed with a given probability ρ. All required visits to determined customers will be removed from the column.
- `rr_worst_tc`: this operator removes all required visits to the customer in the column that has the biggest arrival time differential across the planning horizon.
- `rr_worst_dual`: this operator removes all required visits to the customer in the column that has the biggest dual value.
- `rr_worst_reduced_cost`: this operator removes all required visits to the customer in the column that has the biggest reduced cost contribution $\bar{c}_i$.
- `rr_related`: this operator takes as input the percentage of customer visits to be removed from the column and determines the total number u of customer visits to be removed. This operator aims to remove customers with similar characteristics. The similarity of two customers i and j is computed using equation given in Kovacs et al. (2015). The related removal operator starts with a randomly selected customer in the column and removes all its required visits. The customer is then added to set S. In following iterations, a customer is randomly chosen from S and similarity values between the chosen customer and all other customers in the column are computed. The customer with the smallest $\mathcal{R}(i, j)$ is chosen as the next customer to be removed. The removal process continues until at least u removals have been made.
- `rr_cluster`: the same customer clustering approach given in `rm_cluster` is used to identify a set of customers and all the required visits to the identified customers are removed from the column.

- `rr_taboo_arc_day`: this operator checks the vehicle route on each day separately and removes all customers whose incoming arcs are declared taboo after branching.
- `rr_taboo_arc_cus`: this operator removes all customers from the column whose connecting arcs are declared taboo on any day of the planning horizon.

Two reinsertion operators are defined:

- `rr_insert_day`: In this operator, the reinsertion of removed customer visits to a column is conducted for each day separately. On a day, the order in which the removed visits are reinserted is randomized. For a visit to be reinserted, the operator identifies its best insertion position on the route that has the smallest travel cost increase.
- `rr_insert_cus`: in this operator, the reinsertion of removed customer visits is carried out for each customer sequentially. The order in which customers are considered for reinsertion is randomized. For a customer who has removed visits on one or more days of the planning horizon, its removed visits are reinserted back to the column on each day sequentially. For a visit to be reinserted, the operator identifies its best insertion position on the route that has the smallest travel cost increase.

Note that any of the seven removal operators can be paired with one of the reinsertion operators, which creates a total of fourteen removal-reinsertion operators.

4.4.5.4 Operator selection

At each iteration of the proposed heuristic algorithm, a neighborhood search operator is selected from the four types of operators. Operator selection is based on the historical performance of each individual operator. Specifically, the total number of times an operator is selected and the total number of times it successfully replaces the current column are recorded. The quotient of these two numbers is set to be the success rate of the

operator. The initial success rates of all operators are set to 1.0 and they can not drop below 0.20 at each performance update, this is to ensure that every operator has a positive probability to be selected.

The operator selection consists of two steps: operator type selection and operator selection. In the first step, the average operator success rate is computed for each operator type and the roulette wheel method is used to select the operator type. Operator types with greater average success rates have higher chance to be chosen. In the second step, an operator is selected from the determined type using the roulette wheel selection method with the individual success rate in the operator type.

Note that only removal operators and swap operators are considered for columns with vehicle capacity constraint violations. These two types of operators help to restore feasibility. All four types of operators can be applied on a column without vehicle capacity constraint violations. The operator success rate is recomputed every T iterations.

4.4.6 Branching rule

At a tree node P_x, branching is required if there exist fractional values in the obtained optimal solution from the column generation process. To branch, all columns appearing in the optimal solution are used to construct the original x_{ijkd} variable. To this end, let $\theta^* = \{\theta^*_1, \ldots, \theta^*_{|V^*|}\}$ denote the optimal values associated with the optimal column set V^*. Then, the x_{ijkd} can be computed as $\sum_{v \in V^*} x^v_{ijkd} \theta^*_v$ where x^v_{ijkd} equals 1 if arc (i, j) is traveled by vehicle k on day d in column v, and equals 0 otherwise.

For every arc (i, j, k, d) with fractional x_{ijkd} value, the value $b = \min(x_{ijkd}, 1 - x_{ijkd}) * t_{ij}$ is computed. All such (i, j, k, d) arcs are sorted in non-increasing order of b and an arc (i^*, j^*, k^*, d^*) with smallest b value is used to branch. After branching, two child tree nodes are generated, namely, P_l and P_r, and the graph associated with each tree node must be modified accordingly. In P_l, the arc (i^*, j^*, k^*, d^*) must be used and the underlying graph $\mathcal{G}_l$ is defined as

- if $i^* = 0$ and $j^* \neq 0$, then $\mathcal{G}_l = \mathcal{G}_{P_x} \setminus \{(i, j^*, k^*, d^*), \forall i \in \mathcal{N}^0, i \neq i^*\}$
- if $i^* \neq 0$ and $j^* = 0$, then $\mathcal{G}_l = \mathcal{G}_{P_x} \setminus \{(i^*, j, k^*, d^*), \forall j \in \mathcal{N}^0, j \neq j^*\}$
- if $i^* \neq 0$ and $j^* \neq 0$, then
  $\mathcal{G}_l = \mathcal{G}_{P_x} \setminus \{\{(i, j^*, k^*, d^*), \forall i \in \mathcal{N}^0, i \neq i^*\} \cup \{(i^*, j, k^*, d^*), \forall j \in \mathcal{N}^0, j \neq j^*\}\}$

In P_r, the arc (i^*, j^*, k^*, d^*) must not be used and the underlying graph $\mathcal{G}_r$ is defined as $\mathcal{G} \setminus (i^*, j^*, k^*, d^*)$. Note that the subproblem of a tree node is defined on its underlying graph. For example, any feasible column of P_l must contain arc (i^*, j^*, k^*, d^*) and any feasible column of P_r must not contain arc (i^*, j^*, k^*, d^*). After branching, the master problem $MP(\Omega)$ (defined in (4.22)–(4.25)) and the restricted master problem (defined in (4.26)–(4.29)) associated with a tree node P are then defined on the updated graph $\mathcal{G}_P$.

4.5 Computational experiments

This section describes the computational experiments conducted to verify the performance of the proposed branch-and-price algorithm. Both MIP1 and MIP2 are solved in CPLEX 12.6 using C++ Concert Technology and B&P is also implemented in C++. A time limit of one day is set for all algorithms. For the procedure `FindInitialCols(`$|M|$`,` I_{max}`)`, the population size $|M|$ is set to 200 and the maximum number of non-improving iterations I_{max} is set to 20. The LNS is applied with probability 0.05. For the column generation heuristic, maximal number of iterations η^{max} is set to the product of the instance size and 50000. The operator performance update interval T is set to 200. These parameters are set based on experience and no attempt is made to find the best parameter values.

4.5.1 Existing instances

Ten small-sized instances are taken from Groër et al. (2009) to test the performance of the proposed B&P algorithm. Five of them have ten customers and the other five have twelve customers. Both coordinates of all customer locations are generated randomly following a

continuous uniform distribution $U(0, 10)$ and the depot is located at $(0, 0)$. The planning horizon is set to three days and customers require service on each day with probability 0.7. Customer demand is uniformly distributed in $[1, 3]$ and all service duration are set to one unit. The maximum travel time limit is $T = 35$ and vehicle capacity is $Q = 15$. The maximum arrival time differential is $L = 5$.

Table 4.1 shows the computational results comparison between the MIPs and the B&P algorithm. The first column gives the number of customers in each instance. The second column indicates the instance number for each instance size. The next six columns show for MIP1 and MIP2 their corresponding best integer solution (column **Best**), the relative percent optimality gap (column **Gap**) and time used in seconds (column **Time**). The last five columns report for the B&P the best integer solution (column **Best**), the lower bound (column **LB**), the relative percent optimality gap (column **Gap**), the time used in seconds (column **Time**) and the indicator of whether the corresponding instance requires branching at the root node (column **Branch**). The last row provides the averages across all ten instances.

It can be seen from the table that all the three compared algorithms can find the optimal solutions, but MIP1 fails to prove optimality of the obtained solutions for two twelve-customer instances. MIP2 outperforms MIP1 for all instances with respect to computational times and significant reduction in the total time required to prove optimality is observed for all instances. Take the instance 1 with 12 customers for example. MIP1 stops at a 30.47% gap after a time limit of one day, while MIP2 proves optimality within 962 seconds. On average, MIP1 returns a 6.28% gap for this set of benchmark instances with an average runtime of 27884 seconds. On the other hand, MIP2 is able to prove optimality for all instances within 669 seconds.

Table 4.1 also proves the superior performance of the proposed B&P algorithm when compared to MIP1 in the literature. First, the B&P is able to prove optimality for all instances within the time limit of one day. Second, the time required for the B&P to prove

optimality is less than that of MIP1 for six out of ten instances. Last, the B&P uses an average time of 6534 seconds to close the gap for all instances, while MIP1 takes 27884 seconds to reach an average gap of 6.28%.

It can be observed from Table 4.1 that, while both MIP2 and the B&P can prove optimality for all instances, MIP2 outperforms the B&P for eight out of ten instances with respect to computational times used. On average, the times required for the B&P to prove optimality, 6534 seconds, are ten times more than that of MIP2, which is 669 seconds. It can be seen from the last column that nine out of the ten instances require branching at the root node. The instance 2 with 12 customers takes the least time to prove optimality among all the benchmark instances.

Table 4.1: Computational results on literature instances

size	id	MIP1			MIP2			B&P				
		Best	Gap	Time	Best	Gap	Time	Best	LB	Gap	Time	Branch
10	1	122.032	0.00%	203	122.032	0.00%	34	122.032	122.032	0.00%	232	YES
	2	99.069	0.00%	470	99.069	0.00%	9	99.069	99.069	0.00%	4595	YES
	3	123.411	0.00%	9006	123.411	0.00%	380	123.411	123.411	0.00%	2707	YES
	4	126.886	0.00%	1904	126.886	0.00%	18	126.886	126.886	0.00%	2904	YES
	5	109.313	0.00%	2535	109.313	0.00%	48	109.313	109.313	0.00%	245	YES
12	1	145.025	30.47%	86400	145.025	0.00%	962	145.025	145.025	0.00%	786	YES
	2	89.5416	0.00%	3583	89.5416	0.00%	82	89.5416	89.5416	0.00%	216	NO
	3	119.687	37.70%	86400	119.687	0.00%	507	119.687	119.687	0.00%	47437	YES
	4	141.37	0.00%	84114	141.37	0.00%	4753	141.37	141.37	0.00%	1071	YES
	5	117.418	0.00%	4528	117.418	0.00%	195	117.418	117.418	0.00%	5145	YES
Average		119.375	6.28%	27884	119.375	0.00%	669	119.375	119.375	0.00%	6534	-

4.5.2 New instances

To further compare the performance of the three algorithms, we randomly generated a new set of instances using the same procedure and parameters given in Groër et al. (2009). The parameter values are used to generate the benchmark instances tested in Section 4.5.1. Ten instances are generated for each problem size. Tables 4.2 and 4.3 show the computational results on the newly generated instances (using a 24-hour time limit). In these tables, the first and second columns give the instance size and number, respectively. The best integer solution, the relative percent optimality gap and the time used in seconds are reported for

MIP1 and MIP2 in the next six columns. The best integer solution, the lower bound, the relative percent optimality gap, the time used in seconds and the indicator of whether branching is required at the root node are given for the B&P. The last row in Table 4.3 gives the averages across all instances.

It can be seen from Table 4.2 that MIP2 outperforms MIP1 for all instances with ten or twelve customers. Although both MIP1 and MIP2 can find the optimal solutions for all these instances within the time limit, MIP1 fails to prove optimality for five instances with twelve customers, while MIP2 is able to close the gap for all these instances. Also MIP2 achieves significant reduction in computational times required to prove optimality for these instances. Take the instance four with twelve customers as example. MIP1 requires 37793 seconds to prove optimality, while MIP2 only needs 107 seconds to close the gap, which is a 99.72% reduction in computational time. For the ten instances with fourteen customers, MIP2 outperforms MIP1 for seven instances. MIP1 and MIP2 perform the same for the rest three instances with fourteen customers.

For all of the instances with more than fourteen customers, MIP2 is able to achieve smaller gaps than MIP1. Take the instance one with sixteen customers as example. MIP2 reaches a 34.24% gap after the time limit of one day, while MIP1 achieves a 55.23% gap for the same instance. However, MIP1 outperforms MIP2 with respect to the best integer solution found within the time limit. On average, MIP1 takes an average time of 66940 seconds to achieve a 38.75% gap and the average best integer solution objective value is 155.752, while MIP2 requires an average time of 49549 seconds to reach a 17.35% gap with an average best integer solution objective value of 152.347. In conclusion, compared to MIP1, MIP2 requires 25.98% less time to achieve a 21.40% reduction in optimality gap and a 2.18% improvement in the best integer solution objective value.

Table 4.2 also reveals that the proposed B&P algorithm outperforms MIP1 in six out of ten instances with 10 customers and is outperformed by MIP1 for the rest four instances with the same size. For all the instances with twelve to eighteen customers, the B&P algorithm

achieves superior performance to MIP1 with respect to the best integer solution objective value, the optimality gap and the total time requirement. Take the instance one with eighteen customers as an example. The B&P reaches a 7.02% gap within the time limit, while MIP1 has a 58.80% gap for the same instance. For the instances with twenty customers, the B&P is able to outperform MIP1 with respect to the best integer solution objective value. But the B&P fails to find the lower bound for four out of ten instances. However, for the six instances for which the B&P is able to find the lower bonds, the B&P achieves smaller gaps than MIP1. On average, the B&P requires an average time of 43517 seconds to achieve a 1.24% gap and the average best integer solution objective value is 151.166, while MIP1 takes an average time of 66940 seconds to achieve a 38.75% gap and the average best integer solution objective value is 155.752. In conclusion, compared to MIP1, the B&P requires 34.99% less time to achieve a 37.51% reduction in optimality gap and a 2.94% improvement in the best integer solution objective value.

Table 4.2 shows that MIP2 outperforms the B&P in nine out of ten instances with ten customers. While both of them can find the optimal solutions, MIP2 requires significantly less time for the majority of these instances. Similar comparative performance of MIP2 and the B&P is also observed for instances with twelve customers in which the B&P is outperformed by MIP2 in seven out of ten instances. However, for instances with fourteen customers, MIP2 is outperformed by the B&P in eight out of ten instances and the B&P is able to close the optimality gap for the majority of these instances. Furthermore, the B&P outperforms MIP2 for all instances with sixteen or eighteen customers. For the largest instances with twenty customers, the B&P still outperforms MIP2 whenever it can obtain the lower bounds. In the cases when the B&P fails to find the lower bounds, it is able to identify a better integer solution objective value in three out of four times. On average, compared to MIP2, the B&P requires 12.17% less time to achieve a 16.11% reduction in optimality gap and a 0.77% improvement in the best integer solution objective value. It can also be observed from Tables 4.2 and 4.3 that the B&P uses significant less time in

proving optimality when no branching is required at the root node.

4.6 Conclusion

This paper investigates methods for exact or near-exact solution of the consistent vehicle routing problem (ConVRP). We first present a MIP formulation for the ConVRP from the literature and then show how that formulation can be improved with new constraints. Testing with ordinary branch-and-bound shows the constraints introduce substantial benefits.

The main contribution of the paper is the development of a branch-and-price (B&P) algorithm using columns for routes serving the same customers across the time horizon. LP relaxations of partial master problems are solved over a subset of known columns, and a subproblem produces new columns of interest or demonstrates none exist. Heuristics are derived for all steps including creation of the first set of columns, generation of new ones as needed, and branching with the partial master optimum is fractional. Comparative computational testing over randomly generated instances paralleling available ones in the literature validates the merit of the (B&P) approach. Algorithm (B&P) outperforms the MIP methods on all but the smallest instances. Indeed it offers the only approach that is competitive for larger instances. In summary, MIP2 achieves the same or better performance than MIP1 for all instances and the B&P outperforms MIPs for medium size instances with fourteen to eighteen customers. Therefore, MIP2 can be used to solve small ConVRP instances quickly and B&P can be used to obtain a good near-optimal solution. In the future, heuristic or exact algorithms can be explored to solve the subproblems more efficiently. Also, cutting planes can be incorporated after the column generation process at each tree node to tighten the lower bound.

Table 4.2: Computational results on new instances

size	id	MIP1			MIP2			B&P				
		Best	Gap	Time	Best	Gap	Time	Best	LB	Gap	Time	Branch
10	1	136.619	0.00%	1694	136.619	0.00%	78	136.619	136.619	0.00%	2568	YES
	2	138.692	0.00%	1327	138.692	0.00%	40	138.692	138.692	0.00%	1250	YES
	3	87.4332	0.00%	134	87.4332	0.00%	100	87.4332	87.4332	0.00%	4199	YES
	4	132.993	0.00%	7626	132.993	0.00%	403	132.993	132.993	0.00%	2993	YES
	5	140.091	0.00%	4731	140.091	0.00%	48	140.091	140.091	0.00%	426	YES
	6	129.79	0.00%	2488	129.79	0.00%	53	129.79	129.79	0.00%	20	NO
	7	111.241	0.00%	3017	111.241	0.00%	157	111.241	111.241	0.00%	3044	YES
	8	154.778	0.00%	20061	154.778	0.00%	1560	154.778	154.778	0.00%	7834	YES
	9	143.801	0.00%	1404	143.801	0.00%	66	143.801	143.801	0.00%	3991	YES
	10	101.777	0.00%	3639	101.777	0.00%	82	101.777	101.777	0.00%	202	YES
12	1	120.98	44.92%	86400	120.98	0.00%	67	120.98	120.98	0.00%	50899	YES
	2	133.453	0.00%	15574	133.453	0.00%	884	133.453	133.453	0.00%	7802	YES
	3	122.56	40.41%	86400	122.56	0.00%	4379	122.56	122.56	0.00%	9656	YES
	4	122.249	0.00%	37793	122.249	0.00%	107	122.249	122.249	0.00%	3778	YES
	5	138.442	0.00%	11976	138.442	0.00%	2674	138.442	138.442	0.00%	380	YES
	6	137.97	44.15%	86400	137.97	0.00%	3686	137.97	137.97	0.00%	7681	YES
	7	132.653	41.15%	86400	132.653	0.00%	1442	132.653	132.653	0.00%	14720	YES
	8	140.046	0.00%	6276	140.046	0.00%	221	140.046	140.046	0.00%	1524	YES
	9	139.536	0.00%	10690	139.536	0.00%	178	139.536	139.536	0.00%	95	NO
	10	148.294	52.05%	86400	148.294	0.00%	3063	148.294	148.294	0.00%	406	NO
14	1	126.886	49.44%	86400	126.886	0.00%	387	126.886	123.045	3.12%	86400	YES
	2	164.356	51.94%	86400	155.225	23.16%	86400	155.225	155.225	0.00%	28750	YES
	3	159.065	40.04%	86400	159.065	40.04%	86400	158.358	158.358	0.00%	48107	YES
	4	151.269	54.76%	86400	151.269	25.55%	86400	151.269	151.269	0.00%	2989	YES
	5	137.578	52.69%	86400	137.578	0.00%	8659	137.578	137.578	0.00%	79035	YES
	6	140.214	46.74%	86400	140.214	0.00%	2986	140.214	140.214	0.00%	62102	YES
	7	158.209	49.31%	86400	158.209	49.31%	86400	155.025	155.025	0.00%	28001	YES
	8	152.033	44.93%	86400	152.033	44.93%	86400	152.033	152.033	0.00%	44147	YES
	9	127.475	45.11%	86400	127.476	0.00%	6388	127.709	121.931	4.74%	86400	YES
	10	143.865	42.57%	86400	143.865	0.00%	15392	143.865	143.865	0.00%	5540	YES
16	1	198.96	55.23%	86400	179.209	34.24%	86400	179.209	179.209	0.00%	59097	YES
	2	178.053	54.94%	86400	177.885	24.26%	86400	177.885	170.641	4.25%	76400	YES
	3	136.405	36.05%	86400	136.405	0.00%	31141	136.405	136.405	0.00%	10279	YES
	4	152.472	49.24%	86400	147.591	8.71%	86400	147.592	145.159	1.68%	86400	YES
	5	148.146	51.36%	86400	148.146	23.01%	86400	148.146	146.666	1.01%	86400	YES
	6	134.328	45.04%	86400	132.065	24.25%	86400	132.065	132.065	0.00%	198	NO
	7	124.001	50.05%	86400	124.001	0.00%	56788	124.001	124.001	0.00%	295	NO
	8	117.696	52.36%	86400	116.999	0.00%	67097	116.999	116.999	0.00%	356	NO
	9	156.827	53.90%	86400	161.093	27.43%	86400	156.827	156.827	0.00%	3586	YES
	10	164.275	59.52%	86400	163.483	22.79%	86400	163.483	148.513	10.08%	86400	YES
18	1	207.639	58.80%	86400	210.136	40.81%	86400	207.639	194.016	7.02%	86400	YES
	2	196.964	58.14%	86400	173.886	32.97%	86400	167.26	167.26	0.00%	11782	YES
	3	159.143	58.04%	86400	160.041	43.96%	86400	157.759	153.789	2.58%	86400	YES
	4	205.657	66.41%	86400	180.916	33.69%	86400	180.916	180.673	0.13%	86400	YES
	5	140.571	49.25%	86400	138.916	30.79%	86400	138.84	136.544	1.68%	86400	YES
	6	142.59	51.24%	86400	137.623	40.44%	86400	135.535	128.63	5.37%	86400	YES
	7	148.873	46.84%	86400	147.906	16.06%	86400	147.906	144.741	2.19%	86400	YES
	8	152.064	56.19%	86400	152.064	33.55%	86400	152.064	142.838	6.46%	86400	YES

Table 4.3: Computational results on new instances

size	id	MIP1			MIP2			B&P				
		Best	Gap	Time	Best	Gap	Time	Best	LB	Gap	Time	Branch
20	1	198.336	54.44%	86400	217.817	46.55%	86400	198.336	190.047	4.36%	86400	YES
	2	219.878	63.71%	86400	205.274	52.44%	86400	200.339	NA	NA	86400	NA
	3	207.173	60.02%	86400	203.635	38.35%	86400	203.635	199.632	2.01%	86400	YES
	4	208.82	56.99%	86400	206.609	42.42%	86400	198.384	NA	NA	86400	NA
	5	217.151	59.99%	86400	220.95	38.25%	86400	217.151	213.058	1.92%	86400	YES
	6	208.622	54.84%	86400	209.872	31.75%	86400	208.622	194.345	7.35%	86400	YES
	7	178.283	60.31%	86400	135.464	18.95%	86400	135.464	NA	NA	86400	NA
	8	199.08	50.59%	86400	203.1258	42.84%	86400	199.08	NA	NA	86400	NA
	9	200.023	58.58%	86400	162.399	32.68%	86400	162.399	159.246	1.98%	86400	YES
	10	215.525	53.01%	86400	210.307	37.00%	86400	210.307	210.307	0.00%	82589	YES
Average		155.752	38.75%	66940	152.347	17.35%	49549	151.166	146.795	1.24%	43517	-

Reference

Kris Braekers and Attila A. Kovacs. A multi-period dial-a-ride problem with driver consistency. Transportation Research Part B: Methodological, 94:355–377, 2016.

Marilne Cherkesly, Guy Desaulniers, Stefan Irnich, and Gilbert Laporte. Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and multiple stacks. European Journal of Operational Research, 250(3):782–793, 2016.

Jamison M. Day, P. Daniel Wright, Tobias Schoenherr, Munirpallam Venkataramanan, and Kevin Gaudette. Improving routing and scheduling decisions at a distributor of industrial gasses. Omega, 37(1):227–237, 2009.

Guy Desaulniers. Branch-and-price-and-cut for the split-delivery vehicle routing problem with time windows. Operations Research, 58(1):179–192, 2010.

Alan L. Erera, Martin Savelsbergh, and Emrah Uyar. Fixed routes with backup vehicles for stochastic vehicle routing problems with time constraints. Networks, 54(4):270–283, 2009.

Chris Groër, Bruce Golden, and Edward Wasil. The consistent vehicle routing problem. Manufacturing & Service Operations Management, 11(4):630–643, 2009.

Irina Ioachim, Jacques Desrosiers, Franois Soumis, and Nicolas Blanger. Fleet assignment and routing with schedule synchronization constraints. European Journal of Operational Research, 119(1):75–90, 1999.

Attila A. Kovacs, Bruce L. Golden, Richard F. Hartl, and Sophie N. Parragh. Vehicle routing problems in which consistency considerations are important: A survey. Networks, 64(3):192–213, 2014a.

Attila A. Kovacs, Sophie N. Parragh, Richard F. Hartl, and Sophie N. Parragh. A template-based adaptive large neighborhood search for the consistent vehicle routing problem. Networks, 63(1):60–81, 2014b.

Attila A. Kovacs, Bruce L. Golden, Richard F. Hartl, and Sophie N. Parragh. The generalized consistent vehicle routing problem. Transportation Science, 49(4):796–816, 2015.

Zhixing Luo, Hu Qin, ChanHou Che, and Andrew Lim. On service consistency in multi-period vehicle routing. European Journal of Operational Research, 243(3):731–744, 2015.

Diego Pecin, Artur Pessoa, Marcus Poggi, and Eduardo Uchoa. Improved Branch-Cut-and-Price for Capacitated Vehicle Routing, pages 393–403. Springer International Publishing, Cham, 2014.

David Russell, Robert J. Rosati, and Evie Andreopoulos. Continuity in the provider of home-based physical therapy services and its implications for outcomes of patients. Physical Therapy, 92(2):227–235, 2012.

Anirudh Subramanyam and Chrysanthos E. Gounaris. A branch-and-cut framework for the consistent traveling salesman problem. European Journal of Operational Research, 248 (2):384–395, 2016.

C. D. Tarantilis, F. Stavropoulou, and P. P. Repoussis. A template-based tabu search algorithm for the consistent vehicle routing problem. Expert Systems with Applications, 39(4):4233–4239, 2012.

Richard T. Wong. Vehicle Routing for Small Package Delivery and Pickup Services, pages 475–485. Springer US, Boston, MA, 2008.

Christel A. Woodward, Julia Abelson, Sara Tedford, and Brian Hutchison. What is important to continuity in home care?: Perspectives of key stakeholders. Social Science and Medicine, 58(1):177–192, 2004.

5. Conclusion and Future Research Directions

This thesis studies service consistency in the context of multi-period vehicle routing problems. Chapter 2 studies the impact of improving service consistency on the total travel cost in the context of multi-period vehicle routing problems using a multi-objective optimization approach. An improved multi-directional local search algorithm is proposed to approximate the Pareto frontier of the problem. The performance of the proposed algorithm is validated on a set of benchmark instances taken from the literature. Trade-off analysis using the technique of level diagram shows that approximately 60% and 75% improvement in driver consistency and time consistency can be achieved at the cost of 5% increase in total travel cost.

Chapter 3 considers service consistency in the context of the periodic vehicle routing problems for the first time in the literature. The trade-offs between the objectives of service consistency maximization and travel cost minimization are investigated using a multi-objective optimization approach. Various multi-objective optimization algorithms are employed to approximate the Pareto frontier of the problem. Competitive strengths of these algorithms are verified on a set of benchmark instances taken from the literature. Trade-off analysis is conducted on the super set of the non-dominated solutions obtained by all algorithms to facilitate managerial decision making. Computational results show that approximately 34% and 30% improvement in driver consistency and time consistency can be achieved at the cost of 9% increase in total travel cost.

Chapter 4 examines the impact of enforcing service consistency on the total travel cost in the context of the consistent vehicle routing problems. Service consistency is treated as hard constraints in a single-objective optimization framework. An improved mixed-integer programming formulation is proposed for this problem and a branch-and-price algorithm is developed. The problem is reformulated as a set covering problem and the associated subproblem is defined. An efficient subproblem heuristic is proposed to identify new columns with negative reduced costs quickly. The performance of the new mixed-integer

programming formulation and the branch-and-price algorithm is validated on a set of benchmark instances taken from the literature and a new set of randomly generated instances. Computational results show that the new mixed-integer program outperforms the existing one from the literature and the branch-and-price algorithm is able to obtain better gaps than mixed-integer programs for instances with more than fourteen customers. Studying the impact of enforcing strict service consistency in PVRP is a possible research direction. Chapter 3 represents a first stride in the literature to consider service consistency in the context of PVRP. It is worthwhile to evaluate the travel cost increase if strict driver consistency and time consistency is enforced in PVRP as in the ConVRP studied in Chapter 4, due to the wide applicability and versatility of PVRP. The Pareto frontier approximation obtained in Chapter 4 for the MoConPVRP may not contain a solution that satisfies the service consistency constraints defined in Chapter 4 and therefore can not be used to study the impact of enforcing service consistency on the travel cost. Efficient solution methods must be defined for this problem since PVRP is more constrained than ConVRP, mainly because the number of available vehicles is limited at the depot. Another research direction is to develop efficient exact solution methods for the generalized consistency vehicle routing problems in which multiple drivers are allowed to visit a customer across a planning horizon and time consistency is enforced as in the ConVRP.