A matheuristic for the inventory routing problem with divisible pickup and delivery

Einar Aastveit
Tuva Toftdahl Staver
Simen Tung Vadseth

Industrial Economics and Technology Management
Submission date: June 2018
Supervisor: Magnus Stålhane, IØT
Co-supervisor: Henrik Andersson, IØT

Norwegian University of Science and Technology
Department of Industrial Economics and Technology Management

Problem description

This thesis considers an inventory routing problem with divisible pickup and delivery (IRP-DPD). A customer may have both delivery and pickup demand in this problem type and is allowed to have separate servings for pickup and delivery. The last part is contrary to what is common and is what constitutes a so-called divisible option. The goal of this thesis is to develop a model formulation for the IRP-DPD and to find good and reliable solution methods for the problem. Two different solution methods are proposed and tested:

- An exact solution approach which is strengthened with valid inequalities and a branch and cut algorithm.
- An iterative matheuristic with a final improvement search.

The different solution approaches are optimized, evaluated and compared to each other.

Preface

This master thesis is the final work for our Master of Science degree at the Norwegian University of Science and Technology. The degree is written within the field of Applied Economics and Operations Research at the Department of Industrial Economics and Technology Management. The thesis was written during the spring semester of 2018 and is a continuation of our work for the project report in the fall semester 2017.

In this thesis, we introduce a model formulation to solve an inventory routing problem with divisible pickup and delivery. The exact solution approach is strengthened with valid inequalities and a branch and cut algorithm. Also, a matheuristic is developed in order to solve the problem for larger instance sizes.

We would like to thank our supervisors Magnus Stålhane and Henrik Andersson for their helpful feedback and valuable guidance. Their enthusiasm and knowledge in the problem have been motivating and of great importance for our work.

Trondheim, June 10th, 2018

[illegible] Simen Vadseth Tuva T. Staver

Einar Aastveit Simen Tung Vadseth Tuva Toftdahl Staver

Abstract

The management of *reverse logistics* has received growing attention in recent years due to new regulations, environmental aspects and an eagerness to obtain further operational benefits. Reverse logistics can be modeled as a closed-loop supply chain, where significant improvements can be made through inventory routing optimization.

This thesis considers an inventory routing problem with divisible pickup and delivery (IRP-DPD). A customer may have both delivery and pickup demand in this problem type and is allowed to have separate servings for pickup and delivery. The last part is contrary to what is common and is what constitutes a so-called divisible option. A single depot with a fleet of homogeneous vehicles is considered. The vehicles are restricted by capacity and a maximal duration for a route. An arc-flow formulation of the problem, formulated as a mixed integer linear program, is proposed as an exact solution approach. The formulation is strengthened with valid inequalities and an extended branch and cut algorithm is suggested. The branch and cut algorithm dynamically adds subtour cuts for each strong component of the solution that violates subtour elimination constraints. The exact approach with the extended branch and cut algorithm solves larger instances to optimality, produces lower dual gaps and finds feasible solutions for larger instances.

To solve larger instances, a matheuristic with a two-phase construction approach followed by an improvement search is proposed. To construct a solution the matheuristic decomposes the problem into an inventory problem and a routing problem. The constructed solution is further improved with a set of operators. The matheuristic embeds the construction and the improvement operators into an iterative scheme. When the iterative loop is terminated, a final improvement search is suggested. The matheuristic gives solutions with lower dual gaps than the exact method for all larger instances. It also finds good solutions faster and produces feasible solutions for instances where the exact method does not.

Sammendrag

Omvendt logistikk har fått økt oppmerksomhet i senere tid på grunn av nye forskrifter, miljøaspekter og for å oppnå driftsfordeler. Problemet kan modelleres som en lukket verdikjede, der betydelige forbedringer kan oppnås ved bruk av optimeringsteknikker for ruteplanlegging og lagerstyring.

Denne oppgaven tar for seg et problem der både ruteplanleggingen av kjøretøy og lagerstyring optimeres over en planleggingshorisont. Hos kundene skal varer både hentes og leveres. Problemet er formulert slik at en kunde kan ha to separate besøk hver dag, ett for henting og ett for levering, noe som utgjør et såkalt *divisible option*. Dette skiller seg fra tidligere modeller av problemet der det kreves at henting og levering gjøres samtidig. Problemet tar for seg et depot og en flåte av homogene kjøretøy med en gitt varekapasitet, samt en restriksjon knyttet til maksimal kjøretid. En *arc - flow* formulering av problemet er foreslått som en eksakt metode. Formuleringen styrkes med en serie gyldige ulikheter. I tillegg er en *branch and cut algoritme* foreslått. Algoritmen legger til subtur kutt for hvert sterke komponent i løsningen som bryter subtur-eliminasjonsrestriksjoner. Den eksakte metoden med denne algoritmen løser større instanser til optimalitet, gir lavere dual gap og finner lovlige løsninger for større instanser.

For å kunne løse problemet for større instanser, er det foreslått en *matheuristic*. For å konstruere en løsning dekomponeres problemet i et lagerproblem og et ruteproblem. Den konstruerte løsningen blir videre forbedret med et sett av operatorer. Heuristikken itererer mellom å konstruere en løsning og å forbedre denne løsningen. Når iterasjonen brytes gjøres et siste forbedringssøk. Heuristikken gir løsinger med lavere dual gap enn den eksakte metoden for alle større instanser. I tillegg finner den gode løsninger raskere og gir lovlige løsninger for instanser der den eksakte metoden ikke gjør det.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An inventory routing problem with pickup and delivery (IRP-PD) integrates pickup and delivery problems (PDPs) with inventory management (Archetti et al., 2017b). The problem is an extension of the inventory routing problem (IRP) where the supplier decides when to serve a customer, how much to serve in a given visit and how to combine customers into vehicle routes. The supplier develops a distribution strategy that both minimizes the distribution costs and the inventory holding costs of the customers. The entire supply chain benefits from such comprehensive planning. Suppliers can efficiently coordinate shipments made to different clients, while the customers do not need to allocate efforts to inventory control (Coelho et al., 2013). In an IRP-PD, the complexity of dealing with both pickup and delivery decisions is added to the problem. Here, the distribution and holding costs are minimized by letting the supplier coordinate pickups and deliveries to the customers over a planning horizon.

Real life examples of IRP-PDs can be found in *reverse logistics* where new products and returned goods are transported in the same vehicles. This happens for instance when products are distributed in reusable containers which later are collected, and when full bottles and cans are distributed, while empty bottles and cans are collected and recycled (Iassinovskaia et al., 2017). Lately, the reverse logistics field has gained increasing attention due to growing environmental awareness in many industries, both as a result of an increased sense of responsibility and as a consequence of being faced with new laws and regulations. When reverse logistics is done correctly, the amount of waste produced and energy consumed are reduced, which can lead to reduced costs for many supply chains and the society as a whole. This is why several countries have imposed regulations on recycling rates of materials and recovery goals for sales packaging in later years. The enlarged prevalence of reverse logistics due to these regulations has led to increased pickup demands at the end of many supply chains, which again has led to an increased flow of products back up the same supply chains. To make sure that this increased magnitude of backflow is treated efficiently, we need smart and powerful ways to model and solve problems in reverse logistics (Dethloff, 2001).

Our planet is still undergoing a considerable population growth and an increasing number of people are taking part in the middle class lifestyle and the mass consumerism that comes with it. McKinsey Global Institute predicts global household consumption to rise by more than $20 trillion dollars by the year 2025 compared to 2012 levels, where more than $14 trillion of this will come in emerging markets (Dobbs et al., 2012). This increase in consumption, even though it provides substantial economic growth, will put an increased strain on the world's natural resources. Moreover, the consequences of global warming are getting more severe and landfills all over the world are filling up (Hoornweg et al., 2013). Consequentially, we need to use our resources more wisely and the importance of recycling and reuse will increase. Thus, it is necessary to develop more efficient ways to recycle and reuse resources in order to meet the consumption of tomorrow. Our contribution to this is to further the development of optimization techniques in reverse logistics.

The focus area of this thesis is the inventory routing problem with divisible pickup and delivery (IRP-DPD). The most common versions of the problem only allows customers to have either pickup demand or delivery demand, and if they are allowed to have both demands the pickup and delivery must happen simultaneously. In IRP-DPDs, we allow a customer to be visited twice in the same time period, once for pickup and once for delivery, which is described as a divisible option. This can be advantageous since it allows for solutions that cannot be obtained with a simultaneous model. As an example, some customers may prefer that pickup and delivery happen at different times of the day, and it can therefore be beneficial for them to allow for multiple visits. Further, it can from a practical point of view be preferable for the supplier to have divisible solutions. This has to do with the need to ease merchandise handling within a vehicle. Although there are vehicles with several dropsides, most only have one door at the back. Consequently, if a vehicle is full and deliveries and pickups are performed simultaneously, quite a lot of rearranging within the vehicle could be required (Hoff et al., 2009). In addition, the larger solution space allows for solutions which may have lower total costs than in the simultaneous case. However, a divisible model may be considerably harder to solve than a model where pickup and delivery have to happen simultaneously. Yet, the more general structure of the divisible model easily adapts to other versions of the IRP with pickup and delivery. For instance, one can describe cases where customers only have either positive pickup or delivery demand. The model formulation can also be transformed into a simultaneous model, which might be beneficial for larger instances which are hard to solve. To the best of our knowledge this is the first attempt to integrate the divisible option into an IRP-PD. This has previously been done with great success in VRP-PDs.

The inventory routing problem with pickup and delivery is a problem with high complexity. It is therefore essential to develop solution methods appropriate for different instance sizes. Due to the latest development in hardware technology, many mixed integer linear programs can now be solved close to optimality or to optimality within an acceptable amount of time. For smaller instances, an exact solution method is therefore applicable. For larger instances, a tradeoff must be made between the accuracy of the solution and the solution time. The majority of earlier papers related to the problem type described in this thesis have used simple heuristics to solve the problem for larger instances. Though, motivated by the development of hardware technology, some researchers have started to design heuristics that incorporate phases where mixed integer linear programs are solved, named matheuristics. However, the literature on matheuristics related to IRPs with pickup and delivery is limited. It is therefore advantageous for the progression in research to develop a matheuristic for our problem. The increased importance of recycling and reuse also makes it a motivating topic to work with.

The thesis is organized as follows. In Chapter 2, we review relevant literature that exists on pickup and delivery problems, considering both VRPs and IRPs. Literature on improvement of solution methods, including branch and cut algorithms and a classification of matheuristics for IRPs, are also considered. In Chapter 3, we give a detailed problem description of the inventory routing problem with divisible pickup and delivery that is modeled in this thesis. Chapter 4 contains the model formulation used for the exact solution approach. The chapter also includes modeling choices and assumptions used for both the exact and heuristic approach. Chapter 5 presents how we develop the exact solution method further. Both valid inequalities and a branch and cut algorithm are presented. The branch and cut algorithm adds subtour elimination constraints dynamically. In Chapter 6, a matheuristic is presented and thoroughly described. Chapter 7 covers how the different instances are generated. The results of the computational study from both the exact and the heuristic solution method are presented in Chapter 8. Lastly, in Chapter 9, we conclude and make suggestions for further research.

Chapter 2

Literature review

Problems that integrate inventory routing problems with pickup and delivery have been of interest in later years. Though the number of papers related to this particular extension of the IRP is limited, there has been elaborate research on similar problems such as the vehicle routing problem with pickup and delivery. The purpose of this chapter is to present an overview of the literature related to IRP-PDs.

We begin by describing the classification of pickup and delivery problems in the literature in Section 2.1. Various distribution networks and different solution types for the problem are described. In Section 2.2 a selection of papers covering important aspects of the IRP-PD is presented. The papers are classified according to the dimensions of routing problems we consider most essential for this thesis. The classification provides a framework to evaluate differences and similarities between the problem presented in this thesis and prior research. Section 2.3 presents literature relevant for the development of branch and cut algorithms that add subtour cuts dynamically. Lastly, Section 2.4 presents literature relevant to heuristic solution methods for different routing problems.

2.1 Distribution networks for pickup and delivery problems

The distribution network of pickup and delivery problems (PDPs) depends on the movement of commodities between pickup nodes and delivery nodes. Berbeglia et al. (2007) distinguish and classify three PDP structures: Many-to-many (M-M), where the goods have multiple origins and multiple destinations, one-to-one (1-1), where each good has a single origin and a single destination, and one-to-many-to-one (1-M-1), a combination of the two. The structures are more complex than the 1-M structure commonly used in VRPs and IRPs. In reverse logistics, the 1-M-1 is a common structure. Delivery commodities are picked up at one origin and delivered to multiple destinations, while pickup commodities are picked up at multiple origins and delivered to one destination.The different structures are illustrated in Figure 2.1.

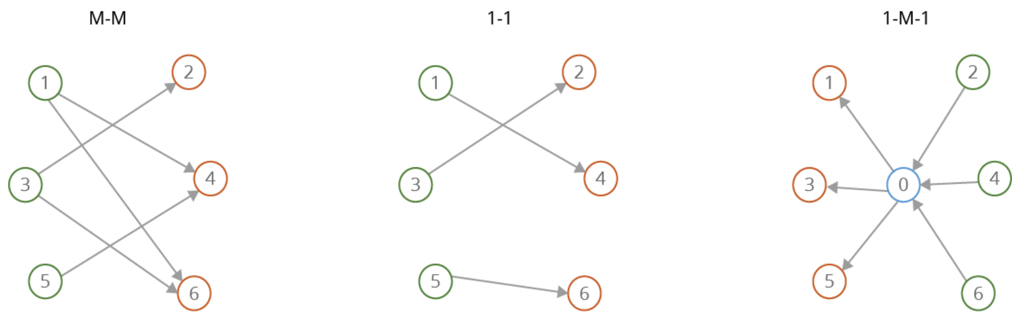


Figure 2.1: Classification of PDP structures

In PDPs, customers demand or supply commodities, or in some cases both. Problems where customers can be divided into two mutually exclusive sets; the pickup customers and the delivery customers are defined to have *single demand.* Delivery customers are also denoted as linehaul customers and pickup customers as backhaul customers. If linehauls and backhauls are allowed to occur in any order on a vehicle tour, the problem is classified as a *routing problem with mixed linehauls and backhauls* (RP-MB) (Parragh et al., 2008). *Combined demand* means that at least one customer has both positive demand and supply. Considering combined demand, Berbeglia et al. (2007) identify four solution types based on the order of pickup and delivery in the route and on how many visits the customer allows. The general solution enforces no restrictions on the order, and a customer can have up to two visits in a given time period, one for pickup and one for delivery. Note that it is still required that all of the delivery or pickup to a customer is made in a single visit. This problem type is classified as the *routing problem with divisible pickup and delivery* (RP-DPD) by Parragh et al. (2008). The lasso solution does not allow any single delivery visit after a pickup. That is, it requires all deliveries to happen before any pickup, but also allows multiple simultaneous pickup and deliveries. The same logic holds if pickups are performed first. The double-path solution is a special case of the lasso-solution with only one simultaneous pickup and delivery. The Hamiltonian solution type requires all pickups and deliveries to happen simultaneously. PDPs that enforces this solution type is also known in the literature as the *routing problem with simultaneous pickup and delivery* (RP-SPD). The different solution types are illustrated in Figure 2.2.

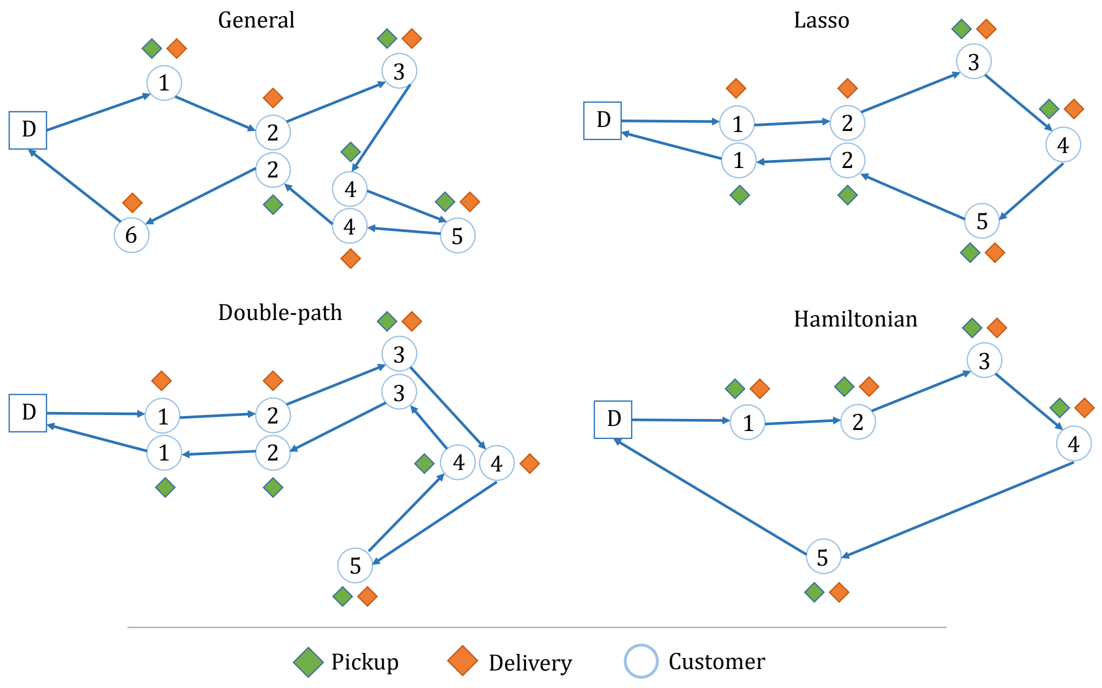


Figure 2.2: Solution types of a PDP problem with a 1-M-1 structure

2.2 Dimensions of the IRP-PD

The IRP-PD integrates inventory routing with pickup and delivery. The general nature of the problem makes it applicable in many contexts, and different papers cover various forms of it. Different dimensions help us to distinguish the characteristics of each variant and to align the problem presented in this thesis with prior research. We consider the following dimensions to be of greatest importance for the IRP-PD presented in this thesis:

- Problem type
- The structure of the distribution network
- The number of products handled
- Fleet size
- Time restrictions

Table 2.1 lists the papers presented in this literature review and classifies them according to the dimensions described above. The table presents a selection of papers and is not intended to be a comprehensive list of all relevant research. Due to the limited research on IRP-PD, the list does not present this problem type exclusively.

Table 2.1: Articles related to IRP-DPD

Paper	**Problem type**	**Structure**	**Products**	**Fleet size**	**TW**
Our problem	IRP-DPD	1-M-1	Dual	Multiple	No
Iassinovskaia et al. (2017)	IRP-SPD	1-M-1	Dual	Multiple	Yes
Van Anholt et al. (2016)	IRP-PD	1-M-M-1	Single	Multiple	No
Archetti et al. (2017b)	IRP-MB	M-M	Single	Single	No
Dethloff (2001)	VRP-SPD	1-M-1	Dual	Multiple	No
Privé et al. (2006)	VRP-SPD	1-M-1	Multiple	Unconstrained	No
Ai and Kachitvichyanukul (2009)	VRP-SPD	1-M-1	Dual	Unconstrained	No
Nagy et al. (2015)	VRP-DPD	1-M-1	Dual	Multiple	No
Hoff et al. (2009)	VRP-DPD	1-M-1	Dual	Multiple	No
Coelho and Laporte (2013)	IRP	1-M	Multiple	Multiple	No

Problem type

The problem type dimension distinguishes between IRPs and VRPs as well as problems with different solution types regarding the order of pickup and delivery. Iassinovskaia et al. (2017)

describe an IRP with simultaneous pickup and delivery in their paper. The paper considers a distributor located at a depot that has to distribute products packed in returnable transport items to a set of customers. The distributor also picks up empty returnable transport items and brings them back to the depot. A customer is represented with one single node that has separate inventories for loaded and empty returnable transport items. Nagy et al. (2015) describe a VRP with divisible pickup and delivery. The paper concludes that serving a customer twice, once for pickup and once for delivery, often reduces costs and the number of vehicles required. Notably, the benefit of splitting is more significant for customers with large differences between delivery and pickup demand and for customers located close to the depot or in clusters. Also any divisible visit are likely to be split across routes rather than having a single vehicle serving a customer twice in a period.

It is most common for IRP-SPDs to describe a customer as *one* customer node, with separate inventories for pickup and delivery and a binary variable related to both actions. The divisible option makes it hard to model the problem with only one customer node and therefore two customer nodes are proposed in the paper of Nagy et al. (2015). The paper of Hoff et al. (2009) considers a problem where breweries deliver mineral water bottles to a set of customers and collect empty bottles from the same customers. The problem allows divisible visits in order to postpone pickup until the vehicle has sufficient free capacity. The problem is formulated such that the vehicle should be partly unloaded before any empty bottles are picked up, inducing lasso solution types. In the article a tabu search heuristic is developed, capable of producing lasso and general solutions for the VRP-PD. The test results from Hoff et al. (2009) show that the general solution type outperforms other solution types in terms of cost, however the problem can be very time-consuming to solve. The best lasso solution generated within a given time limit is often better than the best general solution produced with the same computing effort for their VRP with pickup and delivery.

The structure of the distribution network

The articles related to a 1-M-1 structure are usually motivated from an environmental perspective. Both Iassinovskaia et al. (2017) and Ai and Kachitvichyanukul (2009) use this structure to solve a closed loop supply chain problem for returnable transport items. Similarly, Privé et al. (2006) consider the problem of delivering sodas and collecting empty bottles. All these papers assume simultaneous pickup and delivery. Other structures than 1-M-1 have also been researched, and an example is the paper of Archetti et al. (2017b) that describes an M-M problem with single demand. In this paper, a single commodity is made available at several pickup locations and consumed at several delivery locations, with one vehicle available for the distribution. The vehicle starts and ends its tour at the depot which also plays the role of a warehouse. It is also possible to combine structures, however, this may increase the complexity. As an example, Van Anholt et al. (2016) utilize both the 1-M-1 structure and the M-M structure in a recirculating automatic teller machine (RATM) problem. The 1-M-1 structure deals with the back and forth transportation from the depots to multiple RATMs. The M-M structure accounts for the redistribution of commodities between the RATMs.

The number of products handled

In an IRP the vendor handles the routing of either a single product, dual products, or multiple products. In the papers of Van Anholt et al. (2016) and Archetti et al. (2017b), a single commodity is redistributed among multiple nodes. Additional products complicate the problem but are often seen in real-life applications. A 1-M-1 structure often applies to reverse logistics and therefore dual products. With dual products, we mean that one type of product is delivered and another type picked up. Dethloff (2001) for instance, considers this twofold division of products in his paper where he focuses on reverse logistics. Further, there are situations where more than two products need to be handled. This is the case in both Privé et al. (2006) and Coelho and Laporte (2013), where the former considers the distribution and recycling of several different types of beverages.

Fleet size

In most land-based transportation problems, the fleet of vehicles consists of identical, capacitated homogeneous vehicles, where products are transported as cargoes that easily can be stocked together. Privé et al. (2006) however, consider a heterogeneous fleet where costs and capacity vary among different types of vehicles. All papers representing IRPs with pickup and delivery in Table 2.1 consider constrained fleets, where the use of a vehicle incurs no fixed cost. The costs are only dependent on travel time and distance, and not the number of vehicles used. Ai and Kachitvichyanukul (2009) on the other hand, add a fixed cost on the usage of each extra vehicle in their VRP-SPD. Consequently, there is an incentive to limit the number of vehicles used.

Time restrictions

Time can be regarded as continuous or discrete. All papers in Table 2.1 consider discrete time periods where vehicles are assigned closed loop routes for each time period. In a land-based IRP, it is common to require the vehicles to return to a depot at the end of each time period. Ai and Kachitvichyanukul (2009) and Privé et al. (2006) set an upper limit on the total duration of a route for a given time period. Iassinovskaia et al. (2017) impose time windows for each customer. This means that customers can only be visited at certain time intervals each time period. Instead of using time windows or restrict the problem with a total time length of a route, Van Anholt et al. (2016) introduce a variable representing overtime for a given vehicle in each time period. The overtime cost is a part of the objective function in their paper.

2.3 Branch and cut

In this section, we present and review literature that considers how branch and cut algorithms can improve the exact solution method. A branch and cut algorithm combines branch and bound with cutting planes to tighten the LP relaxations. Archetti et al. (2007) developed the first branch-and-cut algorithm to solve a single vehicle IRP according to Coelho et al. (2013). The paper describes a branch and cut algorithm where violated subtour elimination constraints are added to the current subproblem. The subproblem is then re-optimized. If no violated subtour elimination constraints are found, branching is performed. The paper also introduces a number of other valid inequalities. Experiments in the paper show that the dynamic handling of these inequalities slows down the execution, so these valid inequalities are only handled in the root node. Only subtour constraints are added dynamically. The branch and cut algorithm have also been used to solve problems with more than one vehicle. Coelho and Laporte (2013) solve a multi-product vehicle inventory routing problem using branch and cut. The algorithm is used together with a solution improvement algorithm with the purpose of improving the primal bound.

One of the most challenging parts when implementing branch-and-cut algorithms is the identification of violated subtour elimination constraints (SECs). This identification process is commonly referred to as separation. Both Archetti et al. (2007), Archetti et al. (2017b) and Coelho and Laporte (2013) use the separation procedure described by Padberg and Rinaldi (1991) to identify the connected components. This standard way identifies violated SECs by a maximum flow algorithm, but there also exist papers that describe alternative approaches. One example is the paper of Drexl (2013) that proposes a procedure based on computing strong components of a supported digraph. The method both has a better worst-case time complexity than the standard separation algorithm and is also easier to implement.

2.4 Heuristics

Most of the earlier papers describing IRPs use simple heuristics to reduce the complexity of the problem. These heuristics typically explore the solution space through the use of operators that change the solution. The solution space reachable by the operators is defined as the neighborhood of the solution. Decomposition of the problem into subproblems is also done, where the solution to a subproblem is used in the next step (Coelho et al., 2013).

Because of the advances in hardware technology and exact solution methods, many mixed integer linear programs can now be solved close to optimality or to optimality within a reasonable amount of time. Motivated by this advancement, several researchers have lately designed heuristics that incorporate phases where mixed integer linear programs are solved, the so-called matheuristics. The paper of Archetti and Speranza (2014), classify matheuristics in three different classes; decomposition approaches, improvement heuristics, and column generation based approaches. The recent literature mainly focuses on column generation based approaches and improvement heuristics. Though, for complex problems such as IRPs, the survey shows that the decomposition approaches still are popular. A further description of literature related to the three different classes is given in the following section. The discussed papers are classified in Table 2.2.

Table 2.2: Classification of relevant literature considering matheuristics

Paper	**Decomposition**	**Improvement**	**Column generation**
Van Anholt et al. (2016)	✓		
Yu et al. (2008)	✓		
Hemmati et al. (2016)	✓		
Savelsbergh and Song (2008)		✓	
Stålhane et al. (2012)		✓	
Archetti et al. (2017a)		✓	
Raa and Aghezzaf (2009)			✓
Danna and Pape (2005)			✓

Decomposition

A decomposition approach means to divide the problem into smaller and simpler subproblems. Next, a specific solution method is applied to each of the subproblems. In a matheuristic at least one of these subproblems have to be solved by a mathematical programming formulation (Archetti and Speranza, 2014). For routing problems, the mathematical programming model used will typically be a mixed integer linear program (MILP). A routing problem often decomposes into two subproblems; the clustering of customers which assign groups of customers to different vehicles and the sequencing of customers into vehicle routes. Van Anholt et al. (2016) use a decomposition approach to solve an IRP-PDP arising in the replenishment of automated teller machines (RATMs). The problem consist of 6 377 RATMs that are replenished by 32 vehicles each based at different depots, over a six-day planning horizon. To reduce the high complexity and the number of variables a clustering procedure is applied and the problem is decomposed into 32 subproblems of about 200 RATMs each. This type of decomposition is known as a *cluster first - route second approach* in the literature and are used in a number of papers.

The cluster first-route second approach is the most widely used subgroup for routing problems of the *two-phase approach*, an approach that decomposes the problem into two phases and solves them separately (Archetti and Speranza, 2014). An example of a two-phase approach that does not include the cluster first - route second method is the paper of Yu et al. (2008). The paper proposes an approximation algorithm which uses Lagrangian relaxation to solve an IRP with split deliveries. The relaxed problem is decomposed into a routing problem solved by a minimum cost flow algorithm and an inventory problem that is solved with linear programming. More advanced approaches have embedded the two-phase idea in an iterative scheme in later years. An example is the paper of Hemmati et al. (2016), a paper that proposes an iterative two-phase hybrid matheuristic to solve a multi-product short sea inventory routing problem. In the first phase of the algorithm, the inventory-routing problem is converted into a ship routing and scheduling problem. This is done with mathematical programming by generating cargoes subject to inventory limits. In phase two, an adaptive large neighborhood search solves the resulting routing and scheduling problem. The heuristic iteratively modifies the cargoes generated based on information obtained during the process.

Improvement heuristics

Matheuristics that belong to this class uses the exact solution from a MILP model to improve a solution found by a different heuristic approach. If the MILP models are solved only once, the improvement heuristics is named a one-shot improvement heuristic. MILP models can also be used for local optimization over a number of iterations (Archetti and Speranza, 2014).

The paper of Savelsbergh and Song (2008) describes a one-shot approach that solves an IRP with continuous moves. Continuous moves means that a vehicle can start a trip from one facility and end it at a different one. A greedy randomized heuristic is used to construct solutions. Further, a flow formulation based on a time-expanded graph is used to improve the solution. Another example is the paper of Archetti et al. (2017a) that proposes a three-phase matheuristic for a multivehicle inventory routing problem. The first phase generates an initial solution by relaxing the integrality of the routing variables in the IRP, before a routing problem is solved for each time period in order to determine routes. Phase 2 is a tabu search. The last phase consists of an improvement heuristic that uses the vehicle routes collected during the tabu search to focus the search on the most promising parts of the solution space. Stålhane et al. (2012) use a matheuristic consisting of a construction and improvement heuristic, followed by an intensification phase to solve an IRP arising in maritime logistics. In the intensification phase a subset of the variables in the original problem is fixed to the value found by the construction heuristic, and a MILP is used to optimize the value of the free variables.

The basic idea behind the use of MILPs is to use them as operators in a searching procedure, either to explore the neighborhood or as an intensification tool. Archetti et al. (2012) use this approach to solve an IRP problem with a single vehicle. The algorithm solves a travelling salesman problem combined with an intensification search which consists of solving a sequence of MILP models. This intensification search is applied whenever the incumbent solution is improved with the purpose to improve it further. The effectiveness of the heuristic is proved through a set of benchmark instances with known optimal solutions.

Column generation approach

Column generation is a general method for solving linear programs and is often used for problems containing many variables. The main idea is to restrict the problem by removing a large part of the variables and then only generate the variables that are needed. Iterations are done between a master problem, which is the restricted version of the original problem, and a subproblem. For a routing problem, the subproblem typically consists of finding new routes, that are combined in the master problem. The dual solutions from the master problem is sent to the subproblem where a search for variables with the best reduced cost is done. Column generation used together with the branch and price algorithm, an algorithm that integrates branch and bound and column generation, have been successfully proved as an exact solution method for variations of the routing problem. The use of branch and price and/or column generation have therefore been extended to obtain efficient heuristic algorithms with high performance where column generation is used to build heuristic solutions (Archetti and Speranza, 2014).

The column generation heuristic approach is also used for IRPs. Raa and Aghezzaf (2009) describe a cyclic IRP with constant customer demand where all routes start and end at a depot. Computational experiments presented in the paper showed that a column generation based heuristic approach was capable of finding good solutions for varying circumstances. In the paper, the heuristic algorithm is used to generate columns in the subproblem of the column generation. In the subproblem, two different construction heuristics and one improvement heuristic are used.

Danna and Pape (2005) describe a branch and price heuristic to solve a VRP-TW. The branch and price algorithm is embedded with a local search in order to find good integer solutions faster. Local search algorithms use operators to explore the neighborhood around a given set of solutions. This region is explored to generate better solutions iteratively. First, a heuristic is used to build an initial solution and secondly another heuristic is used to improve the solution. The scheme is tested on the VRP-TW, and it is shown that the algorithm improves the solution time of finding good initial integer solutions, without destroying the capability of the branch and price algorithm to produce good lower bounds.

Chapter 3

Problem description

The problem is described over a planning horizon discretized in time periods where a single commodity is transported from a depot and consumed by a set of customers. Another commodity is produced by the same set of customers and delivered to the depot. In the given planning horizon, the demand and supply for each customer are considered known for all time periods. At a customer location, the two commodities are stored separately, where both upper and lower inventory limits are known. Each customer has a holding cost related to the quantity of inventory.

A homogeneous fleet of vehicles is available for the transportation of commodities. Each vehicle has a single compartment with a known capacity, where both commodities are stored. The load on board can never exceed the vehicle's capacity. Each vehicle can perform at most one route in a given time period, that both starts and ends at the depot. During a route, a vehicle can visit any sequence of locations, and pickup and delivery at a given location do not need to happen simultaneously. However, each location can be visited at most once for pickup and once for delivery in a given time period. This corresponds to the general solution type described in Section 2.1.

The cost of driving from a location to another is only dependent on distance and a fixed service cost. As long as the inventory is kept within limits, the customers do not need to be visited in all time periods. Neither does a vehicle need to perform a route in every time period.

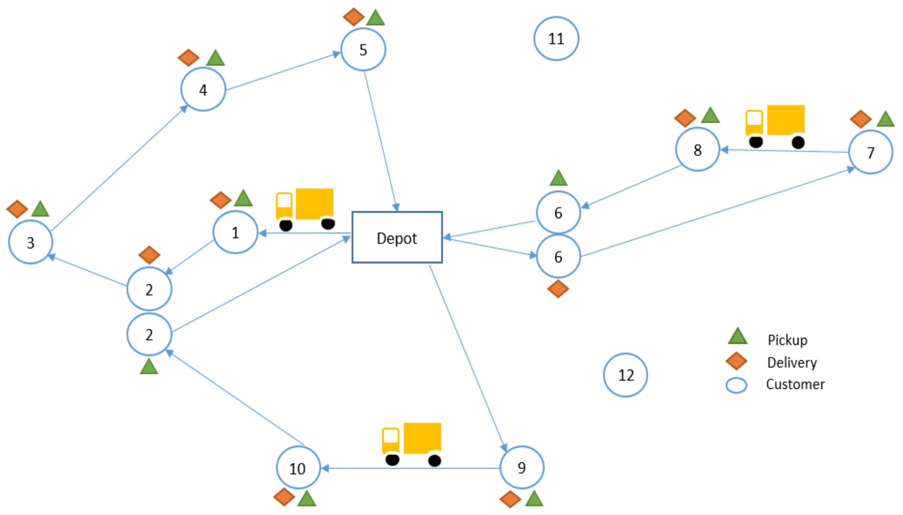


Figure 3.1: An example of a solution of the IRP-DPD

Each arc in the network is associated with a duration and there is a restriction on the total time of a route.

The problem consists of deciding the routing of the vehicles for each time period, and how much to deliver and pickup at the visited locations. The objective is to minimize the transportation and holding costs over a finite time horizon. An illustration of a solution to the described problem is shown in Figure 3.1.

Chapter 4

The mathematical model

In this chapter, we present a mixed integer linear program formulation. We found that a flow-based model formulation, a formulation without a vehicle index, was the most efficient to solve the IRP-DPD. In our project report, we also tested other model formulations that were found less efficient. These model formulations are listed in Appendix A.

4.1 Modeling choices and assumptions

To handle the divisible pickup and delivery option, a pair of fictitious co-located customer nodes, one purely pickup and one purely delivery, are created for each customer. There exist no arcs from pickup nodes to their co-located delivery nodes. In the mathematical model formulation, a delivery node i is co-located with pickup node $i+n$, where n is the number of customers. Elsewhere, there are arcs between all nodes, in both directions, as illustrated in Figure 4.1. Here D and P symbolize a delivery node and its co-located pickup node respectively.

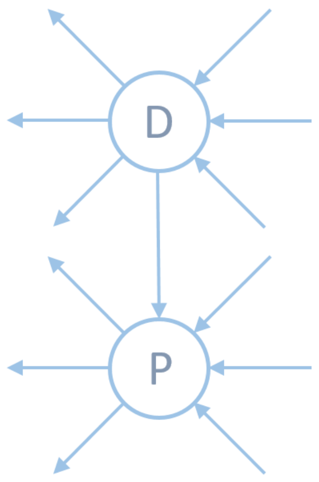


Figure 4.1: Arc network between customers and co-located nodes

The time scheme is discrete. For all pickup and delivery nodes, the commodity is produced or consumed after the node is served as shown for the inventory balance, i, in Figure 4.2. The inventory, i is never allowed to temporarily exceed inventory limits. The inventory i_t is measured at the end of each time period.

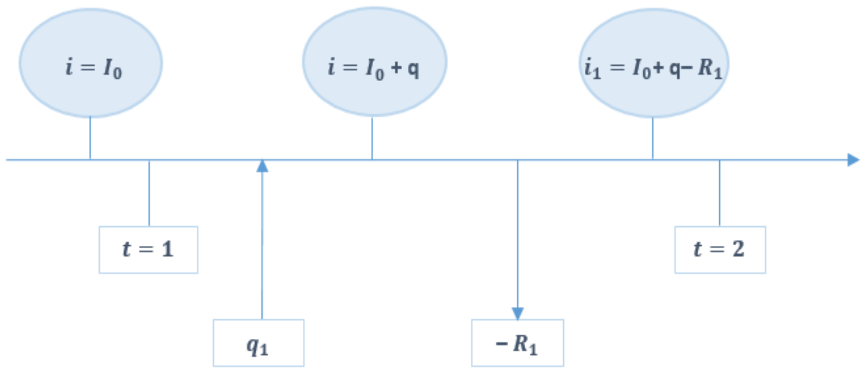


Figure 4.2: Inventory balance at a delivery node

All information is considered deterministic and known a priori. The information consists of customer locations, travel times and costs, initial inventory and pickup and delivery demand over the time horizon. The depot has sufficient capacity and inventory to perform all deliveries and pickups during the planning horizon. Every route in a time period starts and ends at the depot.

4.2 IRP-DPD flow formulation

The followings sets, parameters and variables are defined for the flow model formulation.

Sets	
$\mathcal{N}^D$	Set of delivery nodes, $\{1, 2, \ldots, n\}$
$\mathcal{N}^P$	Set of pickup nodes, $\{n+1, n+2, \ldots, 2n\}$
$\mathcal{N}$	Set of all pickup and delivery nodes, $\{1, 2, \ldots, 2n\}$
$\mathcal{N}'$	Set of all nodes, $\{0, \ldots, 2n\}$
$\mathcal{A}$	Set of arcs $\{(i, j) \mid i \in N', j \in N', i \neq j, i \neq j+n\}$
$\mathcal{T}$	Set of time periods, $\{1, \ldots, T\}$

Indices	
i, j	nodes
t	time periods

Parameters	
C_{ij}^T	Transportation cost from node i to node j
C_i^H	Holding cost for node i
T_{ij}	Travel time from node i to node j, includes service time in node j
R_{it}	Production/Consumption at node i, in time period t
I_{0i}	Initial inventory at node i
L_i	Lower inventory limit at node i
U_i	Upper inventory limit at node i
K^Q	Capacity on a vehicle
K^T	Max duration on a route for a vehicle
V	Size of the vehicle fleet

Variables

x_{ijt}	$\begin{cases} 1, & \text{if arc } (i,j) \text{ is traversed in time period } t \\ 0, & \text{otherwise} \end{cases}$
y_{it}	$\begin{cases} 1, & \text{if a vehicle visits node i in time period } t \\ 0, & \text{otherwise} \end{cases}$
t_{it}	Time node i is visited in time period t
q_{it}	Amount of commodity picked up or delivered at node i, in time period t
i_{it}	Inventory level at node i at the end of time period t
l^{P}_{ijt}	Load of pickup commodity on arc (i,j) in time period t
l^{D}_{ijt}	Load of delivery commodity on arc (i,j) in time period t

Flow formulation

$$\min \sum_{(i,j)\in\mathscr{A}} \sum_{t\in\mathscr{T}} C^T_{ij} x_{ijt} + \sum_{i\in\mathscr{N}} \sum_{t\in\mathscr{T}} C^H_i i_{it} \tag{4.1}$$

$$\text{s.t.} \sum_{j\in\mathscr{N}'} x_{ijt} - \sum_{j\in\mathscr{N}'} x_{jit} = 0 \qquad i\in\mathscr{N}', t\in\mathscr{T} \tag{4.2}$$

$$\sum_{j\in\mathscr{N}'} x_{ijt} - y_{it} = 0 \qquad i\in\mathscr{N}', t\in\mathscr{T} \tag{4.3}$$

$$y_{0t} \le V \qquad t\in\mathscr{T} \tag{4.4}$$

$$i_{i0} = I_{i0} \qquad i\in\mathscr{N} \tag{4.5}$$

$$i_{it} - i_{it-1} - R_{it} + q_{it} = 0, \qquad i\in\mathscr{N}^P, t\in\mathscr{T} \tag{4.6}$$

$$i_{it} - i_{it-1} + R_{it} - q_{it} = 0, \qquad i\in\mathscr{N}^D, t\in\mathscr{T} \tag{4.7}$$

$$i_{it-1} + q_{it} \le U_i \qquad i\in\mathscr{N}^D, t\in\mathscr{T} \tag{4.8}$$

$$i_{it-1} - q_{it} \ge L_i \qquad i\in\mathscr{N}^P, t\in\mathscr{T} \tag{4.9}$$

$$i_{it} \ge L_i \qquad i\in\mathscr{N}, t\in\mathscr{T} \tag{4.10}$$

$$i_{it} \le U_i \qquad i\in\mathscr{N}, t\in\mathscr{T} \tag{4.11}$$

$$t_{0t} = 0 \qquad t\in\mathscr{T} \tag{4.12}$$

$$t_{it} - t_{jt} + T_{ij} \le (K^T + T_{ij})(1 - x_{ijt}) \qquad (i,j)\in\mathscr{A}, t\in\mathscr{T} \tag{4.13}$$

$$t_{it} + T_{i0} x_{i0t} \le K^T \qquad i\in\mathscr{N}, t\in\mathscr{T} \tag{4.14}$$

$$\sum_{j\in\mathscr{N}'} l^D_{jit} - q_{it} - \sum_{j\in\mathscr{N}'} l^D_{ijt} = 0 \qquad i\in\mathscr{N}^D, t\in\mathscr{T} \tag{4.15}$$

$$\sum_{j\in\mathscr{N}'} l^P_{jit} + q_{it} - \sum_{j\in\mathscr{N}'} l^P_{ijt} = 0 \qquad i\in\mathscr{N}^P, t\in\mathscr{T} \tag{4.16}$$

$$\sum_{j\in\mathscr{N}'} l^P_{jit} - \sum_{j\in\mathscr{N}'} l^P_{ijt} = 0 \qquad i\in\mathscr{N}^D, t\in\mathscr{T} \tag{4.17}$$

$$\sum_{j\in\mathscr{N}'} l^D_{jit} - \sum_{j\in\mathscr{N}'} l^D_{ijt} = 0 \qquad i\in\mathscr{N}^P, t\in\mathscr{T} \tag{4.18}$$

$$l^P_{ijt} + l^D_{ijt} \le K^Q x_{ijt} \qquad (i,j)\in\mathscr{A}, t\in\mathscr{T} \tag{4.19}$$

$$i_{it} \ge 0 \qquad i\in\mathscr{N}, t\in\mathscr{T}\cup\{0\} \tag{4.20}$$

$$q_{it} \ge 0 \qquad i\in\mathscr{N}, t\in\mathscr{T} \tag{4.21}$$

$$l^P_{ijt}, l^D_{ijt} \ge 0 \qquad (i,j)\in\mathscr{A}, t\in\mathscr{T} \tag{4.22}$$

$$0 \leq y_{it} \leq 1 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{4.23}$$

$$0 \leq y_{0t} \qquad t \in \mathcal{T} \tag{4.24}$$

$$x_{ijt} \in \{0,1\} \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{4.25}$$

The Objective function 4.1 minimizes the sum of routing and inventory holding costs. The flow conservation is described by Constraint 4.2, while Constraint 4.3 links the x_{ijt} and the y_{it} variables. Constraint 4.4 restricts the number of vehicle routes to be less than the size of the vehicle fleet. Constraint 4.5 initializes the inventory level. The inventory balance for pickup and delivery nodes are defined by Constraints 4.6-4.7. Constraints 4.8-4.9 ensure that the inventory is within the upper and lower limit within each time period. This is consistent with the assumption that the commodity is produced or consumed after the node is served. The inventory limits at the end of each time period are defined by Constraints 4.10-4.11. Constraint 4.12 initializes the time variable. Constraint 4.13 provides the time balance, while Constraint 4.14 ensures that the total time length of a route is not exceeded. The loading and allocation of goods on the vehicles are covered by Constraints 4.15-4.16. The load balance at the pickup and delivery nodes is provided by Constraints 4.17-4.18. Constraint 4.19 ensures that the vehicle capacity is not exceeded. Finally, the requirements for the variables are defined by Constraints 4.20-4.25

4.3 Formulation for simultaneous pickup and delivery

The divisible model is a generalization of the simultaneous model, allowing a greater number of solutions, but with a cost of higher complexity. In order to reduce the complexity, especially for larger instances, a simultaneous formulation is introduced. The divisible flow formulation is flexible and can easily be transformed into a simultaneous model with small changes in the model formulation. Figure 4.3 illustrates how the flow formulation in Section 4.2 is adjusted to only allow simultaneous pickup and delivery. All arcs incident to pickup nodes and incident from delivery nodes are removed, except for the arc between co-located nodes. Binary variables are introduced to only account for the service cost at those nodes where service is applied, i.e., commodities are picked up or delivered. This is to ensure that the two models have the same optimal objective value if there is no divisible pickup and delivery. The costs associated with the arcs in the network are described further in Chapter 7 under the generation of test instances.

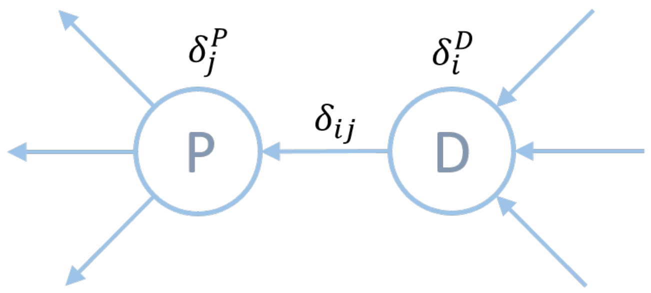


Figure 4.3: Simultaneous model

Mathematically the following new sets, variables and constraints are defined in the model

$\mathscr{A}'$	Set of arcs incident from pickup nodes and incident to delivery nodes. $\{\,(i,j) \mid i \in \mathscr{N}' \setminus \mathscr{N}^D, j \in \mathscr{N}' \setminus \mathscr{N}^P\,\}$
$\mathscr{A}^{co} \subset \mathscr{A}$	Set of arcs between co-located nodes
δ_{it}^D	1 if commodities are delivered in node $i \in \mathscr{N}^D$, 0 otherwise
δ_{it}^P	1 if commodities are picked up in node $i \in \mathscr{N}^P$, 0 otherwise
δ_{ijt}	1 if actions are performed in both co-located nodes, 0 otherwise

$$\sum_{t\in\mathscr{T}}\Big(\sum_{(i,j)\in\mathscr{A}'} C_{ij}^T x_{ijt} + \sum_{(i,j)\in\mathscr{A}^{co}} C_{ij}^T \delta_{ijt}\Big) \tag{4.26}$$

$$q^P_{it} - \min(K^Q, U_i - L_i)\delta^P_{it} \leq 0 \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \qquad (4.27)$$

$$q^D_{it} - \min(K^Q, U_i - L_i)\delta^D_{it} \leq 0 \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \qquad (4.28)$$

$$\delta^D_{it} + \delta^P_{jt} - \epsilon\delta_{ijt} \leq (2 - \epsilon) \qquad (i,j) \in \mathcal{A}^{co} \qquad (4.29)$$

Expression 4.26 represents the transportation costs in the simultaneous formulation. It only accounts for the arc cost between co-located nodes if actions are performed in both nodes. Constraints 4.27-4.28 link pickup and delivery to corresponding binary variables which indicate whether an action is performed or not. Constraint 4.29 sets δ_{ijt} to 1 if actions are performed in both co-located nodes. Here, ϵ is smaller than 1 by multiple orders of magnitude.

Chapter 5

The exact solution method

The purpose of this chapter is to strengthen the flow model formulation presented in Chapter 4. In Section 5.1, a number of valid inequalities are suggested for the flow formulation. Some of these valid inequalities are based upon the work of Archetti et al. (2017b) and Coelho and Laporte (2014). Section 5.2 proposes a branch and cut algorithm that dynamically add cuts when the subtour elimination constraints are violated.

5.1 Valid inequalities

For a general integer program, an inequality is valid if it does not remove any feasible integer solutions. The valid inequality is called a cut if the inequality cuts off the optimal LP relaxation of the problem instance. The purpose of adding valid inequalities is to improve the dual bound of the IP problem, and as a consequence solve the problem faster and for larger instances.

To evaluate and discuss the efficiency of the valid inequalities it is helpful to know what characterizes an optimal LP solution. Any optimal solution to our problem strives to minimize the combination of transportation costs and holding costs. The two objectives are diverging because minimizing the inventory implies substantial transportation costs. On the opposite, less transportation leads to greater amounts of inventory.

Figure 5.1 illustrates the solution of the LP relaxation(left) and an integer problem (IP) solution for the same instance(right). The solution value of the arc variables is represented by a blue color scale. A darker color means greater value. Orange circles represent customer nodes. Co-located nodes are separated horizontally by a small distance for illustrational purposes. A customer's delivery node is to the left and its co-located pickup node to the right. The depot is represented by a green circle in the center of the figure. All illustrations in this thesis adhere to this convention when illustrating nodes and arcs.

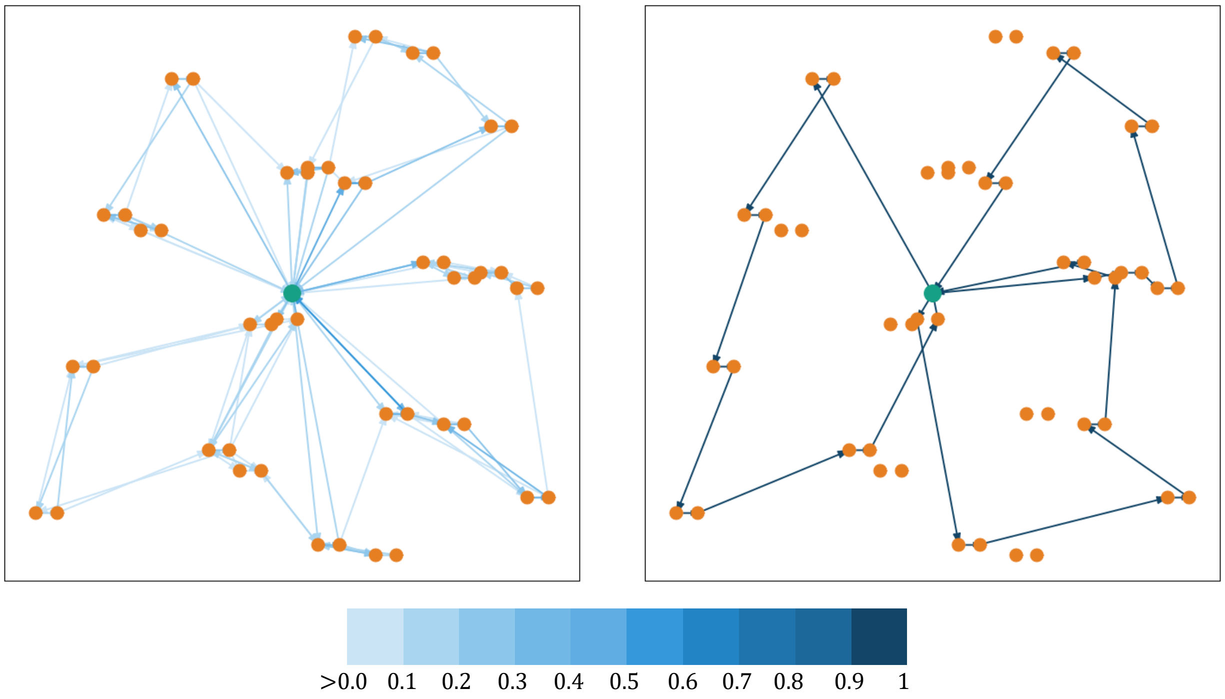


Figure 5.1: The LP relaxation and an IP solution for an instance with 24 customers

The test instance in the figure was run with the arc-flow formulation without any valid inequalities. The figure illustrates typical characteristics of the LP relaxation. The LP relaxation makes the arc-variables as small as possible to minimize the transportation costs. If the capacity is large relative to the load carried, the arc-variables are thus close to zero. In an integer solution, it is often optimal to visit several customers on the same route to minimize the distance traveled. In the relaxation, however, more customers mean greater load on the vehicle which increases the arc-variables on the route. Hence, the LP relaxation tends to have many fractional arcs connected to the depot.

The optimal solution also strives to keep the inventory as low as possible to minimize the holding costs. Considering holding costs only, the customers should be visited as often as possible. In the LP relaxation, where routes are assigned fractional vehicles, the majority of customers are often visited every day indeed. In contrast, in the IP solution, fewer customers are visited. In the following sections, a number of valid inequalities for the IRP-DPD model is introduced.

Inventory levels

In Chapter 4, we defined the inventory level to be measured at the end of each time period, after the node is served and after production and consumption. Therefore, if a delivery node is not visited in a time period t, then the inventory level at period $t-1$ is at least equal to the quantity consumed at the node in period t plus the minimum inventory level. The inventory level constraint related to delivery nodes is stated in Inequality 5.1. Similarly, Inequality 5.2 states that the inventory must be below a certain limit at pickup nodes, which have an increasing level of inventory if not visited.

$$i_{i(t-1)} \geq R_{it}(1-y_{it}) + L_i \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{5.1}$$

$$i_{i(t-1)} \leq U_i - R_{it}(1-y_{it}) \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{5.2}$$

Inequalities 5.3-5.4 consider the case when node i is not served in and between time period $t' = t_1$ and $t' = t_2$, and is thus a generalization of Inequalities 5.1-5.2.

$$i_{i(t_1-1)} \geq \left(\sum_{t'=t_1}^{t_2} R_{it'}\right)\left(1-\sum_{t'=t_1}^{t_2} y_{it'}\right) + L_i \qquad i \in \mathcal{N}^D, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \tag{5.3}$$

$$i_{i(t_1-1)} \leq U_i - \left(\sum_{t'=t_1}^{t_2} R_{it'}\right)\left(1-\sum_{t'=t_1}^{t_2} y_{it'}\right) \qquad i \in \mathcal{N}^P, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \tag{5.4}$$

The concept of Inequality 5.3 is illustrated in Figure 5.2. The figure shows the inventory balance at a delivery node when not visited in and between t'= t_1 and t'= t_2. Then the inventory, i_{t_1-1}, must be greater or equal to I_A, if node i is not visited in and between $t' = t_1$ and $t' = t_2$. Else the inventory constraint for lower limit will not hold.

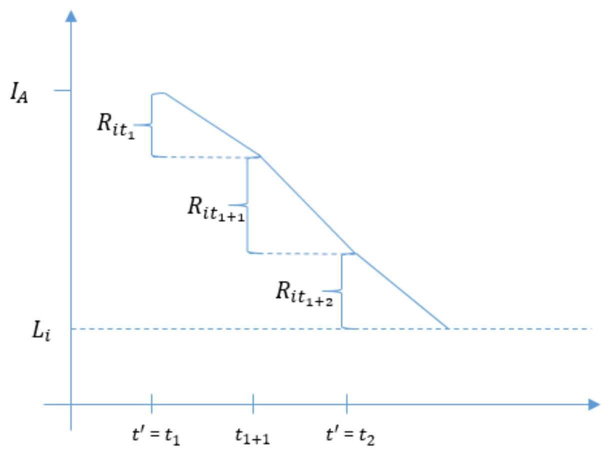


Figure 5.2: Illustration of the minimum inventory inequality

Excess consumption and excess production

This paragraph is an introduction to the notation used to define the valid inequalities in Inequality 5.5-5.8. The inventory level increases or decreases monotonically if a customer node is not visited. Given that a delivery node, i, is not visited in and between time period t_1 and t_2, we define its excess consumption as $E^D_{it_1t_2}$. The excess consumption, $E^D_{it_1t_2}$, is the difference between the inventory level and the node's lower limit if the node is not visited between t_1 and t_2. If there is no violation, i.e., the inventory level is above the lower limit, the excess consumption is set to 0. Likewise, the excess production at a pickup node, $E^P_{it_1t_2}$, is defined as the inventory in excess of the upper inventory limit. In the expressions for $E^D_{it_1t_2}$ and $E^P_{it_1t_2}$ the inventory in t_1, which is unknown, is set to the value that minimizes the excess consumption and excess production. That is the upper limit for delivery nodes and the lower limit for pickup nodes. If t_1 is defined as time period 1, the formulation can be tightened because the inventory at the beginning of time period 1 is known as I_{i0}. The concepts are illustrated in Figure 5.3 and 5.4.

$$E^D_{i t_1 t_2} = \begin{cases} \max\left(0, \sum_{t'=t_1}^{t_2} R_{i t'} - (U_i - L_i)\right) & i \in \mathcal{N}^D, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \\ \max\left(0, \sum_{t'=1}^{t_2} R_{i t'} - (I_{i0} - L_i)\right) & i \in \mathcal{N}^D, t_2 \in \mathcal{T} \end{cases}$$

$$E^P_{i t_1 t_2} = \begin{cases} \max\left(0, \sum_{t'=t_1}^{t_2} R_{i t'} - (U_i - L_i)\right) & i \in \mathcal{N}^P, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \\ \max\left(0, \sum_{t'=1}^{t_2} R_{i t'} - (U_i - I_{i0})\right) & i \in \mathcal{N}^P, t_2 \in \mathcal{T} \end{cases}$$

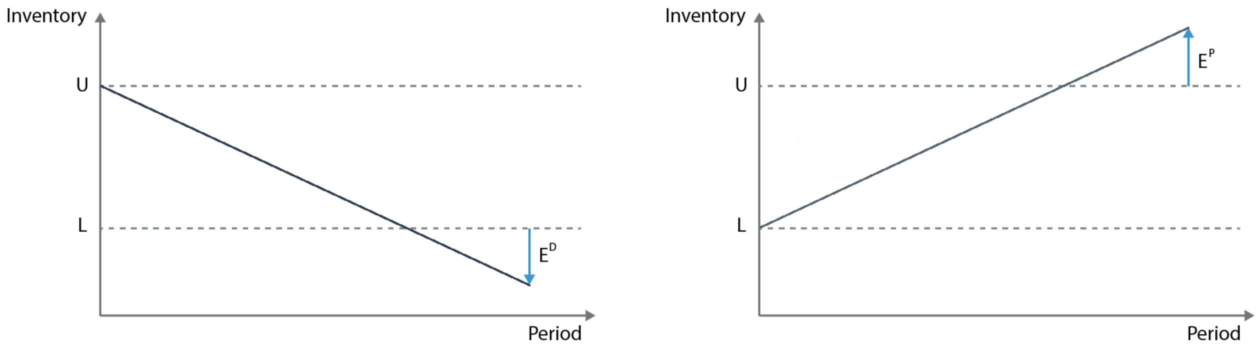


Figure 5.3: Excess consumption and excess production calculated from period t

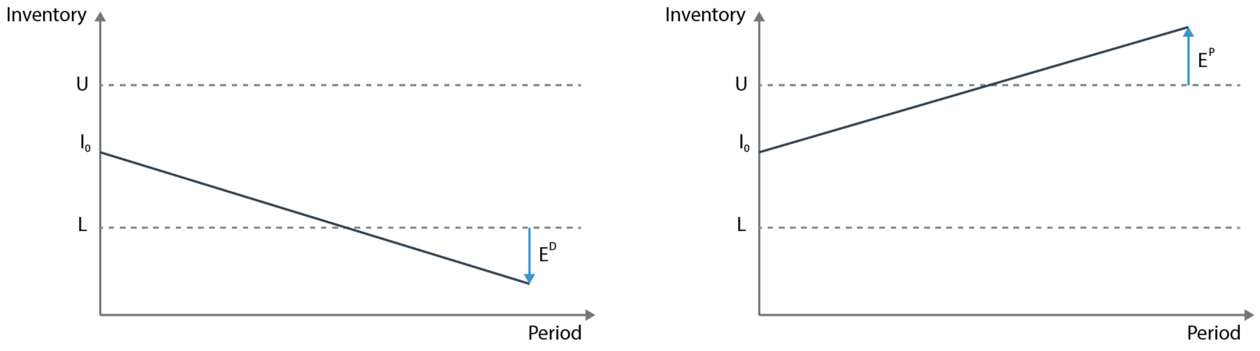


Figure 5.4: Excess consumption and excess production calculated from period 1

Minimum number of node visits

The minimum quantity that has to be delivered and picked up at node i between time period t_1 and t_2 is stated by $E^D_{it_1t_2}$ and $E^P_{it_1t_2}$ respectively. We also have that the maximum quantity to be delivered/picked up at a node is the minimum value of the capacity of a vehicle and the difference between the upper and lower inventory limit. Inequality 5.5 states the minimum number of visits to a delivery node that must occur between two time periods, such that the inventory is sufficient to fulfill future demands. Likewise, Inequality 5.6 sets the minimum number of node visits required to keep the inventory at a pickup node below its upper limit.

$$\sum_{t'=t_1}^{t_2} y_{it'} \geq \left\lceil \frac{E^D_{it_1t_2}}{\min(K^Q, U_i - L_i)} \right\rceil \qquad i \in \mathcal{N}^D, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \tag{5.5}$$

$$\sum_{t'=t_1}^{t_2} y_{it'} \geq \left\lceil \frac{E^P_{it_1t_2}}{\min(K^Q, U_i - L_i)} \right\rceil \qquad i \in \mathcal{N}^P, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \tag{5.6}$$

Minimum flow through a subset of customers

The excess production and excess consumption can also be calculated over a subset of customers. The minimum flow through the subset can then be constrained. On a vehicle, the load of delivery and pickup commodities is monotonically decreasing and increasing respectively. Therefore the minimum flow through the subset is also the minimum number of routes or vehicles that need to visit the subset within a period of time. This logic is captured in Inequalities 5.7 and 5.8. However, these inequalities do not prevent the same vehicle from visiting the subset several times on a route. Hence, it does not capture the fact that each flow through the subset must belong to different routes.

$$\sum_{t'=t_1}^{t_2} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{N}' - \mathcal{S}} x_{ijt'} \geq \left\lceil \frac{\sum_{i \in \mathcal{S}} E^D_{it_1t_2}}{K^Q} \right\rceil \qquad \mathcal{S} \subseteq \mathcal{N}^D, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \tag{5.7}$$

$$\sum_{t'=t_1}^{t_2} \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{N}' - \mathcal{S}} x_{ijt'} \geq \left\lceil \frac{\sum_{i \in \mathcal{S}} E^P_{it_1t_2}}{K^Q} \right\rceil \qquad \mathcal{S} \subseteq \mathcal{N}^P, t_1, t_2 \in \mathcal{T}, t_1 \leq t_2 \tag{5.8}$$

In this thesis, Inequalities 5.7- 5.8 are implemented for all subsets consisting of two and three customer nodes. For subsets consisting of only one customer node, the inequalities reduce to 5.5- 5.6 respectively. The valid inequality is, therefore, a generalization of the minimum number of node visits inequality. Subsets consisting of more than three customer nodes will not be considered.

Subtour elimination

In routing problems with a single destination and origin, the routes of a feasible solution must form a connected graph. Subtour elimination constraints (SECs) can be used to ensure that solutions are connected. In our problem, however, subtour elimination is fulfilled by the model formulation itself because of the monotonically increasing time resource for integer solutions. Yet, the LP relaxation of the problem may still generate subtours. To eliminate subtours, a valid inequality for subsets of customers is introduced. A general method to eliminate subtours is to require the arcs traversed internally within a subset of nodes to be at least one less than the number of nodes in the particular subset. For instance, in the special case where a subset consists of three nodes, the sum of the arcs traversed between the nodes needs to be less or equal to two. Mathematically the inequality is described below in 5.9. Here, $\mathscr{S}$ is a subset of nodes.

$$\sum_{i\in\mathscr{S}}\sum_{j\in\mathscr{S}} x_{ijt} \leq \sum_{i\in\mathscr{S}} y_{it} - y_{mt} \qquad \mathscr{S}\subseteq\mathscr{N}, m\in\mathscr{S}, t\in\mathscr{T} \tag{5.9}$$

5.2 The branch and cut algorithm

The IRP-DPD problem presented in this thesis is a generalization of the vehicle routing problem and is naturally NP-hard. In fact, the pickup and delivery structure combined with the inventory decisions increases the complexity and the solution space of the problem significantly. For small instances, the flow formulation presented in Chapter 4 can be solved directly with a MILP solver by branch and bound within a reasonable time. However, for larger and more realistic instances we propose a branch-and-cut algorithm. During the algorithm, the subtour elimination constraints described by Inequality 5.9 are dynamically generated and added to the problem when violated. The current subproblem is then re-optimized. The process is repeated until no more cuts can be added or until a feasible or dominated solution is reached. The branch-and-cut algorithm proposed is described in Algorithm 1.

Algorithm 1 Branch-and-cut algorithm

```
1   Add valid inequalities to the flow formulation at the root node.
2   $z^* \leftarrow \infty$
3   Termination check.
4   if there are no more nodes to evaluate then
5       Stop with the optimal solution $z^*$.
6   else
7       Select a node from the branch-and-bound tree.
8   end if
9   do Solve the LP relaxation of the selected node and let the cost be $z$.
10      if the current solution is feasible then
11          if $z \geq z^*$ then
12              Prune the node and go to termination check.
13          else
14              Update the current solution of $z^*$ and go to termination check.
15          end if
16      end if
17      Call the Add-Subtour-Cut algorithm.
18  while cut is added
19  Branch on fractional variables.
20  Go to termination check.
```

The branch and cut algorithm can be combined with different combinations of the valid inequalities presented in 5.1-5.8. These inequalities are only inserted in the root node and are not added dynamically. To identify and add subtour cuts for violated subtour elimination constraints during the branch-and-cut algorithm, a call to an algorithm named Add-Subtour-Cut is done.

The subtour elimination algorithm

The algorithm to find and eliminate subtours is described in Algorithm 2. The algorithm first builds a *directed graph* for each period where the solution values of the arc variables correspond to the edge values in the graph (line 2). It then finds the *strong components* of this directed graph (line 3). For each strong component that also contains violated SECs the algorithm adds subtour cut 5.9 to eliminate that particular subtour (line 6). Each step is discussed in details in the following sections.

Algorithm 2 Add-Subtour-Cut

```
1   for each period $t \in$ Periods
2       G = Build-Graph(N, A)
3       StrongComp = Strong-Components(G)
4       for each strong component $S \in$ StrongComp
5           if S.Size $\geq$ 2 and S.ViolateSubtours
6               Add subtour cut
7           end if
8       end for
9   end for
```

The solution graph

Any integer or fractional solution forms a simple, directed graph (digraph) that is referred to as the solution graph. That is a graph with directed edges and no loops or multiple edges. The solution graph $G(V,E)$, is defined by a vertex set $V = V(G)$, that corresponds to the customer nodes, and an edge set $E = E(G)$ that corresponds to the arc flow variables. The solution graph is created by the algorithm *Build Graph*, with the current problem solution as input. For each visited node in the solution, a vertex is added to an initially empty graph. Arcs with a positive flow in the solution are added as edges to the graph. Only edges incident to and from vertices in the graph are considered. Note that the depot node and arcs connected to the depot are not included. The algorithm considers each customer node and potentially each arc once and therefore runs in linear time $O(|\mathscr{N}| + |\mathscr{A}|)$. The function lastly returns the solution graph.

Algorithm 3 Build-Graph

```
for each node $n \in N$
    if y[$n$][period] > 0
        add $n$ to $G.V$
    end if
for each vertex $u \in G.V$
        for each vertex v $\in G.V$
            if $(u, v) \in A$ and x[$u$][$v$] > 0
                add edge $(u, v)$ to $G.E$
            end if
return $G$
```

Strong components

For a digraph, G(V, E), let a path p be defined as a sequence $(i_1, i_2, ..., i_{n-1}, i_n)$ with all $i_k \in V$. When $i_1 = a$ and $i_n = b$, a path from a to b in G is named an a–b–path. "The digraph, $G(V,E)$ is *strongly connected* if it contains an i–j– path and a j–i– path for each pair of vertices $i, j \in V$. [...] A *subdigraph* $G'(V', E')$ of G, is a digraph with $V' \subseteq V$ and $E' = (i, j) \in E : i, j \in V'$. [...] A strongly connected subdigraph G' is called a *strong component* of G if there is no vertex $i \in V'$ and no vertex $j \in V \backslash V'$ for which G contains an i–j–path and a j–i–path" (Drexl, 2013). In other words, a strong component is the maximum set of vertices where at least one oriented path between every two vertices in the set exists. Figure 5.5 illustrates the strong components of a directed graph.

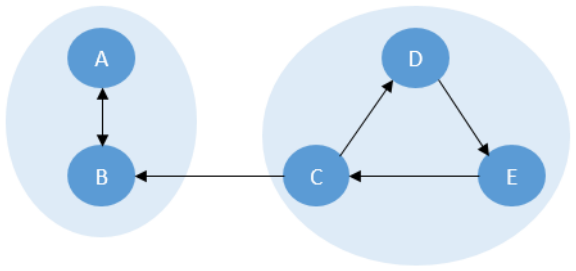


Figure 5.5: A directed graph and its corresponding strong components.

To identify and return strong components the Add-Subtour-Cut algorithm calls function Strong-Components with the digraph, G, as input. To identify the strong components, we use the algorithm proposed in Tarjan (1972). The algorithm performs a depth-first traversal of the graph where it considers each vertex and edge once and therefore have a linear worst-case running time of $O(|V| + |E|)$. Tarjan's original algorithm does not modify the graph, but our version of the algorithm also removes edges that connects strong components in the input graph. After the strong components of the digraph are found, all strong components that violate subtour elimination constraints are identified. A subtour cut is added to the problem if both the strong component includes at least two nodes and the subtour elimination constraint is violated. The details of the implementation of the Strong-Component algorithm is described further in Appendix B.

Figure 5.6 illustrates the subtour elimination algorithm. The upper left shows a solution graph of an LP relaxation. The upper right in the figure shows the directed graph returned from Build-Graph where the depot and edges connected to the depot are not included. The lower left is the graph after a call to the Strong-Component function where the strong components are identified, and edges between them removed. Finally, in the lower right, a strong component with a violated subtour constraint is identified.

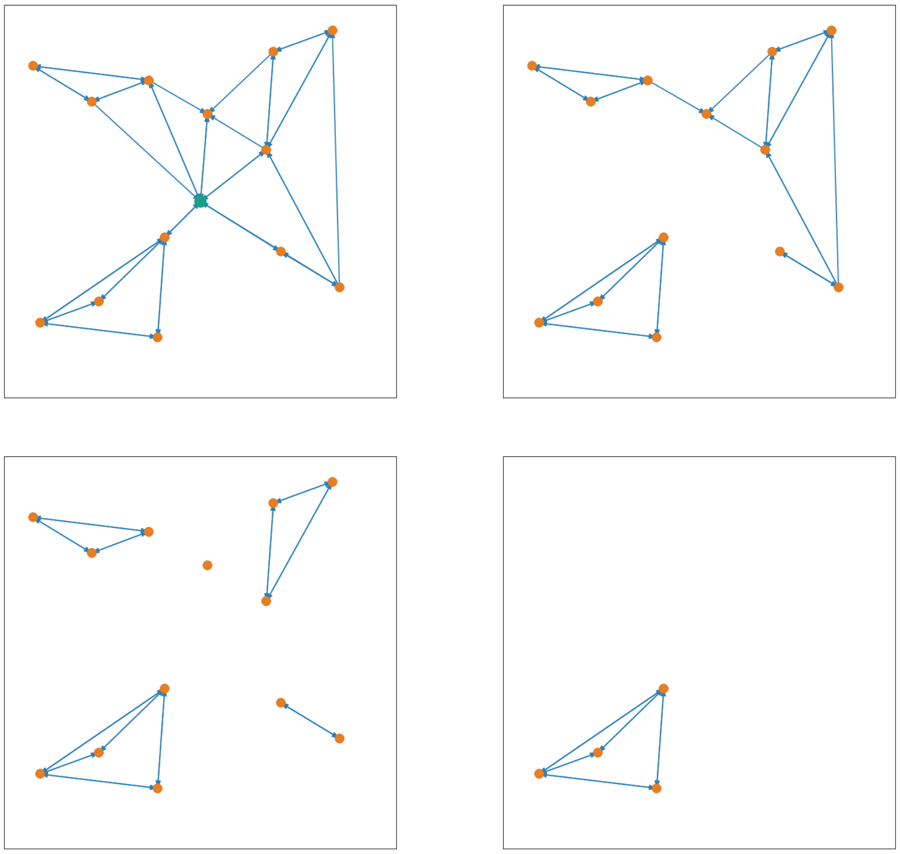


Figure 5.6: Illustration of the dynamic subtour elimination algorithm

For each strong component the add subtour algorithm needs to check every edge yielding a worst-case running time of $O(|E|)$. The entire algorithm then runs in linear time $O(|\mathcal{N}| + |\mathcal{A}| + |V| + |E| + |E|)$. For a simple graph we have that $|V| \leq |\mathcal{N}|$ and $|E| \leq |\mathcal{A}| \leq |\mathcal{N}|^2$ giving a worst case complexity of $O(|\mathcal{N}|^2)$. That is two order of magnitudes better than separation by maximum flows with a worst time complexity of $O(|\mathcal{N}|^4)$ (Drexl, 2013).

Although the worst case complexity is improved, subtour elimination by strong components does not guarantee to find all subtours in the solution graph. The paper of Drexl (2013) summarizes this as: "There are fractional solutions that contain a violated SEC but that do not contain a non-trivial strong component whose vertex set defines a violated SEC." An example of such a strong component is shown in Figure 5.7. The figure to the left is the strong components of a fractional solution from a period for a $C_{24}V_5$ instance. The figure to the right is the strong components that also violate SECs. The edges are color scaled according to their fractional values. A darker color means a greater value. Subtours are present in the upper strong component, for instance in the upper left corner, but the algorithm fails to detect them because the strong component is not a violated SEC itself.

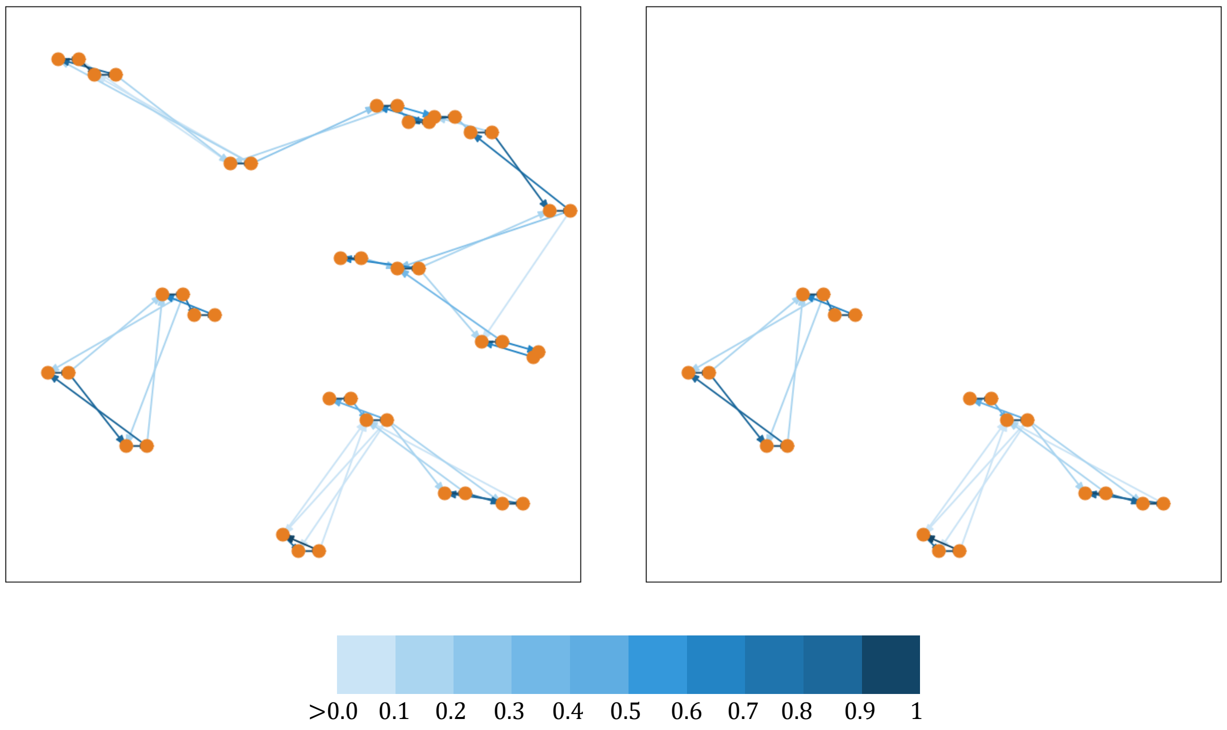


Figure 5.7: Strong components and the corresponding violated SECs.

When the strong components are checked for subtours all edges in the strong component are considered. To overcome some of the shortcomings of the strong components approach, we propose to be selective about the edges we include in the graph that are searched for strong components. Figure 5.8 illustrates the strong components of the same graph if only edges with a value greater than or equal to 0.25 are included. The graph is more dispersed containing several more strong components. The right figure shows the strong components that also contain violated SECs. As illustrated, the algorithm succeeds to find subtours that were not discovered when all edges were included. Also, it manages to divide strong components into several more subtours, yieldings stronger subtour elimination constraints. In the computational study, we present the algorithm's results for different edge weights.

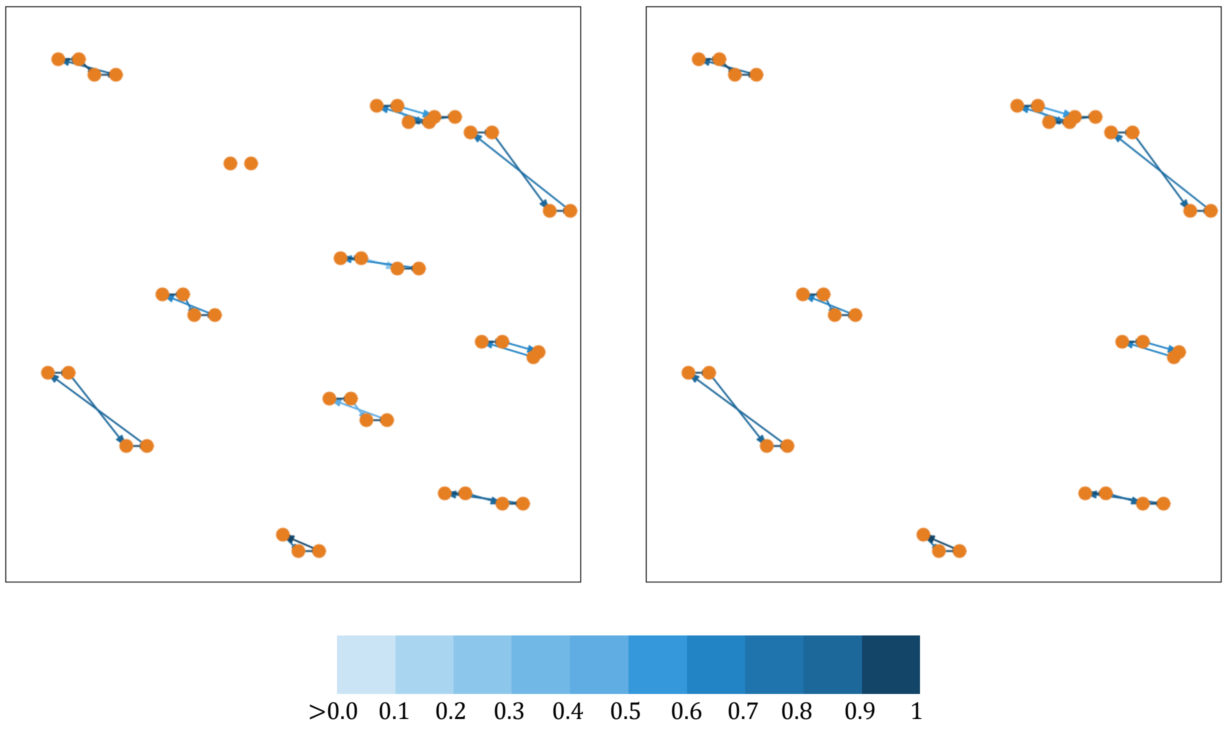


Figure 5.8: Strong components of a solution graph with edge weight above 0.25

If no subtours are allowed to appear in any solution in the branch and bound tree, the number of cuts may be extensive, increasing the overall solution time. In order to reduce the number of added cuts the definition of what is a subtour may be relaxed. To relax the definition of a subtour a parameter α is added to Inequality 5.9 with the resulting Inequality 5.10.

$$\sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}} x_{ijt} \leq \sum_{i\in\mathcal{S}} y_{it} - y_{mt} + \alpha \qquad \mathcal{S}\subseteq\mathcal{N}, m\in\mathcal{S}, t\in\mathcal{T} \tag{5.10}$$

α is set in the interval $[0,1]$. Setting $\alpha = 0$ is the original subtour elimination constraint. Setting $\alpha = 1$ means that subtours are allowed, and no cuts are added. In the computational study results for different values of α are presented.

Chapter 6

The heuristic solution method

The IRP-DPD is a highly complex problem, and an exact solution method is therefore likely to be too time-consuming for larger instances of the problem type. Hence, a matheuristic is proposed in this chapter. The mechanisms of the matheuristic are novel but partly inspired by Archetti et al. (2017a). The matheuristic consists of a two-phase approach that constructs a solution and is followed by an intensification search. The matheuristic then adds diversification constraints in order to construct a new solution. After a number of iterations, a final improvement search is performed.

For the two-phase approach described in this thesis, we decompose the problem into an inventory problem and a routing problem. In the first phase, a relaxed IRP-PD model is solved where the binary requirement on arc flow variable, x_{ijt}, is relaxed. The initial inventory decision of the iteration is decided in this phase. This part is described in Section 6.1. In the second phase, described in Section 6.2, we introduce a VRP-DPD formulation with the purpose of constructing vehicle routes based on the inventory decisions made in phase 1. A separate VRP-DPD is solved for each period in order to construct a solution. In Section 6.3, an *intensification search* is proposed. In the search, a set of operators with the purpose of improving the inventory decisions from phase 1 and finding more efficient routes than the ones generated in phase 2, are described. When the intensification search is finished, the current iteration is completed, and we initiate a new iteration. We call this process *diversification*, and the process is described in Section 6.4. When the iterative loop is terminated, a one-shot improvement heuristic is suggested and described in Section 6.5. In this phase, a route selection problem is solved, finding the optimal combination of routes from all iterations. Lastly, in Section 6.6 we discuss how the matheuristic easily can be transformed to solve other variants of the IRP. The matheuristic is illustrated in Figure 6.1. A further description of the matheuristic's different phases and steps is presented in the following sections.

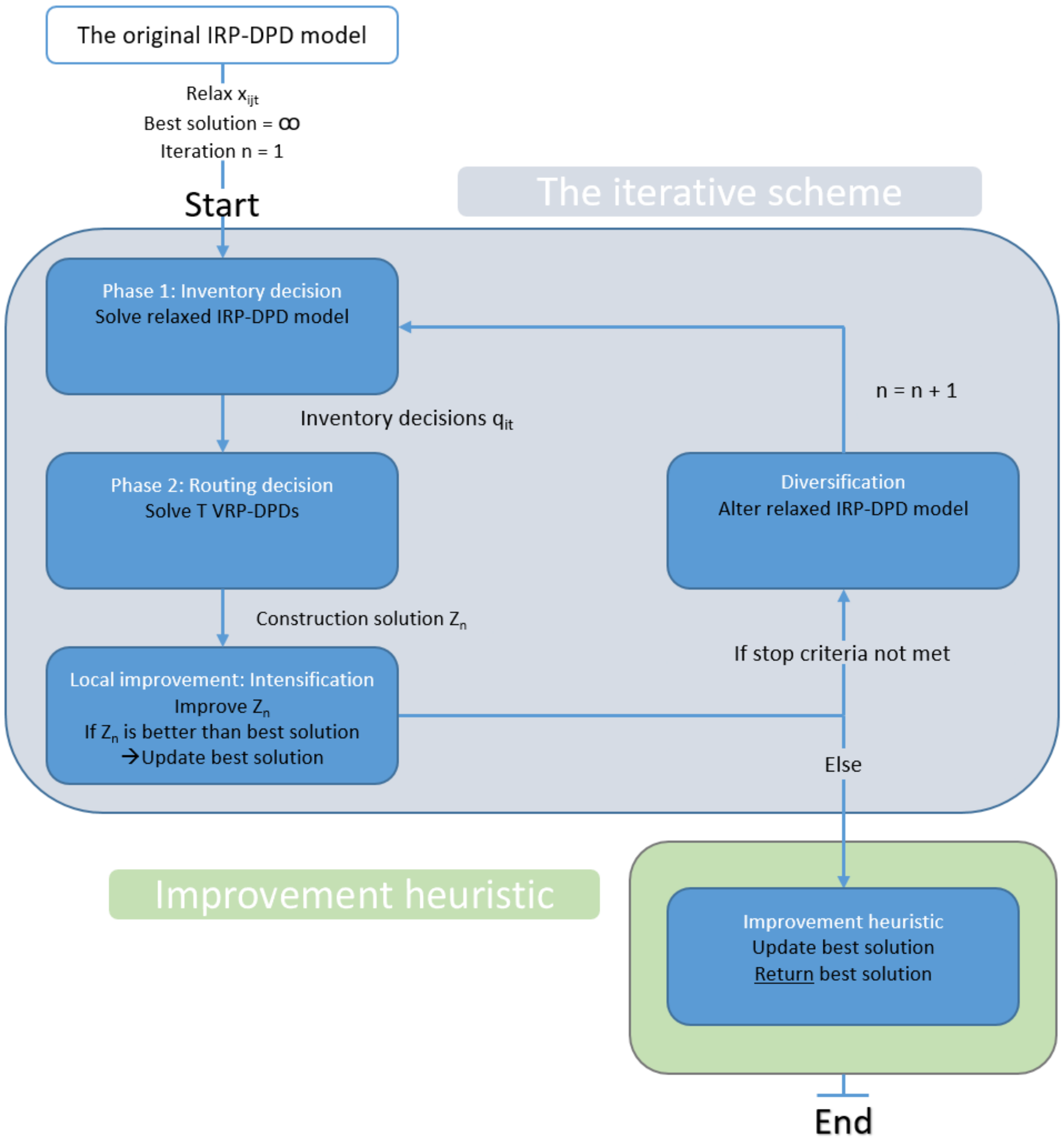


Figure 6.1: Solution methodology for the matheuristic

6.1 Phase 1

In the first phase, a relaxed IRP-PD model formulation is used to solve the inventory problem. The model formulation is equal to the flow formulation presented in Chapter 4, except for Constraints 4.23 and 4.25 which are now changed to:

$$0 \le x_{ijt} \le 1 \qquad (i,j) \in \mathscr{A}, t \in \mathscr{T} \tag{6.1}$$

$$y_{it} \in \{0,1\} \qquad i \in \mathscr{N}, t \in \mathscr{T} \tag{6.2}$$

The pickup and delivery quantities, q_{it}, are saved from the solution of the relaxed model which relates to the inventory decisions. In a feasible integer solution to the original problem, subtours cannot appear because of the monotonically increasing time resource. However, for the relaxed IRP, subtours can be present in a feasible solution. This is likely to occur in the relaxed model since subtours may reduce the transportation costs substantially. Subtours are unwanted because we want the solution from the model, and consequently the inventory decisions, to resemble the characteristics of an integer solution. The branch and cut algorithm with subtour elimination constraints can, therefore, be added to the relaxed model formulation to improve the inventory decisions in phase 1.

However, if no subtours are allowed to appear in a solution, it is harder to construct a feasible solution in the relaxed IRP-DPD, and the solution time may increase substantially. A viable alternative is, therefore, to relax the arc flow variables, x_{ijt}, in the simultaneous model instead. The simultaneous model has a smaller arc set and is likely to produce fewer subtours than the divisible model. For instance, there are no subtours of size two since there is no cycle with only two nodes for this set of arcs. Hence, a potential approach is to solve the relaxed IRP with simultaneous pickup and delivery. The VRPs in phase 2 still have the divisible option available. This version of the relaxed IRP is easier to solve and with SECs introduced it may produce better inventory decisions in acceptable running time. In the computational study, the different suggested solution approaches for the relaxed IRP are tested. Notably, an advantage of using a relaxed IRP with divisible pickup and delivery is that the solution of this relaxed problem is also a dual bound (lower bound) to the original problem. The lower bound can, for instance, be used to evaluate the final solution. This is not the case for the simultaneous alternative.

6.2 Phase 2

Phase 2 constructs vehicle routes based on the inventory decisions made in phase 1. A VRP-DPD formulation is proposed, which solves the routing problem for each day of the planning horizon. More precisely, a VRP-DPD is solved separately for each day. The delivery/pickup variable, q_{it}, from phase 1 describes the quantities that must be delivered/picked up at customer i when the VRP-DPD is solved for time period t. These pickup and delivery decisions are now represented as parameters, Q_i, in the VRP-DPD formulation. As opposed to the relaxed model in phase 1, the VRP-DPD formulation has binary requirements on the arc flow variables, x_{ijt}. To reduce the complexity of the problem further, a subset of customers and arcs are introduced for each time period, only considering the customers that were visited in the given time period in phase 1. The first phase can construct infeasible VRP-DPDs in phase 2, due to the relaxation of the arc flow variable. To ensure feasibility of the VRPs, additional vehicles can be added to the constrained fleet. These vehicles are associated with a penalty cost. The penalty cost is set high, resulting in infeasible solutions only if the problem is impossible to solve with the given vehicle fleet. The mathematical formulation of the VRP-DPDs is described below.

The VRP-DPD formulation

Sets	
$\mathscr{L}$	Subset of all customer nodes visited in the given period in phase 1
$\mathscr{L}'$	Subset of all customer nodes and depot visited in the given period in phase 1
$\mathscr{L}^D$	Subset of all delivery nodes visited in the given period in phase 1
$\mathscr{L}^P$	Subset of all pickup nodes visited in the given period in phase 1
$\mathscr{A}^{\mathscr{L}}$	Subset of all arcs between the customers visited and depot
Parameters	
Q_i	Pickup/delivery for customer i in the given period decided in phase 1
C^V	Cost of each extra vehicle
Variables	
r_0	Extra vehicles required in addition to the constrained vehicle fleet

$$\min \sum_{(i,j)\in\mathscr{A}^{\mathscr{L}}} C^T_{ij} x_{ij} + C^V r_0 \tag{6.3}$$

$$\text{s.t.} \sum_{j\in\mathscr{L}'} x_{ij} - \sum_{j\in\mathscr{L}'} x_{ji} = 0 \qquad i \in \mathscr{L}' \tag{6.4}$$

$$\sum_{j\in\mathscr{L}'} x_{ij} = 1 \qquad i \in \mathscr{L} \tag{6.5}$$

$$\sum_{j\in\mathscr{L}} x_{0j} \leq V + r_0 \tag{6.6}$$

$$t_0 = 0 \tag{6.7}$$

$$t_i - t_j + T_{ij} \leq (K^T + T_{ij})(1 - x_{ij}) \qquad (i,j) \in \mathscr{A}^{\mathscr{L}} \tag{6.8}$$

$$t_i + T_{i0} x_{i0} \leq K^T \qquad i \in \mathscr{L} \tag{6.9}$$

$$\sum_{i\in\mathscr{L}'} l^D_{ji} - Q_i - \sum_{j\in\mathscr{L}'} l^D_{ij} = 0 \qquad i \in \mathscr{L}^D \tag{6.10}$$

$$\sum_{i\in\mathscr{L}'} l^P_{ji} + Q_i - \sum_{j\in\mathscr{L}'} l^P_{ij} = 0 \qquad i \in \mathscr{L}^P \tag{6.11}$$

$$\sum_{i\in\mathscr{L}'} l^P_{ji} - \sum_{j\in\mathscr{L}'} l^P_{ij} = 0 \qquad i \in \mathscr{L}^D \tag{6.12}$$

$$\sum_{i\in\mathscr{L}'} l^D_{ji} - \sum_{j\in\mathscr{L}'} l^D_{ij} = 0 \qquad i \in \mathscr{L}^P \tag{6.13}$$

$$l^P_{ij} + l^D_{ij} \leq K^Q x_{ij} \qquad (i,j) \in \mathscr{A}^{\mathscr{L}} \tag{6.14}$$

$$l^P_{ij}, l^D_{ij}, r_0 \geq 0 \qquad (i,j) \in \mathscr{A}^{\mathscr{L}} \tag{6.15}$$

$$t_i \geq 0 \qquad i \in \mathscr{L}' \tag{6.16}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in \mathscr{A}^{\mathscr{L}} \tag{6.17}$$

In order to improve the solution efficiency, the relaxed subtour elimination constraint 5.10 is added to the VRP-DPD formulation.

Constructing a solution

The solutions from the inventory decisions of phase 1 and the routing decisions of phase 2 need to be combined into a solution of the original IRP-DPD. This solution, Z_n, is hereafter referred to as *the constructed solution.* The n index refers to which iteration of the two-phase approach the solution is derived from. The total cost of the constructed solution, C^{tot}, is calculated from Equation 6.18. In this expression, C_t^{VRP} represents the routing cost of time period t. The second part of the expression calculates the holding costs of the inventory decisions made in phase 1. Further, the solution with the best-known objective value from all the iterations of the matheuristic is noted Z^*, and known as the upper bound. All other solutions are evaluated against this solution.

$$C^{tot} = \sum_{t\in\mathcal{T}} C_t^{VRP} + \sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} C_i^H I_{it} \tag{6.18}$$

6.3 The intensification search

The purpose of the intensification search is to improve the constructed solution, Z_n, by searching in the neighborhood of the solution. To obtain this improvement, we use different operators which are described in the following sections. The intensification search uses the constructed solution, Z_n, as a starting point for the search. Z_n contains the values of q_{it}, i_{it}, x_{ijt} and y_{0t}, and whether these variables are treated as fixed or alterable varies between the different operators.

The relaxed arc flow variables in phase 1 often lead to inventory decisions that underestimate transportation costs and is the reason why the constructed solutions usually have transportation costs that are significantly higher than what is optimal. As a consequence, the main focus of the first class of operators is to reduce these costs. The reduction can be achieved by reducing the number of customer visits over the planning horizon, better utilization of the vehicles used or by reducing the number of routes traversed. In order to accomplish this, delivery and pickup quantities have to be moved from a given time period and re-inserted in other time periods. A class of operators related to quantity shifting is therefore proposed. In addition, an operator generating new vehicle routes is also proposed. Here, customer insertions and removals along with merging of existing routes are used to construct new routes which may improve the solution.

After the different operators have run to completion, a local route selection problem is solved. Here, all the routes which have been used in a solution found by the operators are pooled together, and we select the best combination of them to improve the solution further.

Algorithm 4 describes how we combine the different operators in the intensification search. As described by the algorithm, the search iterates until a stop criterion is met. A maximum number of iterations, a maximum computational time or reoccurrence of a previously found solution define the different stop criteria we use. The purpose of the last stop criterion is to avoid looping between the same solutions in the intensification search.

Algorithm 4 Intensification search

```
1   Get constructed solution, $Z_n$, of the two-phase approach.
2   Initialize the value $Z^{int}$= $Z_n$
3   Store the routes from $Z_n$ in route pool $P$
4   do
5       do
6           $Z^{int}$ = Shift-Quantity($Z^{int}$)
7           Store any new routes from $Z^{int}$ in $P$
11      while the objective value of $Z^{int}$ is improved
12
13      $Z^{int}$ = Route-Generation ($Z^{int}$)
14      Store any new routes from $Z^{int}$ in $P$
15  while stop criterion not met
16  Solve the route selection problem for the routes in $P$ and update $Z^{int}$
```

The solution Z^{int} is the current solution in the intensification search. This solution gets altered and updated by the different operators. Its objective value at the end of the intensification search serves as the upper bound for the given iteration of the matheuristic. Note that this value may differ from the objective value of Z^*, which is the best upper bound of all the iterations of the two-phase approach. The solution Z^* is updated after the intensification search if the objective value of Z^{int} is better than the current best-known solution.

The constructed solution Z_n is not guaranteed to be feasible as too many vehicles may be utilized in a period. Both intensification operators are used to make sure the constructed solution is feasible prior to the intensification search. First, if the constructed solution is infeasible, the Shift-Quantity operator tries to move quantity from the infeasible period in order to reduce the number of vehicles. If this operator fails, the Route-Generation operator is run for five iterations. This double-safe approach has not failed for any instance in the tests performed in this thesis.

The Shift-Quantity operators

The idea behind the Shift-Quantity operators is to remove delivery and pickup quantities from a time period and re-insert them in other time periods. This leads to changes in the current pickup and delivery decisions q_{it}. The operators can be used to ensure feasibility, to reduce transportation costs or to re-optimize holding costs with respect to fixed vehicle routes. Four different operators are proposed in this class of operators. The first operator re-optimizes the inventory decisions while treating all the routes of the current solution as fixed. The second one uses an LP model to minimize the delivery and pickup quantity for a chosen time period, holding all routes fixed. The VRP-DPD for the chosen time period is then resolved in order to reduce transportation costs by possibly finding new and better routes. The third one minimizes the number of customer visits for a chosen time period, considering the routes in all time periods fixed. The VRP-DPD for the chosen time period is again resolved. The last one uses a MILP model to minimize the transportation cost of a chosen period and the total inventory holding costs of all periods. The routes from all time periods, except the chosen one, are considered fixed. The four different operators are described in further details in the following sections, and their performances are evaluated against each other in the computational study.

The inventory optimization operator

The first Shift-Quantity operator is formulated as an LP problem that minimizes holding costs over the planning horizon considering all vehicle routes from the solution Z^{int} fixed. To obtain the route fixation, the x_{ijt} variables in Z^{int} are described as parameters, A_{ijt}, when the LP problem is solved. The formulation is described below.

$$\min \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}} C_i^H i_{it} \tag{6.19}$$

$$i_{i0} = I_{i0} \qquad i \in \mathcal{N} \tag{6.20}$$

$$i_{it} - i_{it-1} - R_{it} + q_{it} = 0, \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{6.21}$$

$$i_{it} - i_{it-1} + R_{it} - q_{it} = 0, \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{6.22}$$

$$i_{it-1} + q_{it} \leq U_i \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{6.23}$$

$$i_{it-1} - q_{it} \geq L_i \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{6.24}$$

$$i_{it} \geq L_i \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{6.25}$$

$$i_{it} \leq U_i \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{6.26}$$

$$\sum_{j \in \mathcal{N}'} l_{jit}^D - q_{it} - \sum_{j \in \mathcal{N}'} l_{ijt}^D = 0 \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{6.27}$$

$$\sum_{j \in \mathcal{N}'} l_{jit}^P + q_{it} - \sum_{j \in \mathcal{N}'} l_{ijt}^P = 0 \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{6.28}$$

$$\sum_{j \in \mathcal{N}'} l_{jit}^P - \sum_{j \in \mathcal{N}'} l_{ijt}^P = 0 \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{6.29}$$

$$\sum_{j \in \mathcal{N}'} l_{jit}^D - \sum_{j \in \mathcal{N}'} l_{ijt}^D = 0 \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{6.30}$$

$$l_{ijt}^P + l_{ijt}^D \leq K^Q A_{ijt} \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{6.31}$$

$$l_{ijt}^P, l_{ijt}^D \geq 0 \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{6.32}$$

$$q_{it} \geq 0 \qquad i \in \mathcal{N}', t \in \mathcal{T} \tag{6.33}$$

Constraints 6.20-6.30 are identical to the constraints for inventory balance, inventory restrictions and load balances on vehicles in the IRP-DPD flow formulation. Constraint 6.31 ensures that the vehicle capacity is not broken and that the vehicle routes from the solution of Z^{int} are fixed. The solution time for the formulation is short because it does not consider any routing decisions. We, therefore, propose to always run this operator strictly after phase 2, before any other operator is used. Running the operator will result in unchanged transportation costs but, in most cases, reduced holding costs.

The minimize pickup and delivery operator

This operator tries to move as much pickup and delivery quantity as possible from a chosen time period, k, to the other time periods. The main idea is to reduce the transportation costs in period k by improving the vehicle utilization of the other time periods. The problem is formulated as an LP problem that minimizes the amount of delivery and pickup in a chosen time period k, when every route in all time periods are considered fixed. To obtain the route fixation, the x_{ijt} variables in Z^{int} are described as parameters, A_{ijt}, as in the LP formulation of the inventory optimization operator. In fact, all the constraints in both formulations are identical. The objective functions, on the other hand, are different and the objective function for this operator is defined in 6.34. Here, we have defined a subset, $\mathscr{L}$, for all the customers visited in period k in the solution of Z^{int}.

$$\min \sum_{i \in \mathscr{L}} q_{ik} \tag{6.34}$$

After the LP problem is solved, the VRP-DPD has to be resolved for period k with the updated quantities for the given time period. This may generate better routes for period k and subsequently reduce the transportation costs. To improve the solution further, the inventory optimization operator is run right after the VRP-DPD. The new total cost is calculated from Equation 6.18 with respect to the updated vehicle routes and inventory.

There are different approaches to select what time period the operator should minimize pickup and delivery. The operator potentially reduces the transportation cost in a given time period at the expense of increased holding costs. We can, therefore, choose a time period where this increment will be as small as possible. This can be achieved by selecting the time period with the smallest reduction in delivery/pickup quantity required to decrease the number of routes needed by one. Mathematically, the expression of this argument is given by Expression 6.35, where Q_{it} and Y_{0t} are input parameters from the current value of Z_{int}. Only periods where the expression is strictly positive are evaluated.

$$\underset{t\in\mathcal{T}}{\arg\min}\left(\max\Big[\sum_{i\in\mathcal{N}^D} Q_{it} - K^Q(Y_{0t}-1), 0\Big] + \max\Big[\sum_{i\in\mathcal{N}^P} Q_{it} - K^Q(Y_{0t}-1), 0\Big]\right) \tag{6.35}$$

The minimize pickup and delivery operator can be run several times before a new operator is applied. If the time period selection is based on Expression 6.35 and the routes in period k are unchanged, the same time period will be selected over again. A tabu list that ensures that a new time period is selected in the next run is, therefore, suggested.

Another approach for selecting a time period is to choose the time period that allows for the largest changes in pickup up and delivery. The approach is stated mathematically in Expression 6.36, where Q_{ik} is the input parameter and q_{ik} is the variable output of the LP formulation solved for period k. Here, the LP described above must be solved for all time periods, setting each period equal to k in order.

$$\underset{k\in\mathcal{T}}{\arg\max}\sum_{i\in\mathcal{N}}(Q_{ik} - q_{ik}) \tag{6.36}$$

The minimize customer visits operator

Reduced transportation costs can also be achieved by reducing the number of customer visits, which is not necessarily achieved through the minimize delivery and pickup operator. Therefore a new Shift-Quantity operator is proposed which aims to minimize the number of node visits in a given time period. The formulation is similar to the minimize delivery and pickup formulation. The difference is the new Objective function 6.37, and that the Constraints 6.38-6.39 are added.

$$\min \sum_{i \in \mathscr{L}} \delta_i + \sigma \sum_{i \in \mathscr{L}} q_{ik} \tag{6.37}$$

$$q_{ik} - K^Q \delta_i \leq 0 \qquad i \in \mathscr{L} \tag{6.38}$$

$$\delta_i \in \{0, 1\} \qquad i \in \mathscr{L} \tag{6.39}$$

δ is a binary variable which takes the value 0 if a node that previously was visited in period k, is not visited in the new solution. Further, we do not want the deliveries or pickups which are not set to 0 to increase excessively. Therefore, we minimize over all q_{ik} as well. This part of the objective function is multiplied by σ, a small constant, in order to balance the priority of the different parts of the function. For this formulation too, the VRP-DPD must be resolved for time period k. The time period selection follows the same strategies as the minimize pickup and delivery operator. Lastly, a call to the inventory optimization operator is done.

The integration of routing and holding cost operator

This operator is formulated as a MILP that integrates the routing decisions for a chosen period, k, with the delivery and pickup decisions for the planning horizon. The formulation considers the routes from Z^{int} as fixed, for all time period except for period k. The idea is better utilization of vehicle capacity on the fixed routes, such that transportation cost in the chosen period can be minimized. This usually comes with the downside of increased inventory holding costs, however. The objective function and the constraints related to the routing decisions in period k are described by Objective function and Constraints 6.40-6.49. Constraints 6.20-6.33 presented previously are also added to the MIP formulation in order to restrict load balances and inventory balances. Constraint 6.31 is valid for all time periods except period k. For this period the equation is replaced with Constraint 6.46.

$$\min \sum_{(i,j)\in\mathscr{A}^{\mathscr{L}}} C^T_{ij} x_{ij} + \sum_{i\in\mathscr{L}} \sum_{t\in\mathscr{T}} C^H_i i_{it} + C^V r_0 \tag{6.40}$$

$$\text{s.t.} \sum_{j\in\mathscr{L}'} x_{ij} - \sum_{j\in\mathscr{L}'} x_{ji} = 0 \qquad i \in \mathscr{L}' \tag{6.41}$$

$$\sum_{j\in\mathscr{L}'} x_{ij} - y_i = 0 \qquad i \in \mathscr{L}' \tag{6.42}$$

$$t_0 = 0 \tag{6.43}$$

$$t_i - t_j + T_{ij} \leq (K^T + T_{ij})(1 - x_{ij}) \qquad (i,j) \in \mathscr{A}^{\mathscr{L}} \tag{6.44}$$

$$t_i + T_{i0} x_{i0} \leq K^T \qquad i \in \mathscr{L} \tag{6.45}$$

$$l^P_{ijk} + l^D_{ijk} \leq K^Q x_{ij} \qquad (i,j) \in \mathscr{A}^{\mathscr{L}} \tag{6.46}$$

$$y_i \in \{0,1\} \qquad i \in \mathscr{L} \tag{6.47}$$

$$0 \leq y_0 \leq V + r_0 \tag{6.48}$$

$$x_{ij} \in \{0,1\} \qquad (i,j) \in \mathscr{A}^{\mathscr{L}} \tag{6.49}$$

The objective function minimizes the transportation cost in the chosen time period and holding cost over all time periods. Constraints 6.41- 6.49 describes the routing decisions for the chosen period. As described by 6.47 the customers that were visited in period, k, are not restricted to be visited in this period, and customer removal is allowed. For choosing a time period, k, the same approaches as for the minimize pickup and delivery operator are suggested. The reason we do not solve the MILP for all time periods to find the one which improves the solution the most is related to the longer running time of the formulation.

The Route-Generation operator

The Shift-Quantity operators can potentially reduce the pickup and delivery in a given period. However, the operators' ability to shift quantity and thereby produce a new solution diminishes as the excess capacity of the existing routes becomes smaller. The Route-Generation operator overcomes this limitation by producing a solution that potentially has both new routes and an increased number of routes. The idea behind this operator is to use the routes of the existing solution to generate new routes that may improve the solution. The best of the generated routes are selected in a route selection problem. The following section describes a set of operations to alter routes. The operations are then combined in several procedures that generate new routes. Subsequently, the details on how the operator employs the route selection problem defined in Section 6.5 is discussed.

Producing new routes

To produce new routes four operations are defined that alter an existing route: Node removal, subgraph removal, node insertion and merging of routes. We define the subgraph of a route as any part of the route that does not include the depot and the arcs connected to the depot. Thus, a subgraph is a cycle-free directed path. For a route with n node visits and a subgraph of size k, where $1 \leq k \leq n$, there are $n - k + 1$ subgraphs. For instance, if all nodes in a route are included ($n = k$) there is only one subgraph. The subgraphs of a route of size five with a subgraph of size three are illustrated in Figure 6.2. The blue and orange circles illustrate the depot and customer nodes respectively in this and the following figures.

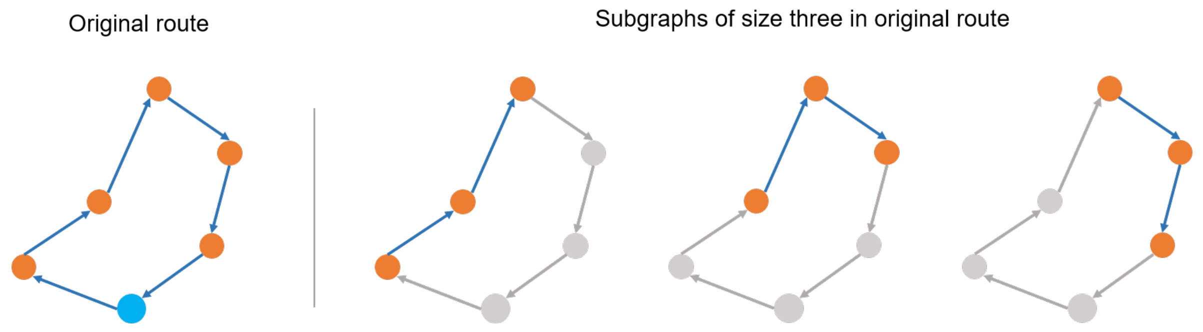


Figure 6.2: Illustration of the subgraphs of a route

All operations result in a path that is shaped as a cycle, in other words, a new route. By combining the different operations in procedures, a variety of new routes can be generated. All operations use a cheapest insertion/removal algorithm to decide which arcs to remove and insert. This algorithm is explained below. A description of the different operations then follows.

Cheapest insertion and removal

First, the algorithm checks all possible insertion and removal positions in a route. For insertion, this involves removing the arc between two connected nodes in the route and add an arc incident to and incident from the node to insert. Similarly, the node removal links the node visit prior to and after the node to be removed in the route. The removal of a node from a route is illustrated in Figure 6.3. The three dashed blue arcs in the figure are evaluated for each alternative route. The dashed grey arcs are removed and the dashed blue arc inserted. When all possible insertions and removals are evaluated, the algorithm inserts or removes the node in the position with the lowest increase or largest decrease in transportation costs.

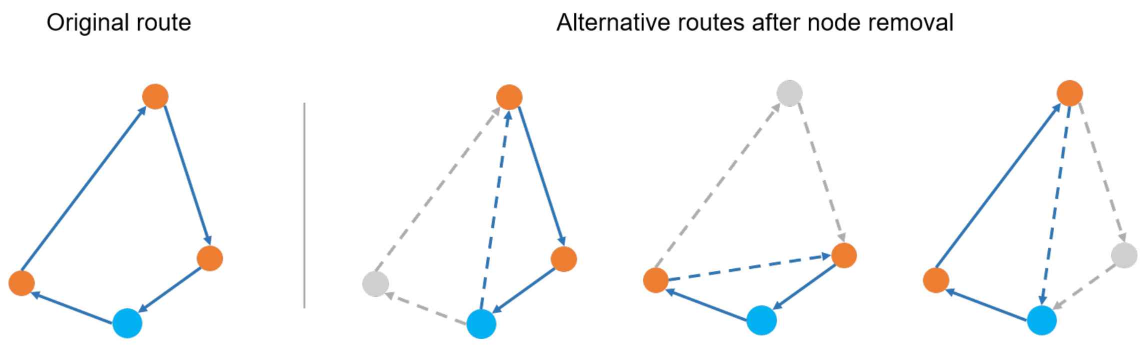


Figure 6.3: Illustration of the cheapest removal algorithm

Node removal and subgraph removal

The removal of a single node from a route is achieved by the cheapest removal algorithm. The node removal operation can be generalized to a subgraph removal operation which removes a larger part of a route. The cheapest removal algorithm then considers the arcs connecting the subgraph to the route. Thus, only three arcs for each position are evaluated no matter the size of the subgraph. This avoids favorizing the shortest subgraphs of a route which is the case if all arcs of the subgraph are considered. The operation with a subgraph of size k is not the same as applying the node removal operation k times because different arcs are considered in the two operations. The number of positions to consider for a subgraph of size k in the cheapest removal algorithm is the same as the number of subgraphs of size k in the route. The operation for a subgraph of two nodes and the corresponding routes after each potential removal is illustrated in Figure 6.4. In the figure, there are four different subgraphs to consider ($n = 5$, $k = 2$, 5-2+1 = 4). The three arcs that are evaluated for cheapest removal are dashed in the illustration.

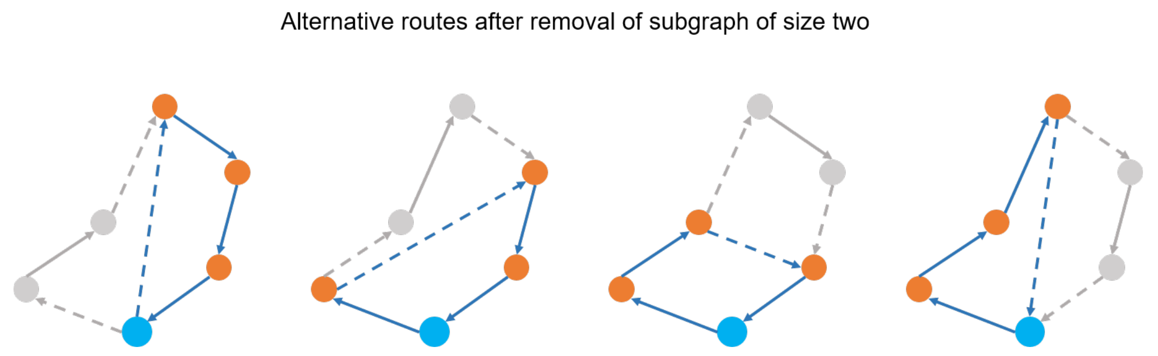


Figure 6.4: Illustration of the subgraph removal operation

Node insertion and merging of routes

The insertion of a single node is achieved by the cheapest insertion algorithm. As for the node removal, the node insertion operation can also be generalized to a subgraph insertion operation. This involves inserting a subgraph into a target route. Potential subgraphs to insert can be found in other routes. Let this be called the insertion route. Because the subgraph insertion involves two routes it is defined as a merge route operation.

A subgraph of the insertion route is connected to the target route by the cheapest insertion algorithm. As for the node insertion operation, the algorithm checks the addition of two arcs and the removal of one arc for every position between two nodes in the target route. The cheapest insertion of a subgraph is illustrated in Figure 6.5. In the figure, the dashed grey arc is removed and the dashed blue arcs inserted. The insertion with the lowest increase in transportation costs is selected. For a subgraph of size k and an insertion route of size n the cheapest insertion algorithm are run for every $n-k+1$ subgraphs of the insertion route and the cheapest insertion overall is selected. For each position between two nodes in the target route three arcs are considered.

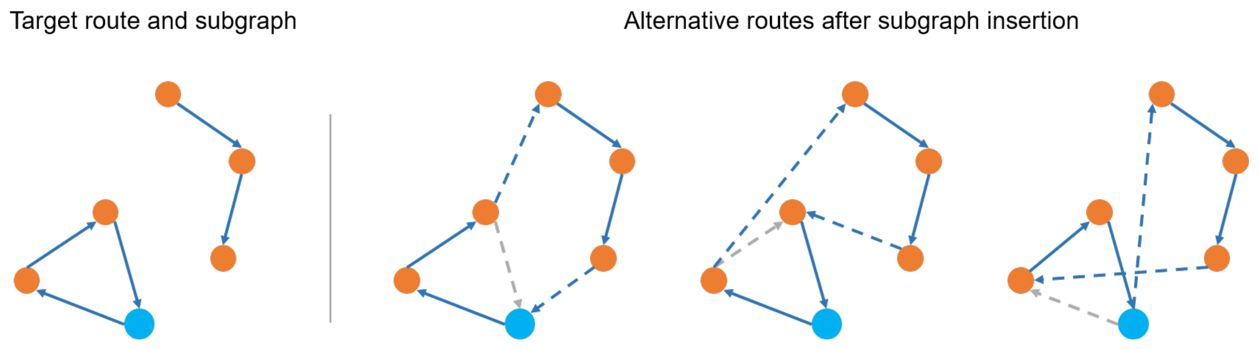

Target route and subgraph
Alternative routes after subgraph insertion

Figure 6.5: Illustration of cheapest insertion of a subgraph in a target route.

Procedures to generate new routes

The Route-Generation operator's efficiency depends on the quality of the new routes. Eight procedures listed below are proposed that combine the operations related to subgraph removal and subgraph insertion in different ways. The first four procedures slightly modify existing routes, as opposed to the more radical changes imposed by the combination of routes procedure. The least served procedures insert or remove nodes from routes where the delivery or pickup quantity is small relative to the other nodes in the route. Before a route from any procedure is added to the route pool, all routes produced are checked for duplicates among the existing routes. This eliminates symmetry in the route selection problem and reduces the problem size.

- Cheapest insertion/Cheapest removal
- Cheapest insertion and removal
- Double cheapest insertion and removal
- Least served removal/Least served insertion
- Restricted least served insertion
- Combination of routes

The first four procedures potentially produce $|\mathscr{R}|$ unique routes each. The cheapest insertion inserts a node in each route by the cheapest insertion algorithm. Likewise, the cheapest removal removes a node from each route by the cheapest removal algorithm. Cheapest insertion and removal inserts a node in each route and removes a different node by the same algorithms. The double version performs two insertions prior to removing a subgraph of size two from the route.

The least served removal procedure removes the least served node in a route in a given time period t. Given that each route visits more than one node the least served removal procedure produce $|\mathscr{R}|$ routes.

The least served insertion procedure is illustrated in Figure 6.6. The procedure considers the set of nodes not visited in a period which is called the insertion period. The same set of nodes are evaluated in the adjacent periods. In the figure, the faded nodes are visited in the insertion period. The opaque nodes are the nodes visited in adjacent periods that are not visited in the insertion period. The nodes with the least, strictly positive serving in each adjacent period are selected. These nodes are inserted in a route in the insertion period by the cheapest insertion algorithm. For a period with two adjacent periods, the least served insertion potentially yields two new routes as illustrated in the figure. However, if none of the nodes not visited in the insertion period are visited in adjacent periods, no new route is generated. All, but the last and the first period, have two adjacent periods. Hence the procedure potentially produces $2|\mathscr{T}|-2$ new routes.

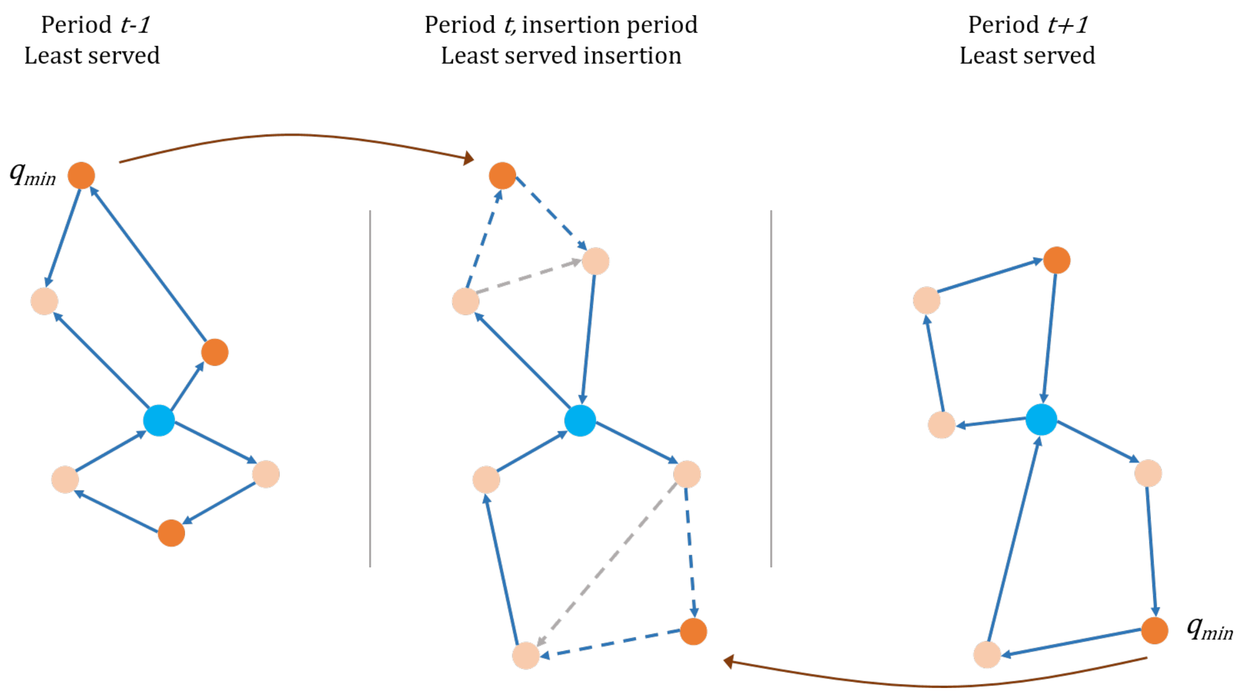


Figure 6.6: Illustration of the least served insertion procedure

A more comprehensive approach is also to create new routes for the periods adjacent to the insertion period as illustrated in Figure 6.7. Here, the least served nodes in $t-1$ and $t+1$ are removed from their respective routes and inserted in an adjacent period by the cheapest insertion algorithm. Here the nodes inserted in period t are also removed from its original route. This is not the same as the least served removal procedure because the set of nodes considered for removals are likely to be smaller. It is therefore called the restricted least served removal procedure. The value of a route depends on how it combines with the other routes, and this node moving approach may produce routes that work well together. Each period with two adjacent periods is considered for removal twice. The first and last period are considered once each. Thus the procedure may produce $2|\mathscr{T}|-2$ routes.

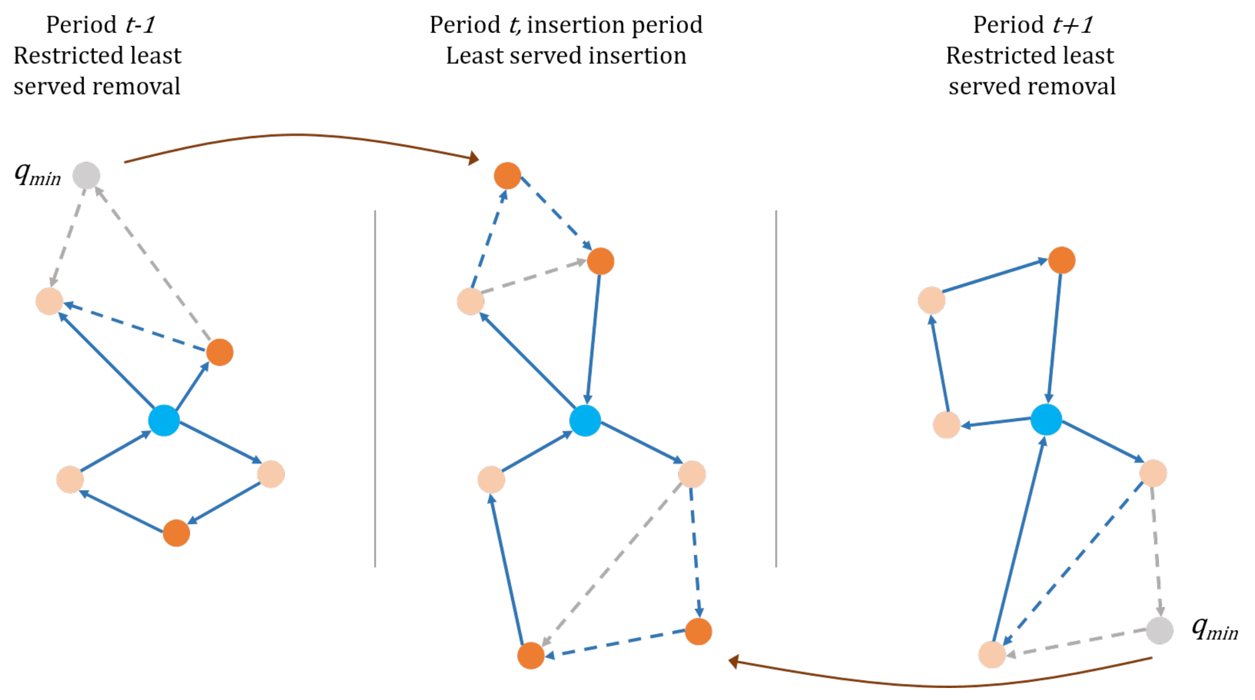

Period *t-1*
Restricted least served removal
Period *t*, insertion period
Least served insertion
Period *t+1*
Restricted least served removal
q_{min}
q_{min}

Figure 6.7: Illustration of the restricted least served insertion

The combination of routes involves a more elaborate series of steps and is summarized in Algorithm 5. In addition to keywords, the operations for subgraph insertion and removal are also bolded in the pseudocode. A pair of routes can consist of routes from two different time periods.

Algorithm 5 Combine routes

```
1    for all pair of routes (r1, r2)
2        remove node visits that coincide in r1 and r2 from r2.
3
4        Let $n$ be the number of node visits in r2
5        Let $k$ be a number s.t. $k \le n$
6
7        for all subgraphs of r2 of size s = 3, 4..k
8            merge(r1, subgraph) to new route r
9            remove subgraph of size s from r
10           If r has no duplicate, add new route to route pool
```

The route combination potentially produces several routes for each pair. In a route pool of size n there are $n(n-1)$ pairs. The procedure uses both the merge route and the subgraph removal operation. First, to avoid merging routes that visit the same node, any node that appears in both routes are removed from the insertion route(r2) in line 2. The procedure proceeds by merging the target route(r1) and a subgraph of the insertion route (line 8). The size of the subgraph is a parameter of the model and affects the number of routes generated. For the tests in this thesis k is set to 5. The merged route is likely to be longer than an optimal route. Hence the subgraph removal operation is applied to the new route (line 9). It removes a subgraph of the same size inserted but is not allowed to remove the subgraph that was inserted. If the resulting route has no duplicate, it is added to the route pool (line 10).

The combine routes algorithm may produce an immense amount of routes if the number of pairs and set of subgraph sizes is large. The implementation employs several rules to limit the number of routes generated. First, if more than 30% of the nodes coincide between the routes, the pair of routes are not combined. The rationale behind merging routes is that the subgraphs that are merged into a target route have appeared in good routes. If too many nodes are removed from a route, it is hard to say anything about the quality of the resulting route. Secondly, only routes that go in the same direction are combined. The direction of a route is defined as being clockwise or counterclockwise based on the angle between the incoming and outgoing arc from the depot as illustrated in figure 6.8. Often routes that go in opposite direction yield a merged route with crossing arcs.

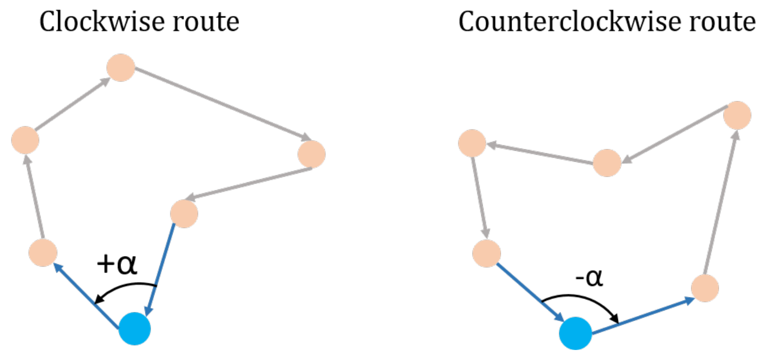


Figure 6.8: Illustration of clockwise and counterclockwise route

Route selection

After the route generation, the route pool contains all routes generated by the different procedures. A route selection problem is solved to select the best routes in the route pool. As this is an intensification search, the original routes are left in the route pool. Therefore, if the problem does not pose any restrictions on the routes to select, the route generation operator can only produce an equal or improved objective value. A potential approach to avoid stagnation in the intensification search and to discover more of the solution space is to require at least one of the routes in the input solution to change. Although the objective value worsens in some iterations good routes may be discovered or even better, the apparent deterioration of the solution could lead to a new best solution in subsequent iterations. However, this approach may lead to cycling between solutions. To avoid this, we introduce a tabu list for the generated routes. Each route that goes out of a solution is tabu to regenerate. That way the operator can never return to the same solution. A too large route pool may slow the route selection problem substantially. A potential approach to overcome this is to impose additional constraints to the problem. Locking routes to periods, forcing routes to be used or reduce the route pool beforehand are some alternatives. The selection is the final step of the route generation operator and the new solution is returned.

6.4 The diversification search

A search for better results in the neighborhood of the current solution is done during the intensification search. There is no guarantee that the best solutions are in the neighborhood of the solution given by the relaxed IRP-DPD and the corresponding VRP-DPDs. Therefore a search in different parts of the solution space is done by altering the relaxed IRP formulation before it is resolved. The concept behind the iterative process of the search is described by Algorithm 6.

Algorithm 6 The iterative matheuristic with a one-shot improvement heuristic

```
1   Objective value of $Z^* \leftarrow \infty$, $n$= 1
2   do
3       Solve phase 1
4       Solve phase 2 and update constructed solution $Z_n$
5       Update tabu list with respect to solution $Z_n$
6       Initialize the value $Z^{int}$= $Z_n$
7
8           do intensification search
9           while intensification stop criterion not met
10
11      Add diversification constraint to the solution of $Z_n$
12      $n++$
13  while diversification stop criterion not met
14
15  Solve the route optimization problem and update $Z^*$
```

The stop criterion in the diversification search is defined by a time limit, depending on instance size. Different diversification constraints are proposed that require changes from the previous constructed solution. The first types require changes to the set of visited and not visited nodes. A combination of the two requires a certain number of changes. The last constraint utilizes more information as it requires changes to routes in the previous construction solution. In addition to constraints that require some degree of change, the problem may be modified by altering the objective function. Excluding holdings costs, allowing it to increase for a fixed amount is an additional option.

Tabu list

The diversification constraints only prevent the last construction solution to reappear. To avoid cycling of the same solutions, another diversification mechanism must be put in place. This is achieved by locking variables which were just changed for a number of iterations in a tabu list. However, by locking all the changed variables, there will be a significant build-up of fixed variables. This will lead to a decreasing number of variables that are free to change after a few iterations which can make the problem infeasible. Therefore, each changed variable has only a 20 % chance of being locked, meaning that an element of randomness is added. A changed variable which got locked will also have a 20 % chance of being locked in the next iteration. Any variable is subject to fixation for two periods only. This tabu list approach makes it less likely that we cycle the same solutions and at the same time give the procedure room for finding good solutions. Every iteration also has its unique tabu list. Therefore, even though the same solution is encountered in subsequent iterations, the different history pushes them in different directions.

The introduction of diversification constraints and the tabu list may induce infeasible relaxed IRP problems. If this happens, the tabu list is redrawn and the change parameters in the diversification constraints are relaxed.

Diversification constraints

Diversification can be achieved by fixating certain y_{it} variables in order to make the solution of the next iteration significantly different. Three methods are proposed for variable fixation.

Force customer visits

One option is to require some customer nodes that were not visited in a period to be visited by fixating certain y_{it} variables to 1. This gives a solution that is unambiguously different from the previous one. Fixating specific nodes makes very hard constraints, and it may, therefore, be smarter to add a more flexible constraint over a set of nodes when the problem is resolved. Hence, a set of all the nodes which were not visited in the previous solution of phase 1 is created for each time period, and a subset of these unvisited nodes must be visited in the next solution. The size of the subset indirectly depends on the size of the instance and directly on a weight parameter α.

Set	
$\mathcal{N}_t^U$	Set of unvisited nodes, i, in period t in the previous solution of phase 1.
Parameter	
α	Weight of fixation

$$\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}_t^U} y_{it} \geq \left\lceil \sum_{t\in\mathcal{T}} \alpha \mid \mathcal{N}_t^U \mid \right\rceil \tag{6.50}$$

Force customer to not be visited

Another alternative is to require some customer nodes that were visited in a period not to be visited by fixating a number of the nodes to 0. As mentioned above, fixation of nodes is very strong constraints. Consequently, we look at the set of nodes which were visited in the previous solution and require a subset of these to be 0.

Set	
$\mathcal{N}_t^V$	Set of visited nodes, i, in period t in the previous solution of phase 1.
Parameter	
β	Weight of fixation

$$\sum_{t\in\mathcal{T}}\sum_{i\in\mathcal{N}_t^V} y_{it} \leq \left\lceil \sum_{t\in\mathcal{T}} \beta \mid \mathcal{N}_t^V \mid \right\rceil \tag{6.51}$$

Minimum number of changes

In order to explore different parts of the solution space, one can also combine the two ideas mentioned above. We will in this case, as seen in 6.52, require a minimum amount of change, $\delta \mid \mathcal{N} \mid$, where δ is the change factor, and $\mid \mathcal{N} \mid$ is the number of nodes in the instance.

$$\sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}_t^N} y_{it} + \sum_{t\in\mathcal{T}} \sum_{i\in\mathcal{N}_t^V} 1 - y_{it} \geq \delta \mid \mathcal{N} \mid \tag{6.52}$$

The change factor, δ, enables us to control how much the new iteration should diversify from the previous solution.

Minimum number of changes to routes

The previous constraints are added in order to produce a different solution after phase 2. However, the constraints can be satisfied by changing only some of the routes in the initial VRPs, leaving a large part of the solution the same. Hence, the intensification search may often find the same best solution after a few iterations. Therefore, constraints that prevent the same initial routes from being produced are introduced. Let $\mathcal{R}$ be the set of routes in the constructed solution. Let $\mathcal{N}_{rt}$ be the set of nodes, depot not included, visited by a route $r \in \mathcal{R}$ in period $t \in T$. Also, let α_r be a binary variable indicating whether route $r \in \mathcal{R}$ is changed or not. The following constraints are added to the IRP-DPD problem.

$$\sum_{i\in\mathcal{N}_{rt}} y_{it} - \left(|\mathcal{N}_{rt}| - 1\right) \leq 1 - \alpha_r \qquad r \in \mathcal{R}, t \in \mathcal{T} \tag{6.53}$$

$$\sum_{r\in\mathcal{R}} \alpha_r \geq \beta|\mathcal{R}| \tag{6.54}$$

$$\alpha_r \in \{0, 1\} \qquad r \in \mathcal{R} \tag{6.55}$$

Constraint 6.53 states that if α_r is equal to 1 at least one of the nodes in the route must be omitted. If α_r is equal to zero the constraint imposes no restrictions on the route. Constraint 6.54 sets the minimum number of routes that need to be changed. β is a parameter of the model and must be set in the interval [0, 1]. Setting β equal to 0 require no route changes. The opposite, letting β be 1 means that all routes need to be changed.

Reduction of transportation costs

One challenge with the heuristic described in this thesis is that the relaxed IRP-DPD tends to underestimate transportation costs. The constructed solutions therefore often have higher transportation costs and usually lower holding costs than the optimal solution. We propose a way of handling this by only minimizing the transportation costs when the relaxed IRP-DPD is resolved in the next iteration of the diversification search. The new objective function is given in 6.56.

$$\min \sum_{(i,j)\in\mathscr{A}} \sum_{t\in\mathscr{T}} C_{ij}^{T} x_{ijt} \tag{6.56}$$

A restriction on how much the costs can increase from the initial solution is added to the formulation. By doing this, some of the information given in the first solution is utilized. The constraint is given in 6.57.

$$\sum_{t\in\mathscr{T}} \sum_{i\in\mathscr{N}} C_{i}^{H} i_{it} \leq \gamma H_{tot} \tag{6.57}$$

H_{tot} is the total holding costs of the first solution, while γ is the increment factor dictating how much the holding costs are allowed to increase. The new formulation is used every time the relaxed IRP-DPD is resolved.

The reduction of transportation cost method is always run in the iteration of the two-phase approach prior to the fixation of variable method. This is due to the fact the first method will most likely significantly alter the initial solution.

6.5 The improvement heuristic

Before the final solution is returned, we propose in improvement heuristic which consists of a route selection problem. The idea behind the route selection is to see if there is a combination of routes found in all iterations which is better than the combination used in the best-found solution. The set of promising routes consists of all routes which improved the current solution of an iteration. This means that each iteration provides several routes to the route set even though the iteration does not find a new overall best solution. The solution of the route optimization problem will always produce a solution which is as good or better than the current best-found solution. In this problem, a route is nothing but a sequence of visited vertices and holds no information regarding quantities delivered or which day the route is performed. The cost of a route is the sum of the costs of the edges used in the route. Naturally, several of the solutions probably use many of the same routes, but a route is still only represented once in the route set. The followings new sets, parameters and variables are defined for the route selection formulation.

Route selection formulation

Sets	
$\mathscr{R}$	Set of all routes
Indices	
r	routes
Parameters	
C_r	Cost of route r
A_{ijr}	Parameter that takes value 1 if arc $i-j$ is used by route r

Variables

$$x_{tr} \quad \begin{cases} 1, & \text{if route } r \text{ is used on day } t \\ 0, & \text{otherwise} \end{cases}$$

The objective function for the route selection formulation is stated by 6.58.

$$\min \sum_{t \in \mathcal{T}} \sum_{i \in \mathcal{N}} C_i^H i_{it} + \sum_{t \in \mathcal{T}} \sum_{r \in \mathcal{R}} C_r x_{tr} \tag{6.58}$$

The objective is followed by Constraints 6.20- 6.33 presented earlier. Though, Constraint 6.31 is replaced with 6.59. In addition we add Equation 6.60-6.62 to the formulation. Note that all routes are valid, so the time restrictions do not need to be handled in the route selection formulation.

$$l_{ijt}^P + l_{ijt}^D \leq K^Q \sum_{r \in \mathcal{R}} A_{ijr} x_{tr} \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{6.59}$$

$$\sum_{(i,j) \in \mathcal{A}} A_{ijr} C_{ij} = C_r \tag{6.60}$$

$$\sum_{r \in \mathcal{R}} x_{tr} \leq V \qquad t \in \mathcal{T} \tag{6.61}$$

$$x_{tr} \in \{0,1\} \qquad t \in \mathcal{T}, r \in \mathcal{R} \tag{6.62}$$

6.6 Other versions of the IRP

The IRP-DPD easily transforms to other variants of the IRP. There is nothing about the matheuristic presented in this section that suggests it is not going to perform equally well for different alternatives of the problem. The divisible model formulation in this thesis can be reduced to the regular IRP if the set of pickup nodes is empty. The problem then becomes to serve a set of delivery customers over a period of time, constrained by customers' inventory limits, duration of the routes and vehicle capacity. The IRP with mixed linehauls and backhauls (IRP-MB) is equal to the divisible problem except that each customer requires either pickup or delivery, but not both. That is each customer is represented by one node, and there are no co-located nodes. Because the IRP-DPD in this thesis is modelled with two nodes for each customer, the heuristic can easily be used to solve the IRP-MB problem. In the computational study we also test the matheurstic for the regular IRP and the IRP with mixed linehauls and backhauls.

Chapter 7

Instance generation

In this chapter, the generation of test instances from which the results are derived is described. In order to test the model formulations and to compare the different solution methods, we have generated a number of test instances. The instances were created by a script programmed in C++, which constructs feasible instances based on predefined rules. The script centers the depot in the origin and assigns each customer an x and y coordinate, randomly generated in a range of -50 to 50. For each customer location (x, y), a pair of co-located nodes is created. The distance, d_{ij}, between two customers is defined as the Euclidean distance between their coordinates. The cost, C_{ij}, between customers is linearly associated with a cost both dependent on the distance, d_{ij}, and a fixed cost, s, related to the service cost at the next node visit. The travel time between customers is linearly dependent on distance and a fixed service time m. Because the distance between co-located nodes is zero, the cost between these nodes is only associated with the service cost, s. Co-located nodes share the same coordinates, so the cost of driving from a node i to a delivery node and its co-located pickup node is the same. The generation of transportation cost and travel time between customers and co-located nodes is described in Figure 7.1. In the figure, D and P symbolize a delivery node and its co-located pickup node respectively.

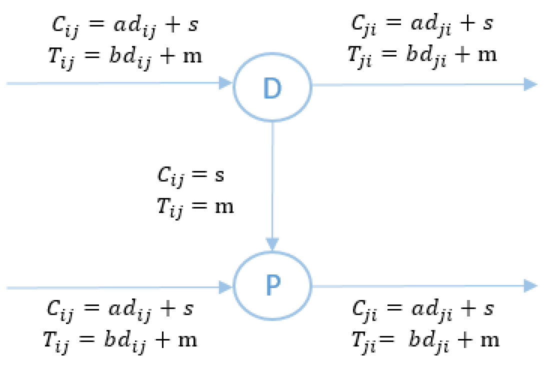


Figure 7.1: Cost and travel time between customers and co-located nodes

For our case we let a take the value of 13 and b take the value of 1. Next we set the service cost s equal 100 and the service time m equal 20. The parameter values are decided from a practical aspect, as a relation between travel cost in Norwegian kroner related to travel time in minutes per km. We have generated a large data set varying the number of customers, vehicles and the length of the planning horizon. The other instance parameters are defined as follows:

- Number of customers C; varying between 6 and 70.
- Number of vehicles V; varying between 3 and 10, dependent on the number of customers.
- Time horizon T; equal to 5 periods.
- Variants of a given combination of C and V is named N: equal to 1,2,3,4,5.
- Customer production/consumption, R_{it}, is drawn randomly from a discrete uniform distribution in the interval [5, 50].
- Inventory holding cost, C_i^H, is drawn randomly from a discrete uniform distribution in the interval [1, 5].
- The vehicle capacity, K^Q, is set to $\lfloor \sum_{i \in \mathcal{N}} \sum_{t \in \mathcal{T}} R_{it} / (|\mathcal{V}||\mathcal{T}|) \rfloor$.
- Maximal duration of a route, T, is set to 480.
- The maximal inventory level is drawn randomly from a discrete uniform distribution in the interval [75, 100].
- The lower inventory limit for pickup nodes, L_i^P, is equal to 0, while delivery nodes, L_i^D, is equal to 5.

- The initial inventory is drawn from a discrete uniform distribution between the upper and lower limit of the specified customer.

Note that the number of customer nodes is doubled from the number of customers. An instance of six customers results in twelve customer nodes, six delivery nodes and six pickup nodes. The length of the planning horizon is set from a practical aspect, equal to a workweek. The test instances are named after the number of customers, the number of vehicles and the length of the planning horizon. For each combination of C and V we have generated a number of five instances. Each variant for a given combination has randomly generated new values for customer locations, demand, upper inventory limits, holding costs and initial inventory. To better evaluate the performance of the different solution methods, most test results presented in Chapter 8 are described from an average of the five variants of test instances for the given combination. As an example, the test instance described as $C_{12}V_3A$ represents an average of five test instances with twelve customers and three vehicles over a planning horizon of five days. An instance named $C_{12}V_3N_1$ describes variant one for the given combination. The vehicle capacity is inspired by the formulation used in Coelho and Laporte (2013), though this formulation only considers delivery demand. The maximal time length of a route is set from a practical point, presenting a working day of eight hours.

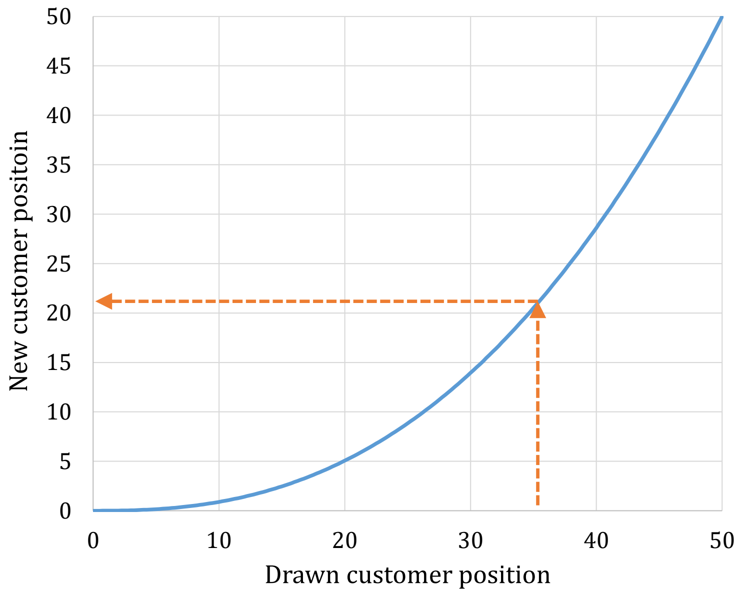
Various geographies

The divisible option may be more beneficial for some types of geographies than others, as pointed out by Nagy et al. (2015). To create instances with clustered customers a set of clusters $\mathscr{C}$ is created. Then a random position is drawn for each cluster. For $\mathscr{N}$ customers $|\mathscr{N}|/|\mathscr{C}|$ customers are assigned each cluster and positioned randomly in a small distance from their respective cluster position. For the tests in the computational study 24 and 30 customers are distributed among 4 and 5 clusters respectively, with random coordinates in a range of -10 to 10 from their cluster position.

To create instances where the majority of customers are located in proximity to the depot a random coordinate c in the interval [0, 50] is first drawn. The number is transformed to a new coordinate by Equation 7.1. The value is rounded up to avoid the same coordinate as the depot. The transformation is illustrated in Figure 7.2. Due to the convex shape of the function, the closer the original coordinate is to the depot the more it gravitates towards the depot. Finally, the coordinate is randomly assigned a positive or negative sign.

$$Coordinate(c) = \left\lceil \frac{50}{50^{2.5}} * c^{2.5} \right\rceil \tag{7.1}$$

Figure 7.2: Illustration of how customers are more likely to be positioned in proximity to the depot



Chapter 8

Computational study

The purpose of this chapter is to present an overview of the results for the IRP-DPD solved by both the exact solution method presented in Chapter 5, and the matheuristic solution method proposed in Chapter 6. The arc-flow formulation, all valid inequalities, the branch and cut algorithm and the matheuristic are implemented in C++ and compiled under Visual Studio 2015. A description of the program design is provided in Appendix D. Mosel Xpress-IVE (64-bit version) is used to solve all MILPs and LPs and are accessed through the Xpress BCL library and its APIs. The computational experiments are carried out on a 64-bit Windows 10 PC with Intel(R) Core(TM) i7-7770 CPU 3.60 GHz processor and 32.0 GB (31.9 GB usable) RAM. The time limit is set to 3600 seconds for instances up to 24 customers. For the largest instances, the time limit is set to 7200 seconds. A test instance is either run to the maximum running time or solved to optimality. The Presolve function integrated with the Xpress Optimizer is turned on for all test instances.

The computational results are organized after the solution method used. First, the results from the exact solution method are presented in Section 8.1. The effectiveness of the proposed valid inequalities and the branch and cut algorithm are evaluated. Following in Section 8.2, the results of the matheuristic solution method are presented. Lastly, in section 8.3, the solution methods are compared, besides the divisible option in the model is evaluated.

8.1 Exact solution method

This section presents the results of adding the valid inequalities introduced in Section 5.1 to the IRP-DPD flow formulation. The valid inequalities are both tested separately and in combinations. The branch and cut algorithm is used to dynamically generate and add subtour elimination constraints to the problem. An overview of the abbreviations used for the valid inequalities is presented in Table 8.1.

Table 8.1: Notation for test instances and valid inequalities

Valid Inequalities	
Inventory levels (Inequalities 5.3- 5.4)	IL
Number of node visits (Inequalities 5.5- 5.6)	NN
Minimum flow through a subset of size 1, 2 and 3 (Inequalities5.5- 5.8)	MF
Minimum flow through a subset of size 2 and 3 (Inequalities 5.7- 5.8)	$MF2$
Subtour elimination constraints (Inequality 5.10)	SEC
All valid inequalities	Ω
No valid inequalities	Ψ

For those instances where optimality is proven within the time limit, the performance of the formulation is evaluated from the *solution time*, the running time for solving to optimality. For those instances where the solution time exceeds the time limit, the performance is described by the dual gap %, the LP gap % and the best-found solution. The dual gap % is defined as the percentage gap between the optimistic bound of the IP solution and the best IP solution found. The LP gap % represents how much a valid inequality or a new solution method improve the LP relaxation. An LP solution with no cuts gives a percentage of 100, while an LP relaxation equal to the optimal IP solution yields a percentage of 0. Thus, the smaller the percentage, the more effective the cut is. More formally, the LP gap % is defined as:

$$\text{LP gap \%} = \frac{\text{Optimal IP solution} - \text{LP relaxation with cut}}{\text{Optimal IP solution} - \text{LP relaxation without cut}} \tag{8.1}$$

There are instances where the optimal IP solution is not known. In these cases, we use the best-found IP solution from the IRP-DPD flow formulation with the subtour elimination constraint added. The LP relaxation in 8.1 is the objective value from the flow formulation where the binary requirement on the arc flow variable is relaxed.

The LP gap % consists of holding costs and transportation costs. The transportation costs and holding costs are decomposed in two terms, T and H, according to equation 8.2. The LP gap in Equation 8.1 is decomposed similarly according to Equation 8.3. LP' refers to the LP relaxation with a valid inequality added.

$$\mathrm{T} + \mathrm{H} = \frac{\text{IP trans. cost-LP trans. cost}}{\text{IP cost-LP cost}} + \frac{\text{IP hold. cost-LP hold. cost}}{\text{IP cost - LP cost}} = 1 \tag{8.2}$$

$$\mathrm{T}\left(\frac{\text{IP trans. cost-LP' trans. cost}}{\text{IP trans. cost-LP trans. cost}}\right) + \mathrm{H}\left(\frac{\text{IP hold. cost-LP' hold. cost}}{\text{IP hold. cost-LP hold. cost}}\right) = \text{LP gap \%} \tag{8.3}$$

First, we present the results from the valid inequalities added separately to the IRP-DPD flow formulation. The performance results for test instances up to 24 customers are presented in Table 8.2. All results in the table are an average of the five different instances generated for the given combination of customers and vehicles. The inequality with the best performance for a given evaluation criterion and instance is made bold in the table. We have also noted the number of instances solved to optimality for those cases where some, but not all of the given instances for a given combination are solved to optimality. For instance sizes where some combinations did not find a feasible solution, the number of solutions found is also listed.

Table 8.2: Results from test cases with single valid inequalities added to the flow formulation

Instance	Info	Ψ	IL	NN	MF	SEC
C_6V_3A	LP gap	100 %	83.5 %	69.8 %	**66.4 %**	100.0 %
	Sol. time	40.2 s	30.6 s	35.4 s	33.8 s	**15.8** s
C_8V_3A	LP gap	100 %	86.8 %	66.3 %	**64.7 %**	100 %
	Sol. time	204 s	228.4 s	141.6 s	168.2 s	**86.4 s**
$C_{10}V_3A$	LP gap	100.0 %	82.5 %	72.6 %	**69.7 %**	100.0 %
	Dual gap	0.56 %	0.75 %	0.62 %	0.44 %	**0.00 %**
	Best sol.	26002.8	26035.2	25994	25996.6	**25992.4**
	# solved to opt.	2/5	3/5	2/5	2/5	**5/5**
	Sol. time	-	-	-	-	**773.6 s**
$C_{12}V_3A$	LP gap	100 %	83.4 %	71.0 %	**67.5 %**	100 %
	Dual gap	3.05 %	2.39 %	3.14 %	3.19 %	**1.15** %
	Best sol.	30400.4	30270.4	30408.8	30440.8	**30156**
$C_{18}V_3A$	LP gap	100.0 %	85.8 %	79.7 %	**75.6 %**	100 %
	Dual gap	-	13.82 %	-	14.39 %	**3.79 %**
	Best sol.	-	44428.6	-	45072.6	**41703.4**
	# sol. found	4/5	**5/5**	4/5	**5/5**	**5/5**
$C_{24}V_5A$	LP gap	100 %	85.0 %	79.1 %	**77.1 %**	100 %
	Dual gap	-	-	26.67 %	23.06 %	**9.94 %**
	Best solution	-	-	69924	66835.2	**58261.6**
	# sol. found	4/5	4/5	**5/5**	**5/5**	**5/5**

Because the subtour elimination constraints are added dynamically, it is hard to evaluate the actual improvement of the LP gap for this constraint. The LP relaxation in the formulation for the LP gap is calculated without binary requirement on the arc variables, and therefore the improvement of the constraint cannot be covered by this evaluation criterion. Though the effect of this inequality is seen from the improved solution time, lower dual gaps and best found IP solutions. As seen in the table, with respect to the dual gap and the best-found IP solution, the subtour elimination constraint performs better than the other valid inequalities for all variants of instances.

The valid inequality related to minimum flow through a subset of customers gives the lowest LP gap for all test instances. The smallest LP gap comes at the cost of the high complexity of the inequality, and the inequality never performs best with respect to solution time, dual gap and best found IP solution. The valid inequality related to inventory level has larger LP gap than

NN and MF, but sometimes performs better than these two valid inequalities with respect to other evaluation criteria. The following is a discussion of how the valid inequalities affect the problem and when they are likely to produce good results.

Inventory level

This valid inequality imposes restrictions on the amount of inventory, given that the nodes are not visited in subsequent periods. At delivery nodes, the inequality sets the minimum inventory required to cover future demand. At pickup nodes, it sets an upper bound on the inventory. From the computational results, this inequality reduces a substantial part of the LP gap. Evident from Figure 8.1 the inequality mainly affects the holding cost part of the gap. In the figure, five variants for each given instance size is described. As illustrated in the figure the inequality performs poorly on instances where nearly the entire gap consists of transportation costs. For some instances, we see that the gap related to holding costs almost is completely closed by the inequality.

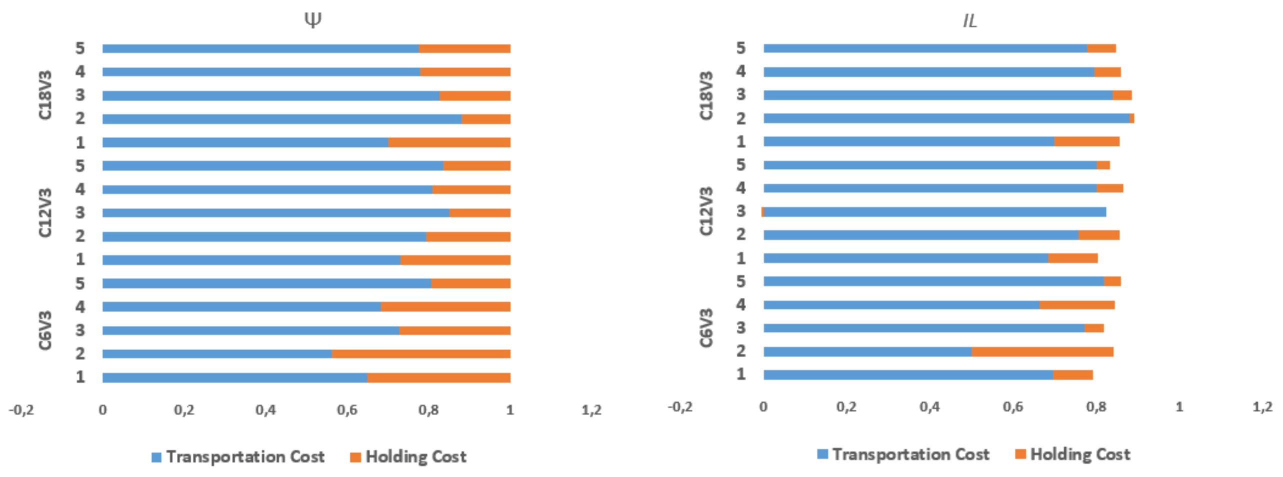


Figure 8.1: Gap composition of the LP relaxation for Ψ and for IL

During tests, we also observed that the inequality worked better at delivery nodes because the inventory requirement from this constraint may be substantially higher than the inventory balance given by Constraint 4.7. In the LP relaxation, the y-variables are minimized to reduce the transportation costs. If the y is close to zero over a period this constraint implies that inventory should cover almost all demand within this period and hence must be increased. When the inventory level is higher, the holding cost increases and results in an LP cut. Because of the incentive to keep inventory low, the upper bound on the inventory at pickup nodes is likely to show little effect.

Minimum number of node visits and flow through a subset of customers

The purpose of the valid inequality NN is to establish a lower limit on the number of visits to each node. The LP relaxation is minimizing transportation costs by setting the y-variables to the smallest fractional value that suffices to cover what is delivered or picked up. Table 8.2 shows that the valid inequality is effective for many instances. This is expected because the vehicles should deliver and pick up an amount close to the excess demand to keep the inventory at a low level. For pickup nodes, the constraint is somewhat less effective because the vehicles pick up more than necessary in the first period to decrease the inventory to its lower limit. As described in Chapter 5 the valid inequality MF is an extension of the valid inequality NN, where NN only considers subsets of size one. In addition, MF also considers flow through subsets of two and three customer nodes. Though, the additional cuts induced by including these subsets seems to affect the dual gap of the solution negatively. As seen in Figure 8.2, MF mainly affects the transportation cost part of the LP gap. The average LP gap for the different valid inequalities is illustrated in Figure 8.3.

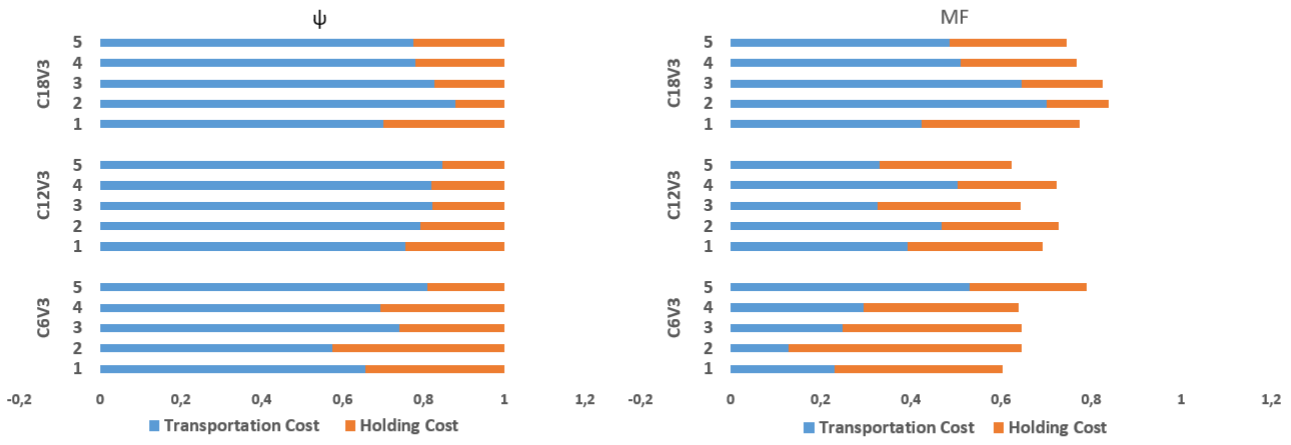


Figure 8.2: Gap composition of the LP relaxation for Ψ and for MF

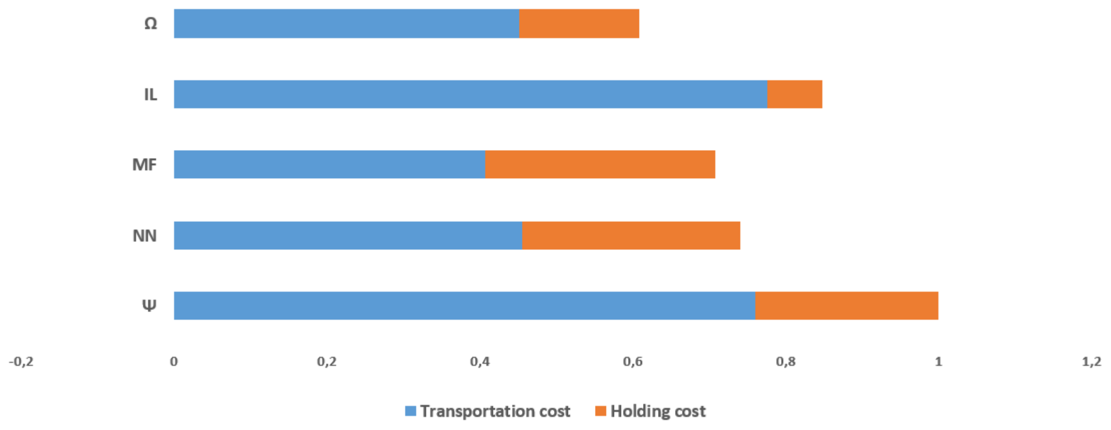


Figure 8.3: Average LP gap for the different valid inequalities

Another consideration is on how many nodes NN should be implemented. If $\frac{E_{it_1t_2}}{min(Q^K, U_i - L_i)}$ in Equation 5.5-5.6 is close to the following integer, the rounding in the constraint has limited effect. Figure 8.4 shows the effect of constraint NN on subsets of nodes. All nodes in the subset have a remainder after dividing excess demand by the capacity that is lower than a certain limit. The graph shows the average cut and number of constraints over all instances as a percentage of implementing the constraint for all nodes. For instance, setting the upper limit on the remainder to 0.4 cuts away 98 percent of the full cut with only half the constraints. Also, the sample variance between instances is low. For all results derived in this thesis, we have set the upper limit on the remainder to 0.3.

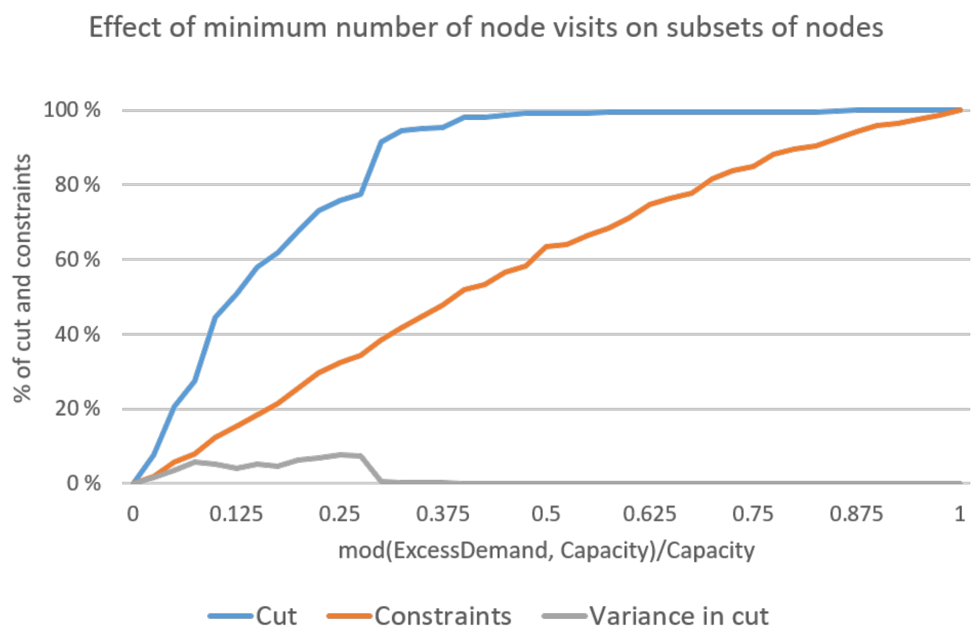


Figure 8.4: Effect of minimum number of node visits for nodes with mod(ExcessDemand, Capacity)/Capacity less than a limit=[0, 1].

The subtour elimination constraint

The subtour elimination constraint given by 5.10 is tested for different parameter settings. First, the lower value η of the edges included in the graph that is searched for strong components are varied. Secondly, the degree of subtour violation α is tested for different values. For varying edge weights, α is fixed to 0.1. From Table 8.3 and 8.4 we see that η and α equal to 0.1 gives the lowest dual gap for $C_{18}V_3A$. These parameter values also worked well when tested for smaller instances. Therefore all results presented in Table 8.2 have used this parameter setting for instances up to 18 customers. For $C_{24}V_5A$ an η value of 0.25 and an α value of 0.1 yield the best results. These parameter settings are used for instances with 24 or more customers in Table 8.2.

Table 8.3: Subtour elimination constraint for different values of η

Instance	$\alpha = 0.1:$	$\eta = \epsilon$	$\eta = 0.1$	$\eta = 0.25$	$\eta = 0.5$
$C_{18}V_3A$	Dual gap	9.23 %	**4.62**%	5.85%	6.41%
	Number of cuts	4832.2	3419.6	2898.6	1166.4
	Number of nodes	4818.8	26921.6	29265	33211.4
$C_{24}V_5A$	Dual gap	14.13%	10.15%	**9.26 %**	11.23%
	Number of cuts	2415.8	1671.2	868.4	514.4
	Number of nodes	1564	2719.6	10307.8	9555.8

Table 8.4: Subtour elimination constraint for different values of α

Instance		$\alpha = \epsilon$	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$
$C_{18}V_3A$	Dual gap	4.62%	**3.95**%	4.22%	4.30%
$\eta = 0.1$	Number of cuts	2175.6	1848.4	1221.0	1006.0
	Number of nodes	36153.6	34345	31577.2	30614.8
$C_{24}V_5A$	Dual gap	11.26%	**8.75**%	10.88%	11.67%
$\eta = 0.25$	Number of cuts	1026.8	917.2	775	604.4
	Number of nodes	12129.8	9660	10473.8	11672.2

Combinations of valid inequalities

Table 8.5 presents the results from tests of combinations of valid inequalities. All instances in the table are described from an average of the five different instances generated for the given combination. Ω in the table refers to all valid inequalities. In addition to this combination, we have excluded the valid inequality from Ω in turn. As an example, $\Omega - IL$ refers to all valid inequalities except the inequality IL. For $\Omega - MF2$, the valid inequality NN is included in the results, but constraints related to flow through subsets of size two and three are removed from Ω.

Table 8.5: Results from test cases with combinations of valid inequalities

Instance	Info	Ω	$\Omega - IL$	$\Omega - MF$	$\Omega - SEC$	$\Omega - MF2$
C_6V_3A	LP gap	**56.42 %**	66.4 %	83.51 %	**56.42 %**	59.57 %
	Sol. time	24.6 s s	21.2 s	**15.2** s	55.6 s	15.4 s
C_8V_3A	LP gap	**56.8 %**	64.7 %	86.8 %	**56.8 %**	58.6 %
	Sol. time	85.6 s	81.4 s	**71.2** s	179.8 s	71.6 s
$C_{10}V_3A$	LP gap	**59.01 %**	69.7 %	82.5 %	**59.01 %**	61.9 %
	Dual gap	**0.00 %**	**0.00 %**	**0.00 %**	0.44 %	**0.00 %**
	Sol. time	**948.2 s**	1048.0	962.2	-	1315.6
$C_{12}V_3A$	LP gap	**57.6 %**	67.5 %	83.4 %	**57.6 %**	61.2 %
	Dual gap	**0.77** %	0.88 %	0.98 %	2.45 %	0.91 %
	Best sol.	30107.2	30122.2	30138.2	30287.8	**30106.6**
$C_{18}V_3A$	LP gap	**62.5** %	75.6 %	85.8 %	**62.5** %	66.6 %
	Dual gap	4.43 %	5.03 %	4.46 %	13.38 %	**3.25 %**
	Best sol.	41927.6	42269.8	41916.8	44321.8	**41510.2**
$C_{24}V_5A$	LP gap	**67.2** %	77.5 %	85.2 %	**67.2** %	69.0 %
	Dual gap	8.10 %	10.96 %	7.51 %	-	**6.92** %
	Best sol.	57064.6	58960.2	56625	-	**56289.6**
	# sol. found	**5/5**	**5/5**	**5/5**	4/5	**5/5**

From Table 8.5 we see that the combination $\Omega - MF2$ is a strong combination, especially for the larger instances. Excluding the extra complexity related to flow through subsets of size two and three is effective. The small extra cut in the LP relaxation related to the constraint seems to come at the cost of higher complexity and solution time. The table also shows the low performance of Ω-SEC and state the importance of including the branch and cut algorithm in the exact method.

Further, it is interesting to compare the results from Table 8.5 with the results presented in Table 8.2. A summary of all test results for the exact solution method is described in Table 8.6. The table lists the valid inequality with the best performance of a given test instance for different solution criteria. From the table, we see that in general $\Omega - MF2$ performs best for the largest instances.

Table 8.6: Best performance for valid inequalities with different evaluation criteria

Criteria	C_6V_3A	C_8V_3A	$C_{10}V_3A$	$C_{12}V_3A$	$C_{18}V_3A$	$C_{24}V_5A$
Sol. time	$\Omega{-}MF$	$\Omega{-}MF$	SEC	-	-	-
Dual gap	-	-	-	Ω	$\Omega - MF2$	$\Omega - MF2$
Best sol.	-	-	-	$\Omega - MF2$	$\Omega - MF2$	$\Omega - MF2$

8.2 Matheuristic

In Chapter 6, a matheuristic was introduced as an alternative to the exact solution method. In this section, test results for different parameter values in the matheuristic are presented. The different phases and operators are tested both separately and combined. First, the test results for different parameter settings in phase 1 and 2 for different instance sizes are presented. Next, the intensification search is tested with varying parameter settings for the different operators. Combinations of the operators with the best performance are also evaluated. Next, the matheuristic is tested and evaluated for a number of iterations through the diversification search. Lastly, the improvement heuristic is applied to the matheuristic. One purpose of the matheuristic is to increase the running time efficiency. Therefore, the different phases and operators are also evaluated with respect to running time. The running time of the different phases and operators are tested when all parameters are set to their determined values. The parameter values and the time budget of the matheuristic are summarized in Appendix C.

Construction of a solution

As stated in Section 6.1, the inventory decisions taken in phase 1 can be improved by adding SECs because it better resembles a true solution to the problem. Figure 8.5 illustrates a period solution for the relaxed IRP for a $C_{24}V_5$ instance. A blue color scale represents the arc variables' relaxed solution values. A darker arc means a greater solution value. The left figure illustrates the arc values in one period, while the right figure shows the subtours in the graph. Figure 8.6 illustrates the solution from the same period with SECs added. In contrast to the solution without SECs, the arc-flow is to a greater extent connected to the depot. Also, any cycle that appears does not form a subtour. As for the exact solution approach we have tested different values for the α in Constraint 5.10. A more thorough presentation of this results and the final parameter settings are described in Appendix C.

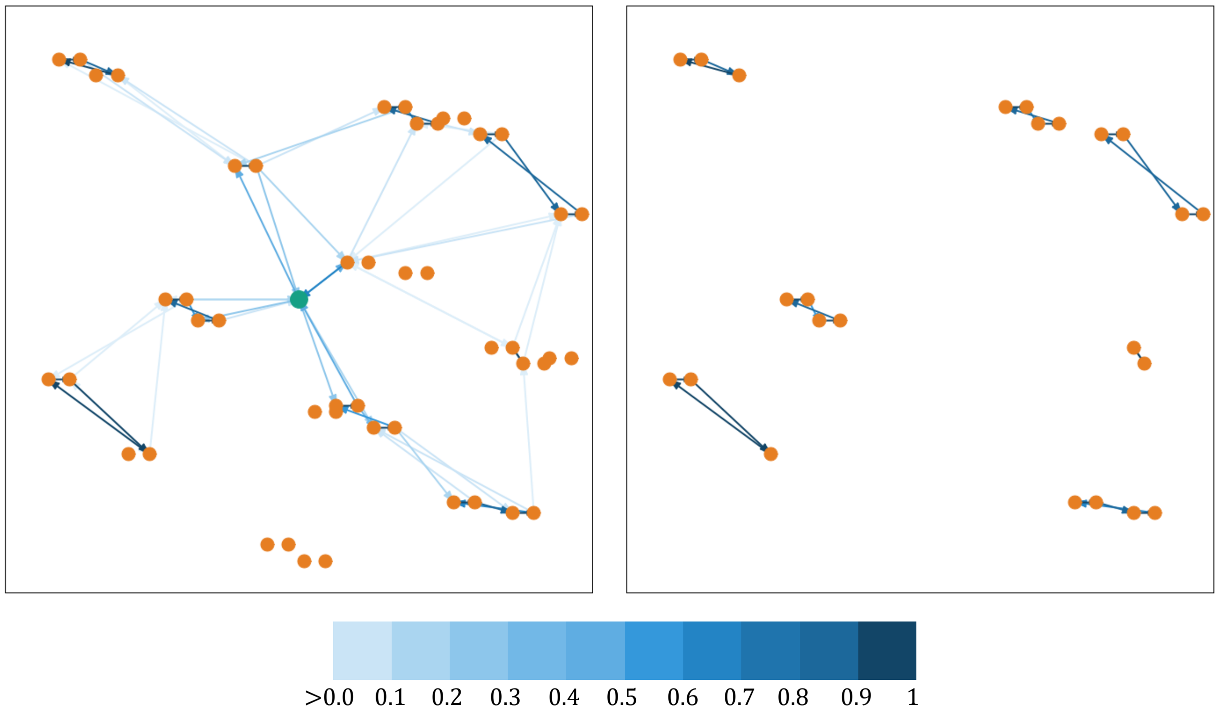


Figure 8.5: Period solution to the relaxed IRP formulation without subtour elimination.

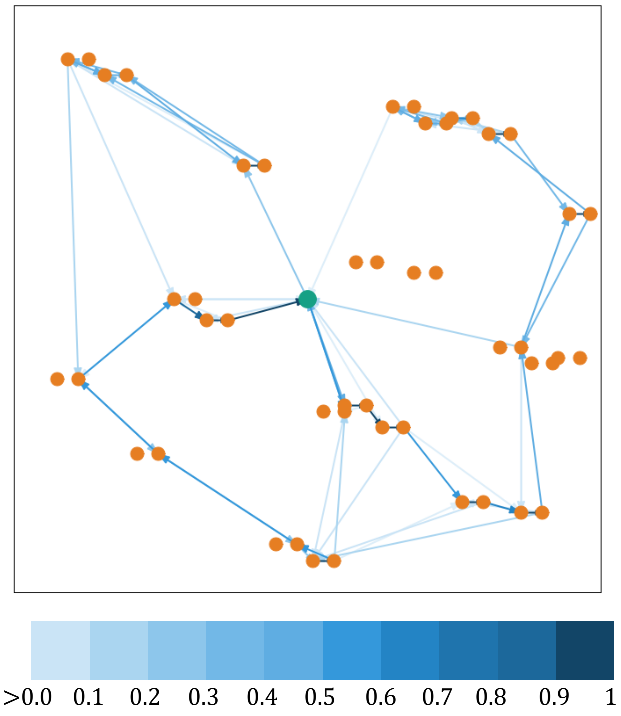


Figure 8.6: Period solution to the relaxed IRP formulation with subtour elimination.

In Section 6.1 we proposed different approaches to solve the relaxed IRP-DPD. The results from these approaches are summarized in Table 8.7. The divisible approach(Div) has relaxed the arc variables of the original problem. The same problem is also solved with subtour eliminating constraints(Div+SECs). The simultaneous approach solves the IRP with the simultaneous arc set and the constraints introduced to track actions at customer nodes(Sim). The same problem is solved with subtour eliminating constraints (Sim+SECs). Also the simultaneous model with the binary variables related to customer visits relaxed is tested(SimRel). The data is the average of the five instances of each instance type in their first diversification iteration. IP dual gap is the percentage gap from the construction solution after the VRPs are solved to the best lower bound from the exact method. The solution time is the time spent on the relaxed IRP problem with a maximum of 240 seconds. Relaxation gap is the dual gap if the relaxed IRP is not solved to optimality. The IP-LP gap is the percentage increase from the relaxed IRP solution to the constructed solution. Note that for the divisible models, given that they solve to optimality, the solution is a lower bound on the problem and thus the IP-LP gap also serves as an IP dual gap for these solution approaches.

Table 8.7: Construction solutions' dual gap from best lower bound

Solution approach		Div	Div+SECs	Sim	Sim+SECs	SimRel
$C_{12}V_3A$	IP Dual gap	5.00%	2.95%	5.28%	**2.90**%	6.19%
	Sol. time	100.2 s	48.8 s	34.2 s	25.8 s	5.6 s
	Relaxation gap	-	-	-	-	-
	IP-LP gap	12.29%	3.86%	11.74%	3.91%	1.57%
$C_{18}V_3A$	IP Dual gap	7.23%	**4.33**%	7.84%	4.47%	8.93%
	Sol. time	-	-	-	181.6 s	33 s
	Relaxation gap	0.95%	2.63%	1.61%	-	-
	IP-LP gap	12.49%	2.94%	13.03%	3.11%	1.46%
$C_{24}V_3A$	Dual gap	9.16%	No sol	9.59%	**7.98**%	10.44%
	Sol. time	-	N/A	-	-	-
	Relaxation gap	1.64%	N/A	4.52%	4.95%	0.39%
	IP-LP gap	12.45%	N/A	12.97%	4.52%	2.13%

Evidently, the models with the SECs added produce better constructed solutions. However, the divisible model with SECs is significantly slower than the simultaneous model with SECs to close the dual gap. For 24 customers the DIV+SECs does not find any feasible solution in 240 seconds and illustrates the costs of a more constrained solution space. SimRel is the simplest model to solve, reflected by the lowest running time. Though, it also produces the worst constructed solutions. Also, note that the IP-LP gap is much smaller when SECs are integrated with the model. This suggests that the relaxed solutions with SECs are much closer to a solution of the original problem, which indeed was the goal of introducing SECs in the first place.

The costs of SECs are illustrated by the relaxation gap. For larger models, in particular, the gap is greater than for the models without SECs. Figure 8.7 illustrates the issue with a too large relaxation gap. The figure shows the relaxation gaps of Div(grey color) and Div+SEC(blue color) for the five instances with 24 customers. The orange points shows the improvement in construction solution from Div to Div+SEC for each instance. The instances in the left figure are run for 400 seconds, the right figure for 600 seconds. The dual gap of Div is rather stable for both solution times. However, the gaps of Div+Sim varies to a great extent. Note how the improvement in objective value tends to be negatively correlated with the dual gap. If the dual gap is too large, as it often is for 400 seconds, the solution from Div+Secs is inferior to the solution of Div. When the solution time increases the relaxation gap is decreased, and so is the IP improvement. Therefore, if the SECs are used with the models, the relaxed model must be given enough time to reduce the relaxation gap in order to produce better solutions than without SECs.

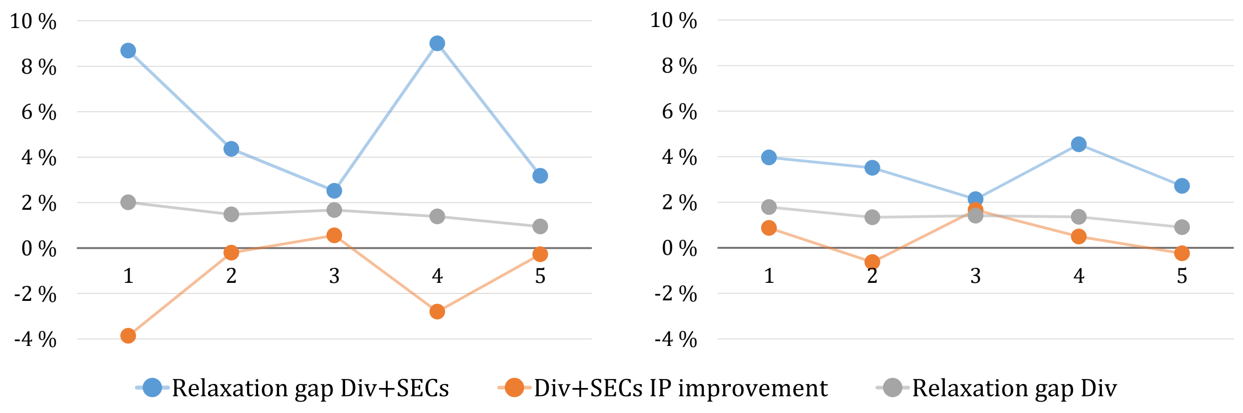


Figure 8.7: LP gap with and without subtour elimination constraint for 400 and 600 seconds.

Given the results, Sim+SECs yields superior construction solutions in acceptable running time. Thus this solution approach is used to solve the relaxed IRP in further tests.

Running time relaxed IRP-DPD and VRP-DPD

Even though the two-phase approach decomposes the original problem into two simpler subproblems, the computational time of each phase can still be problematic. Preliminary tests have shown that the increased computational time of the relaxed IRP-DPD in phase 1 usually is due to the solver having difficulties closing the dual gap. However, it is in general quick at finding good solutions with reasonably small dual gaps. Of course, it is not time-efficient to spend a considerable amount of time trying to close a small dual gap. This is especially true when one considers the fact that there is no guarantee that the optimal solution of phase 1 gives a better starting point for the VRPs in phase 2 than any other feasible solution. Therefore, a thorough analysis has been performed to determine the ideal maximum computational time in phase 1 for instances up to 30 customers. The results are presented in Table 8.8. The table describes how the solution of the relaxed IRP-DPD in phase 1 and the constructed solutions of phase 1 and 2 are affected by different time limits in phase 1. Three different instance sizes have been tested and the numbers presented are an average of five different instances. *Dual Gap relaxed IRP* in the table, refers to the average dual gap of the relaxed IRP-DPD in phase 1. The gap will naturally decrease monotonically as the computational time increases. Likewise, *Total improvement* refers to the percentage improvement in the constructed solution after both phases with the specified solution time for phase 1. The solution after 60 seconds, 120 seconds and 240 seconds serves as the baseline. Here, the percentage will not behave monotonically. All VRPs in phase 2 are run with a maximum computational time of 10 minutes in these tests.

Table 8.8: Improvement of the constructed solution for different time limits of the relaxed IRP-DPD

	Maximum computational time for phase 1 (in seconds)						
$C_{18}V_3A$	60	120	240	600	900	1200	1800
Solution time	60 s	105.8 s	182.6 s	401.2 s	587.6 s	779 s	1131.4 s
Dual Gap relaxed IRP	3.62%	2.53%	2.46%	1.67%	1.20%	1.06%	0.76%
Total Improvement	-	-0.02%	-0.63%	0.13%	-0,02%	0.51%	0.14%
$C_{24}V_3A$							
Solution time	60 s	120 s	240 s	600 s	900 s	1200 s	1800 s
Dual Gap relaxed IRP	No sol	6.47%	5.99%	4.88%	4.73%	4.41%	3.67%
Total Improvement	No sol	-	1.16%	0.89%	1.13%	1.42%	0.95%
$C_{30}V_5A$							
Solution time	60 s	120 s	240 s	600 s	900 s	1200 s	1800 s
Dual Gap relaxed IRP	No sol	No sol	6.78%	6.27%	5.69%	5.59%	5.07%
Total Improvement	No sol	No sol	-	-0.34%	-0.15%	-0.74%	-0.08%

Clearly, there are few advantages of letting the relaxed IRP-DPD in phase 1 run for a large amount of time for instances up to 30 customers. However, the relaxed IRP need sufficient time to find a feasible solution. The improvement of the constructed solution is rather modest and as mentioned above it is not very beneficial to spend time trying to close the dual gap of the relaxed IRP-DPD when the objective value of the resulting solution is rather stable. We believe the time is more wisely spent on trying to improve the constructed solution or to run additional iterations. Based on the results from Table 8.8 we set a maximal running time of phase 1 to 240 seconds for instances smaller than and equal to 24 customers. For instances equal to 30 customers we set the maximum time to be 360 seconds, to ensure that the relaxed IRP finds a feasible solution.

Also, various maximum computational times for the VRP-DPDs in phase 2 are tested. In these tests, the maximum computational time for the IRP-DPD is set to 240 seconds for 18 and 24 customers and 360 seconds for 30 customers. Here, the maximum computational time for phase 2 refers to the time limit of each single VRP-DPD and not the entire phase. The results are presented in Table 8.9.

Table 8.9: Improvement of the solution for different time limits of phase 2

	Maximum computational time for each VRP-DPD in phase 2 (in seconds)										
$C_{18}V_3A$	30	45	60	90	120	180	240	300	450	600	900
Total improvement	-	0 %	0 %	0 %	0 %	0 %	0.16 %	0.16 %	0.17 %	0.26 %	0.38 %
$C_{24}V_3A$											
Total improvement	-	0.15 %	0.22 %	0.29 %	0.29 %	0.30 %	0.34 %	0.38 %	0.41 %	0.42 %	0.44 %
$C_{30}V_5A$											
Total improvement	-	0.31 %	0.57 %	0.66 %	0.79 %	0.99 %	1.04 %	1.15 %	1.24 %	1.25 %	1.27 %

Evident from the results, the improvement in the objective value diminishes for large running times. For each VRP-DPD the maximum running time is set to 60 seconds for instances smaller and equal to 24 customers, and 90 seconds for 30 customers.

The intensification search

In this section, we explore the different operators and search strategies for the intensification search and see how they improve the solution. The notation used in the test results for the different operators, strategies and procedure are presented in Table 8.10.

Table 8.10: Notation for the heuristic

Shift-Quantity	
The minimize pickup and delivery operator	$SQ1$
The minimize customer visits operator	$SQ2$
The integration of routing and holding cost operator	$SQ3$
Selection of time period with highest quantity shifting	MS
Selection of time period with lowest excess in vehicle capacity	LE
Route-Generation	
Generating routes for all procedures of route generation	RG
Original route	ORG
Cheapest insertion	INS
Cheapest removal	REM
Cheapest insertion and removal	$INS+REM$
Double insertion and removal	$2INS+2REM$
Least served removal	$LSREM$
Least served insertion	$LSINS$
Restricted least served removal	$RLSREM$
Combination of routes	$COMB$
Route selection problem	RS

If a number of operators are combined, we name the combination by merging the notation. As an example, the notation $SQ3\_LE\_RG$ describes the $SQ3$ operator that selects the time period with the lowest excess in vehicle capacity and is combined with the Route-Generation operator.

The Route-Generation operator

The Route-Generation operator, RG, is tested for instances up to 24 customers and the results are presented in table 8.11. The results are derived from an average of the five instances of each type, where an instance is run for five diversification iterations. The intensification proceeds as long as an earlier solution is not encountered or the number of intensification iterations exceeds five. The Shift -Quantity operator is not applied in the intensification. The Route-Generation operator requires at least one change in each intensification iteration. Thus the operator may worsen the objective value of the input solution.

The IP improvement in table 8.11 is the average improvement in objective value from the construction solution in each diversification. The improvement tends to decrease for larger instances with the same number of vehicles. The naturally longer routes make it harder to alter the routes in a way that improve the current solution. For 24 customers the number of vehicles is increased, causing shorter routes and more routes. Shorter routes are easier to transform to new good routes. Also, the number of generated routes are greater because the average number of routes in each period is greater. This may explain why the IP improvement increases from 18 to 24 customers. However, for more vehicles, the construction solution may be further away from the optimal solution, making it easier for the operator to improve the solution.

One potential issue with *RG* is the time spent on the route selection problem. For these instances, the number of routes is well below 200 and the route selection problem's solution time is not an issue. However, reduction of the route pool may be appropriate for larger instances or if more route generation procedures are added.

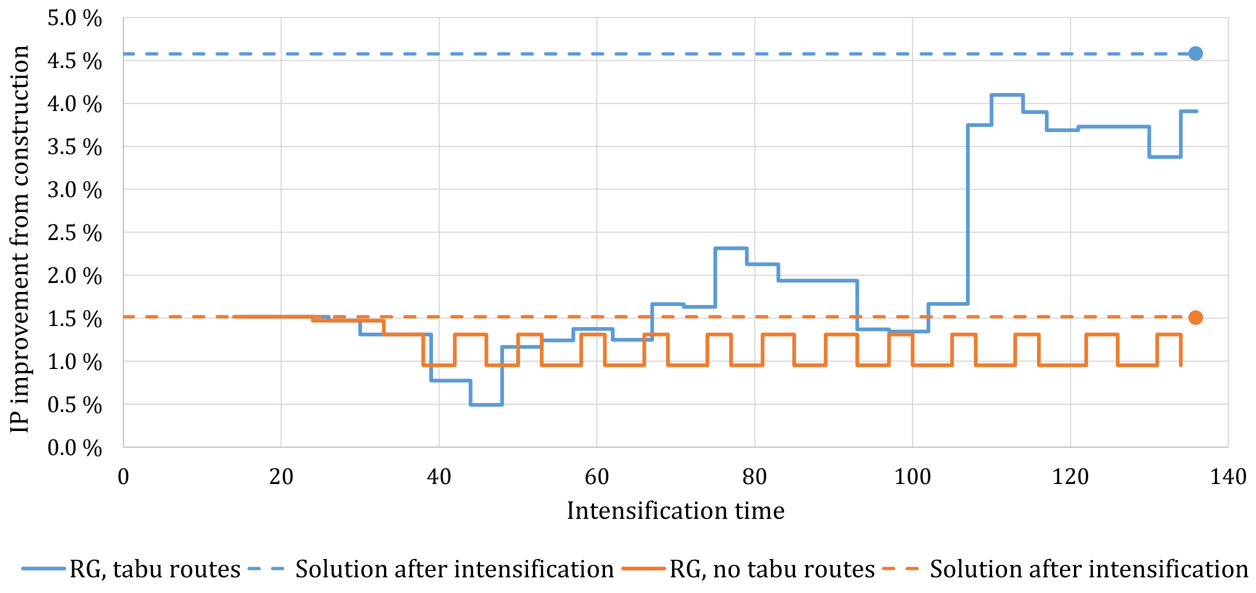
Table 8.11: Results from the route generation test

Instance	C_8V_3A	$C_{10}V_3A$	$C_{12}V_3A$	$C_{18}V_3A$	$C_{24}V_5A$
IP impr.	5.01%	5.31%	4.77%	2.45%	3.83%
Gen. routes	76.4	83.5	88.3	136.5	170.2
$\lvert R_t \rvert$	1.58	1.51	1.53	1.84	2.50
Route length	5.14	5.70	6.25	9.73	8.76

As mentioned in section 6.3, *RG* is likely to be subject to cycling if routes from previous solutions are regenerated. Figure 8.8 illustrates the implication of tabu routes in the intensification iterations. The figure shows the IP improvement from the construction solution. Each endpoint or vertical change on the solid lines represents a new solution. The dashed line is the improvement after the final route selection in the intensification. The orange lines are solution values from the route generation operator with no tabu routes. Evidently, routes are regenerated as the majority of the intensification is spent cycling between two solutions. The blue line shows the intensification of the same instance where routes that leave a solution are tabu to regenerate.

Initially, the solution becomes inferior to the old approach, but in subsequent iterations it significantly outperforms the operator with no tabu routes. Also, note that the solution after the final route selection in the intensification is better than any solution during the intensification iterations. That is, there is a combination of routes from all intensification iterations that is better than the routes that appeared in one particular iteration. With tabu routes more unique routes are generated; hence the final route selection has more routes to select from and better chances to improve the solution.

Figure 8.8: Route generation operator with and without tabu routes



The Route-Generation operator employs the variety of procedures listed in Table 8.10. The number of routes generated by a procedure can be calculated beforehand, but the actual number is less because duplicates are discarded. The average proportion of each route type after the route generation is shown in Figure 8.9.

The number of routes generated is stable for most procedures except the route combination procedure which increases with instance size. The number of vehicles is the same for all instances up to 18 customers, which implies longer routes in the solution for the larger instances. Longer routes give rise to more combinations, explaining the increased number of routes produced by the route combination procedure. Also, the order in which the route generation procedures are applied bias the distribution because subsequent procedures are more likely to have duplicates in the route pool.

The Route-Generation operator finishes by selecting the best combination of the routes generated. Thus, if a route is in the solution after the selection, this indicates that the procedure that generated that particular route improved the objective value. To evaluate the performance of each route generation procedure the proportion of its generated routes in the final solution is measured. These results are summarized in figure 8.9 as well. Evidently, the route selection is stable between the instance types. Note that routes from the simple insertion and simple removal procedures (INS and REM) are selected more often than simple insertion and removal (INS+REM). Given that the average length of the routes does not differ from those in the optimal solution INS and REM will on average produce too long and too short routes respectively. However, these procedures involve the least possible change to already good routes, which could explain why they outperform simple insertion and removal.

The combination route type (COMB) has the largest proportion of routes in the route pool. However, this proportion is substantially less in the selected routes. Combination of routes involves substantial changes to the existing routes, and many of them are therefore not selected. For the same reasons, this procedure may also produce routes that could drastically improve the solution. As long as the size of the route pool is not an issue the procedure should, therefore, be included in the Route-Generation operator.

The restricted least served removal (RLSREM) is selected more often than least served removal (LSREM). This may indicate that it combines better with the least served insertion procedure, which was the intention of this procedure indeed.

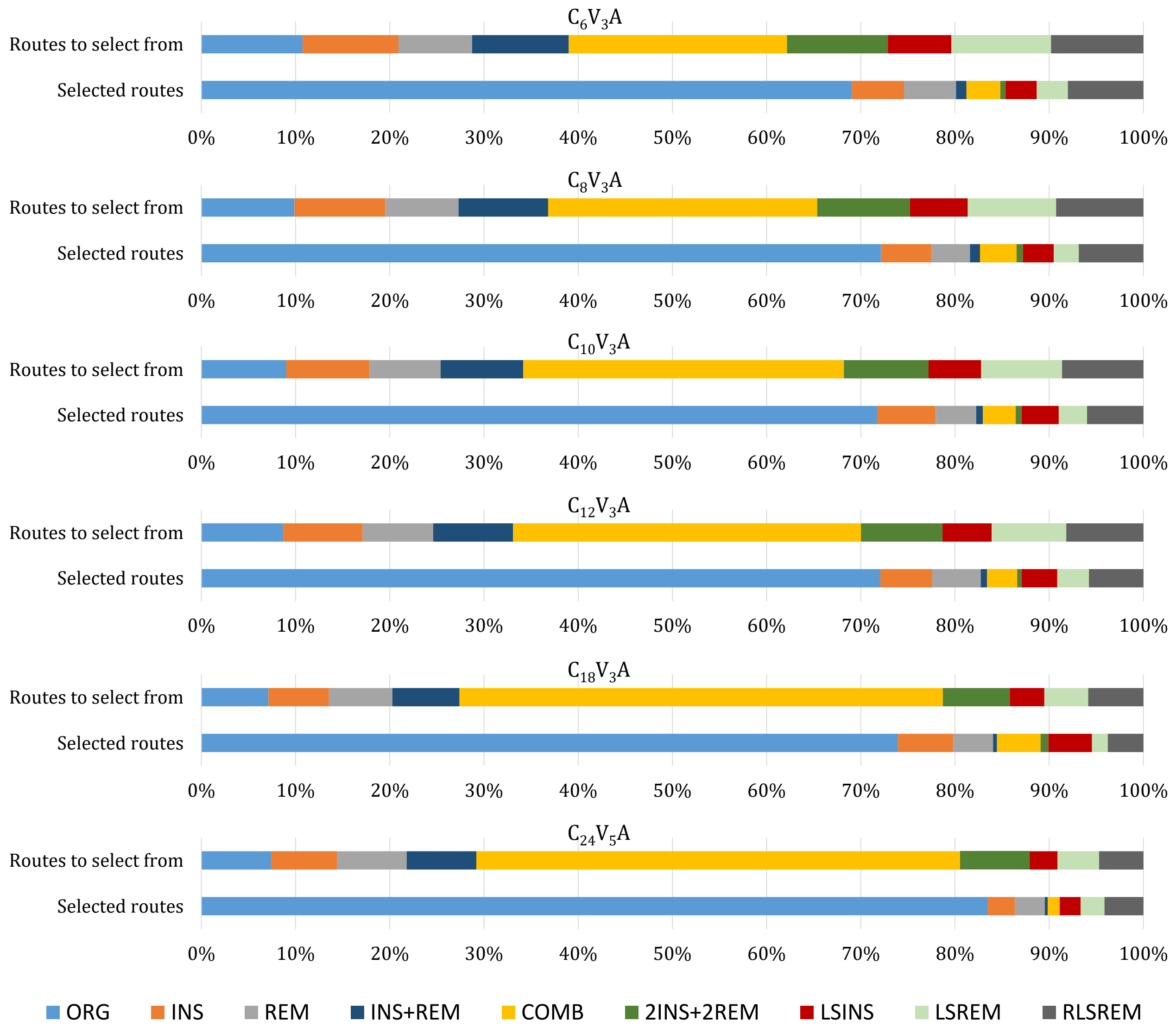


Figure 8.9: Proportion of routes generated and selected

Combinations of operators and time period selections

In Section 6.3 we introduced a number of Shift-Quantity operators and different search strategies for time period selection. These operators and search strategies are evaluated separately and in combination with the Route-Generation operator in this section. The integration of routing and holding costs operator, $SQ3$, is complex and closing the dual gap can be time-consuming for larger instances. A maximum running time of 120 seconds is therefore used for this operator. For the results derived in this section, the first constructed solution, Z_1 initializes the value of Z^{int} in the search. The results are from one diversification iteration.

The first Shift-Quantity operator we introduced was the inventory optimization operator. This operator strives to minimize inventory holding all routes from the current solution as fixed. The inventory holding costs are minimized with respect to inventory limits and load balances on vehicles considering the fixed routes. In preliminary testing we observed that this operator worked as expected. The transportation cost remained unchanged while a small decrease in holding costs occurred. Also, the operator finishes within seconds even for larger instances. This is as expected because no routing decisions are considered. For all results presented in the following section this operator is, therefore, always run at the beginning of the search before the other Shift-Quantity operators are applied.

Figure 8.10 illustrates how the best known IP solution for instance $C_{12}V_3N_2$ is improved with respect to solution time in the intensification search for a selection of different operators, search strategies and combinations of them.

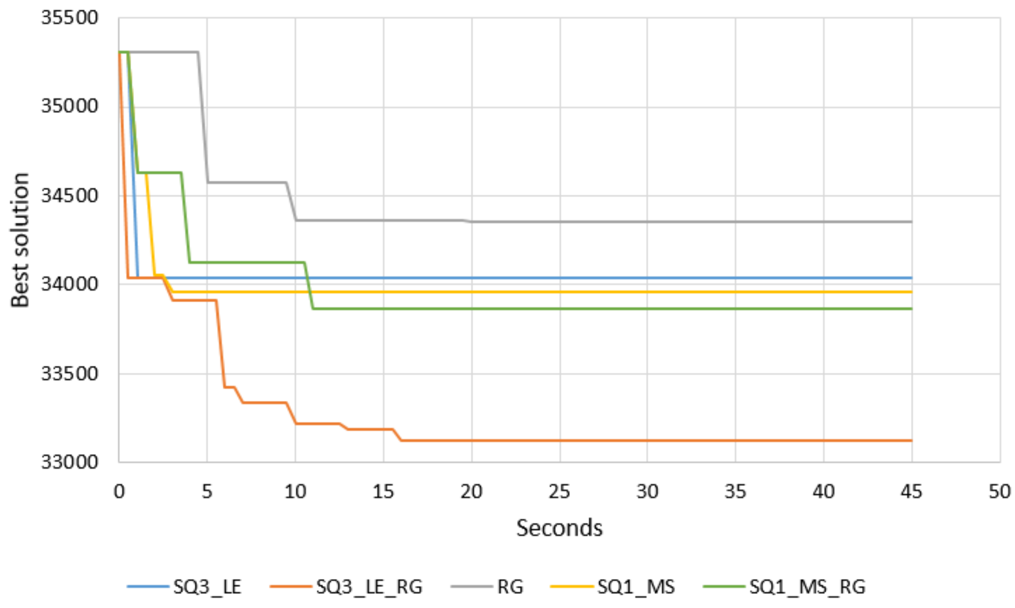


Figure 8.10: Best found IP objective related to running time and operator used

As illustrated in Figure 8.10, running the Shift-Quantity operators over a number of iterations without combining the Route-Generation operator, *RG*, that require changes, is a waste of the time budget of the heuristic. The improvement stops after a few iterations, and for the specific instance, no improvement occurs after about three seconds. The vehicle load on the fixed routes becomes closer to the vehicle capacity in each iteration, and therefore the quantity shifting will not reduce the transportation cost. No customer visits can be removed due to the limited shifting, thus the number of routes in the given time period cannot be reduced.

When the instance size increases we observe a much longer running time before improvement occurs, which indicates that the time budget in the intensification search should increase with instance size in order to improve the IP objective. Figure 8.11 illustrates how combinations of the Route-Generation operator and the different Shift-Quantity operators improve the solution over a running time of 75 seconds for instance $C_{18}V_3N_2$.

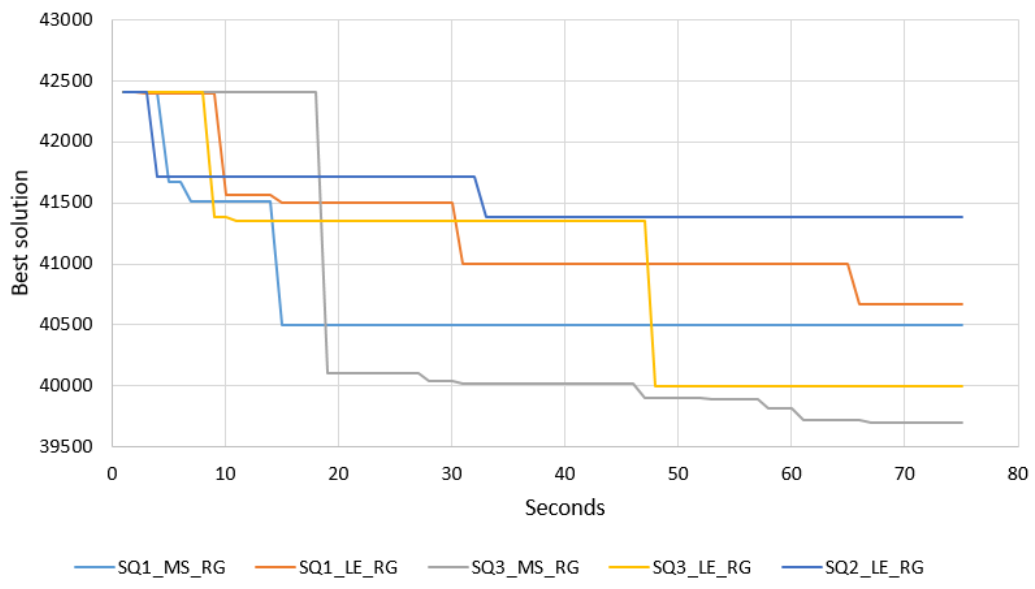


Figure 8.11: Best found IP objective for different operators for $C_{18}V_3N_1$

The average improvement of the IP solution for the five variants of instance $C_{12}V_3$ and $C_{18}V_3$ is presented in the in Table 8.12. We have used a stop criterion of 7 iterations for the results presented in the table. In order to evaluate the time consumed for a number of iterations, we also present the running time of the search for the given stop criterion. If an earlier solution is encountered in the iterations, the search stops in order to avoid looping between solutions.

Due to the obvious higher improvement by combining the Shift-Quantity operators with the Route-Generation operator, we only derived results for the specified combination in Table 8.12.

Table 8.12: Improvement of the IP solution for different operators in the intensification search

	$C_{12}V_3A$		$C_{18}V_3A$	
Operator	IP improvement	Sol.time	IP improvement	Sol.time
$SQ1\_LE\_RG$	3.0 %	24.6 s	3.2 %	88.6 s
$SQ1\_MS\_RG$	2.3%	40.0 s	2.4 %	115.0 s
$SQ2\_LE\_RG$	3.0 %	25.0 s	2.5 %	64.6s
$SQ3\_LE\_RG$	**3.9 %**	21.8 s	**3.4 %**	70.8 s
$SQ3\_MS\_RG$	2.9 %	31.0 s	2.7 %	187.4 s

The best improvement for smaller instances is produced by operator $SQ3\_LE\_RG$.The results in table 8.12 also indicate that the search strategy for selection of time period LE improves the IP more than the search strategy MS. LE also tends to be less time-consuming in each iteration. This can be a result of higher required changes when this search strategy is used. Therefore we have excluded MS in the test results from the larger instances which are described in Table 8.13. The stop criterion used was 5 iterations for $C_{24}V_5A$ and 3 iterations for $C_{30}V_5A$. The improvements are on average smaller than the one presented in 8.12. We therefore propose to increase the running time in the intensification search for the largest test instances in order to obtain a greater number of iterations, on which the improvement tends to be dependent.

Table 8.13: Improvement of the IP solution for different operators in the intensification search

	$C_{24}V_5A$		$C_{30}V_5A$	
Operator	IP improvement	Sol.time	IP improvement	Sol.time
$SQ1\_LE\_RG$	0.9%	417.0 s	0.8 %	245.2 s
$SQ2\_LE\_RG$	1.1%	281.8 s	0.8 %	245.20 s
$SQ3\_LE\_RG$	**2.7%**	356.6 s	**1.9%**	398.80 s

We conclude that the best combination of operators in the intensification search is the integrated routing and holding costs operator, choosing the time period with the lowest excess, and combine the operator with the Route-Generation operator.

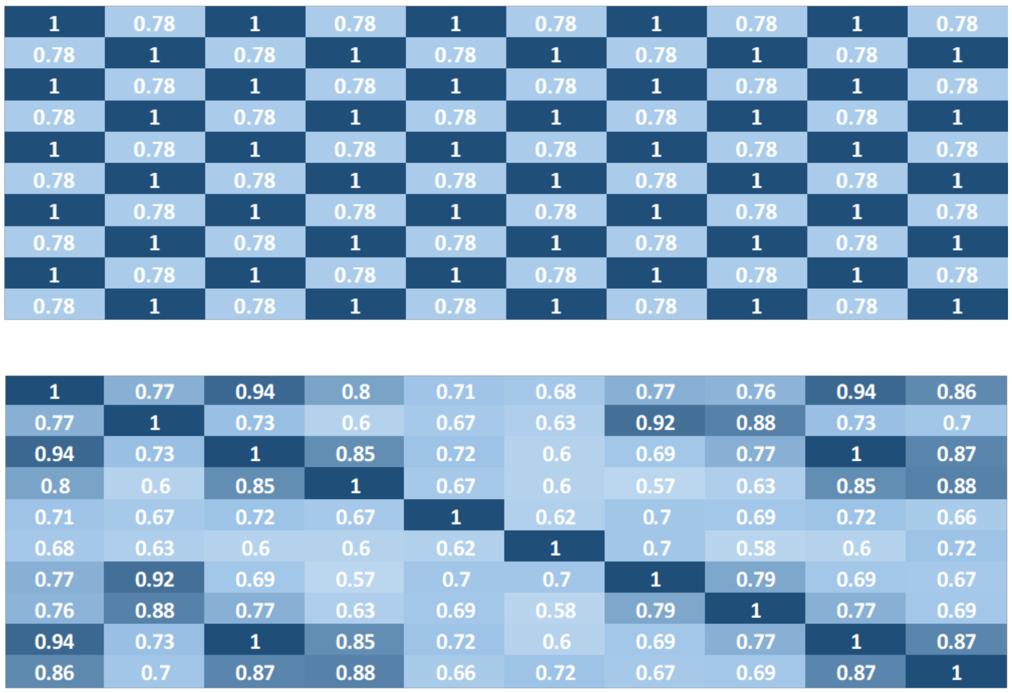
The diversification search

In order to see which of the proposed diversification techniques that is most effective a series of tests have been conducted. First, we tested settings for the tabu list of the search. Next, we tested different diversification constraints. Lastly, we looked at how changing the objective function of the relaxed IRP-DPD to only account for transportation costs, under a restriction for how much the holding costs are allowed to increase, affected the constructed solution.

Tabu list

The goal of the diversification is, as mentioned previously, to explore different parts of the solution space. In order to avoid visiting previous solutions, a tabu list was proposed in Section 6.4. The necessity of this tabu list in the diversification iteration is clearly illustrated in Figure 8.12. The figure shows two matrices holding comparison data between constructed solutions in ten diversification iterations. Row i, column j represent the constructed solutions from iteration i and j respectively. The number in position i, j is the proportion of equivalent arcs in solution i and j. A number of 1 means that all arcs in the two solutions are the same. A number of 0 means that all arcs are unique between the two solutions.

The results are derived from a C_6V_3 instance with no tabu list for the upper matrix and a tabu list applied for the lower matrix. The minimum change constraint 6.52 is applied in every diversification iteration. Evidently, without a tabu list, the constructed solutions loop between two solutions because the minimum change constraint does not prevent earlier solutions. In the second matrix, a tabu list with random locking of variables is introduced to avoid the looping behaviour. Any changed variable is subject to fixation for two periods only. Although solution 3 and 9 are equivalent, the difference in the tabu list prevents that solution 4 and 10 are the same as well.



1	0.78	1	0.78	1	0.78	1	0.78	1	0.78
0.78	1	0.78	1	0.78	1	0.78	1	0.78	1
1	0.78	1	0.78	1	0.78	1	0.78	1	0.78
0.78	1	0.78	1	0.78	1	0.78	1	0.78	1
1	0.78	1	0.78	1	0.78	1	0.78	1	0.78
0.78	1	0.78	1	0.78	1	0.78	1	0.78	1
1	0.78	1	0.78	1	0.78	1	0.78	1	0.78
0.78	1	0.78	1	0.78	1	0.78	1	0.78	1
1	0.78	1	0.78	1	0.78	1	0.78	1	0.78
0.78	1	0.78	1	0.78	1	0.78	1	0.78	1

1	0.77	0.94	0.8	0.71	0.68	0.77	0.76	0.94	0.86
0.77	1	0.73	0.6	0.67	0.63	0.92	0.88	0.73	0.7
0.94	0.73	1	0.85	0.72	0.6	0.69	0.77	1	0.87
0.8	0.6	0.85	1	0.67	0.6	0.57	0.63	0.85	0.88
0.71	0.67	0.72	0.67	1	0.62	0.7	0.69	0.72	0.66
0.68	0.63	0.6	0.6	0.62	1	0.7	0.58	0.6	0.72
0.77	0.92	0.69	0.57	0.7	0.7	1	0.79	0.69	0.67
0.76	0.88	0.77	0.63	0.69	0.58	0.79	1	0.77	0.69
0.94	0.73	1	0.85	0.72	0.6	0.69	0.77	1	0.87
0.86	0.7	0.87	0.88	0.66	0.72	0.67	0.69	0.87	1

Figure 8.12: Diversification without a tabu list(upper) and with a tabu list(lower)

Diversification constraints

In Section 6.4 we proposed different constraints for variable fixation in order to change the relaxed IRP-DPD. These constraints are tested for the 5 variants of instance $C_{12}V_3$ and $C_{18}V_3$ and the result is presented in Table 8.14. The table presents the dual gap when the heuristic was run for 600 seconds for $C_{12}V_3$ and 1200 seconds for $C_{18}V_3$. All constraints are tested with what is found to be the best required number of changes. The dual gap in the table is calculated from the solution found in the heuristic compared to the lower bound from the exact solution method. We conclude that the best performance is given by the diversification constraint that requires a minimum number of changes. All results presented in the forthcoming section use this constraint to diversify.

Table 8.14: Results for variable fixating

Instance		Force visits	Minimum changes	Route changes
$C_{12}V_3A$	Dual gap	3.44 %	**2.76 %**	3.02 %
$C_{18}V_3A$	Dual gap	4.49 %	**4.20 %**	5.56 %

Reduction of transportation costs

The relaxed IRP-DPD tends to underestimate transportation costs, leading to constructed solutions with higher transportation costs and lower holding costs compared to the exact solution. In section 6.4 we proposed to minimize over transportation costs in phase 1 under the restriction of a maximum increment in holding costs given by Equation 6.57 to account for this issue. Table 8.15 summarizes the results for different values for the parameter γ in the equation. To obtain the results we used the same stop criteria as those described for the diversification constraints. The diversification constraint related to a minimum number of changes is used in the tests. The table shows that this diversification approach is not effective. Therefore, we exclude this approach from the tests presented in the forthcoming.

Table 8.15: Results for different parameter settings for reduction of transportation costs

Instance	Holding cost	2.5 %	5.0 %	7.5%	Without
$C_{18}V_3A$	Dual gap	5.85 %	5.90 %	6.13 %	**4.20 %**
$C_{24}V_5A$	Dual gap	6.75 %	6.83	6.23 %	**5.76 %**

8.3 The results of the matheuristic

The parameter settings, valid inequalities for the exact method, and operators and constraints applied to the matheuristic are summarized in Appendix C. These settings are used for the results derived in the forthcoming section, in order to test the complete heuristic and compare it to the exact solution method. We have chosen to increase both heuristic and exact solution time to two hours for the instance sizes greater than and equal to 30 customers.

Note that the maximum running time in the intensification search is checked whenever an iteration of the search is finished. A new iteration is not applied if the maximum time is exceeded. The search will not be stopped in the middle of an iteration, the running time in the intensification search can, therefore, be longer than the maximum time. For the diversification search, a new constraint is not added if the maximum time is exceeded. The search will not be stopped in the middle of a two-phase iteration, the running time of the heuristic can, therefore, be longer than the maximum time.

The results of the matheuristic for the IRP-DPD

Table 8.16 summarizes the dual gaps of the heuristic and exact solution methods. The lower bound produced by the exact method is used to calculate the dual gap. Also, the table shows the number of instances where the heuristic approach produces a better solution than the exact method and the time spent to find the best solution. Additional information regarding the solution process for the two approaches is provided in Table 8.17. The table shows the time each solution method spent to find the best overall solution. Also, the time spent to find solutions n percentage points away from this solution are presented. In these tests, the final route selection of the heuristic is run after each diversification iteration. This is to produce the solution that would emerge if the heuristic exited the loop after that particular iteration. In these tests, the route selection is fast and does not significantly influence the running time of the heuristic. The number of diversification iterations, intensification iterations, size of the final route pool for each instance type and arc-flow solutions for the larger instances are presented in Appendix C.

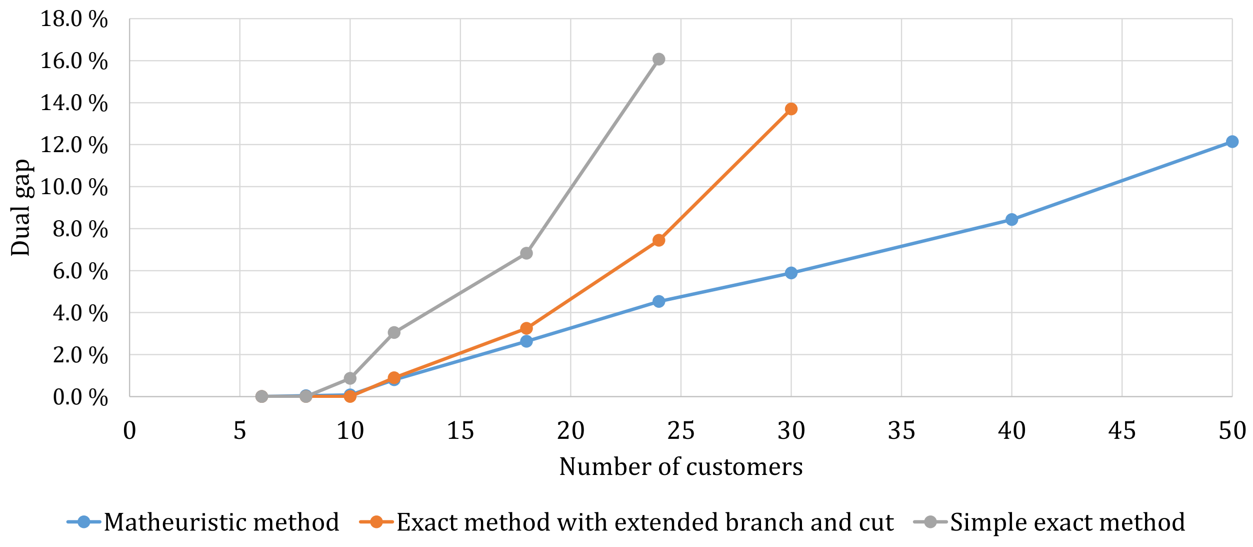
The exact solution method finds the optimal solution for all instances with 10 customers or less. For all larger instances, the exact solution method yields a dual gap. For instances with 6 customers the heuristic solution method finds the optimal solution for all instances. For instances with 8 and 10 customers the heuristic approach finds the optimal solution in 4 out of 5 instances and 3 out 5 of instances respectively. For 12 customers the heuristic and exact method on average produce approximately the same results. For all larger instances, the heuristic solution method gives significantly better solutions than the exact method. The exact model finds no solutions for 40 or 50 customers. However, the model gives a lower bound on the problem used to calculate the dual gap for the heuristic solution.

Table 8.16: Results of the matheuristic and the exact method summarized

Instance	Dual gap$_{Heur}$	Dual gap$_{Exact}$	$Z_{Heur} \leq Z_{exact}$	Best sol$_{Heur}$	Best sol$_{Exact}$
C_6V_3	-	-	5/5	132.4 s	24.6 s
C_8V_3	0.04%	-	4/5	57.8 s	85.6 s
$C_{10}V_3$	0.23%	-	3/5	457.8 s	1625.8 s
$C_{12}V_3$	0.85%	0.89%	3/5	896.6 s	1169.5 s
$C_{18}V_3$	2.63%	3.25%	5/5	1997.6 s	1929.2 s
$C_{24}V_5$	4.53%	7.44%	5/5	2466.6 s	2878.0 s
$C_{30}V_5$	5.89%	13.70%	5/5	4977.6 s	3329.8 s
$C_{40}V_7$	8.43%	No sol	5/5	6507.6 s	No sol
$C_{50}V_8$	12.14%	No sol	5/5	6900.2 s	No sol

Figure 8.13 summarizes the performance of the heuristic and exact solution methods in terms of dual gaps. The dual gap increases exponentially for the exact solution methods. However, the exact method with an extended branch and cut algorithm yields lower dual gaps, solve larger instances to optimality and find feasible solutions to larger instances. The heuristic approach gives a lower dual gap for all instances equal to or larger than 18 customers. For 40 and 50 customers only the heuristic solution method finds a feasible solution in two hours.

Figure 8.13: Dual gap for exact and heuristic solution methods



Another measure of performance is how much time must elapse before finding the best solution or solutions with objective value in proximity to that of the best solution. None of the two solution methods clearly distinguish themselves in the time spent to find the best solution. However, in general, the heuristic method is much faster at finding good solutions. For instance, for 24 customers the heuristic method finds the best solution at an average time of 2466.6 seconds. However a solution with 1% worse objective value is found after a mere 784.4 seconds on average, about 30% of the time spent to find the best solution. On the contrary, the exact method on average spends more than 90% of the time to find solutions with 1% worse objective value than that of the best solution found for the same instances.

Table 8.17: Closing the dual gap for the matheuristic compared to the exact approach

Instance	Sol. method	Best sol	0.5%	1.0%	2.0%	5.0%	10.0%
$C_{12}V_3A$	Heuristic	896.6 s	254.6 s	63.0 s	40.6 s	39.2 s	39.2 s
	Exact	1169.5 s	483.4 s	242.6 s	138.4 s	74.4 s	52.8 s
$C_{18}V_3A$	Heuristic	1997.6 s	917.4 s	300.4 s	241.6 s	240.4 s	240.4 s
	Exact	1929.2 s	1232 s	1224.4	1073.4 s	570.6 s	212.4 s
$C_{24}V_3A$	Heuristic	2466.6 s	1636 s	784.4 s	756.8 s	435 s	433.2 s
	Exact	2878.0 s	2805.6 s	2615.8 s	2075.6 s	1078.4 s	717.2 s
$C_{30}V_5A$	Heuristic	4977.6 s	3529.8 s	1485.6 s	957.4 s	764.8 s	666.8 s
	Exact	3329.8 s	3061.2 s	3056.2 s	3056 s	3012.6 s	3012.6
$C_{40}V_7A$	Heuristic	6507.6 s	5512.2 s	2403.2 s	1966.2 s	1043.2 s	901.0 s
	Exact	No sol	No sol	No sol	No sol	No sol	No sol
$C_{50}V_8A$	Heuristic	6900.2 s	4248.8 s	3211.0 s	2982.6 s	1256.0 s	1256.0 s
	Exact	No sol	No sol	No sol	No sol	No sol	No sol

Figure 8.14 shows the solution process for the exact and heuristic solution method for instances of 24 and 30 customers. A line segment plots the dual gap for an instance type at each second from 0 to 3600 seconds. The results are an average of the five instances of each type and each line segment starts at the solution time where the respective solution method has found a feasible solution for all instances. Evidently, the matheuristic is faster at finding a feasible solution for both instance types. As discussed in the previous paragraph, the figure also illustrates how the heuristic method quickly finds solutions close to the best solution found during the entire search time.

Figure 8.14: Solution process for the exact and heuristic methods

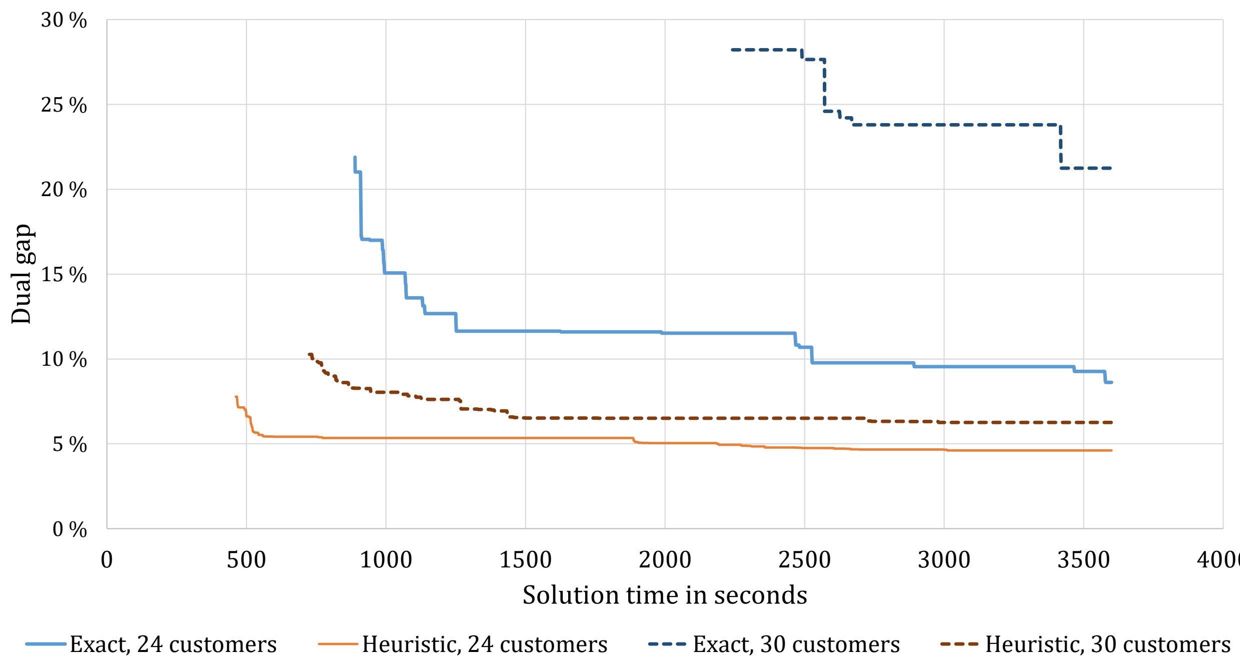
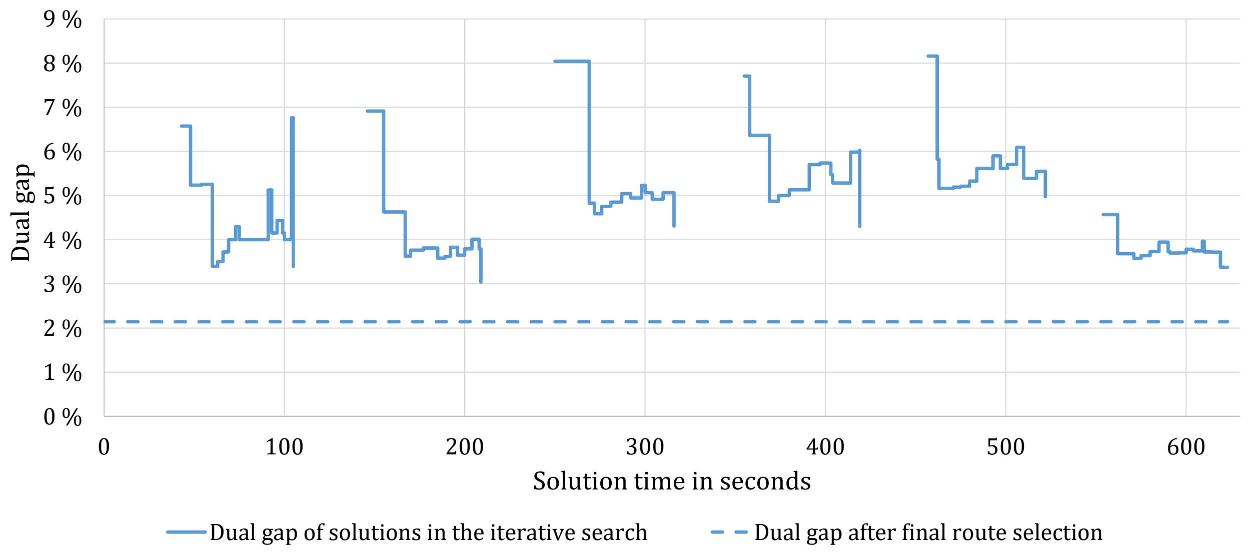


Figure 8.15 illustrates the matheuristic's iterative processes and the final route selection problem which selects between routes from all iterations. The data is from a $C_{12}V_3$ instance run for 600 seconds. The solution time somewhat exceeds 600 seconds because an initiated iteration is not interrupted. Each continuous part of the solid line represents a diversification iteration and the associated intensification search. Thus, in the figure, there are six diversification iterations. Each endpoint or vertical change of a line segment represents a new solution and its corresponding dual gap. Each line segment starts with the construction solution followed by the solutions from the intensification search. Evidently, the route generation operator occasionally leads to worse solutions. However, the final solution in each iteration, i.e., the last point of each line segment, often has the iteration's strictly lowest dual gap. That means combinations of routes in the intensification search produces a better solution than the routes that appeared in a particular solution during the search. The same reasoning holds for the solution from the final route selection problem, which dual gap is lower than that of any iteration.

Figure 8.15: The heuristic's iterative process and final route selection



Structure of the solution

Table 8.18 presents the structure of the heuristic solutions. A divisible visit occurs when two co-located nodes are served in the same period, but the arc between them is not traversed. Likewise, a simultaneous visit occurs when two co-located nodes are served, and the arc between them are traversed. A single visit is when only one of two co-located nodes is served in a period. The node serving ratio is the average proportion of nodes that are served in a period.

For all instances, the majority of customers visited are served simultaneously. However, the proportion tends to decrease for larger instances as the number of single visits increases. For larger instances, a single vehicle must cover more customers which may explain why fewer customers can have both pickup and delivery in the same period. For 12 and 18 customers, there are no divisible visits. For larger instances, the number of divisible visits is somewhat increasing. The lower bound for the simultaneous model is below the solution found in the heuristic; thus no absolute conclusion regarding economic gain for the divisible approach can be made. However, given that the problem is feasible when adding the simultaneous constraints, any substantial economic gain from divisible servings is questionable due to the low proportion of divisible visits. As opposed to vehicle routing models, the more flexible inventory routing approach makes it easier to avoid the additional transportation costs incurred by visiting a customer twice in a period. This may explain why the occurrence of divisible visits is less for inventory routing models. However, as discussed in Section 1, the divisible model in this thesis is a very general version of the divisible model. In real life applications of the model, where constraints such as time windows or order of loading often are present, the divisible option may incur significant cost savings and even be the only feasible approach.

Table 8.18: Structure of the solution

Instance	$C_{12}V_3$	$C_{18}V_3$	$C_{24}V_5$	$C_{30}V_5$	$C_{40}V_7$	$C_{50}V_8$
Divisible visits	-	-	0.9%	1.0%	1.7%	3.2%
Simultaneous visits	71.9%	61.0%	63.7%	57.5%	54.9%	44.2%
Single visits	27.9%	39.0%	35.4%	41.5%	43.4%	52.6%
Routes/period	1.5	1.7	2.4	2.5	3.6	4.3
Route length	7.6	9.4	9.0	10.7	10.0	10.0
Node serving ratio	0.47	0.45	0.46	0.45	0.45	0.43

Various geographies

Geographies with certain characteristics may facilitate more divisible visits. In accordance with the results from Nagy et al. (2015) presented in Section 2.2 two such geographies are customer clusters and customers located in proximity to the depot. Table 8.19 summarizes the results from tests for instances of these types. For each type the result is an average of five instances. Both geographies are tested for 24 and 30 customers. Evidently, the proportion of divisible visits is still very small.

Table 8.19: Divisible visits for various geographies

	Clusters		**In proximity to depot**	
Solution data	$C_{24}V_5$	$C_{30}V_5$	$C_{24}V_5$	$C_{30}V_5$
Divisible visits	0.6%	0.9%	0.5%	1.4%
Simultaneous visits	57.5%	54.7%	57.2%	57.6%
Single visits	41.9%	44.4%	42.3%	41.0%
Routes/period	2.6	2.8	2.6	2.5
Route length	8.6	9.6	9.0	11.3
Node serving ratio	0.47	0.45	0.49	0.47

Figure 8.16 shows the routes of a period solution to a cluster (left) and a proximity to depot (right) instance with 30 customers. In this example, only the cluster instance produces a divisible visit, marked by a blue dashed circle. It occurs when two vehicles visit the same cluster. In general for these cluster instances, when only one vehicle serves a cluster, divisible visits tend not to happen despite the short mutual distance between customers in the same cluster. The chances of divisible visits are somewhat higher if two vehicles visit the same cluster. This coincides with the findings in Nagy et al. (2015) which conclude that a divisible visit is likely to be split across routes. Although the instances with customers in proximity to the depot is well suited for such splitting, only about 1% of the visits happen to be divisible. We conclude that, even with suitable geographies, the flexibility inherent in inventory routing models tend to avoid divisible visits.

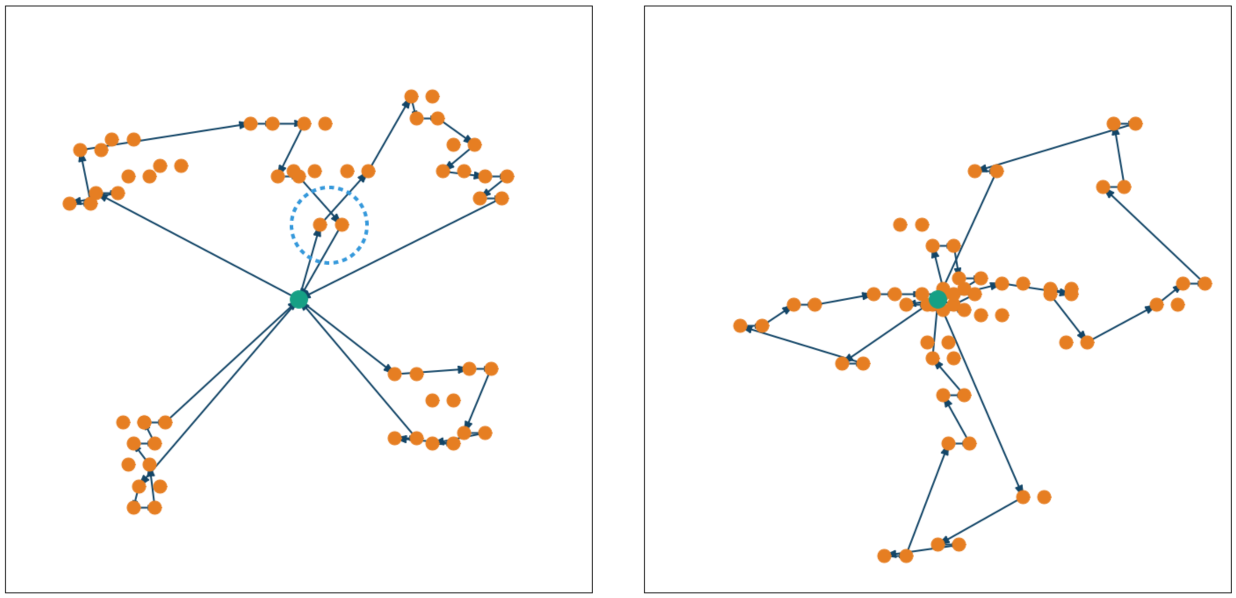


Figure 8.16: Routes in a period for a cluster and a proximity to depot instance.

Results of the matheuristic for IRP variants

The general structure of the IRP-DPD can easily be transformed to other variants of the IRP problem. There is nothing about the matheuristic described in this thesis that suggests it is not going to perform equally well for the different variants of the IRP described in Section 6.6. Table 8.20 presents the exact and heuristic results for instances of the regular IRP and IRP with mixed line hauls and backhauls. Also, the results for the IRP-DPD for 30 and 40 customers are included in the table. Both the regular IRP and the IRP-MB are solved for more customers than the IRP-DPD because each customer is represented by one node only. The regular IRP is the easier of the two because it does not integrate the pickup and delivery structure in the problem. Results for a particular instance type is an average over five instances. The exact method uses the branch and cut algorithm proposed for the IRP-DPD.

Table 8.20: Results for IRP variations

	Regular IRP		**IRP-MB**		**IRP-DPD**	
Instance	$C_{50}V_8$	$C_{70}V_{10}$	$C_{40}V_7$	$C_{60}V_9$	$C_{30}V_5$	$C_{40}V_7$
Dual gap$_{Heur}$	6.66%	9.65%	4.82%	6.80%	5.89%	8.43%
Dual gap$_{Exact}$	15.25%	No sol	8.30%	No sol	13.70%	No sol
$Z_{heur} \leq Z_{exact}$	5/5	5/5	5/5	5/5	5/5	5/5
Best sol$_{Heur}$	2862.2 s	5991.8 s	3600.0 s	3417.2 s	4977.6 s	6507.6
Best sol$_{Exact}$	2470.4 s	No sol	1458.2 s	No sol	3329.8 s	No sol

The matheuristic outperforms the exact method for all instances. For the regular IRP the matheuristic gives a dual gap of less than 10% for instances up to 70 customers and 10 vehicles. The matheuristic also produces low dual gaps for IRP-MB. For the largest instances, the exact method does not find a feasible solution to all instances for either of the problem variants but provides a lower bound on the problem instance.

Chapter 9

Concluding remarks and further research

The work done in this thesis is motivated by the conviction that there is a large potential for operational research in the field of reverse logistics. The problem in this thesis considers an inventory routing problem with divisible pickup and delivery (IRP-DPD). All customers may be visited twice in a time period; once for pickup and once for delivery. An arc-flow formulation of the problem, formulated as a MILP, is proposed as an exact solution approach. The formulation is strengthened with valid inequalities and a branch and cut algorithm. For larger instances, we propose a matheuristic that constructs a solution by decomposing the problem into an inventory problem and a routing problem which are solved separately. The solution is then improved by a set of operators. The matheuristic iterates between construction and improving solutions prior to a final improvement heuristic.

9.1 Conclusion

The branch and cut algorithm drastically improves the exact arc-flow formulation. The enhanced method solves a greater number of instances to optimality, gives lower dual gaps and finds feasible solutions for larger instances. The main contribution to the improvement comes from the dynamically added subtour cuts. Instances up to 10 customers are solved to optimality within an hour. Feasible solutions are found for instances up to 30 customers.

The matheuristic is tested for the same instances. It finds optimal solutions for all instances of 6 customers and yields dual gaps of less than 1% for all instances up to 12 customers within an hour. For all larger instances, the matheuristic gives significantly better solutions than the exact method. The dual gaps of the solutions are less than 10% for instances up to 40 customers and 7 vehicles.

The economic gains in vehicle routing problems have been substantial with the divisible option available. However, in the solution of our instances of the IRP-DPD, the proportion of divisible visits is small. The flexibility inherent in inventory routing models seems to limit the economic gain from the divisible option. Visiting a customer twice in the same period incurs additional transportation costs which can be avoided when the planning horizon consists of several periods. However, the divisible option may be appropriate for many real applications of reverse logistics. With constraints such as time windows or order of vehicle loading, the divisible option may be required to produce feasible solutions.

The IRP-DPD easily transforms to other variations of the IRP. The exact method and matheuristic are thus compared for instances of different IRP problems. As for the IRP-DPD, the matheuristic outperforms the exact method for larger instances also for these problem types.

9.2 Further research

During this master thesis, we have come across various opportunities for further research to our IRP-DPD.

Rolling horizon

Steady businesses are ongoing operations with no end of the horizon. In an optimization model, however, we have to consider decisions over a definite period of time due to the increased complexity of adding more periods. Repeatedly solving a model in a rolling horizon scheme is a way to overcome these limitations. For instance, the model can be rerun every week with a five-week planning horizon. An interesting direction for future work is, therefore, to incorporate the proposed matheuristic into a rolling horizon scheme for handling the ongoing planning problems. However, the uncertainty of the information used as input in the model will increase for future time periods. A stochastic model is likely to be too complex to solve, but different deterministic models which hedge against uncertainty can be evaluated.

Utilizing the divisible option from a practical point of view

From our computational results, we saw that the economic gains from allowing divisible visits were limited. Though, the divisible option can be valuable in several real-life examples, which are not studied in this thesis. For instance, customers may prefer delivery to happen before pickup each day, or the customer may have different time windows for pickup and delivery. It would, therefore, be interesting to further develop the heuristic by constraining the routes or including time windows. Also, the divisible option can be preferable for the supplier in order to ease the merchandise handling within a vehicle. If a vehicle is almost full and delivery and pickup are performed simultaneously, quite a lot of rearranging within the vehicle could be required. Introducing a penalty cost for simultaneous solutions related to extra service time for the supplier makes an interesting problem extension for further research. Setting this cost higher for the customers visited at the beginning of a route may induce lasso solutions. We believe that the strong solution methods we have developed in this thesis will be a good and useful framework for solving problems where these real-life aspects are introduced.

Bibliography

Ai, T. J. and Kachitvichyanukul, V. (2009). A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. *Computers & Operations Research*, 36(5):1693–1702.

Archetti, C., Bertazzi, L., Hertz, A., and Speranza, M. G. (2012). A hybrid heuristic for an inventory routing problem. *INFORMS Journal on Computing*, 24(1):101–116.

Archetti, C., Bertazzi, L., Laporte, G., and Grazia Speranca, M. (2007). A branch-and-cut algorithm for a vendor- managed inventory-routing problem. *Transportation Science*, 41(3):382–391.

Archetti, C., Boland, N., and Grazia Speranza, M. (2017a). A matheuristic for the multivehicle inventory routing problem. *INFORMS Journal on Computing*, 29(3):377–387.

Archetti, C., Christiansen, M., and Grazia Speranza, M. (2017b). Inventory routing with pickup and deliveries.

Archetti, C. and Speranza, M. G. (2014). A survey on matheuristics for routing problems. *EURO Journal on Computational Optimization*, 245(2):223–246.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., and Laporte, G. (2007). Static pickup and delivery problems: a classification scheme and survey. *Top*, 15(1):1–31.

Coelho, L. C., Cordeau, J.-F., and Laporte, G. (2013). Thirty years of inventory routing. *Transportation Science*, 48(1):1–19.

Coelho, L. C. and Laporte, G. (2013). A branch-and-cut algorithm for the multi-product multi-vehicle inventory-routing problem. *International Journal of Production Research*, 51(23-24):7156–7169.

Coelho, L. C. and Laporte, G. (2014). Improved solutions for inventory-routing problems through valid inequalities and input ordering. *International Journal of Production Economics*, 155:391–397.

Danna, E. and Pape, C. (2005). Branch-and-price heuristics: a case study on the vehicle routing problem with time windows. In *Column generation*, pages 99–129. Springer.

Dethloff, J. (2001). Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. *OR Spectrum*, 23(1):79–96.

Dobbs, R., Remes, J., Manyika, J., Roxburgh, C., Smith, S., and Schaer, F. (2012). Urban world: Cities and the rise of the consuming class.

Drexl, M. (2013). A note on the separation of subtour elimination constraints in elementary shortest path problems. *European Journal of Operational Research*, 229(3):595–598.

Hemmati, A., Hvattum, L. M., Christiansen, M., and Laporte, G. (2016). An iterative two-phase hybrid matheuristic for a multi-product short sea inventory-routing problem. *European Journal of Operational Research*, 252(3):775–788.

Hoff, A., Gribkovskaia, I., Laporte, G., and Løkketangen, A. (2009). Lasso solution strategies for the vehicle routing problem with pickups and deliveries. *European Journal of Operational Research*, 192(3):755–766.

Hoornweg, D., Bhada-Tata, P., and Kennedy, C. (2013). Environment: Waste production must peak this century. *Nature*, 502:615–617.

Iassinovskaia, G., Limbourg, S., and Riane, F. (2017). The inventory-routing problem of returnable transport items with time windows and simultaneous pickup and delivery in closed-loop supply chains. *International Journal of Production Economics*, 183:570–582.

Nagy, G., Wassan, N. A., Speranza, M. G., and Archetti, C. (2015). The vehicle routing problem with divisible deliveries and pickups. *Transportation Science*, 49(2):271–294.

Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM review*, 33(1):60–100.

Parragh, S. N., Doerner, K. F., and Hartl, R. F. (2008). A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, 58(1):21–51.

Privé, J., Renaud, J., Boctor, F., and Laporte, G. (2006). Solving a vehicle-routing problem arising in soft-drink distribution. *Journal of the Operational Research Society*, 57(9):1045–1052.

Raa, B. and Aghezzaf, E. H. (2009). A practical solution approach for the cyclic inventory routing problem. *European Journal of Operational Research*, 192(2):429–441.

Savelsbergh, M. and Song, J.-H. (2008). An optimization algorithm for the inventory routing problem with continuous moves. *Computers & operations research*, 35(7):2266–2282.

Stålhane, M., Rakke, J. G., Moe, C. R., Andersson, H., Christiansen, M., and Fagerholt, K. (2012). A construction and improvement heuristic for a liquefied natural gas inventory routing problem. *Computers & Industrial Engineering*, 62(1):245–255.

Tarjan, R. (1972). Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160.

Van Anholt, R. G., Coelho, L. C., Laporte, G., and Vis, I. F. (2016). An inventory-routing problem with pickups and deliveries arising in the replenishment of automated teller machines. *Transportation Science*, 50(3):1077–1091.

Yu, Y., Chen, H., , and Chu, F. (2008). A new model and hybrid approach for large scale inventory routing problems. *European Journal of Operational Research*, 189(3):1022–1040.

Appendix A

Complete model formulations

Basic model formulation

$$\min \sum_{(i,j)\in\mathscr{A}} \sum_{v\in\mathscr{V}} \sum_{t\in\mathscr{T}} C^T_{ij} x_{ijvt} + \sum_{i\in\mathscr{N}} \sum_{t\in\mathscr{T}} C^H_i i_{it} \tag{A.1}$$

$$\text{s.t.} \sum_{j\in\mathscr{N}'} x_{ijvt} - \sum_{j\in\mathscr{N}'} x_{jivt} = 0 \qquad i\in\mathscr{N}', v\in\mathscr{V}, t\in\mathscr{T} \tag{A.2}$$

$$\sum_{j\in\mathscr{N}'} \sum_{v\in\mathscr{V}} x_{ijvt} - y_{it} = 0 \qquad i\in\mathscr{N}', t\in\mathscr{T} \tag{A.3}$$

$$y_{it} \le 1 \qquad i\in\mathscr{N}, t\in\mathscr{T} \tag{A.4}$$

$$y_{0t} \le V \qquad t\in\mathscr{T} \tag{A.5}$$

$$\sum_{(i,j)\in\mathscr{A}} T_{ij} x_{ijvt} \le K^T \qquad v\in\mathscr{V}, t\in\mathscr{T} \tag{A.6}$$

$$i_{it} - i_{it-1} - R_{it} + q_{it} = 0, \qquad i\in\mathscr{N}^P, t\in\mathscr{T} \tag{A.7}$$

$$i_{it} - i_{it-1} + R_{it} - q_{it} = 0, \qquad i\in\mathscr{N}^D, t\in\mathscr{T} \tag{A.8}$$

$$i_{i0} = I_{i0} \qquad i \in \mathcal{N} \tag{A.9}$$

$$i_{it-1} + q_{it} \leq U_i \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{A.10}$$

$$i_{it-1} - q_{it} \geq L_i \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{A.11}$$

$$i_{it} \geq L_i \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{A.12}$$

$$i_{it} \leq U_i \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{A.13}$$

$$\sum_{j \in \mathcal{N}'} l^D_{jit} - q_{it} - \sum_{j \in \mathcal{N}'} l^D_{ijt} = 0 \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{A.14}$$

$$\sum_{j \in \mathcal{N}'} l^P_{jit} + q_{it} - \sum_{j \in \mathcal{N}'} l^P_{ijt} = 0 \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{A.15}$$

$$\sum_{j \in \mathcal{N}'} l^P_{jit} - \sum_{j \in \mathcal{N}'} l^P_{ijt} = 0 \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \tag{A.16}$$

$$\sum_{j \in \mathcal{N}'} l^D_{jit} - \sum_{j \in \mathcal{N}'} l^D_{ijt} = 0 \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{A.17}$$

$$l^P_{ijt} + l^D_{ijt} - K^Q \sum_{v \in \mathcal{V}} x_{ijvt} \leq 0 \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{A.18}$$

$$x_{ijvt} \in \{0,1\} \qquad (i,j) \in \mathcal{A}, v \in \mathcal{V}\, t \in \mathcal{T} \tag{A.19}$$

$$i_{it} \geq 0 \qquad i \in \mathcal{N}, t \in \mathcal{T} \cup \{0\} \tag{A.20}$$

$$q_{it} \geq 0 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{A.21}$$

$$l^P_{ijt}, l^D_{ijt} \geq 0 \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T} \tag{A.22}$$

$$y_{it} \geq 0 \qquad i \in \mathcal{N}', t \in \mathcal{T} \tag{A.23}$$

Extended model formulation

$\mathcal{T}'$	The augmented set of time periods $\{\mathcal{T} \cup 0 \cup \|\mathcal{T}\| + 1\}$
P_{it}	Amount left of initial inventory at node $i \in \mathcal{N}^D$ at the end of period t. $\{max(I_0 - \sum_{k=1}^{t} R_{ik}, 0)\}$
R'_{it}	Demand in excess of initial inventory left at node $i \in \mathcal{N}^D$ in period t. $\{max(R_{it} - P_{it-1}, 0)\}$
f^P_{its}	Amount of product produced at node i in time period t to be picked up in time period s
f^D_{its}	Amount of product delivered at node i in time period t to be used in time period s

$$\min \sum_{(i,j)\in\mathcal{A}} \sum_{t\in\mathcal{T}} C^T_{ij} x_{ijt} + \sum_{i\in\mathcal{N}} \sum_{t\in\mathcal{T}} C^H_i i_{it} \tag{A.24}$$

$$\sum_{j\in\mathcal{N}'} x_{ijt} - \sum_{j\in\mathcal{N}'} x_{jit} = 0 \qquad i \in \mathcal{N}', t \in \mathcal{T} \tag{A.25}$$

$$\sum_{j\in\mathcal{N}'} x_{ijt} - y_{it} = 0 \qquad i \in \mathcal{N}', t \in \mathcal{T} \tag{A.26}$$

$$y_{it} \leq 1 \qquad i \in \mathcal{N}, t \in \mathcal{T} \tag{A.27}$$

$$y_{0t} \leq V \qquad t \in \mathcal{T} \tag{A.28}$$

$$\sum_{t=1}^{s} f^D_{its} = R'_{is} \qquad i \in \mathcal{N}^D, s \in \mathcal{T} \tag{A.29}$$

$$\sum_{s=t+1}^{|\mathcal{T}'|} f^P_{its} = R_{it} \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \tag{A.30}$$

$$\sum_{s=1}^{|\mathcal{T}'|} f_{i0s} = I_{i0} \qquad i \in \mathcal{N}^P \qquad \text{(A.31)}$$

$$f^D_{its} \leq R'_{is} y_{it} \qquad i \in \mathcal{N}^D, s \in \mathcal{T}, t \in \mathcal{T}|_{t \leq s} \qquad \text{(A.32)}$$

$$f^P_{its} \leq R_{it} y_{is} \qquad i \in \mathcal{N}^P, s \in \mathcal{T}, t \in \mathcal{T}'|_{t < s} \qquad \text{(A.33)}$$

$$q^D_{it} = \sum_{s=t}^{|\mathcal{T}'|} f^D_{its} \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \qquad \text{(A.34)}$$

$$q^P_{is} = \sum_{t=0}^{s-1} f^P_{its} \qquad i \in \mathcal{N}^P, s \in \mathcal{T} \qquad \text{(A.35)}$$

$$i_{i\tau} = \sum_{t=1}^{\tau} \sum_{s=\tau+1}^{|\mathcal{T}'|} f^D_{its} + P_{i\tau} \qquad i \in \mathcal{N}^D, \tau \in \mathcal{T} \qquad \text{(A.36)}$$

$$i_{i0} = I_{i0} \qquad i \in \mathcal{N}^D \qquad \text{(A.37)}$$

$$i_{i\tau} = \sum_{t=0}^{\tau} \sum_{s=\tau+1}^{|\mathcal{T}'|} f^P_{its} \qquad i \in \mathcal{N}^P, \tau \in \mathcal{T}'|_{t<|T|'} \qquad \text{(A.38)}$$

$$i_{it-1} + q^D_{it} \leq U_i \qquad i \in \mathcal{N}^D, t \in \mathcal{T} \qquad \text{(A.39)}$$

$$i_{it-1} - q^P_{it} \geq L_i \qquad i \in \mathcal{N}^P, t \in \mathcal{T} \qquad \text{(A.40)}$$

$$i_{it} \geq L_i \qquad i \in \mathcal{N}, t \in \mathcal{T} \qquad \text{(A.41)}$$

$$i_{it} \leq U_i \qquad i \in \mathcal{N}, t \in \mathcal{T} \qquad \text{(A.42)}$$

$$t_{0t} = 0 \qquad t \in \mathcal{T} \qquad \text{(A.43)}$$

$$t_{it} - t_{jt} + T_{ij} \leq (K + T_{ij})(1 - x_{ijt}) \qquad (i,j) \in \mathcal{A}, t \in \mathcal{T} \qquad \text{(A.44)}$$

$$t_{it} + T_{ij} x_{ijt} \le K \qquad (i,j) \in \mathscr{A}, t \in \mathscr{T} \quad \text{(A.45)}$$

$$\sum_{j \in \mathscr{N}'} l^D_{jit} - q^D_{it} - \sum_{j \in \mathscr{N}'} l^D_{ijt} = 0 \qquad i \in \mathscr{N}^D, t \in \mathscr{T} \quad \text{(A.46)}$$

$$\sum_{j \in \mathscr{N}'} l^P_{jit} + q^P_{it} - \sum_{j \in \mathscr{N}'} l^P_{ijt} = 0 \qquad i \in \mathscr{N}^P, t \in \mathscr{T} \quad \text{(A.47)}$$

$$\sum_{j \in \mathscr{N}'} l^P_{jit} - \sum_{j \in \mathscr{N}'} l^P_{ijt} = 0 \qquad i \in \mathscr{N}^D, t \in \mathscr{T} \quad \text{(A.48)}$$

$$\sum_{j \in \mathscr{N}'} l^D_{jit} - \sum_{j \in \mathscr{N}'} l^D_{ijt} = 0 \qquad i \in \mathscr{N}^P, t \in \mathscr{T} \quad \text{(A.49)}$$

$$l^P_{ijt} + l^D_{ijt} \le Q x_{ijt} \qquad (i,j) \in \mathscr{A}, t \in \mathscr{T} \quad \text{(A.50)}$$

$$f^D_{its} \ge 0 \qquad i \in \mathscr{N}, s \in \mathscr{T}', t \in \mathscr{T}|_{t \le s} \quad \text{(A.51)}$$

$$f^P_{its} \ge 0 \qquad i \in \mathscr{N}, s \in \mathscr{T}'|_{s>0}, t \in \mathscr{T}'|_{t<s} \quad \text{(A.52)}$$

$$x_{ijt} \in \{0,1\} \qquad (i,j) \in \mathscr{A}. t \in \mathscr{T} \quad \text{(A.53)}$$

$$i_{it} \ge 0 \qquad i \in \mathscr{N}, t \in \mathscr{T}'|_{t<|\mathscr{T}'|} \quad \text{(A.54)}$$

$$q^P_{it}, q^D_{it} \ge 0 \qquad i \in \mathscr{N}, t \in \mathscr{T} \quad \text{(A.55)}$$

$$l^P_{ijt}, l^D_{ijt} \ge 0 \qquad (i,j) \in \mathscr{A}, t \in \mathscr{T} \quad \text{(A.56)}$$

$$y_{it} \ge 0 \qquad i \in \mathscr{N}', t \in \mathscr{T} \quad \text{(A.57)}$$

Appendix B

Solution methods

Algorithm *Strong-Components* is the pseudocode for Tarjan's algorithm (Tarjan (1972)). It is a depth-first graph traversal algorithm which finds the strong components of a simple directed graph. For each node not investigated the helper procedure *Strong-Connect* is called, which returns the strong component to which the current node belongs (line 5). In Strong-Connect the node is assigned an unique index which indicates when the node is investigated (line 11). The node is pushed on a stack S which tracks which nodes to include in the strong component (line 14). A depth-first search is then performed for all edges incident from the node and Strong-Connect is called recursively from the next node (line 19). If all edges lead to discovered nodes, i.e. the index is defined, the node attribute lowlink, which tracks the lowest index in the current strong component, is set to the minimum of the current node's lowlink and the stacked node's index (line 24). The same value of lowlink is assigned to nodes up the recursive chain (line 20).

When the depth first search from a node finishes, the node's lowlink is compared to its index. If lowlink is equal to the node's index the nodes on the stack form a strong component(line 30). The stack is then emptied and the strong component is returned. The algorithm runs in linear time O(|N|+|E|). It is implemented in C++. Line 22 and 26 are extensions of the original algorithm and remove edges between strong components. This is to ease the subsequent check for violated subtour elimination constraints. In line 22, if v is not on the stack it indicates that v belongs to another strong component and the edge to it should be removed. This is a forward edge in the depth first search. In line 26 cross edges to other strong components are removed. The operations are illustrated in figure B.1.

Algorithm Strong-Components

```
index = 0, $S = \emptyset$, Result = $\emptyset$

**for** each node $v \in G.V$
    **if** $v$.index is undefined
        strongComp = Strong-Connect($v$)
        add strongComp to Result
    **end if**
**return** Result

Strong-Connect(u)
u.index = index
u.lowlink = lowlink
index = index + 1
S.push(u)
$u$.onStack = true

**for** each edge $(u, v) \in G.E$
    **if** index is undefined
        Strong-Connect(v)
        u.lowlink = min(u.lowlink, v.lowlink)
        **if not** v.onStack
            remove $(u, v)$ from $G.E$
    **else if** $v$.onStack
        u.lowlink = min(u.lowlink, v.index)
    **else**
        remove $(u, v)$ from $G.E$
    **end if**
**end for**

**if** $v$.lowlink == $v$.index
        StrongComponent = $\emptyset$
    **do**
        $w = S$.top
        add $w$ to StrongComponent
        $w$.onStack = false
        $S$.pop
    **while** $w \neq v$
**end if**
**return** StrongComponent
```

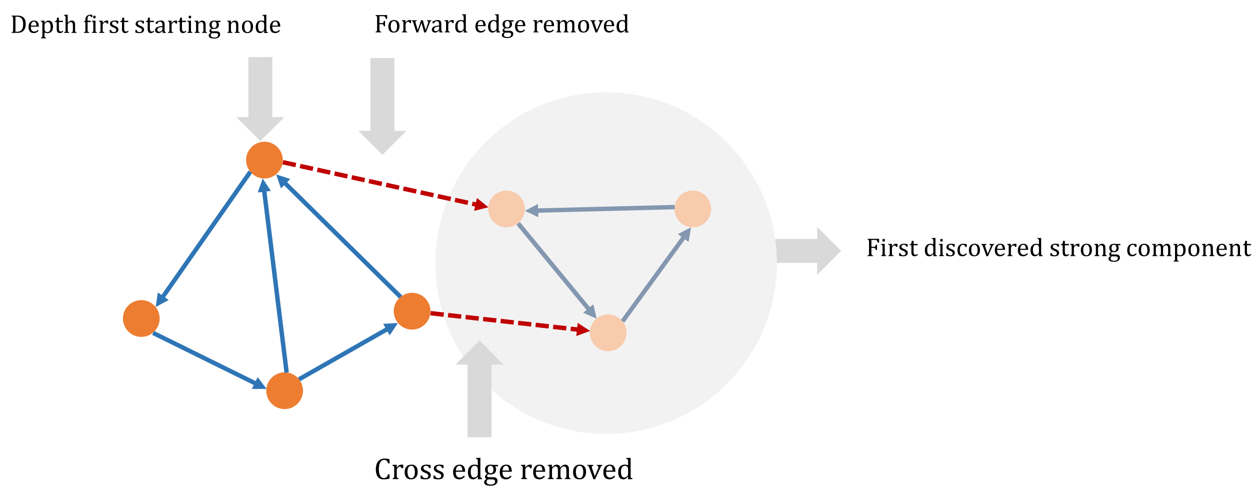


Figure B.1: Illutration of how the Strong-Component algorithm removes edges between strong components.

Appendix C

Computational study

Parameter settings

Table C.1: Parameter settings for different instances

	$C_{6-8}V_3$	$C_{10}V_3$	$C_{12}V_3$	$C_{18}V_3$	$C_{24}V_5$	$C_{30}V_5$	$C_{40}V_7$	$C_{50}V_8$
Maximum time								
Relaxed IRP-PD	240 s	240 s	240 s	240 s	240 s	360 s	480 s	480 s
VRP-DPD	60 s	60 s	60 s	60 s	60 s	90 s	120 s	180 s
MIP SQ_3	60 s	60 s	60 s	60 s	90 s	120 s	120 s	180 s
RG	60 s	60 s	60 s	60 s	60 s	90 s	120 s	120 s
RS	60 s	60 s	60 s	60 s	60 s	90 s	120 s	120 s
Intensification	30 s	45 s	60 s	120 s	300 s	600 s	900 s	900 s
Heuristic	300 s	1200 s	1800 s	3600 s	3600 s	7200 s	7200 s	7200 s
Heuristic Info								
Minimum change	true	true	true	true	true	true	true	true
$SEC + Div$	true	true	true	false	false	false	false	false
$SEC + Sim$	false	false	false	true	true	true	true	false
α SEC	0.1	0.1	0.1	0.1	0.1	0.1	0.1	-
η SEC	0.1	0.1	0.1	0.1	0.25	0.25	0.25	-
Exact Info								
Max time IRP-DPD	3600 s	3600 s	3600 s	3600 s	3600 s	7200 s	7200 s	7200 s
$\Omega - MF2$	true	true	true	true	true	true	true	true
α SEC	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
η SEC	0.1	0.1	0.1	0.1	0.25	0.25	0.25	0.25

Matheuristic solution information

Table C.2 presents information regarding the average number of iterations in the matheuristic search for each instance type. Iterations$_{Div}$ represents the number of diversification iterations. Iterations$_{Int/Div}$ is the average number of intensification iterations in each diversification iteration. Also the table provides information about the average size of the route pool in the final route selection problem.

Table C.2: Number of iterations and size of the final route pool

Instance	Iterations$_{Div}$	Iterations$_{Int/Div}$	Size of final route pool
C_6V_3A	8.25	16.3	33.5
C_8V_3A	6.8	9.5	47
$C_{10}V_3A$	14	9.3	65
$C_{12}V_3A$	24.2	7.54	72.8
$C_{18}V_3A$	11.8	11.4	77.4
$C_{24}V_5A$	6.8	7.4	109
$C_{30}V_5A$	7.4	4.1	140
$C_{40}V_7A$	5.6	2.5	143
$C_{50}V_8A$	4	1	124

Diversification constraint analysis

Table C.3 shows the dual gaps for different values of the change parameter in the diversification constraints described in section 6.4. The results are an average over the five instances with 24 customers. The matheurisitc is run for 1200 s. For the *Minimum number of changes* constraint the change parameter δ in equation 6.51 is varied between 0.1 and 0.2. For the *Minimum number of changes to routes* constraint the change parameter β 6.54 is varied between 0.4 and 0.6. Evidently $\delta = 0.1$ and $\beta = 0.5$ yields the best results for the respective diversification constraints.

Table C.3: Parameter results for diversification constraints

	Min. changes			**Min. route changes**		
Min change param.	0.10	0.15	0.20	0.40	0.50	0.60
Dual gap, $C_{24}V_5A$	8.00%	8.01%	8.48%	8.47%	8.12 %	8.26 %

Table C.4: Results from the subtour elimination constraint included in the relaxed IRP model

Instance	C_6T_5	C_8T_5	$C_{10}T_5$	$C_{12}T_5$	$C_{18}T_5$	$C_{24}T_5$
$\alpha = 0.1$						
Z	2.31%	1.73%	2.72%	2.74%	2.53%	0.61%
C^T	3.95%	3.78%	3.44%	3.96%	3.88%	0.92%
C^H	-1.49%	0.60%	1.54%	0.51%	-0.81%	-0.14%
ΔT	-13.57%	-19.21%	22.69%	-60.62%	-	-
nCuts	129.8	304.2	603.5	646.0	1089.8	1246.4
IRPdiff	-2.62%	-3.13 %	-6.26%	-6.69%	-8.40%	-8.23%
IRP gap	-	-	-	-	3.06%	4.09%
$\alpha = 0.3$						
ΔZ	2.78%	1.72%	2.72%	2.81%	2.06%	0.51%
C^T	5.67%	2.63%	3.44%	3.96%	3.43%	0.59%
C^H	-4.44%	-0.64%	1.05%	0.51%	-1.28%	0.34%
ΔT	-10%	-16%	23%	-46.45%	-	-
nCuts	158.8	288.6	532.0	598.8	1013.3	1142.8
$IRPdiff$	-2.62%	-3.13 %	-6.26%	-6.66%	-7.40%	-6.91%
$IRPgap$	-	-	-	-	3.65	3.49%%
$\alpha = 0.5$						
ΔZ	0.00%	0.19%	2.06%	2.87%	0.97%	0.98%
C^T	0.00%	-0.69%	2.59%	4.00%	1.53%	1.29%
C^H	0.00%	2.63%	0.81%	0.94%	-0.49%	0.21%
ΔT	-21.96%	-14.96%	-37.79%	-54.39%	-	-
nCuts	93.8	313.0	393.3	462.3	826.5	845.4
$IRPdiff$	0.00%	-0.39 %	-2.06%	-6.26%	-7.40%	-6.91%
$IRPgap$	-	-	-	-	2.20%	3.65%
No elimination						
IRP dual gap	-	-	-	-	0.87%	1.75%

Table C.4 shows results for various α in the subtour elimination constraint 5.10 for the relaxed divisible IRP formulation. ΔZ is the constructed solution's improvement from that without SECs. Likewise, C^T and C^H is the improvement for the transportation costs and holding costs respectively. ΔT is the time improvement when SECs are added. nCuts is the average number of subtour cuts added for each instance type. IRPdiff is the improvement in objective value for the relaxed IRP with SECs. IRPgap is the dual gap of the relaxed IRP.

Evidently the constructed solution improves with SECs in the relaxed IRP. Also note that the transportation costs increases substantially, which is what was the intention with the subtour cuts because the relaxed IRP tends to produce solutions where the transportation costs are too low. The idea behind larger values of α was to, in shorter time, construct solutions in proximity to the solutions with more subtour constraints. However, increasing α much, seems to worsen the solution significantly. Therefore the same values as proposed for the exact method is used in the matheuristic.

Arc-flow solutions

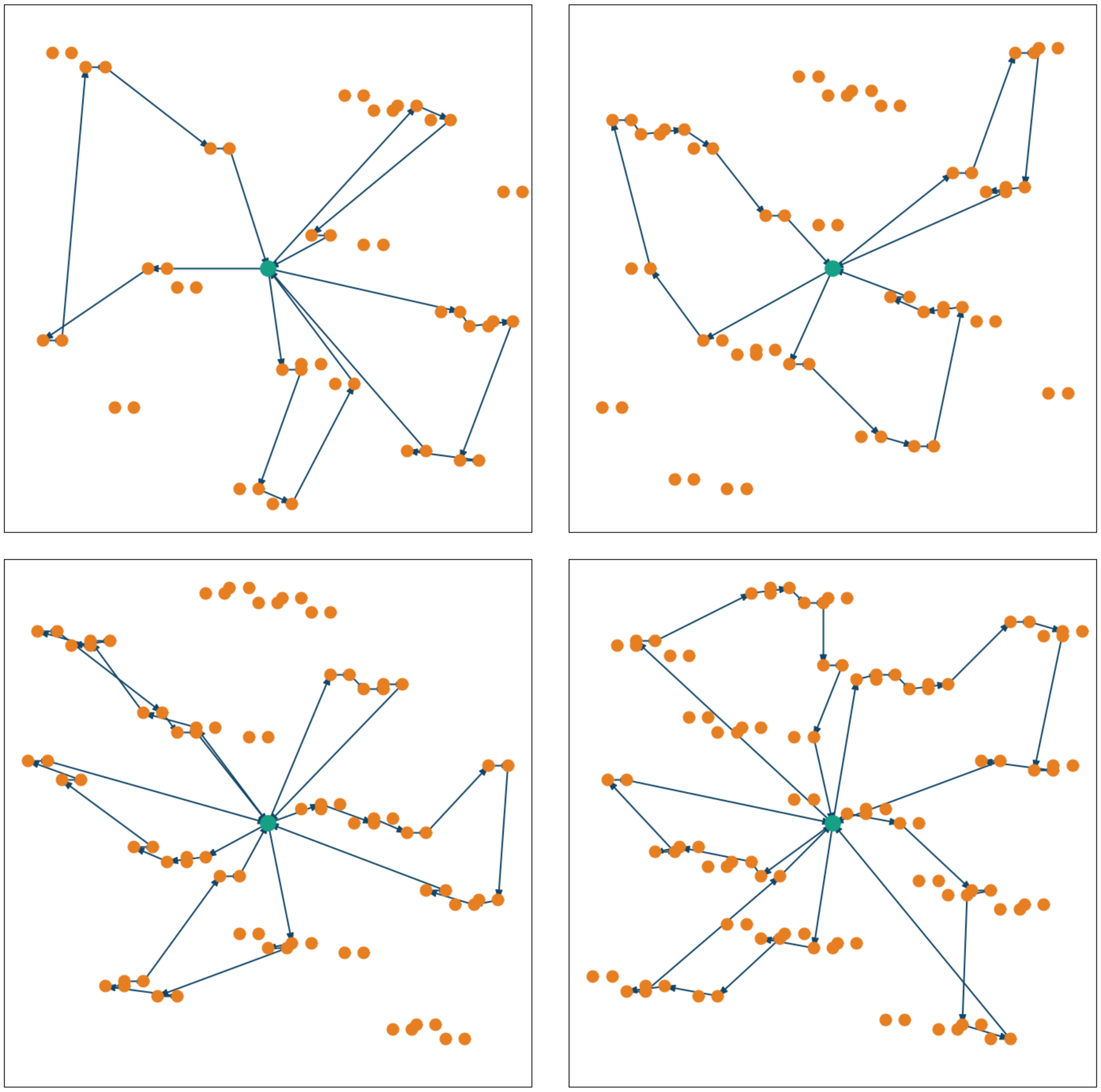


Figure C.1: Illustration of integer arc-flow solutions for 24, 30, 40 and 50 customers

Appendix D

Program design

Classes

Executing class

IRPmodel

Classes for optimization models

IRP

VRPmodel

RouteProblem

Dataholder classes

Node

Node::Edge

NodeIRP

NodeIRP::EdgeIRP

NodeIRPHolder

NodeInstance

NodeInstanceDB

NodeStrong

Route

Classes to store solution information

SolutionInfo

Miscellaneous classes

Solution

GrahpAlgorithm

ModelParameters

The most recent version of the source code is available in the master branch at github.com/einarea/IRPProject. The program provides independent classes for each optimization model. The IRP class is capable of solving various types of IRP, in particular, the IRP-DPD studied in this thesis. It is also used to solve the version where the arc variables are relaxed. The VRPmodel class solves VRP problems for a set of input nodes with given quantities. The RouteProblem class solves various types of route problems, in particular, it is employed by all intensification operators and the final route selection problem.

The program employs various classes to store data in the solution process. These classes are illustrated in figure D.1. The NodeInstance class is used to store predefined customer information such as position, demand and inventory limits. The class NodeIRP represents a customer node in a particular period. This class provides fields for storing variable solution values such as quantity served and inventory. All NodeIRP objects that represent a customer node in any period share the same NodeInstance object from where it accesses the predefined information about a customer.

The solution class is used to store complete information about a feasible solution to an IRP problem. It uses a vector of NodeIRPHolder objects to store data about all customer nodes for all periods. For an instance of n customer nodes the Solution class holds n NodeIRPHolder objects. For T periods each NodeIRPHolder holds T NodeIRP objects. Each NodeIRP object represents either a pickup node or a delivery node in a particular period. Also, each NodeIRP object may have one or several edges to other nodes. The edge class is used to store load and routing information. In an integer solution, only the depot node has more than one edge.

The solution class provides a set of functions which returns information about the current solution, such as objective values, holding cost and transportation cost. Feasibility checks, amount of pickup and delivery, type of visits and number of routes are other pieces of information available. Also, this class has various functions to print the solution in the terminal or plot the arc-flow network. The solution class can also return a vector of Route objects that correspond to the routes of the solution. These routes can be used as input in the Route selection problem. In the matheuristic routes are collected from several solution objects during the iterations before the final route selection problem is solved.

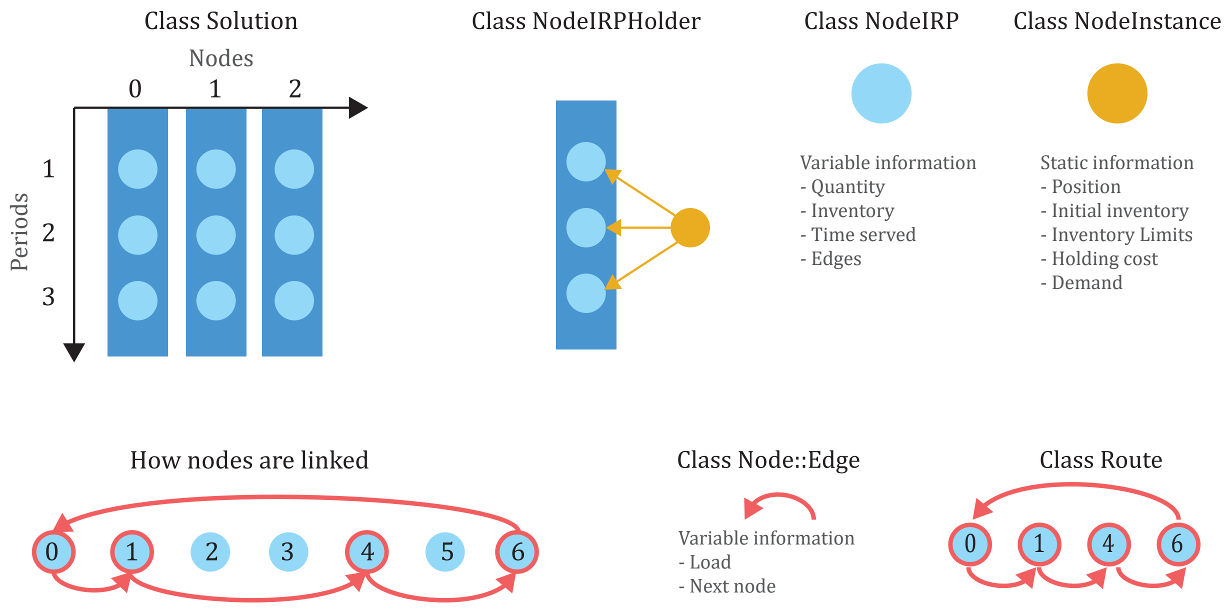


Figure D.1: Illustration of classes that store data in the application

The node and edge structure is well suited for graph traversal algorithms. Collection of routes, calculating costs and various algorithms to modify routes uses the network formed by nodes and edges. Especially the strong component algorithm in class Graph-Algorithm uses the Nodes and edges of a Solution Object to find the strong components in the solution. Subsequently, the corresponding edges are checked for subtour violation.