Optimisation Models and Heuristic Methods for Deterministic and Stochastic Inventory Routing Problems

By

Chanicha Moryadee

Thesis submitted to the University of Portsmouth for the degree of Doctor of Philosophy

Logistics Operational Research and Analytics Group

Department of Mathematics

Supervisors: Professor Dr. Djamila Ouelhadj &

Dr. Graham Wall

September 2017

UNIVERSITY OF PORTSMOUTH

Abstract

The inventory routing problem (IRP) integrates two components of supply chain management, namely, inventory management and vehicle routing. These two issues have been traditionally dealt with problems in the area of logistics separately. However, these issues may reduce the total costs in which the integration can lead a greater impact on overall system performance. The IRP is a well-known NP-hard problem in the optimisation research publication. A vehicle direct delivery from the supplier with and without transhipments (Inventory Routing Problem with Transhipment, IRPT) between customers in conjunction with multi-customer routes in order to increase the flexibility of the system. The vehicle is located at a single depot, it has a limited capacity for serving a number of customers. The thesis is focused on the two main aspects: (1) Development of the optimisation models for the deterministic and stochastic demand IRP and IRPT under ML/OU replenishment polices. On the deterministic demand, the supplier deliveries products to customers whose demands are known before the vehicle arrives at the customers' locations. Nevertheless, the stochastic demand, the supplier serves customers whose actual demands are known only when the vehicle arrives at the customers' location. (2) Development of integrated heuristic, biased probability and simulation to solve these problems. The proposed approaches are used for solving the optimisation models of these problem in order to minimise the total costs (transportation costs, transhipment costs, penalty costs and inventory holding costs). This thesis proposed five approaches: the CWS heuristic, the Randomised CWS heuristic, the Randomised CWS and IG with local search, the Sim-Randomised CWS, and the Sim-Randomised CWS and IG with local search. Specifically, the proposed approaches are tested for solving the deterministic demand IRP and IRPT, namely, the IRP-based CWS, the IRP-based Randomised CWS, the IRP-based Randomised CWS and IG with local search. For the transhipment case are called the IRPT-based CWS, the IRPT-based Randomised CWS, and the IRP-based Randomised CWS and IG with local search. On the stochastic demand, these proposed approaches are named the SIRP-based Sim-Randomised CWS, the SIRPT-based Sim-Randomised CWS, the SIRP-based Sim-Randomised CWS and IG with local search, and the SIRPT-based Sim-Randomised CWS and IG with local search. The aim of using the sim-heuristic is to deal with stochastic demand IRP and IRPT, the stochastic behaviour is the realistic scenarios in which demand is used to be addressed using simulation. Firstly, the Sim-Randomised CWS approach, an

initial solution is generated by Randomised CWS heuristic, thereafter an MCS is combined to provide further improvement in the final solution of the SIRP and the SIRPT. Secondly, the integration of Randomised CWS with MCS and IG with local search is solved on these problems. Using an IG algorithm with local search improved the solution in which it generated by Randomised CWS. The developed heuristic algorithms are experimented in several benchmark instances. Local search has been proven to be an effective technique for obtaining good solutions. In the experiments, this thesis considers the average over the five instances for each combination and the algorithms are compared. Thus, the IG algorithm and local search outperformed the solution of Sim-Randomised CWS heuristics and the best solutions in the literature. This proposed algorithm also shows a shorter computer time than that in the literature. To the best of the author' knowledge, this is the first study that CWS, Randomised CWS heuristic, Sim-Randomised CWS and IG with local search algorithms are used to solve the deterministic and stochastic demand IRP and IRPT under ML/OU replenishment policies, resulting of knowledge contribution in supply chain and logistics domain.

Keywords: Inventory routing problem, Vehicle routing problem, Vendor managed inventory, Transhipment, randomised CWS heuristic, sim-heuristic, Sim-Randomised CWS heuristic, Iterated greedy algorithm and Local search.

Acknowledgements

I would like to express my deep gratitude to my supervisors Professor Djamila Ouelhadj and Doctor Graham Wall, for their patient guidance, enthusiastic encouragement and useful critiques of this research work. I would also like to thank Professor Doctor Djamila Ouelhadj, for firstly giving me the opportunity to study at the University of Portsmouth and her advice and assistance in keeping my progress on schedule. I greatly appreciate numerous staffs in Department of Mathematics and Graduate School for their valuable guidance and encouraging support. They give me fundamental technical advice and the general insight on working of academic research.

I would further like to thank the Suan Sunandha Rajabhat University in Thailand which partially funded and supported me to study PhD and this research.

I would grateful to my examiners Doctor Mahmood Shafiee and Doctor Xiang Song for all the detailed constructive comments and feedbacks that helped to improve this research.

I would also like to extend my thanks to the technicians of the IT support for their help in offering me the resources in running programme. I am grateful to all my friends who were supportive and made my time as a PhD student enjoyable.

Finally, I wish to thank my husband and my parents for their patience, persistent support and encouragement throughout my study.

Declaration

Whilst registered as a candidate for the above degree, I have not been registered for any other research award. The results and conclusions embodied in this thesis are the work of the names candidate and have not been submitted for any other academic award.

Signature: 

Table of Contents

Table of Tables

Table of Figures

Abbreviation	Description
ACA	Ant Colony Algorithm
ALNS	Adaptive Large Neighbourhood Search
CVRP	Capacitated Vehicle Routing Problem
CWS	Clarke and Wright Savings Heuristic
DSS	Decision Support Systems
EOQ	Economic Order Quantity
EPQ	Economic Production Quantity
GA	Genetic Algorithm
GRARP	Greedy Randomized Adaptive Search Procedure
ILS	Iterated Local Search
IG	Iterated Greedy
IRP	Inventory Routing Problem
IRPSD	Inventory Routing Problem with Stochastic Demand
IRPT	Inventory Routing Problem with Transhipment
LNS	Large Neighbourhood Search
LS	Local Search
MCS	Monte Carlo Simulation
MDVRP	The Multi Depot Vehicle Routing Problem
ML	Maximum Level
MILP	Mixed Integer Linear Programming
NP-hard	Non-deterministic Polynomial Hard
OU	Order Up to
Randomised CWS	Randomise version of Clarke and Wright Saving
SA	Simulated Annealing
Sim-heuristic	Simulation optimization using Heuristics
Sim- Randomised CWS	Simulation Randomised Clarke and Wright
TSA	Tabu Search Algorithm
VMI	Vendor Management Inventory
VRP	Vehicle Routing Problem
VRPSD	Vehicle Routing Problem with Stochastic Demands

Table 0.1: List of abbreviations used in this thesis

Symbol	Description
V	Set of customers, vertices or nodes including the depot
V'	Set of customers, vertices or nodes without the depot
E	Set of arcs or edges between all nodes
n	Total number of nodes on a data instance
i, j	Given nodes on a data instance
C_i	Capacity of inventory customer i
h_0	Holding cost of inventory at depot
h_i	Holding cost of inventory customer i
t	Planning horizon time period
p	Total number of time period
r_0^t	Quantity of products made available at depot each time instant
r_i^t	Quantity of products made available at customer i each time instant
I_0^t	Current inventory level of the depot in time period t
I_i^t	Current inventory level of the customer i in time period t
d_i^t	Demand of each customer i in time period t
D_i^t	Random variable for aggregated demand of node i in time period t
Q	Capacity of the vehicle
c_{ij}	Cost associated to the connected arc between nodes $i\ and\ j$
x_{ij}^t	Binary variable indicating that arc (i, j)
y_{ij}	Vehicle load arriving at node j after visiting node i
w_{ij}^t	Quantity of product delivered from the customer i to the customer j
q_i^t	Quantity of product delivered from the supplier to the customer i
b_{ij}	Cost associated with the transhipment from $i\ to\ j$
βc_{ij}	Cost associated with the transhipment from $i\ to\ j$
c_{rt}	Cost of the route performed in period t
l_i^t	Stock-out inventory level of the customer i
p_i	Penalty cost of the customer i
s_i	Level of inventory reorder point of the customer i
z^t	The quantity of product made available at depot can be used for deliveries to customer in time period t

Table 0.2: Mathematical notation used in this thesis

Chapter 1: Introduction

1.1. Background and Motivation

The Logistics and Supply Chain Management field provide a value-added and focus on an organisation with regards to its demand management, distribution planning, production planning and control, order placing, facility location, and inventory management accuracy. The business plays an attention on logistics - essentially an integrated framework that seeks to create a single combination plan, for the flow of material and information through a business. Therefore, supply chain management involves getting the right product to the right place in the right quantity at the right time. It also ensures product arrives in the best condition and is delivered at an acceptable cost and in the most efficient way. This should result in total system costs, (including transportation, distribution and inventories) being reduced.

The fierce competition in local and global markets are pressuring businesses to streamline their logistic systems. Generally, the business organisation controls only small sections of the value chain it is offering. The logistics activities are also concerned with connecting material flow of processes, such as the application for combination of quantitative decisions and shipping, as tackled in the inventory management and routing decisions. The major costs of logistics are due to transportation, this represents about one third of all costs; inventory costs represent about one fifth. Therefore, a supply chain is focusing on planning and taking advantage of depots, customers and vehicle capacity, to minimise the overall logistics costs. It is reasonable that a business is mainly focused on adding value to the chain and it also has the challenge of gaining a competitive advantage by leveraging it (Zhang, Vonderembse & Lim, 2003).

For the company, considering only the inventory management is not the proper way of taking into consideration, where the routing aspects of the transportation are not properly treated and routing on the others with a number of predefined. In orders to serve a natural extension to both problems is better to combine them. In the combined problem, aims to reduce the total costs and improve the service level of the supply chain system. Consequently, companies are critically re-evaluating the common techniques and all areas of operations, looking for methods of optimisation. However, some research has recognised that local optimisation of any activity of the value chain, will not guarantee system wide optimisation (Kheljani, Ghodsypour & Brien, 2009).

The success of the firm will depend on its ability to integrate with other participants in the chain who are responsible for physical, financial flows and information. Consequently, various coordination techniques have risen up for achieving total effectiveness across the entire chain. Vendor Management Inventory (VMI) is one of the most up to date examples of value added through logistics which has been given a lot of attention in recent times. It is an inventory management technique in which the supplier is responsible for optimising the inventory held by a buyer or retailer. Also, it can be described as a win-win situation.

It is remarkable that this thesis also adopted VMI for reducing the total costs and improving the efficiency of the operations. To the best our knowledge, many researchers are actively engaged in the application of studies related to VMI, but they have limited to replenishment decision, contracts, trust and relationships as well as strategic implication of a mechanism (Marques et al., 2017). Traditionally, the supplier-retailer relationship has been characterised by the stronger party strong arming the weaker party and attempting to extract the maximum out of the engagement. VMI reformulates the problem as a case of ensuring optimal supply chain coordination (Disney & Towill, 2003). Thus, the focus is on the financial and operational well-being of all the partners. This is a fundamental redrawing of the nature of the transaction, and it affects both the vendor as well as the retailer. The retailer must provide the vendor access to all relevant information and the latter, on this part, must make the retailer feel comfortable regarding the reliability and durability of the relationship (Al-ameri, 2008).

Today, business drives the competitive advantage in global markets with a focus on reducing costs in the activities of the supply chain. The continuous evolution of the techniques to manage their operations effectively, is motivated by the customers' heightened expectation on products and services from suppliers. Operational Research can be used to focus on the integration of two components of supply chain management: inventory control and vehicle routing. These two issues have been traditionally dealt with separately, but their integration may provide a dramatic impact on overall system performance. This results in a very complex problem called the Inventory Routing Problem (IRP). It captures the essential characteristics of a VMI policy and designs the methodologies to solve this problem for logistics planning systems. This thesis presents the IRP assumes application of VMI concept where each client is served by at most one vehicle in each time unit. Here, the supplier is

responsible for all decisions in which the vendor monitors the inventory and decides the replenishment policy of each customer. The advantage of VMI policy is, more efficient resource utilisation, in which the supplier can reduce its inventories while reduce the routing cost through a more uniform utilisation of shipping capacity. In this thesis case, the supplier acts as a central decision maker and applies maximum level (ML) policy and order-up-to policy (OU) to solve an integrated inventory routing problem.

The integrated inventory routing problem has classical vehicle routing problem structure. In the vehicle routing perspective, the supplier makes decisions in place of the customers, without the customer order, and also makes a decision on the product quantity to be unloaded for the customers each period. However, the suppliers have to be aware that decisions made today will have an impact on future operations and can further decrease the total cost, which is achieved by coordinating their systems with several customers. Therefore, inventory management alone is not sufficient handle the routing problems of transportation. A natural approach will be to combine the two problems. In the combined problem, key components from both inventory management and routing, are modelled appropriately. The benefit of combining these problems is, it offers a minimisation of system costs while satisfying service level requirements. Additionally, large costs are induced in the event that the unavailability of the product causes interruption to the supply chain flow (Henrik et al., 2010).

Many versions of the problem have been discussed in a large number of papers. Methodologies to model these two activities simultaneously have been proposed in the literature. According to (Campbell, 1997), IRP has since then been one of the most commonly used name for the problem of combining inventory management and routing as illustrated in Figure 1.1. The objective of the integrated inventory problem is to decide when and how many units to order for each item, so as to minimise inventory holding and ordering costs, over the planning horizon (Chakravarty, 1985 and Bastian, 1986).

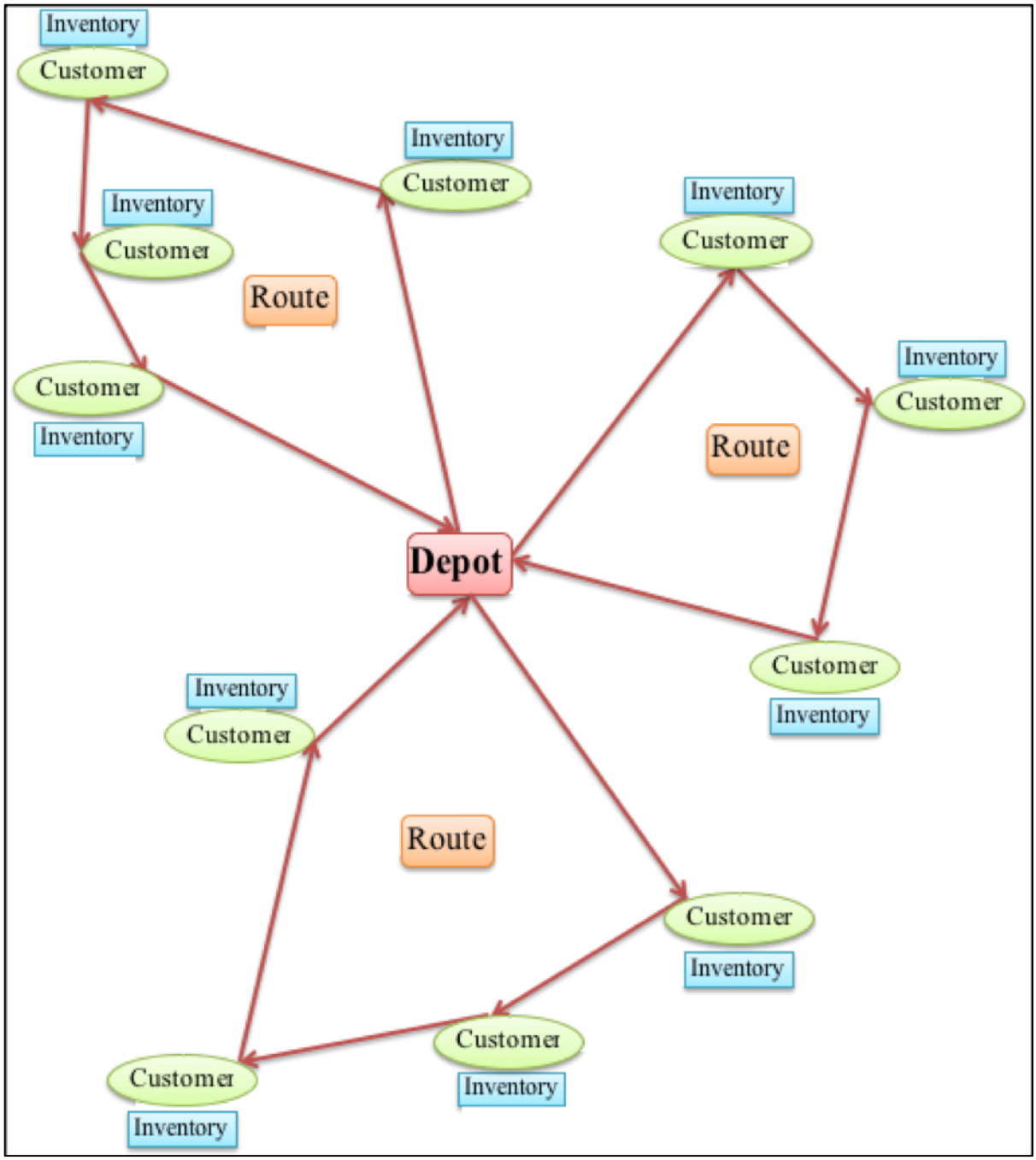

Figure 1.1: An example of the IRP concept

Another significant variant of the IRP is the problem associated with future demand which is uncertainty. This is referred to as a stochastic IRP (SIRP) (Campbell et al., (1998); Kleywegt, Nori & Savelsbergh (2002, 2004) and Hvattum, Lokketangen & Laporte (2009)).

The stochastic IRP has received significant attention in recent years, with important variants of the problem being studied, such as stochastic demand with stock-out and transhipment. Several methods have been proposed to solve the stochastic IRP. Some

authors focused on the IRP with stochastic demand, allowing stock-outs and rollout periods, with replenishment policies implemented to resupply the customers (Bertazzi et al., 2013 and Juan et al., 2014). Coelho et al. (2014), studied a dynamic version of the IRP where customer demand was gradually revealed over time, and planning was made at the beginning of each period. The use of lateral transhipment between customers was allowed, in order to avoid the customer facing stock-out when demand was high.

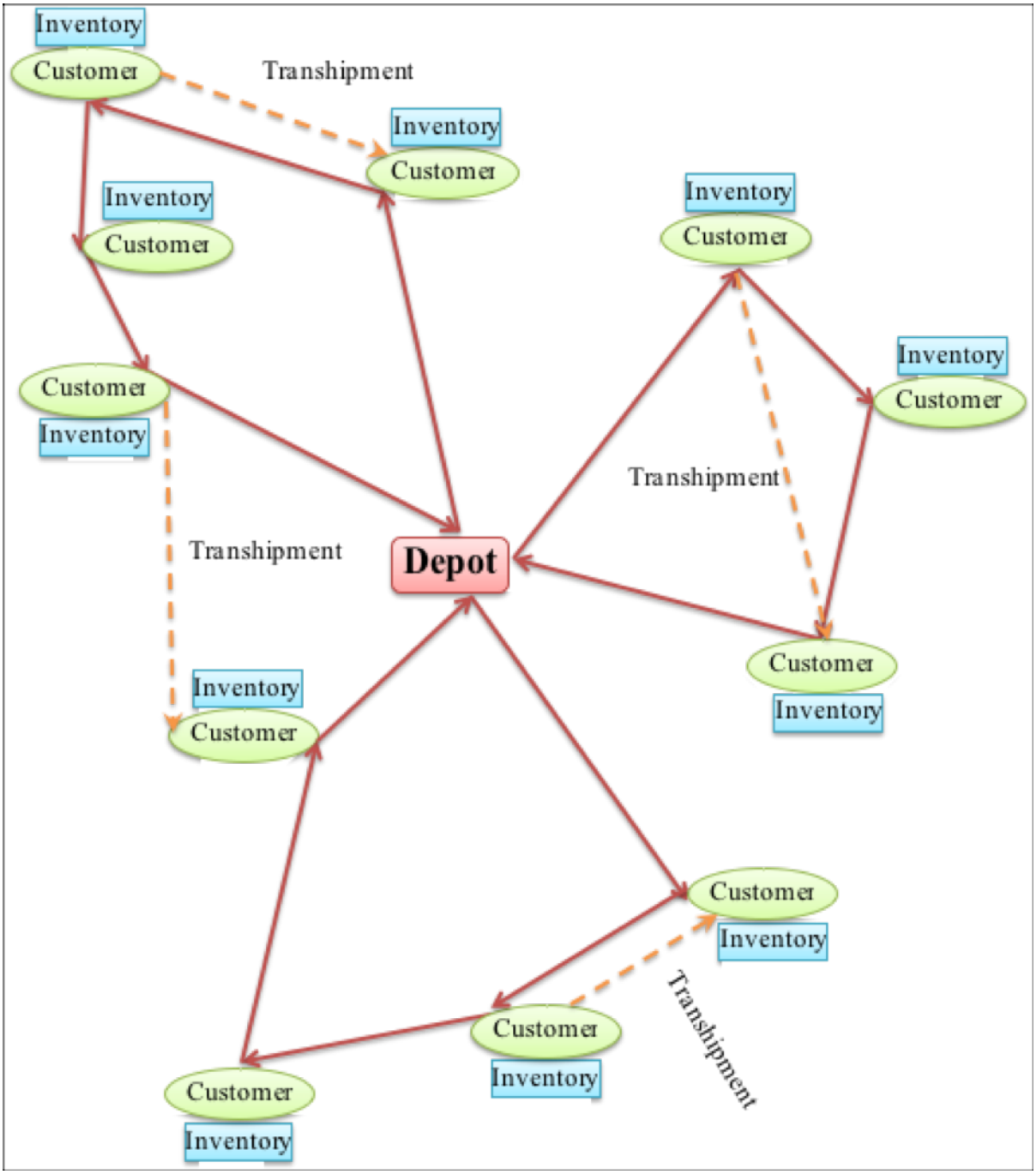

Figure 1.2: An example of the IRPT concept

Later, the IRP with lateral transhipment (IRPT) was introduced so that products may be shipped from either the supplier or customer, to the requested location as shown in Figure 1.2. This happens, for example, between stores belonging to the same chain, which can ship merchandise to one another when unforeseen demand variations occur. However, transhipments may be beneficial in a deterministic context, in which no shortages occur, because they may yield an overall reduced distribution and inventory holding cost. Mercer et al. (1996) provide an example of an inventory routing system used by the supermarket chain Tesco, in the United Kingdom, where deliveries are made from a factory to several warehouses, and lateral transhipments can take place between warehouses. Using lateral transhipment is an option where the supplier selects a customer to delivery products to another customer. The transhipments within the IRP was introduced by Coelho et al. (2012), this allows transhipments either from the supplier to the customers, or between customers. Due to the literature review (chapter 2), this research has taken the lateral transhipment to this problem account. This is an alternative for reducing the total costs and increasing the performance of the supply chain.

According to the literature review in Chapter 2, there are many number of OR method is used to solve the IRPs, including exact algorithms, heuristics, meta-heuristics and sim-heuristics. Some of these techniques are iterated local search (Ribeiro et al., 2003), variable neighbourhood search (Zhao et al., 2008), greedy randomised adaptive search (Campbell et al., 2004), memetic algorithms (Boudia et al., 2009), Tabu search (Archetti et al., 2012) and adaptive large neighbourhood search (Leandro et al., 2012). There is significant attention for this thesis uses the heuristics and implement an improve heuristics to improve the solution of the IRP and IRPT.

Recently, studies on stochastic IRP are using a combination of simulation and optimisation approaches, which are usually used for stochastic IRP, to find the solution for such complex real-life problems. A simulation optimisation technique is used to solve the particular inventory problem. It also can help to deal with more realistic and complex scenarios. The sim-heuristic approach is a particular case of simulation-optimisation which combines a heuristic or meta-heuristic algorithm with simulation approaches. The potential of simulation techniques has been widely proven especially in the stochastic behaviour in a real system that used to be addressed using simulation (Glover et al., 2001). A stochastic system is a set of dynamic interdependent components where some values of its variables change

randomly. Therefore, the simulation processes with stochastic variables are related to the basic mechanisms (Juan et al., 2014). Examples of the application of simulation-based optimisation can be found in scheduling (Ouelhadj et al., 2009), the supply chain (Ding and Schumacher, 2004; Wang et al., 2006 and Arisha et al., 2010), the telecommunication networks (Garcia et al., 2009; Cabrera & Ros, 2009 and Sarkar, Member, & Halim, 2011) and the city logistics (Barcelo, Grzybowska & Pardo, 2007; Taniguchi, Thompson & Yamada, 2012 and Montoya & Herazo, 2014).

A detailed review of related literature shows that heuristics and metaheuristics have been applied to solve the IRP. However, there is a gap in the exploration of the CWS and the combination of CWS and biased randomized approaches used for the IRP, with and without transhipment. A Biased randomised heuristic allows for the generation of good quality solutions by using a skewed probability distribution to guide the solution construction process. This enhances a deterministic heuristic into a probabilistic procedure that can be run multiple times. This method has been successfully applied in Vehicle Routing Problems (VRP).

Regarding IRP with stochastic demand, there are very few papers proposing the use of sim-heuristics to solve the problem. This research, therefore, encompasses a transhipment method applied to the stochastic problem, in order to reduce total costs with no stock out. Besides, the Sim-Randomised IG heuristics is applied in this research in case the problem size is very large. The solutions from this research are then compared to the previous studies; the results show that the new algorithms here can be more effective when compared to those in previous studies.

Additionally, from the literature reviewed, IRP is an increasingly active area within the academic community and produces considerable cost and performance benefits within supply chain sectors. Therefore, this research adopts the principle of IRP to enhance a real-world problem. Adding transhipment to IRP is (IRPT) expected to have a knowledge contribution to both the academic community and practitioners.

This research work reviews cases where the supplier has to make three simultaneous decisions: (1) When to supply products to customers, (2) How much to deliver and (3) How to combine customers into routes. Considering the quantity of deliveries, the two replenishment policies applied are the order-up-to (OU) and the maximum level (ML) replenishment policies. The OU policy delivers a quantity to the customer up to a pre-set level. The ML policy is where the supplier decides and calculates the quantity to

deliver to the customer, as long as its holding capacity is respected. The transhipment concept within the IRP is introduced later in this thesis. Transhipment is where a delivery is made to the customer, either directly from the supplier or from another customer. This happens when extra demand variations occur. Furthermore, the IRP model makes inventory decisions at a depot only and considers deterministic and stochastic demands. The transportation costs are included from the supplier to customers and from one customer to another customer (transhipment costs). The penalty costs are also added into the total costs when the customer faces stock-out.

1.2. Aims and Objectives

This research aims to improve the existing solution methods for deterministic and stochastic IRP and IRPT in order to minimise the total costs.

The objective of this research as;

- Develop an optimisation model and the algorithm for solving the deterministic and stochastic demand IRP and IRPT in order to minimise the total costs.
- Develop both the OU and ML inventory replenishment policies to determine the quantity for deliveries.
- Develop the lateral transhipment policies with the IRP to reduce the total costs.

These objectives can be achieved by:

- A critical review of literature in the area of IRP with the deterministic and stochastic demand, IRP with and without the lateral transhipment policy and the inventory replenishment policies will be given.
- Formulate optimisation models for the IRP and IRPT. In these models, the total cost to be minimised is the inventory holding costs at the supplier and at the customers, routing costs for the supplier's vehicle and transhipment costs.
- Develop the CWS, Randomised CWS heuristics and Randomised CWS and IG algorithm with local search will be implemented to solve the IRP and IRPT with deterministic demand.
- Develop a stochastic optimisation model and investigate methods to solve IRP and IRPT with stochastic demand. Here, simulation optimisation will be applied with randomised CWS heuristic (Sim-heuristic) to solve the IRP and IRPT with stochastic demand.

- Develop and implement an algorithm which combines a sim-heuristic with Randomised CWS and IG algorithm with local search, to improve the solution.
- Evaluate the performance of optimisation models and algorithm under uncertain scenarios instances from the literature through computational experiments.
- Analyse the results and conduct a comparative analysis with the results in the literature.

1.3. Contributions

The research implemented in the presented thesis and contributions made during this study will help achieve the objectives described in the previous section. The most relevant ones are summarised below and are explained in detail in the study.

A. **Advance optimisation models for the deterministic and stochastic IRP and IRPT:** The optimisation models are proposed for solving IRP and IRPT with deterministic and stochastic demands are presented. The objective is to minimise the sum of inventory holding costs at supplier and at the customers, including routing costs for supplier's vehicle and transhipment costs.

B. **Develop advance heuristic methods for solving the proposed models of the deterministic and stochastic IRP and IRPT:**
   - The deterministic demand:
     - CWS heuristic and a Randomised CWS heuristic: To the best of our knowledge, that this research uses the CWS, Randomised CWS algorithms and Randomised CWS and IG algorithm with local search to solve IRP without and with transhipment (IRPT) under OU/ML policies.
   - The stochastic demand:
     - A Sim-Randomised CWS heuristic, and A Sim-Randomised CWS heuristic with IG algorithm with local search use for solving IRP and IRPT with stochastic demand. This approach has not been applied in the literature for solving IRP and IRPT with stochastic demand under OU/ML replenishment policies.

1.4. Structure of this Thesis

This thesis is divided into eight chapters, discussing the issues concerning the Inventory Routing Problem without and with transhipment (IRP and IRPT) under the replenishment policies. The general presentation will provide a research background, which focuses on a theoretical basis for studying the IRP. The policies are provided for improving the methodology and also, the algorithm techniques are implemented for solving the mathematical model of IRP and IRPT. The chapters are as follows:

Chapter 1 contains the introduction and provides the research background and the aims and objectives of the research.

Chapter 2 provides a review of the existing literature dealing with the IRP, followed by the problem definition, the characteristics of IRP, critical current theories, and a discussion of various methodologies. A review of both deterministic and stochastic demand of IRP and the lateral transhipment policy. Moreover, it provides an explanation of the exact methods and heuristic approaches used to solve to IRP with different objective problems. The application of simulation optimisation with heuristic is based on the use of biased randomisation for solving complex optimisation problems.

Chapter 3 introduces the main part of the thesis, whereby the mathematical model of the IRP and IRPT are presented. An explanation of mathematical model is given for the IRP and IRPT with deterministic demand and this proposes methodologies which are used, such as the classical CWS heuristic. The order up to (OU) and maximum level (ML) replenishment policies and lateral transhipment are also introduced. Some results from the experimental work are presented.

Chapter 4 proposes the use of the Randomised CWS heuristic for IRP and IRPT with deterministic demand. The methodology has been introduced to be useful and apply biased randomisation with another heuristic for solving complex problems such as the Clark & Wright Saving algorithm and biased-randomised heuristic. This chapter aims to implement these algorithms to improve the solution of the IRP and IRPT with deterministic demand.

Chapter 5 proposes the combination of Randomised CWS and the Iterated Greedy (IG) algorithm with local search. The algorithm from chapter 4 is implemented, along with a literature review covering the improved and applied heuristic to obtain

solution methods (local search, IG) for deterministic demand. The performances of these methods are compared with benchmark results.

Chapter 6 considers stochastic demand and the Sim-Randomised CWS heuristic. The topic is introduced by a discussion of how to deal with stochastic problems and methodologies needed to solve them. The extension of the deterministic case to a stochastic case by proposing heuristic with MCS to solve this stochastic problem. The mathematical modelling formulation and constraints are described. A simulation technique is used and combined with a heuristic for solving IRP and IRPT with stochastic demand under OU/ML policies. The results of IRP and IRPT for both policies are presented.

Chapter 7 proposes the combination of Sim-Randomised CWS and the Iterated Greedy (IG) algorithm with local search for stochastic demand. The algorithm from chapter 5 is implemented, along with a literature review covering the improved and applied heuristic and meta-heuristic to obtain solution methods (local search, IG). The performances of these methods are compared with benchmark results.

Chapter 8 provides conclusions of the research with recommendations. This includes areas for future work.

1.5. Chapter conclusion

This chapter shows a clear picture of the presented problem and also presented an introduction of the whole thesis. These are the objectives, research gap, contribution and brief explanations of this research structure. Detailed description of all chapters will follow. First is a review of literature, then the problem definition, followed by mathematical models and problem description. The discussion of different algorithms for solving the IRP and IRPT and the experimental results along with comparison of our approaches to previous works is also given. Last but not least, the conclusion is given. More details of the main contribution of this thesis will also be highlighted in each chapter.

Chapter 2: Literature Review

2.1. Introduction

The inventory management and vehicle routing decisions are two key decisions traditionally made sequentially. This is a type of sequential decision making traditionally lacks association between the two components in which it does not allow for the optimal performance of the logistics and supply chain. Therefore, combining them both is key interested concept, which attracted several authors to study it from a research perspective. It is reasonable for taking the deterministic and stochastic demands into the key concept of integrated two problems. Including, the study for the efficiency of planning along with similarities and dissimilarities between different company sectors. In general, the main aim is looking for efficient use of an optimisation algorithm with simulation to solve the IRP and IRPT in which it is designed to find near-optimal, good or high-quality solutions, be simple to configure, flexible to be adapted to new constraints, and easy to understand and implement. The overall objectives are to improve on the existing current solution methods for the small, medium and large size of the deterministic and stochastic IRP and IRPT, with relatively simple constraints.

The vendor managed inventory (VMI) strategy is based on solving the complex mathematical problem called the Inventory Routing Problem (IRP), which combines the two-common problem: Inventory management and vehicle routing. VMI, involves combining the distribution process and the replenishment policy which can lead to an overall reduction in logistics costs. The OU and ML inventory replenishment policies are used to determine the deliveries' quantity. The actual demand is shown only when the vehicle arrives at the customer's location. In this type of problem, the supplier plays an important role in the planning, decision making and scheduling aspects. This study considers an environment in which products are shipped from a single depot to many customers by using a single vehicle. The total costs are encountered for distance travelled and those costs are included in the objective function. The single vehicle case was solved exactly by branch and cut (Archetti & Laporte, 2007; Solyali, 2011) and heuristically in (Coelho, Laporte, & Cordeau, 2012).

2.1.1. **Background**

IRP is a well-known research area that simultaneously considers inventory allocation and routing problems in order to optimise delivery plan in a given planning horizon, and to achieve a better overall performance in supply chain systems (Bertazi et al., 2012). The first studies published on the IRP were mostly variations of models designed for the Vehicle Routing Problem (VRP) and extended to take inventory costs into consideration. The IRP normally consists of one supplier and many geographically dispersed retailers each with a different demand. The products are shipped from the supplier to retailers using a set of available vehicles over a given time horizon while the supplier also makes replenishment decisions based on specific inventory and supply chain policies. Therefore, each retailer is served by at most one vehicle in each time unit used by the VMI mode. VMI is one of the most up to date examples of value added through logistics. The essence of IRP comes from VMI such as suppliers and supermarkets, store chains, automotive industries, clothing industries, by integrated approach of the IRP (Campbell & Savelsbergh, 2004). Thus, inventory management and every supply chain accomplishment increase their effectiveness and improve performance by VMI design (Campbell et al., 1998)

Besides, the supplier determines the replenishment policy of each retailer as well as the vehicle routes, guaranteeing no stock outs. Cetinkaya and Lee (2000) and Chaouch (2001) proposed an analytical model for joint inventory and transportation decisions in VMI systems. They also, considered a vendor realising a sequence of random demands from a group of retailers located in a given geographical region. Chrysochoou and Ziliaskopoulos (2012) opined that VMI systems seem to be at the core of most global supply chains. The concept behind this business model is that the supplier monitors the inventories of each retailer and determines the replenishment policy, guaranteeing that no stock out will occur. The IRP constitutes the backbone of the VMI model. The IRP aims to jointly optimise transportation and inventory decisions and IRPs not only find the delivery routes, but also the quantity to deliver and the time for delivery (Campbell et al., 2004; Andersson et al., 2010 and Coelho et al., 2013). Only considering the inventory management is not sufficient if the routing aspects of the transportation are not properly treated both scheduling deliveries and routing on the others.

In an integrated system, key components from both inventory management and routing are modelled appropriately (Henrik et al., 2010). An integration of inventory

management and the vehicle routing problem has been discussed in a large number of papers and the methodologies to model these two activities simultaneously have been proposed, with the main aim to minimise the total costs of inventory and transportation. A substantial body of knowledge concerning the IRP has been developed over the past 25 years since the seminal work by Bell et al. (1983). The authors described a case where only transportation costs are considered; the demand is stochastic and customer inventory levels must be not being left unmet. This was followed by a number of variants of the problem. As well as many versions of the problem, there are also a number of different areas covered in the literature. The combination of inventory management and routing problem has reformed the problem as a routing problem with explicit inventory features in the previous work of (Campbell, 1997).

In a basic integrated inventory problem, a single facility replenishes a set of items over a finite horizon. Whenever the facility places an order for a subset of the items, two types of costs are incurred: An integrated set-up cost and an item-dependent setup cost also, called major set-up cost and minor set-up cost. These costs are fixed. The IRPs differ from the classical VRPs because, using demand rates, IRP tries to create an integrated strategy for stock replenishment, defining simultaneously the replenishment cycles and the route to be used in the delivery process (Custodio & Oliveira, 2006). Basically, the problem deals with two logistics function: The distribution integration through the IRP and supplier integration through the VMI policy (Jemai et al., 2012).

Campbell and Savelsbergh (2004), Henrik et al. (2010) and Leandro et al. (2012) described the IRP as a problem consisting of one supplier and many geographically dispersed retailers. Each retailer has a different demand whereby products are shipped from the supplier to retailers using some identical vehicles over a given time horizon. To illustrate the difficulty of IRP, a small deterministic example was given by (Bell and Fisher, 1983). The relevant optimal tour costs can be derived from the network shown in Figure 2.1, e.g., the optimal tour cost for visiting customers 1 and 2, denoted by is equal to £210. The vehicle capacity is 5000 gallons and customer tank capacity and usage data are also in gallons.

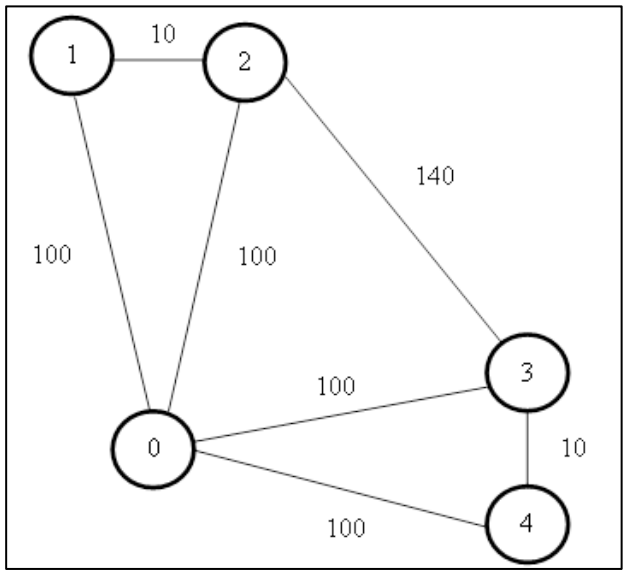

Figure 2.1 : Four customer example with distances shown on edges

A simple schedule jointly replenishes customers 1 and 2 as well as customers 3 and 4 on a daily basis. This schedule is natural because customers (1 and 2) and (3 and 4) respectively have a close proximity to each other. Each customer receives a quantity equal to its daily consumption. The long-run average cost (distance) of this schedule is 420 miles per day. An improved schedule consists of a cycle that repeats itself every two days. On the first day, one trip replenishes 3000 gallons to customer 2 and 2000 gallons to customer 3, at a cost of 340 miles. On the second day, two trips are made. The first trip replenishes 2000 gallons to customer 1 and 3000 gallons to customer 2 and the second trip replenishes 2000 gallons to customer 3 and 3000 gallons to customer 4.

The structure of this chapter is following; firstly, section 2.2 gives an overview the VMI and inventory policies. In section 2.3, a classification of IRPs including an overview of reviewed literature in the IRP with deterministic and stochastic versions is given. The transhipment and inventory policy is also given in this section. The solution methodologies proposed to solve IRP and IRPT with deterministic and stochastic demand, including the combination of Sim-Opt with heuristic/meta-heuristic are in section 2.4. Finally, section 2.5 presents the chapter conclusion.

2.2. The VMI and Inventory replenishment policy

VMI is a supply chain strategy where the supplier has the responsibility of managing the inventories of customers. VMI has become more popular in different industries and market sectors in the last 15 years (Disney & Towill, 2003). The VMI is able to

improve supply chain performance by decreasing inventory level and increasing fill rates. Achabal et al. (2000) presented VMI systems that can be applied for certain manufactures to help improve both retail customer service and inventory turnover. It has also been proven to improve customer service levels by reducing higher fill rate and order cycle times (Yao, Evers, & Dresner, 2007). The VMI development is motivated by the need to establish a collaborative relationship between supplier and customers. The objectives of developing VMI system encompass three key aspects: (1) Give the retailers/customers the best opportunity to purchase the vendor's products. (2) Help the retailers manage their inventory more effectively, and (3) assist the vendor in production scheduling (Achabal et al., 2000).

From the buyer's point of view, establishing good relationships with suppliers can lead to better financial performance, increase supply reliability, lower production costs, and increase the ability to satisfactorily resolve conflicts (Han et al., 1993; Barnes, 1994; Carr & Smeltzer, 1999). Kanna and Tan (2006) examined the outcomes of both operational and strategic levels when a buyer develops close relationships with key supplies. They found that benefits for the buyer includes cost reduction, better quality products and good delivery service on the operational level. On the strategic level, benefits such as enhancements in product/service quality and innovation, improved competitiveness, and increased market share are realisable. From a supplier's perspective, some advantages of establishing collaborative relationships with buyers include higher profitability, reduced uncertainty, managed dependence, exchange efficiency, and insulation from price competition (Hartfield & Olorunniwo, 2001; Ritter & Gemunden, 2001). This shows that good buyer-supplier relationships can result in better buyer's and supplier's performance in both monetary and non-monetary aspects.

The integration of inventory management and routing decisions in supply chain management is essential for logistics managers to lead the logistics planning function of inventory control and transportation planning. The management of these two activities can be an extremely important source of competitive advantage in logistics and supply chain management. The variant of the routing problem that currently gets much attention are the Inventory Routing Problem (IRP) (Vansteenwegen and Mateo, 2014). However, many variants of the problem exist, since different researchers infrequently define the problem in exactly the same way. It is known, the IRP is typically an issue at hand in value added in logistics (VMI). If VMI is designed well it

can reduce the inventory level and raise supply chain integration through reducing system costs (Coelho, Laporte and Cordeau, 2012). Under a VMI policy, the vendor determines the quantity and time delivery of replenishments by accessing customer's inventory and demand data (Darwish & Odah, 2010).

In the recent years, the rate of interest has been increasing in research on VMI to sustain its performance as firms, suppliers, and vendors increasingly find out the interests of more collaboration and integration (Poorbagheri et al., 2014). However, there is some evidence showing that a buyer firm may not necessarily build close relationships with all its suppliers. For instance, Kraljic (1983) classified supply products into four groups: strategic items, bottleneck items, leverage items, and noncritical items. Each classification of supply products may lead to different types of supplier relationships. The model using Kraljic's matrix as presented by Gelderman and Weele (2002) is shown in Figure 2.2.

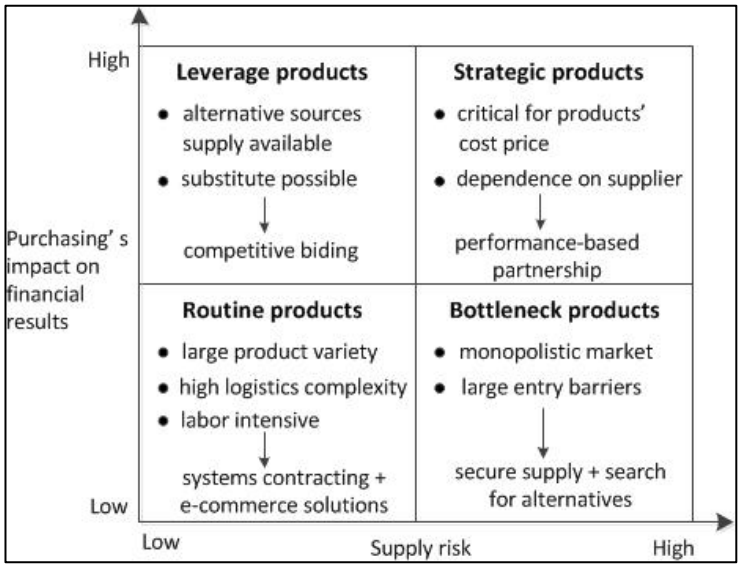

Figure 2.2 : Purchasing product portfolio (Gelderman and Weele, 2002)

According to the Kraljic's matrix, each type of item is defined by different characteristics, and therefore the way to treat the suppliers is also different. For example, the buyer firm must consider establishing close relationships with the suppliers who supply strategic items, while close relationships with the leverage items' suppliers are only optional. Similarly, Olsen and Ellram (1997) proposed the possible strategies for establishing relationships with suppliers based on their attractiveness and difficulty of managing the purchase situation. For instance, it is crucial to establish a collaborative relationship with a supplier especially when the supplier provides a strategically product/service which is very important to the

buyer's business, or its purchase process is complex and difficult to manage, while the buyer firm may reduce the strength of the relationship with a supplier who supplies noncritical or leverage products.

Although using arm's length relationship is considered in some circumstances depending on the importance of supply products, empirical research shows several examples of successful long-term collaborative relationships. It also demonstrates that establishing collaborative relationships is an important approach in several situations for firms to extract competitive edges in the environment of expanding global competition (Sutcliffe & Zaheer, 1998).

Therefore, several papers study factors affecting benefits to be derived from VMI or other cooperative supply chain initiatives and distribution for buyers and suppliers. Coordinating inventory and transportation has been studied by Cetinkaya and Lee, (2000), in which they use an analytical model that order release policy in use with VMI influences the level of inventory required at the vendor, consequently directly affecting a supplier's inventory costs. Yao et al. (2007) improved the analytical model with collaborative initiatives by implementing VMI. They determine how the benefits are likely to be distributed between a buyer and a supplier in supply chain. Therefore, for clarity, the term "distributor" for the customer in VMI relationship and "manufacturer" for the supplier or vendor in the VMI relationship will be used. The VMI has been applied to transportation management with demand uncertainty by using a uniform probability distribution. Moreover, it can reduce holding and routing costs and help balance on-hand inventory costs between the supplier and the retailer.

Demand characteristics can be broadly classified into certain and uncertain. When demand is deterministic, and all its parameters are known, the economic order quantity (EOQ) and its variations are used to solve the problem of replenishment timing and quantities. When uncertainty is significant, it must be explicitly taken into account, and modelling alternatives include one-time decision models, continuous decision models, and periodical decision models. When decisions are taken either continuously or periodically, it is necessary to establish a policy for ordering (Coelho et al., 2016). Therefore, stochastic demand is used in three key parameters to control the inventory policy: when to replenish, how much to replenish, and how often the inventory level is reviewed. This is used to reduce the cost of the optimisation process.

The three policies have been described by Wensing (2011). The first is the order-up-to level (OU) which refers to a system. Here, in each period, the quantity delivered is that to fill the inventory capacity up to. Other policies include (t, s, S) and (t, s, q). In the former, the customer is served if the inventory level is less than *s*. Roldan et al., (2016) reviewed and described the IRP with stochastic demand. They stated that this stochastic demand can be modelled as continuous variables if their average is big enough. However, for small stochastic demand, it is easier to model by using discrete models.

Moreover, in the case where only transportation costs are included, the deliveries must fulfil all inventory levels through the designed route. Dror et al. (1985) and Moshe, Dror and Ball (1987) presented a short-term solution, which was based on the assignment of customers to optimal replenishment periods and on the computation of the expected increase in cost when the customer is visited in another period. Also, Dror et al. (1985) offered the first algorithmic comparison for the IRP with two major simplifications: an OU policy applied and customers are only visited once during the planning period. The OU policy applied on Ball et al. (1987) has been widely used by many researchers.

Lourenco and Riberiro (2003) considered both deterministic and stochastic demand and proposed a multi- period IRP with two types of customers in the model. Firstly, the customer managed inventory (CMI) had deterministic demand and the distributor faces no inventory costs associated with these customers. Secondly, the VMI had stochastic demand and the distributor manages the stock at the VMI. The aim of their work was to determine the best routes for customers, the quantities, and the days to be delivered for the VMI customers over a week (5 days) planning horizon. The model considered a single product in a multi period scenario. The demand of the VMI customers was defined as a continuous random variable. The research proposed an iterated local heuristic to solve the problem. The model was decomposed into two sub problems; inventory and VRP. An initial solution was obtained by solving the inventory problem, and then a good feasible solution was obtained by solving the VRP.

Subsequently, the new quantities and the days to deliver for VMI customers were determined, taking into account the delivery cost calculated from the previous step. The computation study was shown on randomly generated data and it was found that

the reduction on the cost depended on the problem size, proportion of VMI customers and the unit costs chosen for the problem.

2.3. The inventory routing problem

Recently, IRPs have received significant consideration interest from many academic literatures. Moreover, many researchers studied the various versions of the problem as differing in terms of planning horizon, demand characteristics, network topology, inventory policies, and solution methodologies. In which the latest IRP survey is given by Coelho et al. (2013); thirty year of inventory routing after the publication of the first paper by Bell et al. (1983) that simultaneously observed inventories and vehicle routing.

2.3.1. **The classification criteria of IRPs**

To classify IRPs conferring to two structures, firstly the structural variants are presented and the second is related to availability of information on demand (Coelho et al. 2012). In the structural variant, most of the research effort concentrates on the extension of basic versions which are more elegant. Leandro et al. (2012) have classified it according to seven criteria, namely time horizon, structure, routing, inventory policy, inventory decisions, fleet composition and fleet size as shown in Table 2.1. Therefore, this thesis is concerted on all seven criteria as finite time, both demands, one depot to many customers, multiple routing, inventory stock-out is not allowed, homogeneous fleet and a single fleet size.

characteristics	**alternatives**		
Time	Instant	Finite	Infinite
Demand	Deterministic	Stochastic	
Topology	One-to-one	One-to-many	Many-to-many
Routing	Direct	Multiple	Continuous
Inventory	Fixed	Stock-out	Lost sale
Fleet composition	Homogeneous	Heterogeneous	
Fleet Size	Single	Multiple	Unconstrained

Table 2.1 : Classification criteria used for the IRP

2.3.1.1. *Time*

In Table 2.1., time denotes the horizon taken into consideration by the IRP model, which almost use three different modes when classifying the research paper in the time aspect, reflecting the planning periods used;

- Instant horizon situation, this model is so short that at most, one visit per consumer is desired. The main conclusions are balancing between the inventory and transportation cost with the costs related with stock-outs at the consumers. If the planning horizon consists of more than one period, visiting a customer earlier or postponing a visit may also be considered.
- Finite horizon situation where more than one visit at a client may be required. This approach can be further divided into sub modes (short-time, intermediate-time and long-time). In addition, a fixed horizon can be used when there is an expected and finite end to the horizon. Meanwhile, it is assumed that there is no interaction between the time before and after the horizon. If the state after the horizon depends on the decisions made then there is no need to handle the long-term effects, for example, through the inventory levels, the long-term effects must be handled one way or another. The most common way is to use a rolling horizon and solve the problem for a longer period than is actually needed for the immediate decisions.
- Infinite planning horizon refers to a situation where many customers have to make decision of distribution strategies rather than schedules. For an example, see the study of Andersson et al., (2010) where they elucidate that fleet sizing and coordination of route frequencies is usually overlooked.

2.3.1.2. *Demand*

Since inventory management and routing problems have been integrated, practical problems are emphasized rather than theoretical constructs. The demand is considered as one of these IRPs. Then a classification, trying to focus on both stochastic and deterministic realisation, is used in the algorithm. There are two diverse cases of how to predict demand. First, demand is notified by retailers or customers before delivery. Second, demand is unknown at the starting point and it will be informed when the vehicle arrives customers' destination. The uncertain demand case is called stochastic, while the case that demand is assumed and known is called deterministic. As mentioned earlier, it depicts that two types of demand are important in IRPs. Both deterministic and stochastic are studied in this research.

2.3.1.3. *Topology*

To classify the topology problem (Baita, Ukovich & Pesenti, 1998), there are three different modes of transport strategies; one-to-one, one-to-many, and many-to-many. The many-to-one case is also comprised in the one-to-many mode;

- One-to-one mode; this is a situation of direct shipping, the vehicle travelling from node A directly to node B. The delivery time is short and mostly used to transport high value products and deal with emergency cases.
- One-to-many mode; this is an important strategy for road-based inventory routing, where a single facility is used to serve a set of customers by using a fleet of vehicles. The depot is the central distribution where the vehicles start and finale their routes and where the goods are kept before delivered to the customers.
- Many-to-many mode; this is the dominant one in maritime implication. Usually, there is no central distribution, but the vehicles can load and unload at any ports. There is no any fixed positions where the routes start and end, this is called sink location.

2.3.1.4. *Routing*

The routing factor of a combined inventory management and routing problem is classified either as a VRP or as a Pickup Delivery Problem (PDP). In the IRPs, routing categories are classified into three different circumstances. These circumstances depend on the trips which are direct, multiple, and continuous. Firstly, direct refers to a situation that products are distributed completely to each customer from the central warehouse and return to such warehouse. In the multiple, there are more than one customers on a routing visited; this is called "multiple visit". Lastly, the continuous case has no fixed starting point and destination for pick-up and delivery.

2.3.1.5. *Inventory*

The key important role in a combined inventory and routing problems is decision making. There are many different inventory management decisions due to performance of whole distribution. The activities have been influenced by sufficient decision. In this classification, the decision is dependent on demands of the customer. Firstly, the main decision making is whether to allow the inventory level to become negative. In many applications, this case is not allowed since the lowest inventory level is fixed either to zero or based on the safety stock. However, retaining the lowest

inventory level, stock-out, which is insufficient stock, possibly occurs. As a result, the customer demand could fail to satisfy customer requirement and then lost sale could occur. Finally, back order is a resolution step commonly used to solve unsatisfied demand.

2.3.1.6. Vehicle fleet

The vehicle fleet is divided into two types including fleet composition and fleet size. The fleet composition composes of two substitutes, which are homogeneous and heterogeneous. For a homogeneous fleet, each vehicle has the same characteristics, for example, variable cost, fixed cost, size, speed, and equipment. On the contrary, some characteristics of heterogeneous fleet are different. It has different types of vehicle fleet, which are utilized for pick-up and delivery of different kinds of products. The size of vehicle fleet consists of three alternatives, which are single, multiple, and unconstrained. Firstly, the term single is used to describe the fleet comprising only one vehicle. Secondly, multiple refers to the fleet consisting of a number of vehicles and this could be a constraining factor. To illustrate, an example is when in some conditions, if the provider owns the fleet and cannot purchase extra distribution capacity. The extra capacity applications may often hire on short-term basis and where highly specialised vehicles are needed. If it is possible to purchase extra distribution capacity, the planer will always have vehicles enough to organise the products and extra capacity, then unconstrained is set for these situations.

In general, IRPs are long-term dynamic control problems. Those types of problems are very difficult to formulate. In large practical cases, which are common in real world applications, IRPs are almost impossible to achieve optimality, even with accurate data. Therefore, most approaches that deal with real world problems address them in a short-term planning horizon. Additionally, the long-term effects use some approximation, but some of more complex features of IRPs are ignored. Even though previous researches on IRPs shared some common elements, most of them mainly focus on different characteristics. Table 2.2 shows some previous researches that have been studied at some issues, which are classified according to the specific characteristics. The details of reviews are presented in the works of Campbell et al. (1998) (Campbell et al., 1998; Bertazzi, Paletta & Speranza, 2002; Campbell & Savelsbergh, 2004; Archetti et al., 2012 and Leandro et al., 2012a).

Recently, Leandro et al. (2012a) and Bertazzi and Speranza (2013) focused differently from the proposed works by Savelsbergh and Song (2008) and Huang and Lin (2010).

Their work related to the fleet size and vehicle load problems, to the frequency of the deliveries, and to the quantities delivered. Their aims were different. Firstly, their works presented how to solve these problems and introduce the concept of consistency in IRP solutions for increasing quality of service. Another aim was introducing the IRP classification which this case where the demand of multiple customers has been satisfied. Therefore, a formulation of the multi-vehicle IRPs, with and without consistency requirements and the present tutorial was consistence with the work of Bertazzi and Speranza (2012). After that the multi-products and multi-period IRPs were presented by Mirzapour et al. (2014), where multiple capacitated vehicles distribute products from multiple suppliers to a single plant to meet the given demand of each product over a finite planning horizon. The demand associated with each product is assumed to be deterministic and time varying. Consequently, this study extends the research of Archetti et al., 2012 and Coelho and Laporte (2014) in terms of the different demand and methods.

Researchers	Time horizon	Demand	Routing	Structure	Fleet composition	Fleet size	Inventory policy
Bell et al. (1983)	Finite	Deterministic	Multiple	One-to-many	Heterogeneous	Multiple	ML
Federgruen and Zipkin (1984)	Finite	Stochastic	Multiple	One-to-many	Heterogeneous	Multiple	ML
Golden et al (1984)	Finite	Stochastic	Multiple	One-to-many	Heterogeneous	Multiple	ML
Dror and Ball (1987)	Finite	Stochastic	Multiple	One-to-many	Homogeneous	Multiple	ML
Campbell et al. (1988)	Infinite	Stochastic	Multiple	One-to-many	Homogeneous	Unconstrained	ML
Christiansen (1999)	Finite	Deterministic	Multiple	Many-to-many	Heterogeneous	Multiple	ML
Bertazzi et al. (2002)	Finite	Deterministic	Multiple	One-to-many	Homogeneous	Single	OU
Campbell and Savelsbergh (2004)	Finite	Deterministic	Multiple	One-to-many	Homogeneous	Multiple	ML
Kleywegt et al. (2004)	Infinite	Stochastic	Multiple	One-to-many	Homogeneous	Multiple	ML
Aghezzaf et al. (2006)	Infinite	Stochastic	Multiple	One-to-many	Homogeneous	Multiple	ML
Savelsbergh and Song (2007)	Finite	Deterministic	Continuous	Many-to-many	Homogeneous	Unconstrained	ML
Solyali and Sural (2011)	Finite	Deterministic	Continuous	Many-to-many	Homogeneous	Single	OU
Geiger and Sevaux (2011)	Finite	Stochastic	Multiple	One-to-many	Homogeneous	Unconstrained	ML
Liu and Lee (2011)	Finite	Stochastic	Multiple	One-to-many	Homogeneous	Multiple	ML
Archetti et al (2012)	Finite	Deterministic	Multiple	One-to-many	Homogeneous	Single	ML and OU
Solyali et al. (2012)	Finite	Stochastic	Multiple	One-to-many	Homogeneous	Single	ML
Adulyasak et al. (2012)	Finite	Deterministic	Multiple	One-to-many	Both	Multiple	ML and OU
Coelho and Laporte (2012)	Finite	Deterministic	Multiple	One-to-many	Both	Multiple	ML and OU
Coelho et al. (2012)	Finite	Deterministic	Multiple	One-to-many	Homogeneous	Single	ML and OU
Bertazzi et al. (2013)	Finite	Stochastic	Multiple	One-to-many	Homogeneous	Single	OU
Juan et al. (2014)	Finite	Stochastic	Multiple	Many-to-many	Homogeneous	Single	Finite
Coelho and Laporte (2014)	Finite	Stochastic	Multiple	One-to-many	Homogeneous	Single	ML and OU

Table 2.2 : An example of previous research classified according to specific characteristics the IRPs

2.3.2. **Deterministic IRP**

In the IRP with deterministic demand, customer demands are assumed to be known and constant. Chandra (1993) proposed a single depot delivery to customers based upon inventories and vehicle routes in order to meet their non-stationary demand. The demand was assumed to be known for the entire planning horizon. When the retailer orders the products, the supplier has to determine the size of delivery to each customer in every period and the delivery routes. Chandra (1993) developed an integrated model to determine replenishment policies at the retailer's warehouse that is also responsible for devising efficient plans. Both single-depot and multi-depot IRP have been extensively studied over the last 10 years. For example, Savelsbergh and Song (2008) studied the IRP with continuous moves which incorporates two important real-life complexities such as limited product availability at facilities and customers that cannot be served using out-and-back tours. They designed delivery tours spanning several days covering huge geographic areas and involving product pickups at different facilities. Thereafter, Zachariadis et al. (2009) proposed similar problem with a different solution approach. They applied two innovative local search operators for jointly dealing with the inventory and routing aspects of the examined problem, using Tabu Search for further reducing the transportation costs.

Savelsbergh et al. (2009) introduced a problem faced by an Austrian blood bank. The blood bank was faced with a situation in which a set of customer hospitals, clinics and medical institutes require regular deliveries of certain product(blood conserves) which were consumed at different rates. The situation was complicated by the fact that product consumption varies over time and therefore the blood bank wants to minimize its delivery costs. About 250,000 blood products were sold and delivered every year. Delivery routes were planned manually, no routing software or geographic information system was used. The hospitals were grouped into four regions and fixed routes for visiting the hospitals in a region have emerged over time. Hospitals that had requested a delivery of blood products the previous day were visited in the order of these fixed routes.

Guerrero et al. (2013) considered that vehicles might deliver products to more than one retailer per route. In addition, inventory management decisions were included for a multi-depot, multi-retailer system with storage capacity over a discrete time planning horizon. The problems were to determine a set of candidate depots to open, the quantities to ship from suppliers to depots and from depots to retailers per period,

and the sequence in which retailers were replenished by a homogeneous fleet of vehicle. They also proposed a mixed-integer linear programming model and provided bounds on the solutions.

Recently, Cordeau et al. (2014) focused on a multi-product multi-vehicle IRP (MMIRP)under a maximum-level policy. Their work dealt with a multi-period problem in which multiple products were resupplied to a set of customers from a common vendor and deterministic demands occur at the customers for each product. This problem was similar to Laporte et al. (2013), but they assumed that each customer has a maximum inventory level for each product instead of a shared capacity for all products. In order to solve the problem, they decomposed the decisions as planning and routing. Also, they integrated these two parts using a mixed-integer linear programming model.

2.3.3. **Stochastic IRP**

The IRP with stochastic demand has been considered as significant by researchers because it is assumed that the probability distribution of demand is known for the customer. The stochastic IRP differs from the deterministic. For example, IRP with deterministic demand is implemented when demand is known in advance whereas, IRP with stochastic demand is applied when demand is randomed. The stochastic demand IRP with single-depot and multi-retailers was first presented by Barnes and Bassok (1997). The problem is concerned with the distribution of goods, from a warehouse to multi-retailers, where no inventory was kept at the warehouse. The transportation or other lead times from the depot to the retailers were known but the demands were stochastic, stationary and independent with linear inventory costs, backlogged at the retailer over an infinite horizon. For a given day, inventory should be allocated to the retailers, by minimising transportation and inventory costs. They considered a specific policy of direct shipments, i.e. delivery strategies such as the lower bound on the long run average cost per period.

There are many published papers of the IRP with stochastic demand with different characteristics such as time, topology, routing, inventory, fleet composition, and fleet size. For instance, Bard et al. (1998) focused on the customer demand in which demands were unknown with certainty and routing decisions taken over the short run in satellite facilities. A unique aspect of the short-run sub-problem was the presence of satellite facilities where vehicles could be reloaded and customer deliveries continued until the closing time was reached. They developed the Randomised Clarke

and Wright GRASP and modified sweep to solve the problem. Jaillet et al. (2001) studied the problem further using Bard et al. (1998), but the main difference was that customer selection was routed on a specific day of a given week. They introduced a comprehensive decomposition scheme, in which the vehicle routing problem was repeatedly solved over a two-week period. In addition, they proposed incremental cost approximations to be used in a rolling horizon framework for the problem of minimizing the total expected annual delivery costs.

Adelman (2004) proposed a linear programming method to solve IRP with stochastic demand by using the approximation. They obtained two linear programmes by formulating the control problem as a Markov decision process and then replacing the optimal value function with the sum of the single-customer inventory value function. Gaur and Fisher (2004) studied the case study on a supermarket chain in the Netherlands. They analysed a periodic version of the problem which assumes that time varying demand is repeated over a week-long period. They considered the fixed partitioning policies and used a randomised sequential matching algorithm to solve the routing. The inventory was handled by stating a maximum time between deliveries. However, Kleywegt et al. (2004) allowed multiple deliveries per trip and used policies for IRP with stochastic demand by extending the problem proposed by Kleywegt et al. (2002).

The Robust distribution planning for IRP with stochastic demand was proposed by Aghezzaf (2008). They investigated the case where customer demand rates and travel times are stochastic but have constant averages and bounded standard deviations. The approach was proposed to obtain and deploy these robust plans by combining optimization and Monte Carlo simulation. Optimisation was used to determine the robust distribution plan and Monte Carlo simulation was used to fine-tune the plan's critical parameters such as replenishment cycle times and safety stock levels. Huang and Lin (2010) extend the problem developed by Qu et al., (1999). They solved an integrated model that schedules multi-item replenishment with stochastic demand by an ant colony optimisation algorithm to determine which customers to visit based on their degree of urgency, before solving the routing problem heuristically by Clarke and Wright (1964).

Savelsbergh and Song (2008) focused on the inventory routing problem with continuous moves, which incorporates two important real-life complexities: limited product availabilities at facilities and customers that cannot be served using out-and-

back tours. This typical inventory routing problem deals with the repeated distribution of a single product from a single facility with an unlimited supply to a set of customers that can all be reached with out-and-back trips. However, this is not always the reality. They plan delivery tours spanning several days, covering huge geographic areas, and involving product pickups at different facilities. Toptal (2009) presented a replenishment decision model with the consideration of a stepwise freight cost and an all-unit quantity discount. Then, the model was applied to a single-period problem under several scenarios.

Bertazzi et al. (2013), focused on an IRP with stochastic demand with stock-outs allowed. The main differences between their work and the previous references are, firstly, only one vehicle can be used. Secondly, an order-up-to level (OU) policy was used to resupply the customers. Here, a penalty cost was charged and the excess demand was not backlogged whenever the inventory level is negative. Thirdly, a rollout approach was used for solving the IRP with stochastic demand, which was the first approach. Finally, in the rollout scheme, a MIP problem and an effective branch-and-cut algorithm were applied to evaluate the cost-to-go values. The goal was to determine a distribution strategy in order to minimise the expected total cost. Juan et al. (2014) proposed the IRP with stochastic demand and considered a single period which allows stock-out. They used the same algorithm, which was proposed for a different problem, especially that it is considering stochastic demands, stock-outs, and rollout periods. They proposed a Sim-heuristic algorithm, which combined simulation with a randomized heuristic for optimising system performance. The algorithm initially made use of Monte Carlo simulation to estimate the expected inventory costs associated with each RC-policy combination. Next, it employed a fast routing heuristic to compute the total costs, inventory and routing, associated with several refill strategies.

Additionally, Coelho et al. (2014), extended the problem from their previous work by considering a dynamic version of IRP, where customer demands are gradually revealed over time and planning must be made at the beginning of each period. They took advantage of historical information in order to account for stochastic demands. The inventory heuristic policies were used to handle customer's inventories. In their study, lateral transhipments are allowed between customers as a means of avoiding stock-outs. However, is was found that using a longer rolling horizon step did not help to improve the solution, as claimed by Jaillet et al. (2001). Bertazzi eat al. (2015)

focused on the IRP with stochastic demand of the retailers. Whereby, they are satisfied by procurement of transportation services and the supplier, which has a limited production capacity. By assuming that, the size of all deliveries is defined in accordance to the OU policy. They proposed a stochastic dynamic programming formulation of the problem to find an optimal policy in small instances. Their approach integrates a rollout algorithm and an optimal solution of mixed-integer linear programming models, which was able to solve the realistic size problem instances.

2.3.4. **IRP with Transhipment**

The IRP allowing transhipment (IRPT) is advantageous in term of logistics costs and supply chain performance. Transhipment means to transferring goods from customer to another customer. However, in practice, transhipment is less likely to occur when collaborative relationships are not established in the supply chain, although the principle can improve the supply chain performance in theoretical studies. Transhipment can happen as a result because of establishing supply chain relationships among supplier. Sharing information is increasingly allowed in the collaborative association, leading to a better supply chain competitive edge that can produce a win-win outcome from the upstream to downstream. Regarding transhipment, two aspects are always considered when shipping goods between customers is needed; VRPs and IRP. The integration of such two principles is an important ingredient for improving transhipment effectiveness, according to several studie. For example, Kleywegt et al. (2002) and Bertazzi et al. (2002) presented that IRPT is taken into account in some research when a supplier has to deliver goods to a large number of customers.

The IRPT concept was introduced that products may be shipped from either the supplier or customer to the requested location. This happens, for example, between stores belonging to the same chain, which can ship merchandise to one another when unforeseen demand variations occur. However, Transhipments may be beneficial in a deterministic context in which no shortages occur because they may yield an overall reduced distribution and inventory holding cost. Mercer et al. (1996) provide an example of an inventory routing system used by the supermarket chain Tesco, in the United Kingdom, where deliveries are made from a factory to several warehouses, and lateral transhipments can take place between warehouses. Similarly, transhipment has been incorporated into VRP (VRPT).

Yang and Xiao, (2007) described that the transhipment centre plays an important role in connecting suppliers and customers. However, it depends on the retailers who are collaborating with the supplier. For example, the retailer who is collaborating with the supplier to take on the task of sending goods to the other retailers is known as a transhipment point. Before handing over the task to a transhipment point, the supplier ensures that the inventory at the transhipment point is enough to accommodate demand. Other than that, transhipment is allowed when the location of transhipment point with the retailer are nearer compared to the location of the supply. This can shorten the travel distance and at the same time, it can save transportation cost.Shen et al. (2011) studied the IRP in crude oil transportation. They approached a Lagrangian relaxation for a multi-mode inventory routing problem with multiple transhipment ports. They considered a heterogeneous fleet of tankers and a pipeline and multiple types of routes, although, both inventory level and shortage level at each customer harbour were limited. As their problem was a mixed integer-programming problem, a Lagrangian relaxation approach was developed for finding a near optimal solution of the problem. Their approach showed it outperformed an existing meta-heuristic algorithm, especially for large instances.

Jemai et al. (2012) proposed the logistics value chain by integration and coordination of the inventory management and vehicle routing decisions. They extended the works of Zavanella et al. (2009) and Jarugumilli and Grasman (2006), who studied a dynamic IRP model, by adding a VR (vehicle routing) to their proposed inventory problem under a stochastic demand setting. They considered the inventory routing framework in a supplier integration context and showed that the transhipment brings more benefits than the classical context. Furthermore, the transhipment permits to better optimise the distribution tours and to improve the global performance of the supply network. Ahmad et al. (2014) studied an integration of three key logistics decision problems; location, vehicle routing and inventory management. They proposed the Location Routing Inventory Problem with Transhipment as a transhipment point in the logistics system. Selection of the transhipment point was done using p-median. They showed that LRIPT gave satisfactory results compared with a LRIP.

The transhipments applied with the IRP was introduced by Coelho et al. (2012), which allowed transhipments either from the supplier to the customers or between customers. They proposed an adaptive large neighbourhood search heuristic to solve the problem. Mirzapour et al. (2013) considered a transhipment as a possible solution

to increase the performance of the supply chain. They presented multi- products and multi-period IRP where multiple capacitated vehicles distribute products from multiple suppliers to a single plant to meet the given demand of each product over a finite planning horizon.

2.4. Solution methods

This research presents a review of relevant literature on IRP and IRPT solved by using different OR solution methods. Since the first work published in 1980, more than one hundred papers have studied the classical version of the IRP and its variants. Some of which were inspired from real life applications. This complex problem leads to difficulty in finding solutions, and therefore approximate methods are the preferred tools for solving these problems.

Due to the complexity of these problems, this thesis proposes the IRP and IRPT that deals with direct deliveries from the supplier and transhipments between customers themselves in conjunction with multi-customer routes in order to increase the flexibility of the system. This problem is similar to Leandro et al. (2012), which introduced transhipments within the IRP framework. They have included planned transhipment decisions within a deterministic framework as a way of reducing distribution costs. However, the major difference is that the methods proposed in the research includes minimising not only transhipment cost but the the total costs consisting of stockout penality costs and holding costs.

2.4.1. Exact methods

Very little research in the IRP proposed exact algorithm for solving IRP of reasonable size. Talbi (2009) stated that "Exact methods obtain optimal solutions and guarantee their optimality". This type of method is often applied to small size instances. In exact methods, one can find the following classical algorithms: dynamic programming, branch and X family of algorithms, branch and bound, branch and cut, branch and price) developed in the operations research community). There are a few papers published on exact method to solve IRPs because this method is unable to solve complex problem and large instances. They require large amount of computational time to solve.

Archetti and Laporte, (2007) focused on the single vehicle case solved exactly by using branch and cut algorithm. They proposed branch and cut algorithm to solve the deterministic order up to level inventory routing problem which introduced by

Bertazzi et al. (2002). The authors presented a problem where a product is shipped from a common supplier to several retailers. The stock is depleted in a deterministic and time-varying way over a given time horizon. Each retailer can be visited several times over the planning horizon but for our work, the supplier can visit to retailer only once. The product is shipped from the supplier to the retailers by a capacitated vehicle.Also, shipments can only be performed at the discrete time instants within the planning horizon, for example, a day. The problem is to determine a shipping policy minimising the sum of transportation and inventory costs both at the supplier and at the retailers in such a way that no stock-out occurs.

Archetti and Laporte, (2007) also, introduced a mixed integer linear programming model for solving the problem, derive new additional valid inequalities to strengthen the linear relaxation of the model and present an exact branch-and-cut algorithm. The algorithm which they proposed was the first exact approach for IRP and they presented a set of valid inequalities which exploit the problem structure. Some papers presented exact methods used to solve the IRP such as a branch and cut algorithm using a robust formulation and an a-priori-tour-based heuristic for the IRP. These problems addressed by Solyali et al., (2011) where a supplier receives a given amount of a single product in each period and distributes it to multiple retailers over a finite time horizon by using a capacitated vehicle. External dynamic demand is faced by each retailer, which is controlled by a deterministic order up to level policy requiring that the supplier raises the retailer's inventory level to a predetermined maximum in each replacement. The problem occurs when a decision is required on when and in what sequence to visit the retailers, such that system wide inventory holding and routing costs are minimized. They proposed a branch and cut algorithm and a heuristic based on an a-priori tour using a robust formulation and also showed the fact that this method generates good quality solutions.

Coelho et al. (2012) proposed a branch and cut algorithm for the exact solution of several classes of IRPs. Specifically, they solved the multi-vehicle IRP with a homogeneous and a heterogeneous fleet, the IRP with transhipment options, and the IRP with six added consistency features. These features can be represented through extra constraints or as penalties in the objective function.

Laporte et al. (2013) also presented exact methods to solve the multi-product multi vehicle IRP (MMIRP). In their work, the supplier is responsible for the distribution of several products to a set of geographically dispersed customers using a fleet of

vehicles. Customers and vehicles have a maximum inventory capacity which is shared by all products. The problem deals with a deterministic version of the problem in which the supplier has full knowledge of future demands, such that no stock out at the customers is allowed to occur. They also proposed an exact algorithm for this problem and this method able to solve this problem. After that, Adulyasak et al. (2014) have extended the formulation from Coelho et al. (2012) under the OU and ML policies to account for the several classes of MIRPs. They also used a branch and cut algorithm to solve the problem by assuming the transportation cost matrix is symmetric. They proposed undirected model in order to reduce the number of variables. A branch and cut able to be solved their formulation by making use of the capabilities of modern MIP solver.

2.4.2. **Approximate methods**

According to Talbi (2009), approximate methods can be divided into two subclasses of algorithms namely: heuristics and meta-heuristics algorithms. Heuristics usually find reasonably good solutions in a reasonable time and they are tailored to solve specific problems and/or instance. while meta-heuristics are general purpose algorithms that can be applied to solve almost any optimisation problem. Beside that, approximation algorithms provide provable solution quality and provable run time bounds. This thesis will discuss these two algorithms: heuristics and meta-heuristics below:

2.4.2.1. Heuristics

Heuristics were first introduced by Bell et al. (1983), to solve the IRP with different variations either based on sub gradient optimization of a Lagrangian relaxation or constructive and improvement heuristics. Reeves (1996) defined a heuristic as a rule of thumb based on domain knowledge from a particular application that gives guidance in the solution of problem. Heuristics may thus be very valuable most of the time but their performance cannot be guaranteed. Though, a heuristic is a method which seeks good solutions at reasonable computation time without being able to guarantee either feasibility or optimality, or even in many cases to state how close to optimality a particular heuristic solution.

Jaillet et al. (2002) considered IRPs with stationary demands and solved the problem with constant punishment interval. The uncertainty of future demand is a significant variant of problem in the stochastic IRP or SIRP (Campbell et al., (1998); Kleywegt et al., (2002); Adelman (2004); Savelsbergh et al., (2004) and Hvattum et al., (2009)).

Federgruen and Zipkin (1983) iteratively applied interchange heuristics for constructing a better set of traveling salesman tour and an optimization procedure for improving the inventory allocation. The algorithm proposed by Federgruen and Zipkin terminates when no more improvement in the total inventory and routing costs is possible. Chien et al. (1989) formulated the integrated problem as a mixed integer programme and develop a Lagrangian based procedure to generate both good upper bounds and heuristic solutions. Computational results show that the procedure was able to generate solutions with small gaps between the upper and lower bounds for a wide range of cost structures.

Several heuristics have been devised for VRPs and applied with other methods to solve IRPs, only some of which are sufficiently well known to be truly viewed as classical. The Clark and Wright Savings (CWS) algorithm is an example classical heuristic.

The CWS algorithm was developed by Clarke and Wright (1964) and it is applied to problems for which the number of vehicles is not fixed (it is a decision variable), and it works equally well for both directed and undirected problems. The CWS algorithm is the one of three best known heuristics concentrated and improved by Gaskell (1967), Yellow (1970) and Cordeau et al. (2002). They found that it usually results in a sharp deterioration in solution quality which can be explained by the fact that the algorithm is based on a greedy principle and contains no mechanism to undo early unsatisfactory route merges. Besides that, it has at least the distinct advantage of being very quick and simple to implement (Cordeau et al., 2002) and also, it can provide good quality solutions for small size instances. However, large instances calculate the savings considering large savings values which affect the solution. Hence, combining domain reduction with the CWS algorithm is far better approach than using it alone to solve large instances (Tu et al., 2013 and Straka & Besta, 2015).

There are a few papers published in which CWS is in the stand-alone algorithm which practically it applied and combine with another algorithm to solve IRPs. In Bard et al. (1998), they studied the IRP with satellite facilities (depots geographically scattered throughout the service area). Interestingly, the authors use a randomized version of the classical CWS heuristic (Clarke and Wright, 1964) to solve routing instances with up to 500 nodes in about two hours. Juan et al. (2014) proposed an integrated VRP with several inventory problems with stochastic demands. The aim of their work to find the personalized refill policies and associated routing plan that minimize, at each

single period, the expected total costs of the system, i.e., the sum of inventory and routing costs, where the routing costs belonging to each of these strategies are computed using CWS heuristic could find a good solution.

Nevertheless, heuristics for solving IRP is increasing in number of problems along with more attention in the literatures. For example, Bertazzi et al. (2002) presented a heuristic to determine the vehicle route at each discrete time point and an order up to inventory policy is also adopted. They considered a deterministic model of the inventory and routing problem with a single capacitated vehicle over long term period. Each customer has a specified minimum and maximum inventory level. Various objective functions from different levels of decision makers are considered and numerical computations on a set of randomly generated problem instances have been conducted

For this research, we focus mainly on these solution construction huesristics: Clarke and Wright and sweep algorithm. Furthermore, we implement some modification to the CWS heuristic by way of randomisation and apply some solution improvement methods mainly: IG algorithm and a local search algorithm. These were selected partly because of their popularity and also because they operate on vastly different principles.

2.4.2.2. Meta-heuristics

Meta-Heuristics are designed to find a good feasible solution. These methods have become one of the most practical approaches to solving hard combinatorial problems in the real world. They also provide acceptable solutions to complex problems in a reasonable time. Further, these methods provide a robust optimisation strategy that is able to solve large and complex problems. Although, there is no guarantee to seek the optimal solution, but most of the time a near-optimal solution is often found and these methods could prevent getting stuck in local optima (Talbi, 2009). In comparison to classical heuristics, meta-heuristics perform a much more thorough search of the solution space, allowing inferior and sometime infeasible moves, as well as recombination of solution to create new ones. Meta-heuristics consists a numbers of solution approaches such as genetic algorithms, simulated annealing, ant colony optimisation and tabu search.

Qin et al. (2014) solved the inventory problem by proposing a local search method which is achieved by four operators on delivery quantity and retailer's demand. Also, they proposed a tabu search method to solve the routing problem. The authors

considered a periodic IRP in which once the delivery time, quantity and routing are determined, they remained the same in the following periods. The problem was modelled concisely and then decomposed into two sub problems: inventory problem and routing problem. The computational results show that the method is efficient.

Abdelmaguid et al. (2009) proposed CWS and neighbourhood algorithms to solve the IRP with backlogging in which it considers a multi-period inventory holding. They used saving algorithm to solve the associated VRP and neighbourhood algorithm is used to improve CWS heuristic. The problems also permits back logging and the vehicle routing decisions are made for a set of customers who receive units of a single item from a depot with infinite supply. They considered a dynamic routing and inventory problem (DRAI) that addresses the integrated inventory and vehicle routing decisions in the time domain at the operational planning level. Their problem, referred to as the IRP with backlogging (IRPB), considers multiple planning periods, both inventory and transportation costs, and a situation in which backorders are permitted. Then, they proposed to develop constructive and improvement heuristics by using neighbourhood local search to obtain an approximate solution for this NP-hard problem. They have shown the potential benefit of their method for solving larger problem sizes, where they also presented experimental results.

Recently, some of published papers considered the dynamic and stochastic IRP. Several algorithms have been proposed for the infinite horizon case in which one assumes that demand distributions are stationary and the objective is to minimise the discounted cost of delivery at each time step such as the work by Laporte et al. (2008). They proposed to solve a stochastic inventory routing problem. where stochastic demands are specified through general discrete distributions. Heuristics based on finite scenario trees are developed and computational results confirmed the efficiency of these heuristics.

Huang and Lin (2010) proposed the development of a modified ant colony optimisation (ACO) algorithm to solve the IRP with stochastic demand. They considered the multi-item replenishment problem to minimise the total cost and stockout costs indicated by the attraction of pheromone values on nodes. They investigated the performance of an evolutionary computing approach based on ACO algorithm and included an extensive comparison of the conventional ACO. The results of their simulations and optimisations revealed that the modified ACO

algorithm achieves highly significant improvements compared to the conventional ACO.

Moin et al. (2011) and Sofianopoulou (2014) both proposed GA to solve a multi period IRP. Moin et al. (2011) studied a many-to-one distribution network consisting of an assembly plant and many distinct suppliers where each supply a distinct product. They considered a finite horizon, multi periods, multi suppliers and multi products. A fleet of capacitated homogeneous vehicles housed at a depot, is available to transport products from the suppliers to meet the demand specified by the assembly plant in each period. They used GA to generate the inventory routing problem on a finite horizon, multi period and multi product problem, which a new set of crossover and mutation operators are presented. Sofianopoulou (2014) studied similar problem and they proposed a GA based heuristic which they opined that it can also provide interesting insights for solving other problems, especially in an outbound logistics where the demand pattern from one period to the other changes significantly. The best solutions for all their algorithms were found in significantly low cpu times.

Coelho et al. (2012b), avoided stock outs when demand is high by using transhipments within a dynamic and stochastic IRP framework as a means of mitigating stock outs when demand exceeds the available inventory by using an adaptive large neighbourhood search (ALNS) heuristic to solve this problem. Their heuristic manipulates vehicle routes while the remaining problem of determining delivery quantities and transhipment moves is solved through a network flow algorithm. Their approach can solve four different variants of the problem: the IRP and the IRPT, under maximum level and order up to level policies and this algorithm was able to find good quality solutions in reasonable cpu times compared to lower bound and upper bound on most of all the instances tested. Moreover, they have also applied their algorithm to the traditional IRP (without transhipment) by setting the transhipment cost sufficiently large ($b_{ij} = c_{ij}$) so as to avoid the use of transhipment in the final solution.

Hassanvand and Salehsohrabi (2013) proposed GA to improve the initial solution through inventory routing improvement procedure and pricing improvement procedure. They mainly concentrated on the area of operational issues of a two-echelon supply chain under linear demand function for each buyer. This is to find out the optimal price and the optimal transportation quantity, for each buyer from several suppliers. Further, Razavi and Nik (2013) presented the multi depot inventory routing problems allowing order backlogs, which considered the problem of new hypotheses

to come closer to the real conditions(allowing orders to be backlogged). The goal of the IRP is finding the appropriate balance between inventory costs (holding and shortage) and shipping costs. They proposed a parallel genetic algorithm and solved the IRPB solved with two cases: single depot IRPB and multi depot IRPB. In the proposed method, the problem is searched at the macro level by a parallel GA. Their algorithm computational results in the single depot compare with the lower bound obtained by the multiple- depot solution of Abdelmaguid et al. (2009) showed that the efficiency of parallel GA was suitable.

Bertazzi et al. (2013) focused on an IRP with stochastic demand, where stock-outs may occur during the time horizon. These situations can be observed, for instance, in the supermarket industry when the consumption of a specific product is quite high so that the regular resupplying policy is not able to satisfy all the customer requirements in the same period during the time horizon. This is especially true when the demands are stochastic and the vehicle capacity is quite limited with respect to the volume required at the retailers. Each retailer defines a maximum inventory level. An order up to level policy is applied, i.e. the quantity sent to each retailer is such that its inventory level reaches the maximum level whenever the retailer is served. Also, they provided a dynamic programming formulation of the problem that allows to design a hybrid rollout algorithm aimed at finding good quality solutions. Furthermore, rollout algorithms are a class of heuristics that can be used to solve deterministic and stochastic dynamic programming problems. The computational results showed that these algorithms were able to solve instances with a realistic number of retailers in a few minutes, providing significantly better solutions than the ones obtained by a benchmark algorithm.

Moreover, Sofianopoulou (2014) studied on a multi-period IRP where a vendor serves multiple geographically dispersed customers who receive units of a single product from a depot, with adequate supply, using a capacitated vehicle. This problem is the inventory routing problem with backorders (IRPB) and involves determining inventory levels when backorders are allowed. In order to minimise the total cost (comprising of holding cost, transportation and backorder penalty cost) for the planning period, while ensuring that inventory level capacity constraints are not violated. This author implemented GA with suitably designed genetic operators in order to obtain near optimal solutions. the computational results presented demonstrated the effectiveness of the proposed procedure. Papageorgiou et al. (2014)

proposed the ACO approach for solving IRP similar to the one proposed in Moin et al. (2011). They considered a many to one, part supply network where the retailer's demands are assumed to be known for all periods. The IRP problem was first modelled as an equivalent vehicle routing problem. Then, they used an ant-based algorithm that combines elements from some of the most successful ACO variants, namely (Elitist) Ant System (E-AS) and Max-Min Ant System (MMAS), to solve the corresponding problem. For this reason, the solution gaps between the algorithm and CPLEX solutions were kept in reasonably low values, while offering perspective for further improvement by proper parameter tuning. The results obtained were assessed with respect to the optimal solutions found by established linear solvers such as CPLEX.

In addition, Li et al**. (**2014**)** studied an IRP in a large petroleum and petrochemical enterprise group**.** The objective was to find the minimise the largest route travel time**.** They presented a tabu search algorithm to tackle the problem. The method builds an efficient and effective procedure to improve the search quality at each iteration**.** It is also capable of providing near optimal, close**-**to**-**lower**-**bound solutions in a computational timely and effective manner.

Cordeau et al. (2014) studied the dynamic version of the IRP and different policies to handle the DSIRP. They proposed a method that can make use of historical data to forecast an approximation of future unknown demand. The transhipment is allowed to avoid stock-out when demand is high. Therefore, the objective function is implemented for solving the DSIRP in order to minimise the total inventory, shortage, routing and transhipment costs over the planning horizon. An implementation of the adaptive large neighbourhood search (ALNS) algorithm is proposed for solving the problem. Then, the policies are compared by performing an extensive computational analysis on randomly generated instances. They have successfully solved the dynamic and stochastic version of the IRP under different policies. However, they considered the inclusion of consistency features in the solutions of the DSIRP, so their experiments show that ensuring consistent solutions over time under a dynamic and stochastic environment is much more expensive than under a deterministic setting.

Mirzaei and Seifi (2015) presented a mathematical model for IRP to allocate the stock of perishable goods. The proposed model balances the transportation cost, the cost of inventory holding and lost sales. The model is solved to optimality for small instances and is used to obtain lower bounds for larger instances. They have devised an efficient

meta-heuristic algorithm to find good solutions for this class of problems based on simulated annealing (SA) and tabu search (TS). The algorithm can find good solutions in a reasonable time with a maximum average optimality gap for the objectives.

2.4.2.3. Simulation optimisation-Heuristic/Meta-heuristic

This subsection presents a review of relevant literature on IRPs solved by using sim-heuristic/meta-heuristic methods. After more than 20 years, the simulation-based optimization field is still a promising research area. Several studies have been done in this area for different purposes (Glover & Kelly, 1996;1999). Numerous studies have combined simulation and optimization approaches to find resolution techniques for complex real problems. Cai (2010) studied on chanel coordination on the supply chain process. The highlights of his work was the sensitive influence of stochastic demands from a supplier and retailer perspective. proposed a decision support tool based on simulation-optimisation. With a simulation optimisation techniques and modelling can help deal with more realistic and complex scenarios. A simulation-optimization technique is used to solve some inventory routing problems.

The potential of simulation techniques has been widely proven (Carson & Maria, 1997). Actually, the stochastic behaviour in real systems used to be addressed using simulation (Glover et al., 2001). A stochastic system is a set of dynamic interdependent components where some values of its variables change randomly. As real systems require to be optimised in order to provide better solution quality. Therefore, simulation-based optimisation is a research emerging field that involves the integration of optimization techniques and simulation analysis (Deng, 2007).The simulation processes with stochastic variables are related to the basic mechanisms **(**Grasman et al**.**, 2014**).** The stochastic problem includes the following steps; firstly, the random behaviour selection of a specific variable, which can follow a uniform or non-uniform distribution**.** Secondly, the probability distribution is defined; several parameters need to be settled such as the expected value and the standard deviation**.**

Sajadifar et al. (2012) used inventory models with deteriorating items, stochastic lead times, and Poisson demands. Their work allowed shortages and backorders. They proposed three stochastic parameters in their simulation model; item lifetime, demands, and lead-time. Whereas, their objective is minimising the long run total expected cost. Some other examples of the application of simulation based optimization can be found in scheduling (Ouelhadj et al., 2009), and the city logistics

(Barcelo, Grzybowska & Pardo, 2007; Taniguchi, Thompson & Yamada, 2012 and Montoya & Herazo, 2014).

Generally, sim-heuristic is a particular case of Simulation-based optimisation (Sim-Opt), which combines a heuristic/meta-heuristic algorithm with simulation methodologies as shown in Figure 2.3. An example of Sim-heuristics applications proposed by Juan et al. (2012), is a combination Monte-Carlo simulation (MCS) and metaheuristic for solving the IRP with stock-out and stochastic demand. This method is an iterative process which aims to find feasible solutions. Obviously, this approach does not guarantee optimality.

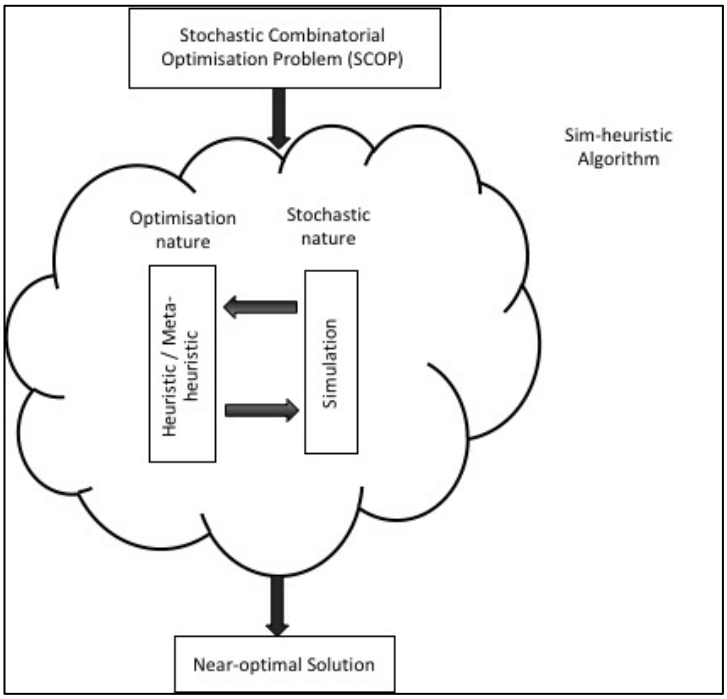

Figure 2.3 : Overview scheme of Sim-Opt with heuristc/meta-heuristc approach (Juan et al., 2014)

Sim-heuristic principles can be applied in simple routing problems where the random values are integrated at the end of the optimisation process. For example, the routing costs are initially defined by using a randomise CWS algorithm. Then a simulation of random demand is performed under some specific conditions (Juan et al., 2011). Bear in mind that this simulation can affect the previous results. Therefore, the idea is to define how these routing costs have changed under certain conditions. The creation of this relationship depends on the studied problem and the proposed algorithm. Earlier work has shown that this methodology can be easily applied in many research areas such as the inventory routing problem (IRP) and logistics (Glover & Kelly, 1996; Carson & Maria, 1997; Arisha, 2010). For instance, the IRP with Stochastic Fuel Delivery Problem (Popovic, Bjelic & Radivojevic, 2011) and the Stochastic Inventory

Routing Problem with Stock-out (Grasman, et al., 2014) are a family of well-known delivery or routing problems characterized by the randomness of at least one of their parameters or structural variables. This uncertainty is usually modelled by means of suitable random variables, which in most cases, are assumed to be independent.

2.5. Chapter Conclusion

In this chapter, the basic overview of the VMI, Inventory replenishment policies and relationship between supplier and customer are presented. The definitions and classification of IRPs are illustrated. The problems are usually classified according to their practical characteristics and are defined in terms of decision variables, constraints, objectives and cost factors. and also on the proposed approach to the solution. With regard to demand, it can either be deterministic or stochastic which will be taken into account in this thesis. Even Though, most of the existing literature has considered the IRP as a long-term, multi-period problem with random demands and a high variability, these are inherent in the real business. This makes it difficult to forecast future inventory levels and therefore a real-life IRP needs a particular method to find its solution.

A review of the solution methods,which have been proposed to solve the IRP and IRPT with deterministic and stochastic demand are also presented. These heuristic, meta-heuristic and sim-heuristic methods for IRPs are widely discussed in this chapter. Knowledge derived from the literature reviewed and the performance of the studied works will be implemented as direct extension methodologies and proposed for solving the presented problems in this thesis will be detailed in next chapter.

Chapter 3: Clark and Wright Saving (CWS) heuristic for deterministic IRP and IRPT

3.1. Introduction

The inventory routing problems (IRPs) have received a lot of attention in the operational research community because it integrates two components of supply chain management: inventory control and vehicle routing. IRPs are complex logistic problems in which these two components are applied into an incorporated framework. Traditionally, these logistics problems have been dealt with separately, but their integration can have an overwhelming impact on overall system performance. IRP and VRP have a very distinct description. The VRP is in place when customers make orders, and the supplier consigns the orders by the means of vehicles to the customers on a daily basis. With respect to IRP, instead of customers, the supplier makes a decision regarding the item quantity to be dispatched for the customers each day. By doing this, the restriction is that the customers' inventories are not allowed to run out; otherwise the supplier will have a penalty. Another difference is the planning horizon. VRPs typically deal with demand, customer request, or travel time within a single day; it is required that all orders have to be delivered by the end of the day. On the other hand, the supplier makes day-to-day decisions about which customers to visit and how many items to deliver to each customer, while also bearing in mind that the decisions made each day may affect the future logistics management. In the case where only transportation costs are included, the deliveries must fulfil all inventory levels through the designed route. Dror et al. (1985) and Moshe, Dror and Ball (1987) presented a short-term solution, which was based on the assignment of customers to optimal replenishment periods, and on the calculation of the expected increase in cost when the customer is visited in another period. Also, Dror et al. (1985) offered the first comparison for the IRP with two major simplifications: an OU policy was applied and, customers are only visited once during the planning period. The OU policy applied in Ball et al. (1987) has been widely used by many researchers. So, the IRP involves trading off between depot and customer inventory holding costs, and vehicle routing cost.

The IRP in this research is conducted under two policies: order-up-to (OU) and maximum level (ML) replenishment policy. With respect to the OU policy, the quantity delivered to the customer fills its specific inventory level, while the ML policy describes that the supplier decides the level of quantity to fulfil the customer inventory capacity. Consequently, the resulting problems are solved by determining the customer demands to be delivered, the vehicles and delivery routes to be used.

Later, the IRPT concept was introduced detailing that products may be shipped from either the supplier or customer, to the requested location. This happens, for example, between stores belonging to the same chain that can ship merchandise to one another when unforeseen demand variations occur. Moreover, transhipments may be beneficial in a deterministic context in which no shortages occur, because they may yield an overall reduced distribution and inventory holding cost. Therefore, this chapter considers deterministic demand of the IRP without and with transhipment (IRPT), where, only one vehicle leaves and returns to a depot after serving a set of customers. The reason for using a single vehicle is that its results will be able to benchmark with the previous study. In this instance, transhipment is allowed when customer demand is unexpectedly requested. Also, this problem includes direct deliveries from the supplier and transhipments between the customer networks in conjunction with multi-customer routes, in order to increase the flexibility of the system.

The chapter is structured as follows: section 3.2 presents this chapter contributions, and the IRP and IRPT modelling is proposed in section 3.3. Section 3.4 presents the proposed approach to solve IRP and IRPT with deterministic demand. All the computational experiments of this chapter are shown in section 3.5 and the conclusion of this chapter is given in section 3.6.

3.2. Contribution

When reviewing literature, the most updated publications on the IRP, were mainly variations of models designed for the VRP which were extended to take inventory costs into consideration. However, there are only a few studies that used the lateral transhipment as an alternative policy to ensure the customer does not face stock-out and reduced the total costs of IRP under OU/ML policies (IRP_OU and IRP_ML). These is one contribution of our knowledge study from literatures the IRPT deals with direct deliveries from the supplier and transhipments between customers themselves, in conjunction with multi-customer routes, in order to increase the flexibility of the

system. In this thesis, lateral transhipments are allowed to take place when the inventory level of the customer has become negative. This problem is similar to Leandro et al. (2012) where they introduced transhipments within the IRP framework. They also included planned transhipment decisions within a deterministic framework as a way of reducing distribution costs. Nonetheless, the major difference is that the model proposed in the research includes minimising the total costs (Holding Cost and Transportation Cost).

Several VRPs solution methods have been extended to the IRPs. Of such methods are the sweep algorithm, CWS heuristic and some neighbourhood search. However, the CWS algorithm is one of the best known, and has been applied to solve vehicle routing problems. There are a few studies published in which CWS is the stand-alone algorithm which has been applied and in some published works, it has been combined with other algorithms to solve IRPs. The contributions of this research are: (i) propose a novel idea employing the use of heuristic methods, such as classical CWS to solve the deterministic demand of the IRP and IRPT under a deterministic environment with OU/ML replenishment policies (IRP_OU, IRP_ML, IRPT_OU and IRPT_ML). (ii) Present an optimisation model with the objective of minimising the total cost consisting of (holding cost and transportation cost). (iii) Propose a classical CWS heuristics to handle the proposed model. The reason behind the choice of the CWS heuristic as the solution method is due to its efficiency and simplicity of application.

3.3. IRP and IRPT optimisation models

The IRP and IRPT are mathematical models formulated for minimising the total costs. The proposed optimisation models are defined as a graph $G = (V, E)$ where $V = \{0, \dots, n\}$ is the vertex set and $E\{(i,j): i,j \in V, i \neq j\}$ is the arc set. Vertex 0 is the depot where the supplier is located and the vertices of $V' = V|\{0\}$ represent customers. Both the supplier and customers incur unit inventory holding costs h_i per period ($i \in v$) and each customer has an inventory holding capacity C_i. The duration of the planning horizon is p and, at each time period $t \in \mathrm{T} = \{1, \dots, p\}$. The problem is defined by the quantity of product made available at the depot is z^t. Also, we assume that the quantity z^t becoming available at the supplier in period t can be used deliveries to customers in the same period and that the quantity q_i^t received by customer i in time period t.

This is assuming that the inventory level of the customer at the end of a period cannot exceed the maximum available inventory capacity. Also, the supplier has enough inventories to meet all the demand during the planning horizon and inventories are not allowed to be negative. From the beginning of the planning horizon, the decision maker knows the current inventory level of the depot I_0^0 and of the customers I_i^0 and receives information on the demand $d_i^t$of each customer i for each time period t (Leandro et al., 2012).

This case considers a vehicle of capacity Q. This vehicle can perform one route to deliver products from the supplier to a subset of customers in each time period, and c_{ij} is the routing cost which is associated with arc $(i, j) \in E$. The binary variable: x_{ij}^t is equal to 1 and only if customer j immediately follows customer $i\,on$ the route of supplier's vehicle in period t. Let I_i indicate the inventory level at vertex $i \in V$ at the end of period. Finally, q_i^t is the quantity delivered from the supplier to customer i in time period t and is able to be used to meet the demand in that period.

Firstly, we will describe the products delivered by the vehicle from the depot to the customer according to the OU policy or the ML policy. Therefore, the total quantity delivered to a customer in a given period, guarantees that no shortages occur and that the capacity is not exceeded at the end of the period. However, the customer's capacity may be temporarily exceeded during that period. This case also assumes that all orders and deliveries can be performed during the same time period, which means that lead times are insignificant (Archetti et al., 2007; Leandro et al., 2012 and Coelho et al., 2013). The objective of the problem is to minimise the total cost while meeting the demand for each customer. The OU or ML replenishment policies plan is used under the following assumptions:

- The inventory level for each customer at the end of a period cannot exceed the maximum available inventory capacity.
- The inventories are not allowed to be negative, which means that all demand must be met by previous inventory plus deliveries performed during the time period.
- The OU/ML replenishment policies applies when the supplier decides a vehicle needs to deliver to a customer in each time period.

- The vehicle must start a trip from the depot and end that trip at the depot
- The vehicle from supplier can perform only one route in each time period.
- The vehicle has a limited capacity which cannot be exceeded.

In the next section, mathematical formulation for IRP and IRPT are provided.

3.3.1. **The IRP optimisation model**

The IRP optimisation model here is based on the work of Coelho et al. (2012). The objective function to minimise the total cost is as follows:

$$Minimize \sum_{t\in T} h_0 I_0^t + \sum_{i\in V'}\sum_{t\in T} h_i I_i^t + \sum_{i\in V}\sum_{j\in V, i<j}\sum_{t\in T} c_{ij} x_{ij}^t \quad (1)$$

Subject to the following constraints:

$$I_0^t = I_0^{t-1} + z^t - \sum_{i\in V'} q_i^t, \quad t \in T \quad (2)$$

$$I_0^t \geq 0, \quad t \in T \quad (3)$$

$$I_i^t = I_i^{t-1} + q_i^t - d_i^t, \quad i \in V', \quad t \in T \quad (4)$$

$$I_i^t \geq 0, \quad i \in V', \quad t \in T \quad (5)$$

$$q_i^t \geq C_i \sum_{j\in V'} x_{ij}^t - I_i^{t-1}, \quad i \in V', \quad t \in T \quad (6)$$

$$q_i^t \geq C_i - I_i^{t-1}, \quad i \in V', \quad t \in T \quad (7)$$

$$q_i^t \leq C_i \sum_{j\in V} x_{ij}^t, \quad i \in V', \quad t \in T \quad (8)$$

$$\sum_{i\in V'} q_i^t \leq \mathrm{Q}, \quad t \in T \quad (9)$$

$$\sum_{i\in V} x_{ij}^t = \sum_{i\in V} x_{ji}^t, \quad j \in V, \quad t \in T \quad (10)$$

$$\sum_{i\in V} x_{i0}^t \leq 1, \quad t \in T \quad (11)$$

$$x_{ij}^t \in \{0,1\} \quad q_i \geq 0, i, j \in V, i \neq j, \quad t \in T \quad (12)$$

- Constraints (2) is the inventory level at the supplier (I_0) at the end of period t by its inventory level (I_0^{t-1}) at the end of period $t-1$, plus the quantity of product z^t made available in period t, minus the total quantity of product shipped to the customers by supplier's vehicles in period t.
- Constraints (3) avoid stock-outs at the supplier in which the inventory of the supplier cannot be negative.
- Constraints (4) and (5) are similar to constraints (2) and (3) and apply to customers.
- Constraints (6), (7) and (8) define that the quantity delivered by a supplier's vehicle to each customer $i \in V'$ will fill the customer's inventory capacity if the customer is served and will be zero otherwise. These set of constraints impose to the Order-Up-To (OU) policy. If customer i is not visited throughout the period t, then constraint (8) means that the amount delivered to it will be zero (while constraints (6) and (7) are still respected). Otherwise, if customer i is visited during the period t, constraint (8) limits the quantity delivered to the customer's inventory holding capacity, and this bound is tightened by constraints (7), making it impossible to deliver more than what would fill this capacity. Constraints (6) models of OU inventory replenishment policy, ensuring that the quantity transported will be exactly within the bound provided by constraints (7). Therefore, to solve the IRP under the ML replenishment policy, this model can be modified to coerce the ML policy by dropping constraints (6).
- Constraints (9) state that the vehicle's capacity is not exceeded.
- Constants (10) and (11) guarantee that a feasible route is designed to visit all customers served in the period t. Constraint (10) enforces the number of arcs entering and leaving a vertex to be the same. Constraint (11) shows where a single vehicle is available.
- Constraints (12) is integrally and non-negativity conditions on the variables.

3.3.2. **The IRPT optimisation model**

The inventory routing problem with transhipment (IRPT) model is based on the model proposed by (Leandro et al., 2012). Transhipments can be made later during the time period. The transhipment can begin at the depot or at any customer in a subset $R \subseteq V'$. For instance, products can be delivered from one customer to another customer as

they request, or from customer $i \in R$ to customer $j \in V'$. The model also uses continuous variables v_i^t to enforce the VRP sub tour elimination constraints, which presents the sum of deliveries made by a vehicle in time period t after visiting customer i.

The transhipments also can take place when it is profitable to deliver products from the distribution centre to a customer on a "special- needs" basis. That is, for instance, when there is a special request or need for products. This can be done by outsourcing to a courier, who will pick up products either at the supplier or from any transhipment point. It is possible that both the supplier's vehicle and the outsourced vehicle visit the same customer in the same time period. Let w_{ij}^t be the amount of product delivered directly from $i \in R \cup \{0\}$ to customer $j \in V'$ at period t using the outsourced carrier. These authorised transfers are only made by direct shipping and this generates cost b_{ij} associated with transhipping from i to j. The formulation of IRP with transhipment consists of minimising the total cost is given below:

$$\text{Minimize} \sum_{t\in T} \mathrm{h}_0 I_0 + \sum_{i\in v'} \sum_{t\in T} h_i I_i^t + \sum_{i\in v} \sum_{j\in v, i<j} \sum_{t\in T} c_{ij} x_{ij}^t + \sum_{i\in R\{0\}} \sum_{j\in v'} \sum_{t\in T} b_{ij} w_{ij}^t \qquad (13)$$

The total costs of IRPT, are the sum of inventory holding costs at the depot and at the customers, the routing/delivery costs of the supplier's vehicle and the transhipment costs. In classical vehicle routing, travel costs are distance dependent and are unrelated to the vehicle capacity. However, transhipment costs are distance and quantity dependent because this is the outsource transports to customer which can be defined in term of their contracts.

The IRPT optimisation model use the following constraints:

$$I_0^t = I_0^{t-1} + z^t - \sum_{i\in V'} q_i^t - \sum_{i\in V'} w_{0i}^t, \qquad t \in T \qquad (14)$$

$$I_0^t \geq 0, \qquad t \in T \qquad (15)$$

$$I_i^t = I_i^{t-1} + q_i^t + \sum_{j\in R\{0\}} w_{ji}^t = \sum_{j\in V'} w_{ij}^t - d_i^t, \qquad i \in V' \quad , \quad t \in T \qquad (16)$$

$$I_i^t \geq 0, \qquad i \in V, \qquad t \in T \qquad (17)$$

$$I_i^t \leq C_i, \qquad i \in V, \qquad t \in T \qquad (18)$$

$$q_i^t \geq C_i \sum_{j \in V'} x_{ij}^t - I_i^{t-1}, \quad i \in V', \qquad t \in T \tag{19}$$

$$q_i^t \leq C_i - I_i^{t-1}, \quad i \in V', \qquad t \in T \tag{20}$$

$$q_i^t \leq C_i \sum_{j \in V'} x_{ij}^t, \quad i \in V', \qquad t \in T \tag{21}$$

$$\sum_{i \in V'} q_i^t \leq Q, \qquad t \in T \tag{22}$$

$$\sum_{i \in V} x_{ij}^t = \sum_{i \in V} x_{ji}^t, \quad j \in V \ , \quad t \in T \tag{23}$$

$$\sum_{i \in V} x_{i0}^t \leq 1, \quad t \in T \tag{24}$$

$$V_i^t, q_i^t, w_{ji}^t \geq 0, \quad i \in V', j \in R\{0\}, t \in T \tag{25}$$

$$x_{ij}^t \in \{0,1\} \qquad i, j \in V, i \neq j \ , \quad t \in T \tag{26}$$

- Constraints (14) represents the current inventory level at the supplier (I_0) at the end of period $t \in T$given by the previous inventory level (I_0^{t-1}), plus the quantity (z^t) made available in the period t, and minus both the total quantity shipped and transhipped to customers by supplier's vehicles in period t.
- Constraints (15) defines the stock outs at the supplier such that the inventory of the supplier cannot be negative.
- Constraints (16) the existing inventory level at every customer (I_i^t) in the period t is given by its preceding inventory level along the period $t-1$ plus the quantity (q_i^t) transported by depot's vehicle in the period t, plus the total transhipment amount to customers within the period t minus its demand (d_i^t) in the period t.
- In constraints (17), the restrictions guarantee that for every customer $i \in v'$ the inventory level I_i^t remains non-negative at all times.
- Constraints (18) state limitations to ensure that for each customer $i \in v'$ the inventory level (I_i^t) remains below the maximum level C_i at the end of each period.

- Constraints (19), (20) and (21) ensure that these sets confirm that the quantity delivered by a supplier's vehicle to each retailer $i \in V'$ will fill the customer's inventory capacity if the customer is served and will be zero otherwise. Therefore, if customer i is not visited throughout the period t, then constraints (21) ensures that the amount delivered to it will be zero, while constraints (19) and (20) are still considered. However, if customer i is visited during the period t, constraint (21) limits the quantity distributed to the customer's inventory holding capacity, and constraint (20) makes sure this bound is tightened, making it possible to deliver more and exceed this capacity. Constraints (19) models of the Order-Up-To (OU) inventory replenishment policy, ensuring that the quantity transported will be exactly within the bound provided by constraints (22). Therefore, to solve the IRP and IRPT under the ML replenishment policy, it significant does to drop constraints (19) and (21).
- Constraints (22) certifies that the vehicle's capacity is not exceeded. The routing constraints are a guarantee that a feasible route is designed to visit all customers served in the period t.
- Constraints (23) enforces the number of arcs entering and leaving a vertex to be the same.
- Constraints (24) means a single vehicle is available.
- Constraints (25) and (26) are integrality and nonnegativity conditions on the variables.

3.4. Proposed CWS heuristic for solving IRP and IRPT

This study proposes a solution method to solve the IRP and IRPT under OU or ML replenishment policy by using CWS heuristic. The CWS is one of the best-known heuristic algorithms, which has been widely applied in many routing problems. To the best our knowledge, this is first study that the CWS heuristic is used as a stand-alone heuristic in order to obtain the solution for IRP and IRPT. The CWS is probably one of the best cited and successful heuristics for solving the VRPs (Juan et al., 2011). The reason for choosing the CWS heuristic is because it has been proven to be an efficient heuristic, it is also successful in terms of solution convergence and it is also flexible enough to be combined with other solution methods. This research also proposes to extend CWS heuristic with replenishment policies to solve IRP and IRPT.

The main idea behind the CWS is that it is able to take into account the distance saved i.e. if two customers who were independently served, are instead served by one delivery vehicle, savings occur. This simply means that the heuristic is based on an estimation of savings from merging routes. The only Euclidean distance is used in this experiment in which the results are compared with in the literature. The co-ordinates have taken from benchmark problem. The distance matrix is calculated with the following equation;

$$c_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} \tag{27}$$

where $x_i, y_i \; and \; x_j, y_j$are the geographical location of customer$s \; i \; and \; j$. The saving value between customers $i \; and \; j$ is calculated as

$$s_{i,j} = c_{0,j} - c_{i,j} \tag{28}$$

where $c_{0,j}$ is the travelling distance between depot and customer j and $c_{i,j}$ is the travelling distance between customer $i \; and \; j$. So, the saving value are collected in the saving list using the equation (33) as shows in Figure 3.1, where point 0 represents the depot.

$$\mathrm{s_{i,j}} = \mathrm{c_{0,i}} + \mathrm{c_{j,0}} - \mathrm{c_{i,j}} \tag{29}$$

Figure 3.1 represented by (a) shows customers i and j who are visited on separate routes. An alternative to this is to visit the two customers on the same route, for example in the sequence $i - j$ as illustrated in Figure 3.1 (b). Because the transportation costs are given, the savings that result from driving the route (see right-hand side in Figure 3.1) represented by (b) instead of the two routes (see left-hand side in Figure 3.1) represented by (a) can be calculated. These savings are estimated between all nodes, and then decreasingly sorted. The bigger saving (at the top of the list) is always taken and used to merge the two associated routes. After that, the values in the saving list are sorted in decreasing order, then the rout merging procedure starts from the top of the saving list (the largest $s_{i,j}$). So, both customers $i \; and \; j$ are combined into the same route if the total demand does not exceed the capacity of a vehicle and no other route constraints exist.

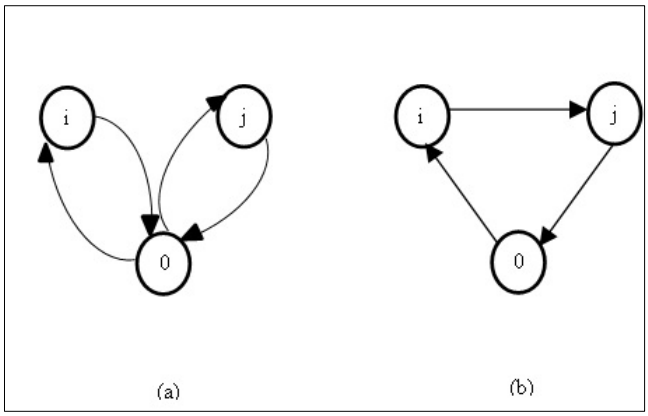

Figure 3.1 : Illustration of the saving concept use for IRPs

3.4.1. **IRP-based CWS heuristic**

The proposed IRP-based CWS heuristic is an iterative approach designed to find good solutions to the integrated inventory and routing problem. This problem is to estimate the quantity of each customer by applying the OU/ML replenishment policies. When deciding on the allocation of the customers along the route, the aim is to ensure that only one vehicle is needed to cover a route. Consequently, this solution is generated by using the classical CWS. For the feasible solutions, a round-trip route starting and ending from the depot, by satisfying all the customer demands, each demand node must be visited by the vehicle exactly once. Whereas, the CWS heuristic always selects the edge with the largest saving value in each step. The proposed CWS algorithm applied to the IRP under OU policy which aims to avoiding an infeasible solution where a stock-out would occur at a customer due to limited deliveries. Therefore, all the transhipment arcs from the supplier, and from all customers to all other customers, are kept with large penalty costs. This means that a feasible solution can always be reached, but it is at a very high cost if transhipments are used. These costs act as penalties in the objective function when the vehicle capacity is exceeded.

In this case, CWS algorithms are set as a routing network flow problem (i.e., Traveling Salesman Problem: TSP) with a saving list to enable the connection of arcs into one route. From the supplier node, the vehicle starts a trip for each period. It visits all the selected customer nodes. The vehicle carries requested units of product from the supplier with up to Q units of flow. The OU replenishment policy is modelled to solve the flow on the arcs, connecting customers in successive time periods when customer i is visited in period t, then the arc linking to it on the next period has a flow equal to $U_i - d_i^t$. Moreover, the ML replenishment policy is applied as the same as the OU policy, except that the arc connecting the customers in successive time periods have a

minimum flow equal to 0. The vehicle vertex is served from the supplier with up to Q units, and the minimum cost routing algorithm decides how much to serve to each customer. Dummy arcs are again inserted as a penalty for an unvisited customer and for solutions that would require an exceeded vehicle capacity. Hence, the IRP under OU yields upper bound on the IRP under ML then one constraint is relaxed in the former problem (Leandro et al., 2012b).

The procedure of IRP-based CWS heuristic is described by Algorithm 3.1 showing the Pseudo-code of IRP-based CWS heuristic. This procedure provides an estimate of the total distribution cost under the ML/OU replenishment and transhipment policies with a known demand. The holding cost is calculated by multiplying the value of the assigned holding weight and the inventory level for each client. Therefore, both ML/OU replenishment policies can be obtained by how much each customer needs to refill.

This procedure contains essential input parameters as follows; inventory policies, the nodes to be served, and a set of constraints. Firstly, the OU/ML replenishment policies are applied for determining product quantities to be delivered in each period, in order to fulfil each customer inventory (Pseudocode: line 2 to line 13), which will be used to calculate the inventory holding cost. Then the vehicle starts from the depot to all customer nodes. Later, the associated routing costs are estimated by using CWS (Pseudocode: line 20 to line 24), that finds the near-optimal route by choosing the edge with the largest cost saving, this edge is then chosen for merging two routes successively into one new route. Formerly, all edges are saved in the list in which it is sorted from highest to lowest saving based on distance. For the selection, it is logical that the value that has the greatest saving is more likely to be selected from the savings list.

Algorithm 3.1. Pseudocode for IRP-based CWS heuristic.

1: **Procedure IRP based CWS** *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, l_{n,}, n$*)*
▷ d_n: *Demand*
▷ I_n: Current inventory level
▷ p_n : *Penalty costs*
▷ h_n : Inventory holding cost
▷ l_n: Lost demand
▷ n: Customer node

2: **for** each customer n **do**
3: h_n←inventory holding cost of each Node n ▷ ML/OU policy
4: *nodeList* ← getNodeList (inventoryPolicy) ▷ List of node n
5: oversuppliedList (*OSL)* ← declare an empty list to store nodes which have stocks in the inventory
6: undersuppliedList *(USL)*← declare an empty list to store nodes which have not enough stocks in the inventory
7: **while** stopping criterion is not satisfied **do**
8: **for** each node in the list of available nodes n **do**
9: d_n ← demand of Node n
10: I_n ← I_{n-1} - d_n
11: **if** $I_n > 0$ **then**
12: **add** *n* into *OSL*
13: **else if** $I_n < 0$ **then**
14: **add** *n* into *USL*
15: **end if**
16: **end for**
17: **for** each undersupplied node in undersupplied list *n* **do**
18: $penaltyCosts \leftarrow p_n * l_n$
19: **end for**
20: *bestSol* ←*CWS (nodeList)*
21: *sol* ← *Ranking saving list and selected edge (input)*
22: *sol* ← *generate route (sol)*
23: **if** $sol < bestSol$ **then** ▷ *Compare the total cost of the obtainedSol with the bestSol*
24: *bestSol* ← *sol* *(total cost=inventory cost +transportation cost)*
25: **end if**
26: **end while**
27: **end for**
28: **return** *bestSol*
29: **end procedure**

3.4.2. IRPT-based CWS heuristic

This research focuses on the CWS heuristic for solving the IRP with transhipment (IRPT); this is advantageous in term of logistics costs and supply chain performance. For this case, the transhipment refers to transferring products from a customer to another customer, or from a supplier to customer during a time period if it is possible. However, in practice as seen in the literature that, transhipment is less likely to occur

when collaborative relationships are not established in a supply chain, although the principle can improve the supply chain performance in theory studies (Waller, Johnson, & Davis, 1999). Therefore, the procedure of IRPT-based CWS heuristic are the same as that mentioned in the above section of IRP-based CWS heuristic. The main difference between these two problems is that lateral transhipment is used when the supplier cannot meet customer demand and the customer may face stock-outs. So, the status of the inventory level at each node is calculated. With these cases, the following may happen: a node has enough stock in the inventory, and a node does not have enough stock in the inventory.

Algorithm 3.2: Procedure for TRANSHIPMENT

1: **Procedure Transhipment** (*underSuppliedList (USL), overSuppliedList (OSL),* ns, l_n, w_n, b_n, n)
- ▷ ns: *Node supplier*
- ▷ I_n: Current inventory level
- ▷ b_n: Transhipment cost/ *the round trip distances*
- ▷ w_n: Quantity transhipment
- ▷ n: Customer node

2: **for** each undersupplied node in *USL* n **do**
3: **while** the current inventory level $I_n < 0$ **do**
4: $ns \leftarrow$ find the nearest node in the oversupplied list to be the supplier
5: $b_{n,ns} \leftarrow$ calculate the round-trip distance between n and ns
6: update the inventory level of n
7: add *transhipmentCosts* $\leftarrow b_{n,ns} * w_n$
8: **end while**
9: **end for**
10: **end procedure**

In case when the supplier not able to satisfy any customer demand in the route, this may lead to customer stock out during that period. Therefore, the transhipment policy can be applied to fulfil the customer's inventory. By doing this, a transhipment node must be identified by selecting the nearest client who has enough inventory capacity to supply as shown in procedure of transhipment (Algorithm 3.2.). However, the supplier must be responsible for the transhipment costs, which consider the quantity of products of the transhipment factor, and the round-trip distance.

This section also describes the IRPT-based CWS algorithm under an OU or ML replenishment policy (Pseudocode: line 2 to line 14). The procedure of IRPT-based CWS heuristic is described by algorithm 3.3 showing in Pseudo-code of IRPT-based CWS heuristic. The problem is modelled as a network flow problem and solved by means of the saving algorithm, as described above. Secondly, the IRPT_ML is similar

to IRPT_OU except that arcs connecting customers between successive time periods, do not force the flow to respect the OU replenishment policy. The saving algorithm solves the network flow, which determines the quantities served (Pseudocode: line 29 to line 34). Thus, it is easy to see that if all transhipments are set to zero, then the problem reduces IRPT_OU and IRPT_ML to IRP_OU and IRP_ML, which yields an upper bound on the IRPT_OU and IRPT_ML optimum.

Algorithm 3.3: Pseudocode for IRPT-based CWS heuristic.

1: **Procedure IRPT based CWS** *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, b_n, l_n, w_n, n$*)*
▷ d_n: *Demand*
▷ I_n: Current inventory level
▷ p_n : *Penalty costs*
▷ h_n: Inventory holding cost
▷ b_n: Transhipment cost Value
▷ l_n: Lost demand
▷ w_n: Quantity transhipment
▷ n: Customer node
2: **for** each customer n **do**
3: h_n ← inventory holding cost of each Node n ▷ ML/OU policy
4: *nodeList* ← getNodeList (inventoryPolicy) ▷ List of node n
5: oversuppliedList (*OSL*) ← declare an empty list to store nodes which have stocks in the inventory
6: undersuppliedList (*USL*) ← declare an empty list to store nodes which have not enough stocks in the inventory
7: **while** stopping criterion is not satisfied **do**
8: **for** each node in the list of available nodes n **do**
9: d_n ← demand of Node n
10: I_n ← I_{n-1} - d_n
11: **if** $I_n > 0$ **then**
12: **add** n into *OSL*
13: **else if** $I_n < 0$ **then**
14: **add** n into *USL*
15: **end if**
16: **end for**
17: **if** policy = TRANSHIPMENT **then** ▷ Transhipment procedure
18: *transhipmentCosts* ← $b_n * w_n$
19: **end if**
20: **for** each n in *OSL* **do**
21: *inventoryCosts* ← $h_n * I_n + p_n * l_n$
22: *nodeToSupply* ← declare an empty list of nodes to record the nodes to be filled in this round
23: **for** each n in *nodeList* **do**
24: **if** I_n is less than the level specified by the policy **then**
25: **add** *(nodeToSupply, n)*
26: **end if**
27: **end for**
28: **end for**
29: *bestSol* ← *solve CWS (nodeList)*
30: *sol* ← *Rank saving list and select edge (input)*
31: *sol* ← *generate route (sol)*
32: **If** *sol < bestSol* **then** ▷*Compare the total cost of the obtained solution with the bestSol*
33: *bestSol* ← *sol*
34: **end if**
35: **end while**
36: **end for**
37: **return** *bestSol*
38: **end procedure**

3.5. Computational experiments

The computational results report on the performance of the IRP-based CWS and IRPT-based CWS heuristics. The optimisation model of these heuristics considers the minimisation of inventory and transportation costs, for the IRP with and without transhipment. Two policies are also considered: an order up to (OU), and maximum level (ML) replenishment policy. In order to evaluate the performance of the IRP-based CWS heuristic under the replenishment policy (OU or ML) the instances used are taken from the benchmark introduced by Leandro et al. (2012b). The instances are divided into two classes according to their inventory cost, following the work of Archetti et al. (2007). Also, the instances are made up of up to (i) three time periods and 50 customers, and (ii) six time periods and 30 customers. These specific time periods are used for testing the approach because they must be compared the results with the previous work.

These instances are the suffix low/high that represent two levels of inventory holding costs. With regard to low cost instances, the inventory holding costs are selected randomly in the interval [0.01, 0.05]. In contrast, the inventory holding costs are selected randomly in the interval [0.1, 0.5] for high cost instances. These instances are divided into two classes according to their inventory cost, and in each set of instances they are divided by the number of customers, each of which contains five instances. For example, absn10 is an instance with 10 customers which has been averaged over 5 instances such as abs1n10to abs5n10. The data set consists of the customer coordinates, holding cost, inventory level (low and max), vehicle capacity and customer demand. The first subsection shows the performance of CWS heuristic and replenishment policies, which are evaluated to solve the IRP and IRPT.

The solutions generated from the proposed IRP-based CWS and IRPT-based CWS are compared to the solutions published in the literature. The instances have been generated on the basis of the following data.

- Number of retailers n: 50;
- Time horizon H: 30;
- Product quantity z_i^t absorbed by retailer i at time t: Constant over time, i.e.,$z_i^t = z_i, t \in T$ and randomly generated as an integer number in the interval [10,100];
- Product quantity of z_0^t made available at the supplier at time t: $\sum_{i \in M} z_i$;

- Maximum level U_i of the inventory at retailer i: $z_i g_i$, where g_i is randomly selected from the set {2, 3} and represents the number of time units needed in order to consume the quantity U_i;
- Starting inventory level I_0 at the supplier: $\sum_{i \in M} U_i$;
- Starting inventory level I_{i0} at retailer i: $U_i - z_i$;
- Inventory cost at retailer $i \in M, h_i$: randomly generated in the intervals $[0.01, 0.05]$ and $[0.1, 0.5]$;
- Inventory cost at the supplier h_o: 0.03 when h_i is generated in $[0.01, 0.05]$ and 0.3 when h_i is generated $[0.1, 0.5]$;
- Transportation capacity C: $\frac{3}{2}\sum_{i \in M} z_i$;
- Transportation cost c_{ij}: $\left[\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}\right]$, where the points (x_i, y_i) and (x_j, y_j) are obtained by randomly generating each coordinate as an integer number in the interval [0,500].

In all the instances, random selections are performed in accordance to a uniform distribution. The algorithms are coded in Java programming language with eclipse software, and all the tests are run on a computer with a Core i5, 2.30 GHz processor and 4GB of RAM. The experiments consider the average of five instances for each combination and then compare different algorithms with these cases. The computational results are shown in all tables, which report an average over five instances (abs1nX to abs5nX, while is the set of X = 5, 10. 15. …, 50) for each approach. Each instance has been run for 20 times. On all the tables, the first column represents the name of the instance, which shows the number of the customers (absn05 stand for five customers).

The cost breakdown of the total costs such as the inventory holding (HC), penalty (PC), transhipment (TrC), routing (RC) and total costs (TC) are shown in Tables 3.1 and 3.3. The average costs for each instance are shown in bold number. For the IRP and IRPT under both the ML/OU replenishment policies solved by this chapter proposed algorithm are represented as IRP_ML-based CWS, IRP_OU-based CWS, IRPT_ML-based CWS and IRPT_OU-based CWS heuristics. Tables 3.1 and 3.3 clearly show one result calculated from five instances for each combination of the cost element of IRP-based CWS and IRPT-based CWS heuristics under OU or ML replenishment policies with a time period (p) equal to three and six respectively. Each

table illustrates the results of the low or high inventory cost in two time periods($p = 3, p = 6$). The first and second column represents the instance name, which includes the number of the customers. The cost breakdowns of the total costs are shown from the third column to the eight columns for IRP-based CWS and IRPT-based CWS under ML/OU policies, respectively.

All tables also, show the average results aggregated in four classes of instances. The first class contains the instances with time period $p = 3$ and low inventory cost ($h_i \in [0.01, 0.05] and h_0 = 0.03$). The second class contains the instances with time period $p = 3$ and high inventory cost ($h_i \in [0.1, 0.5]\ and\ h_0 = 0.3$), while the third class the instances with time period $p = 6$ and low inventory cost ($h_i \in [0.01, 0.05]\ and\ h_0 = 0.03$). Finally, the fourth class contains the instances with time period $p = 6$ and high inventory cost ($h_i \in [0.1, 0.5]\ and\ h_0 = 0.3$).

3.5.1. **IRP-based CWS heuristic**

The computational results in Tables 3.1 show the average costs and the cost breakdown. The average costs are highlighted as bold numbers, the solutions of IRP_OU-based CWS heuristic are slightly larger than those average costs of IRP_ML-based CWS heuristic. The reason is that using either the ML or OU policy can lead to a different impact on inventory holding cost. In the former type of policy, the quantity delivered to a customer is such that the level of the inventory at the customer is not greater than the maximum level in a time period. In the latter type of policy, the quantity shipped is such that the level of the inventory at the customer reaches exactly its maximum level at the time period, as long as its holding capacity is respected. It is obvious that ML replenishment policy gives lower inventory holding cost during period than that OU replenishment policy. Also, ML replenishment policy can reduce routing and inventory holding cost because it is less likely to deliver more products to the customer when its inventory capacity is fully served. On the other hand, adopting OU replenishment policy may lead to multi-trip deliveries and penalty cost due to customer stock out. Therefore, penalty or transhipment costs are not charged as no stock outs are allowed.

Moreover, the low and high inventory holding costs can lead to an increase the inventory holding cost in which the low inventory holding costs have lower holding cost than high inventory holding costs in both time periods ($p = 3, p = 6$). For high inventory holding costs, the holding costs are higher than those holding cots on all

instances sets with low inventory holding costs, but routing costs are not much different. In time period equal three, the averages cost 641.33 (low) and 6046.44 (high) are much different (Table 3.1). In additional, the time period is equal to six, it can lead to higher holding cost than other costs, which can be seen that much of the cost is the inventory holding costs. This may be due to the fact that the longer the duration, the higher inventory holding costs. That will need to stock inventory resulting in more storage charge during the period of time.

Low inventory holding cost and Time period = 3							
Name of Instance		IRP_ML-based CWS heuristic			IRP_OU-based CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	94.33	1202	1295.89	94.33	1312	1406.33
2	absn10	137.30	1753	1890.30	147.94	1989	2136.94
3	absn15	303.65	1966	2269.65	238.90	2214	2452.90
4	absn20	533.48	2233	2766.48	508.61	2404	2912.61
5	absn25	600.56	2531	3131.56	637.70	2930	3567.70
6	absn30	708.63	2890	3598.63	757.88	3185	3942.88
7	absn35	841.77	3145	3986.77	857.77	3336	4193.77
8	absn40	984.66	3497	4481.66	1100.79	3519	4619.79
9	absn45	1043.49	3564	4607.49	1059.67	3797	4856.67
10	absn50	1165.46	3764	4929.46	1187.09	4009	5196.09
	Average	**641.33**	**2654.46**	**3295.79**	**659.07**	**2869.50**	**3528.57**
High inventory holding cost and Time period = 3							
Name of Instance		IRP_ML-based CWS heuristic			IRP_OU-based CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	890.63	1342	2232.63	983.21	1446	2429.21
2	absn10	2389.88	2148	4537.88	2248.45	2182	4430.45
3	absn15	2999.89	2698	5697.89	3540.20	2461	6001.20
4	absn20	4440.61	2997	7437.61	5088.45	2990	8078.45
5	absn25	5788.98	3368	9156.98	6698.50	3398	10096.50
6	absn30	6996.15	4076	11072.15	7090.65	4377	11467.65
7	absn35	7257.11	4453	11710.11	7460.66	4788	12248.66
8	absn40	8281.88	4603	12884.88	8909.45	5094	14003.45
9	absn45	10506.97	4818	15324.97	10510.96	5105	15615.96
10	absn50	10912.28	5233	16145.28	11409.50	5999	17408.50
	Average	**6046.44**	**3573.60**	**9620.04**	**6394.00**	**3784.00**	**10178.00**
Low inventory holding cost and Time period = 6							
Name of Instance		IRP_ML-based CWS heuristic			IRP_OU-based CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	371.20	2908	3279.20	371.29	3076	3447.29
2	absn10	571.13	4263	4834.13	595.88	4398	4993.88
3	absn15	765.70	5328	6093.70	669.88	5412	6081.88
4	absn20	895.36	6654	7549.36	897.97	6554	7451.97
5	absn25	953.45	7050	8003.45	936.76	7025	7961.76
6	absn30	1009.50	7782	8791.50	1030.01	7589	8619.01
	Average	**761.06**	**5664.17**	**6425.22**	**750.30**	**5675.67**	**6425.97**
High inventory holding cost and Time period = 6							
Name of Instance		IRP_ML-based CWS heuristic			IRP_OU-based CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	2395.54	3099	5494.54	2568.63	3083	5651.63
2	absn10	3689.69	5166	8855.69	4096.60	4989	9085.60
3	absn15	5373.35	6849	12222.35	5660.60	6971	12631.60
4	absn20	7952.34	8134	16086.34	7862.05	8207	16069.05
5	absn25	9089.90	9005	18094.90	9095.34	9091	18186.34
6	absn30	10090.53	11010	21100.53	11001.55	10579	21580.55
	Average	**6431.89**	**7210.50**	**13642.39**	**6714.13**	**7153.33**	**13867.46**

Table 3.1 : Average results of the cost breakdown of the IRP-based under ML/OU and p=3 and 6

3.5.1.1. *Comparison of the results of IRP-based CWS with the best results in the literature*

The computational results shown in Tables 3.2 provide the average results for the IRP on five instances generated for each combination of the number of customers (n) and the time horizon(H). Each table consists of the following: the bottom row divides into two columns, which present the Low/High inventory holding cost and the time horizon $p = 3$ $(n = 50)$ and $p = 6$ $(n = 30)$. These are grouping the number of instances, the best results of the branch and cut algorithm (ABLS) proposed by Archetti and Laporte (2007), the best solutions for the Adapted Large Neighbourhood Search (ALNS) introduced by Leandro et al. (2012) and the solution for the IRP-based CWS heuristic (IRP_ML-based CWS heuristic and IRP_OU-based CWS heuristic, respectively).

Table 3.2 shows the average results and the average percentages difference between solutions, when compared with ABLS ("A") for both ("B") IRP_ML-based CWS and IRP_OU-based CWS are shown in bold. Hence, the percentage (%) gap is calculated as follows: the solution of ABLS (A), the solution of IRP-based CWS under ML/OU replenishment policy (B); $Percentage\ (\%) gap = \left(\frac{B-A}{A}\right) * 100$.The average percentage gaps of total costs are approximated 4.89%, 3.04%, 6.29%, 6.39%, 5.87%, 4.69%, 6.77% and 5.10% respectively. These gaps show the significant difference between IRP-based CWS and ABLS (Archetti & Laporte, 2007). The results of IRP-based CWS and IRP-based CWS heuristics under ML/OU replenishment policy also compared to Archetti et al. (2007) results obtained by ABLS. The experiments show that the IRP-based CWS is able to find a good solution, with total costs becoming much closer to those of ABLS (Archetti & Laporte, 2007).

This table also presents the comparison of results between this chapter and Leandro et al. (2012) (ALNS). The experiments show that the IRP-based CWS is able to find good solution with total costs become much closer the solutions obtained by ABLS, but worse than those results obtained by ALNS. They have higher average total costs than those results proposed by Leandro et al. (2012) for on all instances sets. The percentage gaps on average 4.67%, 4.85%, 5.69%, 5.56%, 6.78%, 5.75%, 6.48% and 4.80%, respectively. It is only reasonable, since Leandro et al. (2012) adopted meta-heuristic to solve the problem, whereas a heuristic is used in this chapter.

Moreover, the comparison of average results between IRP-based CWS and those of Leandro et al. (2012) with under both ML/OU replenishment policies and the time

horizon $p = 3, p = 6$ as shows in Table 3.2. The results of both time periods solved by this chapter proposed method are higher than the results solving by ABLS and ALNS. However, from the table shows the computation time, which is considered to be consuming not much time for the large size of instances. When the time period is equal to 3, all instances with up to 50 customers can be solved in less than 40 minutes, while when the time period is 6, only instances with more than 30 customers can be solved in less than one hour. For the IRP_ML and the IRP_OU require a similar computational time.

Low inventory holding cost and Time period = 3								High inventory holding cost and Time period = 3							
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based CWS		Gap (%)	Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based CWS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A		(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	1275.86	1275.99	6.08	0.01	1280.00	2.01	0.32	absn05	2187.30	2201.20	5.82	0.64	2244.34	2.30	2.61
absn10	1910.93	1911.08	18.59	0.01	1992.86	11.89	4.29	absn10	4337.97	4339.35	17.13	0.03	4493.02	12.44	3.57
absn15	2207.77	2208.57	44.69	0.04	2293.74	30.98	3.89	absn15	5435.80	5438.80	46.34	0.06	5557.86	39.77	2.25
absn20	2665.58	2675.58	98.05	0.38	2900.49	60.45	8.81	absn20	7225.70	7262.24	93.08	0.51	7350.80	76.17	1.73
absn25	2987.90	2996.77	171.38	0.30	3026.08	139.99	1.28	absn25	7501.07	9007.11	164.36	20.08	9129.55	179.03	21.71
absn30	3292.93	3330.77	331.45	1.15	3587.37	394.21	8.94	absn30	10918.31	10941.55	315.76	0.21	11128.32	429.56	1.92
absn35	3448.84	3495.04	495.46	1.34	3673.24	709.11	6.51	absn35	11411.67	11472.14	575.54	0.53	11665.37	783.43	2.22
absn40	3703.82	3736.05	793.99	0.87	3945.10	997.89	6.51	absn40	12541.06	12632.72	801.64	0.73	12804.33	1105.76	2.10
absn45	3867.48	3886.26	1405.72	0.49	4000.98	1246.78	3.45	absn45	13865.34	13928.10	1197.02	0.45	14766.56	1334.54	6.50
absn50	4327.16	4366.43	1719.41	0.91	4442.44	1882.09	2.66	absn50	15410.82	15512.49	2109.81	0.66	16012.15	2090.98	3.90
Average	**2968.83**	**2988.25**	**508.48**	**0.55**	**3114.23**	**547.54**	**4.67**	**Average**	**9083.50**	**9273.57**	**532.65**	**2.39**	**9515.23**	**605.40**	**4.85**
Name of insnatce	**ABLS**	**ALNS**		**Gap (%)**	**IRP_OU-based CWS**		**Gap (%)**	**Name of insnatce**	**ABLS**	**ALNS**		**Gap (%)**	**IRP_OU-based CWS**		**Gap (%)**
	(A)	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**		**(A)**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	1418.76	1418.75	10.71	0.00	1496.34	2.98	5.47	absn05	2354.18	2354.17	9.45	0.00	2444.34	3.23	3.83
absn10	2228.67	2228.66	35.28	0.00	2292.86	13.65	2.88	absn10	4690.46	4691.02	34.62	0.01	4699.02	14.09	0.18
absn15	2493.47	2493.47	99.32	0.00	2503.74	36.87	0.41	absn15	5736.91	5740.66	97.09	0.07	6017.86	40.11	4.90
absn20	3053.02	3055.58	239.76	0.08	3100.49	77.08	1.56	absn20	7619.91	7626.94	224.24	0.09	8090.80	81.54	6.18
absn25	3451.15	3451.86	572.28	0.02	3646.08	197.08	5.65	absn25	9460.75	9476.04	446.47	0.16	10115.55	210.15	6.92
absn30	3643.22	3645.70	1072.47	0.07	3947.37	454.09	8.35	absn30	11320.65	11354.66	890.16	0.30	11458.32	489.12	1.22
absn35	3846.87	3850.83	1439.28	0.10	4173.24	981.44	8.48	absn35	11828.82	11848.90	1600.56	0.17	12665.37	1008.88	7.07
absn40	4125.70	4140.16	2755.72	0.35	4445.10	1232.09	7.74	absn40	13011.46	13043.95	2767.76	0.25	13974.33	1309.12	7.40
absn45	4270.61	4283.33	3417.87	0.30	4540.98	1583.45	6.33	absn45	14317.82	14392.04	3010.08	0.52	15566.56	1609.91	8.72
absn50	4810.87	4841.26	2675.47	0.63	5292.44	2009.89	10.01	absn50	15948.78	16077.86	2987.26	0.81	17412.15	2204.50	9.18
Average	**3334.23**	**3340.96**	**1231.82**	**0.16**	**3543.86**	**658.86**	**5.69**	**Average**	**9628.97**	**9660.62**	**1206.77**	**0.24**	**10244.43**	**697.07**	**5.56**
Low inventory holding cost and Time period = 6								**High inventory holding cost and Time period = 6**							
Name of insnatce	**ABLS**	**ALNS**		**Gap (%)**	**IRP_ML-based CWS**		**Gap (%)**	**Name of insnatce**	**ABLS**	**ALNS**		**Gap (%)**	**IRP_ML-based CWS**		**Gap (%)**
	(A)	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**		**(A)**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	3136.90	3288.30	12.90	4.83	3365.73	9.99	7.29	absn05	5354.20	5514.49	12.20	2.99	5544.77	10.07	3.56
absn10	4612.50	4751.96	54.26	3.02	4798.65	43.76	4.04	absn10	8601.92	8757.00	56.27	1.80	8971.40	51.15	4.30
absn15	5418.55	5546.96	160.81	2.37	5876.29	178.88	8.45	absn15	11543.04	11681.58	154.74	1.20	12247.92	200.09	6.11
absn20	6625.35	6762.36	381.15	2.07	7333.95	612.11	10.70	absn20	14602.14	14750.66	348.11	1.02	16046.18	706.15	9.89
absn25	7261.77	7443.39	599.87	2.50	7661.68	1256.15	5.51	absn25	16913.97	17113.84	703.10	1.18	18188.35	1409.11	7.53
absn30	7710.01	7835.02	1415.08	1.62	8072.92	2908.80	4.71	absn30	20410.65	20547.02	1597.86	0.67	21043.26	3298.68	3.10
Average	**5794.18**	**5938.00**	**437.35**	**2.74**	**6184.87**	**834.95**	**6.78**	**Average**	**12904.32**	**13060.77**	**478.71**	**1.48**	**13673.65**	**945.88**	**5.75**
Name of insnatce	**ABLS**	**ALNS**		**Gap (%)**	**IRP_OU-based CWS**		**Gap (%)**	**Name of insnatce**	**ABLS**	**ALNS**		**Gap (%)**	**IRP_OU-based CWS**		**Gap (%)**
	(A)	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**		**(A)**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	3299.98	3299.97	20.92	0.00	3465.73	11.12	5.02	absn05	5538.02	5538.91	22.82	0.02	5644.77	13.05	1.93
absn10	4832.89	4832.87	95.88	0.00	4998.65	61.54	3.43	absn10	8872.41	8872.41	106.84	0.00	9071.40	78.09	2.24
absn15	5566.39	5582.80	337.70	0.29	6026.29	256.08	8.26	absn15	11721.82	11738.50	370.67	0.14	12647.92	307.19	7.90
absn20	6833.29	6857.90	797.63	0.36	7433.95	698.05	8.79	absn20	14863.86	14883.49	1021.13	0.13	16046.18	733.65	7.95
absn25	7454.15	7487.80	1610.54	0.45	7761.68	1252.51	4.13	absn25	17170.82	17223.47	2221.46	0.31	18188.35	1429.02	5.93
absn30	7847.39	7888.56	3031.66	0.52	8572.92	2925.34	9.25	absn30	20657.28	20752.32	3399.49	0.46	21243.26	3400.55	2.84
Average	**5972.35**	**5991.65**	**982.39**	**0.27**	**6376.54**	**867.44**	**6.48**	**Average**	**13137.37**	**13168.18**	**1190.40**	**0.18**	**13806.98**	**993.59**	**4.80**

Table 3.2 : Comparison of average results for the IRP-based CWS under ML/OU, p=3, p=6 and ABLS, ALNS

3.5.2. **IRPT-based CWS heuristic**

In this section, the transhipment option within the context of IRP is an additional challenge of the problem, and therefore the classical CWS heuristic algorithms are applied to solve two variants of the problem: the IRPT_ML and IRPT_OU. The instances are identical to those of Leandro et al.(2012b) and allow transhipment by setting the unit cost associated with transhipping product from $i\ to\ j\ is\ b_{ij}$ ($b_{ij} = 0.01c_{ij}$). This study has also evaluated the impact of the transhipment cost on the solution and its total cost.

Table 3.3 shows the average cost breakdown and total cost in bold number obtained by the IRPT-based CWS heuristic. It is obvious that the average costs of the IRPT under ML policy are less than the IRPT handle by OU policy because of different policies are able to make an effect on the costs. Using the ML policy, the quantity of products supplied fills up to maximum inventory level at the time period. Consequently, ML policy can lead to reduce the transportation cost because it is less likely to deliver more products to the customer when its inventory capacity is fully served. Mainly, ML replenishment policy makes less inventory holding cost during period than that OU replenishment policy. On the other hand, adopting OU policy may lead to multi-trip deliveries due to customer extra demand or stock-out. Then, the transhipment costs are slightly higher than using ML policy in both time periods are equal 3 and 6.

Additionally, when the time periods are equal to 3 and 6, the low inventory holding costs have the lower holding cost than high inventory holding costs, however routing costs are not much different. When the time period is equal to 6, the averages costs are higher than the time period is equal to 3. This is reasonable that the longer the time horizon could be made the higher inventory holding and routing costs. For the average transhipment costs are not much different in both the time periods are equal to 3 and 6.

Low inventory holding cost and Time period = 3									
Name of Instance		IRPT_ML-based CWS heuristic				IRPT_OU-based CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	88.53	0.00	668	756.19	88.53	9.80	668	766.33
2	absn10	134.57	86.90	1375	1596.47	144.33	98.76	1399	1642.09
3	absn15	303.73	84.90	1545	1933.63	310.50	147.50	1579	2037.00
4	absn20	415.68	108.90	1866	2390.58	454.69	174.10	1885	2513.79
5	absn25	517.67	149.77	2061	2728.44	555.43	210.89	2050	2816.32
6	absn30	698.63	178.65	2179	3056.28	727.88	298.19	2307	3333.07
7	absn35	700.42	199.87	2398	3298.29	789.77	367.00	2544	3700.76
8	absn40	776.34	268.90	2509	3554.24	832.91	400.65	2698	3931.56
9	absn45	820.89	292.12	2794	3907.01	909.29	459.25	2998	4366.54
10	absn50	958.56	328.34	3003	4289.90	1003.57	498.50	3144	4645.72
	Average	**541.50**	**169.84**	**2039.77**	**2751.10**	**581.69**	**266.46**	**2127.16**	**2975.32**
High inventory holding cost and Time period = 3									
Name of Instance		IRPT_ML-based CWS heuristic				IRPT_OU-based CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	649.50	0.00	1003	1652.50	650.90	9.68	1113	1773.58
2	absn10	1998.32	99.99	1791	3889.31	1877.99	109.45	2209	4196.44
3	absn15	2899.34	114.50	2097	5110.84	2756.50	165.45	2503	5424.95
4	absn20	3770.88	138.90	2880	6789.78	4198.95	205.50	2980	7384.45
5	absn25	4977.15	209.23	3432	8618.38	5300.95	298.55	3602	9201.50
6	absn30	5999.56	294.50	4287	10581.06	6798.45	305.25	4000	11103.70
7	absn35	7009.98	330.95	4779	12119.93	7305.75	330.35	4423	12059.10
8	absn40	8790.80	399.95	5003	14193.75	8595.55	405.50	4590	13591.05
9	absn45	9302.65	425.25	5195	14922.90	9790.90	500.45	5005	15296.35
10	absn50	10028.88	477.70	5495	16001.58	10112.00	599.65	6027	16738.65
	Average	**5542.71**	**249.10**	**3596.20**	**9388.00**	**5738.79**	**292.98**	**3645.20**	**9676.98**
Low inventory holding cost and Time period = 6									
Name of Instance		IRPT_ML-based CWS heuristic				IRPT_OU-based CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	234.10	9.05	2298	2541.15	241.71	19.05	2298	2558.76
2	absn10	484.02	85.98	3529	4099.00	469.89	105.05	3403	3977.94
3	absn15	551.99	161.90	4105	4818.89	577.77	189.50	4208	4975.27
4	absn20	696.66	200.50	5254	6151.16	700.01	271.01	5390	6361.02
5	absn25	772.50	287.90	5801	6861.40	827.04	330.30	6001	7158.34
6	absn30	990.20	350.01	6440	7780.21	910.90	370.75	6701	7982.65
	Average	**621.58**	**182.56**	**4571.17**	**5375.30**	**621.22**	**214.28**	**4666.83**	**5502.33**
High inventory holding cost and Time period = 6									
Name of Instance		IRPT_ML-based CWS heuristic				IRPT_OU-based CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	2490.87	7.05	2237	4734.92	2473.20	13.60	2231	4717.80
2	absn10	3997.01	91.09	3834	7922.10	3963.09	169.33	3897	8029.42
3	absn15	5905.05	180.02	4990	11075.07	5945.45	208.30	4974	11127.75
4	absn20	7001.98	294.55	7021	14317.53	6994.66	352.75	6970	14317.41
5	absn25	8088.66	321.22	8558	16967.88	8099.08	541.99	8360	17001.07
6	absn30	10156.45	531.99	10098	20786.44	10062.10	684.40	10007	20753.50
	Average	**6273.34**	**237.65**	**6123.00**	**12633.99**	**6256.26**	**328.40**	**6073.17**	**12657.83**

Table 3.3 : Average results of the cost breakdown of the IRPT-based under ML/OU and p=3 and 6

3.5.2.1. *Comparison of the results of IRPT-based CWS with the lower bound and the best results in the literature*

The computational results of these experiments for the variants of the IRPT-based CWS heuristics are reported in Tables 3.4. The average results and the average percentage gaps for both the Low/High inventory holding cost with the time horizon $p = 3$ *and* $p = 6$ of the IRPT_ML-based CWS and IRPT_OU-based CWS are shown in bold numbers. The solutions costs of this chapter proposed algorithm are compared to those of the lower bound (LB) and the upper bound (UB) solution truncated by CPLEX (Leandro et al., 2012) and the solutions obtained by ALNS.

Table 3.4 presents the comparison of average results between IRPT-based CWS, LB and UB obtain by CPLEX with under both ML/OU replenishment policies in the times horizon are equal to 3 and 6. It can be seen that the IRPT-based CWS heuristic under OU/ML policies provided the good solution, but some results are better than UB obtained by CPLEX (see red numbers). Although, the solutions are worse than LB but it has taken less computation time for solving. This can be reasonable that the larger size of instances, the better solution when proposed method is used instead of exact method. Which can be seen from the average percentage gaps between this proposed algorithm and the UB (Leandro et al., 2012).

When the time period is equal to 3 and low/high inventory holding costs, the comparative results between IRPT-based CWS under both ML/OU policies and UB (Leandro et al., 2012) are shown in terms of the average costs. The red numbers show some average results obtained by IRPT-based CWS heuristics better than those UB tests. It is obvious that most of the total costs for a medium size of instances solved by this chapter proposed method have total costs lower than those obtained by Leandro et al. (2012) proposed algorithm. The IRPT-based CWS heuristic is able to give better results than UB obtained by CPLEX for some of medium and large sizes of the instances on all sets. These results are over 50% better than those UB for the IRPT under ML policy, low inventory cost with time period is equal to 3, and the average computation time is less than 15 minutes. They have better on up to six sets of instances on the all sets for the IRPT_ML, low and high inventory holding costs with three time periods, respectively. For the IRPT_OU with low inventory holding cost in the three-time period, they have four sets of instances lower than those solved by CPLEX. They also, have better on the average computation time than CPLEX with less than 12 minutes. This means that the use of transhipments is able to reduce costs and the CWS heuristic is able to solve deterministic case for medium size instances.

Even though this chapter proposed algorithm provided higher solutions costs than LB but it is able to find better solution than UB for most of instances on all sets with six-time period (see red numbers). The solutions solved by this chapter proposed method are on average 67% better than those obtained by Leandro et al. (2012) proposed method for the IRPT_ML and low inventory holding costs. They are better on all five sets of instances on the set with high inventory holding costs for the IRPT_OU with low and high inventory holding costs. For the IRPT_ML with high inventory holding costs, the solutions have also better on all four set of instances. They are on average 50% better than those UB achieved through CPLEX.

In addition, the IRPT-based CWS heuristic under both OU/ML policies is able to find good solution with the less computation time. Some results are better than those UB achieved through CPLEX though they are slightly worse than ALNS (Leandro et al., 2012) but this chapter required less running time than the ALNS. This is reasonable because Leandro's work applied meta-heuristics to solve the problem, which can produce a better solution than heuristics. However, the IRPT_OU-based CWS heuristic with low inventory holding cost in time period is equal to 6, they have five sets of instances better than those ALNS. While, they have average total costs higher than those average total costs presented by Leandro et al. (2012) for the IRPT under ML/OU with low/high holding costs in three and six time periods.

It can be seen that only four or five of the average value costs are slightly less than the average costs provided by ALNS (see green number in Table 3.4). The other total costs solved by the IRPT-based CWS heuristic under ML/OU policies are higher than those solved by ALNS (Leandro et al., 2012). It is remarkable that meta-heuristics perform a much more thorough search of the solution space compared to classical heuristics and heuristics, allowing inferior and sometimes infeasible moves, as well as recombination of solutions to create new ones. Meta-heuristics are usually adopted in complex problems, rather than heuristics, and it is well known from the literature that meta-heuristics are the most effective method (Cordeau et al., 2002).

Low inventory holding cost and Time period = 3												High inventory holding cost and Time period = 3								
	LB	UB	Time (s)	Gap (%)	ALNS		Gap (%)	IRPT_ML-based CWS		Gap (%)	LB	UB	Time (s)	Gap (%)	ALNS		Gap (%)	IRPT_ML-based CWS		Gap (%)
	(A)	(C)		(C-A)/A	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A	(A)	(C)		(C-A)/A	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	741.76	741.76	0.29	0.00	744.80	5.28	0.41	776.38	3.16	4.67	1660.26	1660.26	0.72	0.00	1660.26	5.98	0.00	1667.44	4.01	0.43
absn10	1577.23	1577.30	2.64	0.00	1586.91	14.52	0.61	1605.71	8.79	1.81	3998.8	3999.02	3.17	0.01	**4011.81**	15.36	0.33	3999.05	10.45	0.01
absn15	1839.88	1840.06	58.46	0.01	1849.83	30.64	0.54	1962.67	24.65	6.67	5054.12	5054.58	35.86	0.01	**5061.92**	33.76	0.15	5061.54	27.80	0.15
absn20	2232.38	2278.04	1143.92	2.05	2290.55	56.74	2.61	2498.99	40.19	11.94	6752.82	6822.95	1134.22	1.04	**6869.26**	57.73	1.72	6868.82	47.12	1.72
absn25	2367.55	2657.38	1920.44	12.24	2579.18	92.04	8.94	2796.79	77.04	18.13	8342.03	8714.21	1496.09	4.46	8562.59	91.26	2.64	**8618.38**	80.08	3.31
absn30	2735.52	3116.05	2006.29	13.91	2985.99	165.20	9.16	**3082.55**	102.30	12.69	10319.56	10716.07	1586.59	3.84	10557.63	159.13	2.31	**10591.63**	130.09	2.64
absn35	2742.38	4034.69	1200.07	47.12	**3448.17**	248.74	25.74	**3296.03**	199.98	20.19	10689.39	11511.05	1191.40	7.69	11309.46	282.47	5.80	**11397.84**	235.45	6.63
absn40	2799.85	4480.11	1992.43	60.01	3361.80	348.93	20.07	**3499.58**	289.90	24.99	11573.32	14329.19	1643.60	23.81	12165.28	360.83	5.11	**13185.09**	309.10	13.93
absn45	3034.49	5110.99	1613.87	68.43	**3697.61**	461.71	21.85	**3682.37**	330.45	21.35	12979.04	15822.42	2143.09	21.91	13699.96	627.96	5.55	**14214.48**	456.78	9.52
absn50	3396.46	9172.72	2301.51	170.07	4071.09	760.30	19.86	**4077.12**	515.50	20.04	14406.63	18144.64	2461.43	25.95	15004.18	939.87	4.15	**15826.23**	601.15	9.85
Average	**2346.75**	**3500.91**	**1223.99**	**37.38**	**2661.59**	**218.41**	**10.98**	**2727.82**	**159.20**	**14.25**	**8577.60**	**9677.44**	**1169.62**	**8.87**	**8890.24**	**257.44**	**2.78**	**9143.05**	**190.20**	**4.82**
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based CWS**		**Gap (%)**	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based CWS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**	**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	745.39	745.39	0.06	0.00	745.39	6.56	0.00	776.38	3.67	4.16	1664.38	1664.38	0.07	0.00	1664.38	8.06	0.00	1785.91	5.09	7.30
absn10	1616.08	1616.11	2.29	0.00	1617.51	29.55	0.09	1736.78	9.42	7.47	4043.83	4043.94	2.40	0.00	4043.94	30.12	0.00	4132.11	11.34	2.18
absn15	1851.02	1851.13	19.58	0.01	1864.70	82.24	0.74	2051.57	44.11	10.83	5069.04	5069.51	20.64	0.01	5116.21	74.63	0.93	5362.62	59.06	5.79
absn20	2274.24	2339.63	557.58	2.88	2442.44	183.28	7.40	2498.99	97.56	9.88	6818.37	6877.20	684.51	0.86	6927.36	151.21	1.60	7407.42	102.30	8.64
absn25	2421.25	2710.08	1887.00	11.93	2724.67	389.10	12.53	2796.79	175.78	15.51	8422.54	8611.29	2100.71	2.24	8754.03	350.23	3.94	9103.38	211.01	8.08
absn30	2812.98	3123.22	2037.20	11.03	3341.49	635.79	18.79	3382.55	402.34	20.25	10332.37	10765.01	1362.58	4.19	10867.17	521.52	5.18	10903.66	475.89	5.53
absn35	2772.64	4316.84	1339.51	55.69	**3522.66**	895.11	27.05	**3596.03**	669.05	29.70	10695.61	11608.97	1622.33	8.54	11367.04	930.08	6.28	**11409.36**	753.50	6.67
absn40	2828.15	4531.14	1760.32	60.22	**3795.60**	1577.21	34.21	**3799.58**	1020.40	34.35	11588.44	13117.59	1427.80	13.20	12563.16	1577.64	8.41	13458.98	1133.34	16.14
absn45	3090.45	4540.60	1511.04	46.92	**4078.89**	2350.23	31.98	**4082.37**	1600.90	32.10	13069.51	16468.68	1427.33	26.01	13921.34	2244.01	6.52	**14595.90**	1890.60	11.68
absn50	3376.61	6491.95	1610.72	92.26	**4581.07**	2898.06	35.67	**4597.12**	2005.70	36.15	14418.41	17326.09	2009.60	20.17	15560.06	3375.50	7.92	**16727.93**	2101.67	16.02
Average	**2378.88**	**3226.61**	**1072.53**	**28.09**	**2871.44**	**904.71**	**16.85**	**2931.81**	**602.89**	**20.04**	**8612.25**	**9555.27**	**1065.80**	**7.52**	**9078.47**	**926.30**	**4.08**	**9488.73**	**674.38**	**8.80**
Low inventory holding cost and Time period = 6												**High inventory holding cost and Time period = 6**								
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_ML-based CWS**		**Gap (%)**	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_ML-based CWS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**	**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	2554.37	2554.45	2.62	0.00	**2558.37**	10.90	0.16	2554.46	5.76	0.00	4741.97	4742.19	2.23	0.00	4748.31	13.28	0.13	4749.01	8.99	0.15
absn10	3443.84	4056.36	1660.45	17.79	4095.10	36.70	18.91	4099.42	13.66	19.04	7429.95	8015.35	2035.50	7.88	7961.72	37.91	7.16	8015.25	20.11	7.88
absn15	3619.79	5114.79	976.65	41.30	**4834.73**	80.68	33.56	**4817.97**	52.98	33.10	9933.52	11262.90	813.05	13.38	10949.14	88.45	10.22	**10983.76**	66.33	10.57
absn20	4385.61	9880.24	1367.07	125.29	6020.83	174.24	37.29	**6089.20**	120.78	38.84	12321.4	17437.75	1166.17	41.52	14152.04	179.70	14.86	**14205.09**	156.34	15.29
absn25	4679.71	13786.58	1769.39	194.60	6808.40	295.30	45.49	**6812.54**	205.65	45.58	14333.9	21962.55	1937.04	53.22	16320.18	329.51	13.86	**16991.51**	224.54	18.54
absn30	5547.73	14296.67	2336.32	157.70	7466.89	671.24	34.59	**7758.82**	474.34	39.86	18182.52	29215.44	2064.83	60.68	20235.50	787.47	11.29	**20838.57**	499.07	14.61
Average	**4038.51**	**8281.52**	**1352.08**	**89.45**	**5297.39**	**211.51**	**28.33**	**5355.40**	**145.53**	**29.40**	**11157.21**	**15439.36**	**1336.47**	**29.45**	**12394.48**	**239.39**	**9.59**	**12630.53**	**162.56**	**11.17**
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based CWS**		**Gap (%)**	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based CWS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**	**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	2561.77	2561.84	1.35	0.00	**3299.97**	16.23	28.82	2564.46	9.45	0.10	4759.29	4759.54	2.21	0.01	4760.94	14.98	0.03	4766.69	10.02	0.16
absn10	3524.75	4054.39	2221.58	15.03	**4832.87**	70.29	37.11	**3979.42**	39.67	12.90	7495.05	8077.14	1477.52	7.77	8038.53	65.29	7.25	**8038.54**	44.89	7.25
absn15	3880.76	5042.41	1029.67	29.93	**5582.80**	208.08	43.86	**4881.02**	169.53	25.77	10037.35	11225.20	1109.10	11.83	11027.72	156.07	9.87	**11106.95**	188.02	10.66
absn20	4396.94	7418.80	1382.76	68.73	**6857.90**	491.31	55.97	**6395.20**	354.22	45.45	12332.22	16778.25	1207.34	36.05	14278.32	442.43	15.78	**14326.66**	387.07	16.17
absn25	4641.33	10566.48	1834.25	127.66	**7487.80**	805.16	61.33	**6842.54**	503.34	47.43	14262.97	20807.23	1540.39	45.88	16867.70	906.47	18.26	**16975.62**	545.65	19.02
absn30	5562.96	13917.82	1590.49	150.19	7888.56	1650.37	41.81	**7992.72**	998.26	43.68	18169.95	26411.34	1951.69	45.36	20492.18	1718.08	12.78	**21097.34**	1003.22	16.11
Average	**4094.75**	**7260.29**	**1343.35**	**65.26**	**5991.65**	**540.24**	**44.82**	**5442.56**	**345.75**	**29.22**	**11176.14**	**14676.45**	**1214.71**	**24.48**	**12577.57**	**550.55**	**10.66**	**12718.63**	**363.15**	**11.56**

Table 3.4 : Comparison of average results for the IRPT-based CWS under ML/OU, p=3, p=6 and literature

3.5.2.2. *Comparison of the results of IRPT-based CWS with IRP-based CWS*

For the case with transhipment, to evaluate the performance of IRPT-based CWS heuristic, this sub-section shows the solutions solved by the IRPT-based CWS heuristic compared to the IRP-based CWS heuristic as shown in Table 3.5. This table also, shows the average results aggregated in four classes of instances. The first class contains the instances with time period $p = 3$ and low inventory cost. The second class contains the instances with time period $p = 3$ and high inventory cost, whilst the third class the instances with time period $p = 6$ and low inventory cost. Lastly, the fourth class contains the instances with time period $p = 6$ and high inventory cost.

In the time horizon is equal to 3, IRPT-based CWS is able to provide lower average total costs than IRP-based CWS for all sets of instances with low/high inventory holding costs. These total costs found by IRPT-based CWS are on the average 18.83% with low holding cost and 7.93% with high holding cost for under ML replenishment policy better than IRP-based CWS. Whereas, the solutions obtained by IRP-based CWS under OU replenishment policy are on average 27.73% with low inventory holding cost and 11.38% with high inventory holding cost higher than those solutions solved by IRPT-based CWS. For the IRPT-based CWS in time period is equal to 6, the results have also better on all sets of instances. The average total costs are 17.96% under ML policy and 20.19% under OU policy with low inventory holding costs lower than IRP-based CWS with the same class contains the instances. In high inventory holding cost, the results solved by IRPT-based CWS are on average 10.20% handle by ML policy and 10.83% handle by OU policy have better than IRP-based CWS.

Therefore, transhipment can be advantageous in a deterministic context in which no shortage occurs as they may yield both an overall reduced distribution and inventory holding cost. Also, to solve the IRP without transhipment ($b_{ij} = 1.00c_{ij}$) is setting the transhipment cost sufficiently larger than the IRPT ($b_{ij} = 0.01c_{ij}$). So, the total costs of the IRPs are higher than IRPTs; however, companies decide to arrange a lateral transhipment only in the case of an emergency or in the case of preventing stock-out caused by high demand. With regard to the means of transportation, the companies prefer to use their vehicles for direct delivery. Therefore, the reason for avoiding transhipment is that the companies have to pay more, not only in shipping costs, but also other operational costs.

Low inventory holding cost and Time period = 3				High inventory holding cost and Time period = 3		
Name of insnatce	IRP_ML-based CWS (A)	IRPT_ML-based CWS (B)	Improvement (%) (A-B)/B	IRP_ML-based CWS (A)	IRPT_ML-based CWS (B)	Improvement (%) (A-B)/B
absn05	1280.00	776.38	64.87	2244.34	1667.44	34.60
absn10	1992.86	1605.71	24.11	4493.02	3999.05	12.35
absn15	2293.74	1962.67	16.87	5557.86	5061.54	9.81
absn20	2900.49	2498.99	16.07	7350.80	6868.82	7.02
absn25	3026.08	2796.79	8.20	9129.55	8618.38	5.93
absn30	3587.37	3082.55	16.38	11128.32	10591.63	5.07
absn35	3673.24	3296.03	11.44	11665.37	11397.84	2.35
absn40	3945.10	3499.58	12.73	12804.33	13185.09	-2.89
absn45	4000.98	3682.37	8.65	14766.56	14214.48	3.88
absn50	4442.44	4077.12	8.96	16012.15	15826.23	1.17
Average	**3114.23**	**2727.82**	**18.83**	**9515.23**	**9143.05**	**7.93**
Name of insnatce	**IRP_OU-based CWS (A)**	**IRPT_OU-based CWS (B)**	**Improvement (%) (A-B)/B**	**IRP_OU-based CWS (A)**	**IRPT_OU-based CWS (B)**	**Improvement (%) (A-B)/B**
absn05	1496.34	776.38	92.73	2444.34	1785.91	36.87
absn10	2292.86	1736.78	32.02	4699.02	4132.11	13.72
absn15	2503.74	2051.57	22.04	6017.86	5362.62	12.22
absn20	3100.49	2498.99	24.07	8090.80	7407.42	9.23
absn25	3646.08	2796.79	30.37	10115.55	9103.38	11.12
absn30	3947.37	3382.55	16.70	11458.32	10903.66	5.09
absn35	4173.24	3596.03	16.05	12665.37	11409.36	11.01
absn40	4445.10	3799.58	16.99	13974.33	13458.98	3.83
absn45	4540.98	4082.37	11.23	15566.56	14595.90	6.65
absn50	5292.44	4597.12	15.13	17412.15	16727.93	4.09
Average	**3543.86**	**2931.81**	**27.73**	**10244.43**	**9488.73**	**11.38**
Low inventory holding cost and Time period = 6				**High inventory holding cost and Time period = 6**		
Name of insnatce	**IRP_ML-based CWS (A)**	**IRPT_ML-based CWS (B)**	**Improvement (%) (A-B)/B**	**IRP_ML-based CWS (A)**	**IRPT_ML-based CWS (B)**	**Improvement (%) (A-B)/B**
absn05	3365.73	2554.46	31.76	5544.77	4749.01	16.76
absn10	4798.65	4099.42	17.06	8971.40	8015.25	11.93
absn15	5876.29	4817.97	21.97	12247.92	10983.76	11.51
absn20	7333.95	6089.20	20.44	16046.18	14205.09	12.96
absn25	7661.68	6812.54	12.46	18188.35	16991.51	7.04
absn30	8072.92	7758.82	4.05	21043.26	20838.57	0.98
Average	**6184.87**	**5355.40**	**17.96**	**13673.65**	**12630.53**	**10.20**
Name of insnatce	**IRP_OU-based CWS (A)**	**IRPT_OU-based CWS (B)**	**Improvement (%) (A-B)/B**	**IRP_OU-based CWS (A)**	**IRPT_OU-based CWS (B)**	**Improvement (%) (A-B)/B**
absn05	3465.73	2564.46	35.14	5644.77	4766.69	18.42
absn10	4998.65	3979.42	25.61	9071.40	8038.54	12.85
absn15	6026.29	4881.02	23.46	12647.92	11106.95	13.87
absn20	7433.95	6395.20	16.24	16046.18	14326.66	12.00
absn25	7761.68	6842.54	13.43	18188.35	16975.62	7.14
absn30	8572.92	7992.72	7.26	21243.26	21097.34	0.69
Average	**6376.54**	**5442.56**	**20.19**	**13806.98**	**12718.63**	**10.83**

Table 3.5 : Comparison average results between the IRP and IRPT –based CWS under ML/OU, p=3 and p=6

3.6. Chapter conclusion

This study investigates a multi-period IRP and IRPT with deterministic demand where each agent's demand can be fulfilled directly from supplier and indirectly by lateral transhipments. This chapter proposes the IRP and IRPT optimisation model. An initial approach is to rely on the inventory holding costs at the depot and each customer, and vehicle routing costs to deliver from the depot under both OU and ML replenishment policies. The CWS heuristic is applied in order to solve IRP and IRPT with deterministic demand in different periods. This algorithm is the first heuristic approach ever proposed for an inventory routing problem with and without transhipment. The effectiveness of the IRP-based CWS and IRPT-based CWS heuristics were tested though a set of computational experiments. Instances with up to 50 customers were solved with a time period equal to 3. While the time period is

equal to 6, the largest instances that could be solved involve 30 customers. The results show that lateral transhipment can reduce the overall inventory and routing costs, because no inventory shortage has occurred. This study aims to minimise the total cost and propose one of the basic heuristics to find a better solution for this complex combination optimisation problem. The proposed CWS heuristic provided good solutions which are very close to those obtained from benchmark literature (Leandro et al., 2012). However, this proposed approach IRP_ML, IRP_OU, IRPT_ML and IRPT_OU based on the CWS heuristics is able to be improved when combined with another algorithm. For this reason, in the next chapter, we implement a biased randomised algorithm to the IRP-based CWS heuristic. The randomisation process introduces a random behaviour into the heuristic, which is proven as a beneficial and efficient search mechanism inside the solution generation procedure.

Chapter 4: Randomised CWS heuristic for deterministic IRP and IRPT

4.1. Introduction

This chapter presents an improved CWS heuristic by combining the CWS heuristic with a biased randomisation technique. This is the first time a biased randomised version of CWS heuristic has been used to solve a deterministic IRP and IRPT. Therefore, this study describes the biased randomisation and how this algorithm works and its hybridisation with the CWS heuristic. This chapter focuses on the deterministic case of IRP and IRPT and proposes Randomised CWS heuristic to solve these problems under the replenishment policies. The use of biased randomisation with classical heuristic was first introduced by (Bresina, 1996). This author presents Heuristic Biased Stochastic Sampling, which is a search technique for a scheduling problem. Bresina's work (1996) has shown an outperformance of a greedy search with small size problem instances. Juan et al., (2010) proposed to use randomisation combined with CWS heuristic to solve a CVRP using the Simulation in Routing via the Generalized Clarke & Wright Savings heuristic (SR-GCWS) algorithm. This algorithm presents a skewed (e.g. geometric probability distribution) random behaviour within the classical CWS heuristic. This is used as a guide in the solution search process. Juan et al., (2014) proposed to use biased randomisation of meta-heuristic combined with of simulation. They applied the method to deterministic and stochastic Vehicle and Arc Routing Problems. They presented the basic idea behind biased randomisation and how it can be combined with classical heuristics to obtain solutions the VRP and Arc Routing Problems. This methodology is quite flexible, and it can be easily adapted for other similar problems in the area of routing or scheduling.

Biased randomisation of an algorithm is usually simple and easy to implement, it is also fast with very high probability of generating good solutions (Karp, 1991). The process uses a probabilistic distributions criterion (e.g. geometric and uniform probability distributions) for selecting edges from the saving list. This is a very important behaviour for the generation of solutions in the aspect of the algorithm. From the best our knowledge, randomisation of algorithms has been applied and adapted with other heuristics or meta-heuristics successfully, for solving many combinatorial optimisation problems such as the VRP, ARP and the scheduling problems. However, the biased randomisation version CWS heuristic has not been

applied for solving the IRP and IRPT in a deterministic case. This is the challenge of this study - presenting a use of this method to obtain the solution for these problems. The structure of this chapter is: section 4.2 presents this chapter contributions. Section 4.3 presents the proposed approach to solve IRP and IRPT with deterministic demand. The computational experiments of this chapter are shown in section 4.4 and the conclusion of this chapter is given in section 4.5.

4.2. Contribution

Randomised CWS heuristic is proposed to solve IRP and IRPT with deterministic demand in this chapter. To the best of our knowledge, this is the first time that this method is applied for the deterministic IRP and IRPT. Thus, this is a major contribution of this research. In this chapter, the implementation of a biased randomisation CWS algorithm will be presented. The implementation will be applied to improve the IRP and IRPT with deterministic demand under an OU or ML replenishment policies. The optimisation objective is to minimise the total costs comprising of holding costs and transportation costs. Another contribution of this chapter is to show how the randomisation technique improves the CWS heuristic. The benefits of randomised version CWS is the use of a random number generator during the construction phase of heuristic while keeping the heuristic criterion. This enables to find a different solution that can be generated every time this algorithm is performed, which increases the possibility of outperforming the previous solution found. Also, the fact that the CWS algorithm has very low execution times, introducing biased randomisation does not impact greatly on the computation time. As the technique does not use a uniform sorted list of savings stored in a savings list, the output of the randomized algorithm will provide a good solution. Further in this section, the solution generated by Randomised CWS heuristic will be compared with the solution obtained by the classical CWS heuristic, in order to show the potential of the IRP-based Randomised CWS heuristic and IRPT-based Randomised CWS heuristic.

4.3. Proposed IRP and IRPT based Randomised CWS heuristic

The proposed approach focuses on the routing aspect of the deterministic IRP and IRPT under OU or ML replenishment policies. The proposed algorithm is performed by improvement and modification of the randomisation version of Clark and Wright (1964) saving heuristic (CWS). This study introduces Randomised CWS heuristic in order to generate the best routes to satisfy objectives and constraints of IRP and IRPT.

A feasible solution consists of a round-trip route in which a vehicle starts and ends at the same depot. Furthermore, each customer must be visited by the vehicle exactly once.

4.3.1. **Randomised CWS heuristic Process**

Characteristically, CWS heuristic uses a priority list of potential movements (savings) which is traversed in an iterative manner, i.e., at each iteration, the next constructive movement is selected from the list, which is sorted such that the most saving is on the top list. Also, the criteria employed to sort the list depends on the specific heuristic being considered in which the list is sorted in a decreasing manner. Notice that this is a deterministic process, once the criterion has been defined, it provides a unique order for the list of potential movements. This is the reason for applying the concept of randomising the order in which the elements in the list are selected. Hence a different output is likely to occur each time the entire procedure is executed. However, note that using a uniform distribution for the randomisation of that list will basically destroy the logic behind the greedy behaviour of the heuristic. Therefore, the output of the randomised algorithm is unlikely to provide a good solution. Also, note that, this procedure could be run multiple times, but it is likely that all the solutions generated could be significantly worse than the one provided by the original heuristic. To avoid losing the common sense behind the heuristic, Greedy Randomised Adaptive Search Procedure (GRASP) proposes to consider a restricted list of candidates (Festa & Resende, 2009). This is a sub-list including just some of the most promising movements that is, the ones at the top of the list, and then applying a uniform randomisation in the order of the elements of that restricted list as shown in Figure 4.1 (see left-hand side). This is the logic behind transforming the deterministic procedure into a randomised process which can be encapsulated into a multi-start process. On the other hand, presented the basic idea of the Multi-start biased Randomised of Heuristic with Adaptive local search (MIRHA). This combined classical greedy heuristic with pseudorandom varieties from a different non-symmetric probability distribution, is developed in order to add a biased randomisation into classical heuristics. This proposed method by Juan et al. (2013) goes one step more, instead of restricting the list of candidates, it assigns different probabilities of being selected to each potential movement in the sorted list. In this way, the elements at the top of the list receive more probabilities of being selected than those at the bottom of the list, but potentially all elements could be selected (see left-hand side in Fig 4.1). Notice that by doing this, the process does not only avoid the

issue of selecting the proper size of the restricted list, but it also guarantees that the probabilities of being selected are always proportional to the position of each element in the list.

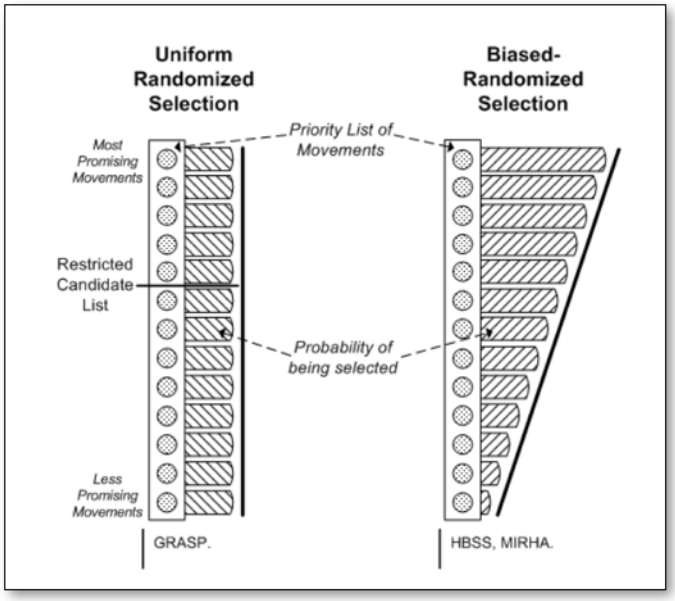

Figure 4.1 : Uniform randomisation vs. biased randomisation (Juan et al. 2013)

The steps for transforming CWS heuristic into a probabilistic heuristic by means of biased randomisation are:

- The CWS heuristic is used for this IRP and IRPT with the characteristic of always choosing the largest saving edge on the top of the saving list at each step.
- Using the geometric probability distribution, it can give a selection of probabilities in each edge in the saving list. The larger saving value is more likely to be selected from the list than the other lesser saving value (Algorithm 4.2: Pseudocode for Randomised edge-selection).

Algorithm 4.1: Pseudocode for RANDOMISED CWS HEURISTIC

1**: Procedure Randomised CWS** (*nodeList, savingList,* c_{ij}*, constraints)*
2: SavlingList←createSavlingList (nodeList)
3: sol←constructInitialSol (nodes, c_{ij}) ▷ $sol = c_{0,i} + c_{j,0} - c_{i,j}$
4: **while** the saving list is not empty **do**
5: *edge* ← randomSample *(savingList)*
6: *node1* ← getOrigin *(edge)*
7: *node2* ← getEnd *(edge)*
8: *route1* ← getRouteByNode *(sol, node1)* ▷ *Find the routes that contain both nodes*
9: *route2* ← getRouteByNode *(soln, node2)*
10: **if** all CWS route-merging conditions are satisfied **then**▷ *Constraints*
11: *sol* ← mergeRoute *(route1, route2, edge)* ▷ *CWS heuristic*
12: **end if**
13: removeEdge *(edge, savingList)*
14: **end while**
15: **return** solution
16: **end procedure**

Algorithm 4.2: Pseudocode for RANDOMISED EDGE-SELECTION

1**: Procedure SelectEdgeAtRandom** (List*, rng)*
▷*rng: random number (a;b)*

2: betat←generateRandomNumber (rng: a;b) ▷*e.g.: a=0.10 and b=0.25*
3: randomValuel←generateRandomNumber (rng: 0;1)
4: *position* ← floor (log (randomValue=log (1<beta))) ▷ *Geometic ditribution*
5: *position* ← listSize ▷*Random positon from the list*
6: **return** getEdgeAtPositon(position)
7: **end procedure**

To construct an initial solution, the CWS algorithm is implemented (Algorithm 4.1: Pseudocode for Randomised CWS heuristic). The algorithm uses the concept of savings associated with each edge to be considered for merging. At each step, the edge with the greatest saving is selected, if and only if the two-corresponding routes can be combined into a new feasible route. Also, the selected edge is composed of nodes that are directly connected to the depot. Thereafter, a new savings list of edges is obtained by randomising the original savings list through the use of a geometric probability distribution as shown in Pseudocode for Randomised edge-selection (Algorithm 4.2). This allows different outputs at each iteration of the multi-start procedure. The logic behind the randomised CWS heuristic is that edges with higher savings are more likely to be selected from the list than those with lower savings.

Then, until the savings list gets empty, an iterative process begins in which the edge at the top of randomised list is most likely to be extracted. For the IRP/IRPT studied in this thesis, the vehicle routing decision is managed by the extended CWS heuristic as explained above after the inventory management decisions have been made.

4.3.2. **IRP-based Randomised CWS heuristic**

The concept behind IRP-based Randomised CWS solution is random selection by using pseudo geometric distribution to assign a selection probability to each candidate in the savings list as have been described in the section 4.3.1. Therefore, a probability distribution is used in this proposed method in order to assign a selection probability to each edge in the savings list and the savings list represents the routing cost. The IRP-based Randomised CWS heuristic considers inventory policies in relation to routing and vehicle load. Both OU and ML replenishment policies use with IRP-based Randomised algorithm, which are modelled as an IRP_OU-based Randomised CWS heuristic, IRP_ML-based Randomised CWS heuristic.

The procedure of IRP_OU/ML- based Randomised CWS is similar to the IRP_OU/ML-based CWS, except for the selection of arcs in connecting network process. This concept for selecting arc has been described and explained in the section 4.3.1. For that reason, inventory policies are used to calculate demand and inventory holding cost for each customer. Pseudocode for IRP-based Randomised CWS heuristic shows in Algorithm 4.3, which are explained below:

- The replenishment policy is applied to estimate the inventory level holding cost in each customer.
- The routing problem is considered, and the associated routing costs are estimated by using Randomise CWS. It is starting from an initial dummy solution in which each customer is served by a dedicated vehicle. Then an iterative merge route is processed from the route in the initial solution. The process for merging a route can be improved by using the random selection (merging criterion), which selects the biggest saving edge of the saving edge list (Algorithm 4.2).
- The total costs for each route with combination of inventory cost are estimated by applying policy after the routing link is calculated.

Pseudocode for IRP-based Randomised CWS heuristic is presented as follows. Firstly, the OU/ ML replenishment policy is applied for determining product quantity to be delivered in each period, in order to fulfil the inventory of each customer (Pseudocode: line 2 to line 18). Secondly, the procedure for generating a list of edges connecting any two nodes is shown in Algorithm 4.3 (Pseudocode: line 18 to line 20). The initial dummy solution is generated by using the classical CWS heuristic, resulting in a delivery route with the vehicle starting and ending at the same depot. The saving associated with a given edge is calculated as the reduction in costs for merging two different routes into a single new route. Then, these edges are stored in a list, which is sorted from highest to lowest savings. After each iteration of the multi start process, a biased randomisation of the saving list is produced (Pseudo-code: line 20 to 30). The selection probability is the logic of the weighted saving value associated with each edge. Hence, each combination of edges has a chance of being selected and merged with previously built routes. At the end of each iteration, a new solution is iteratively constructed by merging routes, and the newly built route costs and status is updated.

Algorithm 4.3: Pseudocode for IRP-based RANDOMISED CWS HEURISTIC

1: **Procedure IRP based Randomised CWS** *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, l_n, n$*)*

▷ d_n: *Demand*
▷ I_n: Current inventory level
▷ p_n : *Penalty costs*
▷ h_n: Inventory holding cost
▷ l_n: Lost demand
▷ n: Customer node

2: **for** each customer n **do**
3: h_n←inventory holding cost of each Node n ▷ *ML/OU policy*
4: *nodeList* ← getNodeList (inventoryPolicy)
5: oversuppliedList (*OSL)* ← declare an empty list to store nodes which have stocks in the inventory
6: undersuppliedList *(USL)*← declare an empty list to store nodes which have not enough stocks in the inventory
7: **while** the saving list is not empty **do**
8: **for** each node in the list of available nodes n **do**
9: d_n ← demand of Node n
10: I_n ← I_{n-1} - d_n
11: **if** $I_n > 0$ **then**
12: **add** n into *OSL*
13: **else if** $I_n < 0$ **then**
14: **add** n into *USL*
15: **end if**
16: **end for**
17: **for** each undersupplied node in undersupplied list n **do**
18: $penaltyCosts \leftarrow p_n * l_n$
19: **end for**
20: *SavingList* ← createSavingList *(nodeList)*
21: *sol* ← createInitialSol *(nodeList)*
22: *edge* ← randomSample *(savingList)*
23: *node1* ← getOrigin *(edge)*
24: *node2* ← getEnd *(edge)*
25: *route1* ← getRouteByNode *(sol, node1)*
26: *route2* ← getRouteByNode *(soln, node2)*
27: **if** route1 and route2 can be merged according to specific merging criteria **then**
28: *sol* ← mergeRoute *(route1, route2, edge)*
29: **end if**
30: removeEdge *(edge, savingList)*
31: **end while**
32: **end for**
33: **return** solution
34: **end procedure**

4.3.3. **IRPT-based Randomised CWS heuristic**

This section presents an extension of the IRP deals with transhipment. The lateral transhipment between customers have been used in the routing problem and conjunction multi-customer routing problems, in order to increase the flexibility of the system and reduce transportation costs. Therefore, the deterministic case of the IRP similar to Coelho et al. (2012b) with transhipment is considered for minimising the distribution costs. However, the main difference between this work and their work is the methodology for solving IRPT. Furthermore, the experimental results obtained by using the Randomised CWS heuristic will be compared with a benchmark problem from the literature. The potential of proposed IRPT-based Randomised CWS heuristic will be shown.

However, in this case it is different from IRP whereby the lateral transhipments are allowed to take place after the demand is realised. Also, the case of the inventory level of customer becomes negative or stock-out. The process of selecting the customer to be the transhipment point starts by checking the updated inventory level of all customers, then finding the nearest customer which has enough inventory level to be the supply node. In addition, the cost of transhipment will be added to the total costs - the supplier has responsibility for this. This solution will provide an estimate of the total cost under the ML/OU replenishment and transhipment policies with known demand.

Algorithm 4.4 shows Pseudocode for IRPT-based Randomised CWS heuristic, the quantity of products to supply are calculated. Then the status of the inventory level at each node is calculated: oversupply is declared when a node has enough stock in the inventory, and undersupply is declared when a node does not have enough stock in the inventory. Later, the holding cost is calculated by multiplying the value of the assigned holding weight and the inventory level for each client (Pseudocode: line 2 to line 13). Once again, the producer's concept of IRPT-based Randomised CWS heuristic under OU/ML replenishment policies similar to IRP-based Randomised CWS heuristic under OU/ML replenishment policies, which has been described in the section 4.3.2. The processes of IRPT and IRP can be differentiated by transhipment procedure, which is an additional procedure of IRPT. Transhipment occurs when the delivery cannot meet demand (undersupply) (Pseudocode: line 9 to line 20). In this case, the transhipment policy is used to serve the customer. The geographically nearest oversupplied nodes are found by checking the updated inventory level, then

selecting that node to fill the demand. The details of these processes, however, are explained in the chapter 3.

Algorithm 4.4: Pseudocode for IRPT-based RANDOMISED CWS HEURISTICS

1: **Procedure IRPT based Randomised CWS** *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, b_n, l_n, w_n, n$*)*
▷ d_n: *Demand*
▷ I_n: Current inventory level
▷ p_n : *Penalty costs*
▷ h_n: Inventory holding cost
▷ b_n: Transhipment cost Value
▷ l_n: Lost demand
▷ w_n: Quantity transhipment
▷ n: Customer node
2: **for** each customer n **do**
3: h_n←inventory holding cost of each Node n ▷ *ML/OU policy*
4: *nodeList* ← getNodeList (inventoryPolicy)
5: oversuppliedList (*OSL*) ← declare an empty list to store nodes which have stocks in the inventory
6: undersuppliedList (*USL*) ← declare an empty list to store nodes which have not enough stocks in the inventory
7: **while** the saving list is not empty **do**
8: **for** each node in the list of available nodes n **do**
9: d_n ← demand of Node n
10: I_n ← I_{n-1} - d_n
11: **if** $I_n > 0$ **then**
12: **add** n into *OSL*
13: **else if** $I_n < 0$ **then**
14: **add** n into *USL*
15: **end if**
16: **end for**
17: **if** policy = TRANSHIPMENT **then**
18: *transhipmentCosts* ← $b_n * w_n$ ▷ *Transhipment*
19: **end if**
20: **for** each n in *OSL* **do**
21: *inventoryCosts* ← $h_n * I_n + p_n * l_n$
22: **end for**
23: *SavingList* ← createSavingList *(nodeList)*
24: *sol* ← createInitialSol *(nodeList)*
25:
26: *edge* ← randomSample *(savingList)* ▷ *Randomised CWS*
27: RCWS*Sol* ← mergeRoute *(route1, route2, edge)*
28: **end while**
29: **end for**
30: **return** solution
31: **end procedure**

4.4. Computational experiments

The performance of the IRP-based Randomised CWS and IRPT-based Randomised CWS heuristics under ML/OU replenishment policies is examined. Most of the findings on the performance of the proposed methodology will be explained and presented in tables. The case study in this chapter focuses on the deterministic IRP and IRPT where demand is known, and the same problem instances as the previous chapter are used. This chapter extends the work from Chapter 3 by applying biased randomisation (geometric probability distribution) technique into the CWS heuristic. The IRP/IRPT-based Randomised CWS heuristics have compared the previous works, while the potential of the proposed method for solving IRP and IRPT is illustrated. All instances used for experiments are taken from the benchmark (Archetti et al., 2007 and Leandro et al., 2012b). All sets of instances are divided and classed the same as described in Chapter 3. The solutions provided by IRP-based Randomised CWS under both the ML/OU replenishment policies will be presented in the first subsection. The next subsection will show the results of IRPT-based Randomised CWS under both the ML/OU policies. All sets of instances and analyses have been generated and run on a computer with a Core i5, 2.30 GHz processor and 4GB of RAM.

This chapter considers the integrated inventory and routing problem where the inventory capacity at depot cannot be negative; it always has enough products to serve all the customers and customer demand is deterministic. The best-found solution for each approach is the total costs which consists of the following costs: inventory holding cost, routing costs and transhipment costs. Transhipment costs are always added in the total cost, when the transhipment (IRPT) is considered. Notice that this strategy always provides good solutions in terms of routing costs. Since stock-out is not allowed, these solutions do not have negative inventory level. This mean that, if lateral transhipment is used to fill up the inventory capacity, stock-out can be avoided. Finally, transhipment help lower expected total cost.

All tables show the average results aggregated in four classes of instances. The first class contains the instances with time period $p = 3$ and low inventory cost ($h_i \in [0.01, 0.05]$ and $h_0 = 0.03$). The second class contains the instances with time period $p = 3$ and high inventory cost ($h_i \in [0.1, 0.5]$ and $h_0 = 0.3$), while the third class the instances with time period $p = 6$ and low inventory cost ($h_i \in [0.01, 0.05]$ and $h_0 = 0.03$). Finally, the fourth class contains the instances with time period $p = 6$ and high

inventory cost ($h_i \in [0.1, 0.5]\ and\ h_0 = 0.3$). For the IRP and IRPT under both the ML/OU replenishment policies with this chapter proposed Randomised CWS heuristic are represented as the IRP_ML-based Randomised CWS, the IRP_OU-based Randomised CWS, the IRPT_ML-based Randomised CWS and the IRPT_OU-based Randomised CWS heuristics.

Table 4.1 and 4.5 present the cost breakdowns of total cost (TC) and its components which are: the inventory holding (HC), transhipment (TrC), and routing (RC). There are clearly illustrated results with the low or high inventory cost in the time period $p = 3\ and\ p = 6$ for the IRP_ML-based Randomised CWS, IRP_OU-based Randomised CWS, IRPT_ML-based Randomised CWS and IRPT_OU-based Randomised CWS heuristics. The first column represents the instance number and the second column represents instance name, which includes the number of the customers. HC, RC, TrC, and TC are presented in the next columns, respectively.

4.4.1. IRP-based Randomised CWS heuristic

Table 4.1 gives the average cost breakdowns and the average total costs of the IRP-based Randomised CWS under ML/OU replenishment policies with low/high inventory holding costs in $p = 3\ and\ p = 6$, which are presented in bold numbers. The total costs provided from IRP-based Randomised CWS heuristic under OU replenishment policy are slightly higher than those costs obtained when using the ML replenishment policy both in low/high inventory holding cost and the time periods are equal to three and six. This is because when using either the OU or ML replenishment policy, it can lead to a different effect on the situation of inventory management, which impacts on the holding cost. Typically, OU replenishment policy delivers a quantity of products to meet the exact level of inventory need in the time period and penalty costs are incurred if the customer faces any shortage. However, ML replenishment policy provides the amount of product to fill up, until it meets the maximum level of inventory capacity in each time period. The ML policy does not incur penalty cost because stock-outs are avoided.

It is obvious that using ML or OU policies will impact to inventory and transportation costs in both time periods $p = 3\ and\ p = 6$. In particularly, in the case of low inventory holding cost, utilising ML policy will result in lower cost than that using OU policy. On the other hand, the OU policy will contribute to cost saving when

inventory cost is high. Also, the class of the low/ high inventory holding costs, the high inventory holding costs provide higher the inventory cost than low inventory holding costs with under both policies and in both time period ($p = 3, p = 6$). However, the routing costs are not much different for the low/high inventory holding costs. When the time horizon is equal to 3, the holding and routing costs less than the time horizon is equal to 6. This may be due to the fact that the shortage of the time period, the lower holding and routing costs.

Low inventory holding cost and Time period = 3							
Name of Instance		IRP_ML-based Randomised CWS heuristic			IRP_OU-based Randomised CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	89.05	1186	1275.05	106.60	1369	1475.60
2	absn10	137.01	1851	1988.01	250.35	2008	2258.35
3	absn15	301.55	2001	2302.55	324.23	2181	2505.23
4	absn20	491.11	2393	2884.11	495.01	2589	3084.01
5	absn25	516.85	2488	3004.85	599.79	2999	3598.79
6	absn30	647.20	2807	3454.20	687.83	3210	3897.83
7	absn35	709.15	3009	3718.15	748.50	3300	4048.50
8	absn40	848.01	3118	3966.01	839.82	3471	4310.82
9	absn45	910.82	3290	4200.82	890.48	3509	4399.48
10	absn50	1000.12	3407	4407.12	1039.15	3888	4927.15
	Average	**565.09**	**2555.00**	**3120.09**	**598.18**	**2852.40**	**3450.58**
High inventory holding cost and Time period = 3							
Name of Instance		IRP_ML-based Randomised CWS heuristic			IRP_OU-based Randomised CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	890.44	1359	2249.44	909.80	1539	2448.80
2	absn10	2280.39	2187	4467.39	2180.57	2352	4532.57
3	absn15	2853.45	2593	5446.45	3201.98	2787	5988.98
4	absn20	4310.41	2990	7300.41	4723.02	3216	7939.02
5	absn25	5784.90	3288	9072.90	6140.99	4001	10141.99
6	absn30	6993.30	4001	10994.30	7008.47	4310	11318.47
7	absn35	7562.20	4387	11949.20	7598.88	4509	12107.88
8	absn40	8392.78	4711	13103.78	8706.79	4787	13493.79
9	absn45	9001.45	5559	14560.45	9489.01	5800	15289.01
10	absn50	10083.25	5809	15892.25	10430.35	6677	17107.35
	Average	**5815.26**	**3688.40**	**9503.66**	**6038.99**	**3997.80**	**10036.79**
Low inventory holding cost and Time period = 6							
Name of Instance		IRP_ML-based Randomised CWS heuristic			IRP_OU-based Randomised CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	397.88	2950	3347.88	399.09	3051	3450.09
2	absn10	593.45	4179	4772.45	594.41	4391	4985.41
3	absn15	642.49	4977	5619.49	701.05	5211	5912.05
4	absn20	799.20	6399	7198.20	879.08	6399	7278.08
5	absn25	938.99	6561	7499.99	949.99	6550	7499.99
6	absn30	1000.72	6991	7991.72	1193.04	7001	8194.04
	Average	**728.79**	**5342.83**	**6071.62**	**786.11**	**5433.83**	**6219.94**
High inventory holding cost and Time period = 6							
Name of Instance		IRP_ML-based Randomised CWS heuristic			IRP_OU-based Randomised CWS heuristic		
		HC	RC	TC	HC	RC	TC
1	absn05	2397.90	3142	5539.90	2699.50	2966	5665.50
2	absn10	3980.20	4840	8820.20	4270.08	4800	9070.08
3	absn15	5180.55	6990	12170.55	5703.10	6548	12251.10
4	absn20	7411.90	8065	15476.90	7755.77	7931	15686.77
5	absn25	8907.88	8999	17906.88	8922.90	9101	18023.90
6	absn30	10720.07	10004	20724.07	10918.22	10132	21050.22
	Average	**6433.08**	**7006.67**	**13439.75**	**6711.60**	**6913.00**	**13624.60**

Table 4.1 : Average results of cost breakdown and total cost of the IRP-based Randomised CWS under ML/OU, p=3, p=6

4.4.1.1. *Comparison of the results of IRP-based Randomised CWS with the best results in the IRP-based CWS*

All tables show the average of the solutions provided by different algorithms. This IRP-based Randomised CWS algorithm approach considers average results of five instances set generated for each combination of customer (n) and time horizon(H). Tables are categorised as: the low/high inventory holding costs for the IRP_ML- based Randomised CWS and the IRP_OU-based Randomised CWS with time horizon is equal to 3. The IRP_ML-based Randomised CWS and the IRP_OU-based Randomised CWS with time horizon is equal to 6.

Low inventory holding cost and Time period = 3						
Name of insnatce	IRP_ML-based CWS (A)	IRP_ML-based Randomised CWS (B)	Improvement (%) (A-B)/B	IRP_OU-based CWS (A)	IRP_OU-based Randomised CWS (B)	Improvement (%) (A-B)/B
absn05	1280.00	1279.20	0.06	1496.34	1493.04	0.22
absn10	1992.86	1986.81	0.30	2292.86	2256.81	1.60
absn15	2293.74	2214.29	3.59	2503.74	2494.29	0.38
absn20	2900.49	2882.35	0.63	3100.49	3082.35	0.59
absn25	3026.08	2998.79	0.91	3646.08	3598.79	1.31
absn30	3587.37	3447.70	4.05	3947.37	3847.70	2.59
absn35	3673.24	3525.52	4.19	4173.24	4025.52	3.67
absn40	3945.10	3910.74	0.88	4445.10	4290.74	3.60
absn45	4000.98	3965.62	0.89	4540.98	4325.62	4.98
absn50	4442.44	4396.36	1.05	5292.44	5126.36	3.24
Average	**3114.23**	**3060.74**	**1.66**	**3543.86**	**3454.12**	**2.22**
High inventory holding cost and Time period = 3						
Name of insnatce	IRP_ML-based CWS (A)	IRP_ML-based Randomised CWS (B)	Improvement (%) (A-B)/B	IRP_OU-based CWS (A)	IRP_OU-based Randomised CWS (B)	Improvement (%) (A-B)/B
absn05	2244.34	2241.04	0.15	2444.34	2441.04	0.14
absn10	4493.02	4456.98	0.81	4699.02	4696.98	0.04
absn15	5557.86	5459.13	1.81	6017.86	5979.13	0.65
absn20	7350.80	7270.10	1.11	8090.80	7930.10	2.03
absn25	9129.55	9045.38	0.93	10115.55	10025.38	0.90
absn30	11128.32	11002.28	1.15	11458.32	11356.28	0.90
absn35	11665.37	11518.85	1.27	12665.37	12109.85	4.59
absn40	12804.33	12644.99	1.26	13974.33	13494.99	3.55
absn45	14766.56	14504.25	1.81	15566.56	15204.25	2.38
absn50	16012.15	15878.17	0.84	17412.15	17058.17	2.08
Average	**9515.23**	**9402.12**	**1.11**	**10244.43**	**10029.62**	**1.72**
Low inventory holding cost and Time period = 6						
Name of insnatce	IRP_ML-based CWS (A)	IRP_ML-based Randomised CWS (B)	Improvement (%) (A-B)/B	IRP_OU-based CWS (A)	IRP_OU-based Randomised CWS (B)	Improvement (%) (A-B)/B
absn05	3365.73	3352.54	0.39	3465.73	3452.54	0.38
absn10	4798.65	4764.47	0.72	4998.65	4964.47	0.69
absn15	5876.29	5565.34	5.59	6026.29	5885.34	2.39
absn20	7333.95	7179.09	2.16	7433.95	7279.09	2.13
absn25	7661.68	7458.28	2.73	7761.68	7498.28	3.51
absn30	8072.92	7967.26	1.33	8572.92	8167.26	4.97
Average	**6184.87**	**6047.83**	**2.15**	**6376.54**	**6207.83**	**2.35**
High inventory holding cost and Time period = 6						
Name of insnatce	IRP_ML-based CWS (A)	IRP_ML-based Randomised CWS (B)	Improvement (%) (A-B)/B	IRP_OU-based CWS (A)	IRP_OU-based Randomised CWS (B)	Improvement (%) (A-B)/B
absn05	5544.77	5531.58	0.24	5644.77	5631.58	0.23
absn10	8971.40	8827.21	1.63	9071.40	9027.21	0.49
absn15	12247.92	12136.19	0.92	12647.92	12236.19	3.36
absn20	16046.18	15598.48	2.87	16046.18	15598.48	2.87
absn25	18188.35	17892.42	1.65	18188.35	18092.42	0.53
absn30	21043.26	20882.21	0.77	21243.26	21072.21	0.81
Average	**13673.65**	**13478.02**	**1.35**	**13806.98**	**13609.68**	**1.38**

Table 4.2 : Comparison between the IRP-based CWS and the IRP-based Randomised CWS heuristic under ML/OU and p=3, 6

Table 4.2 shows the comparison of the average results obtained by the IRP-based CWS and the IRP-based Randomised CWS heuristics and the percentage difference between these approaches. This table gives detail for each category with low/high inventory holding costs and $p = 3, p = 6$ as follows: the first column gives the number

of customers. The second and the fifth columns are the average results solved by IRP_ML-based CWS and IRP_OU-based CWS heuristics respectively. The average total costs obtained by IRP_ML-based Randomised CWS and IRP_OU-based Randomised CWS heuristics are presented in the third and the sixth columns. The forth and the seventh columns show the percentage improvement between the IRP-based CWS and the IRP-based Randomised CWS heuristics respectively.

Regarding the averages total cost and the percentage gap from Table 4.2, the IRP-based Randomise CWS heuristic performs better than the IRP-based CWS heuristic for all of the instances on all sets. The results obtained by Randomised CWS are on average 1.66% and 2.22 % better than solutions solved by CWS for IRP_ML and IRP_OU for low inventory holding cost in $p = 3$, respectively. In the time period is equal to 6 with low inventory holding costs, the average results found by the IRPT-based CWS are improved on average 1.11% and 1.72% under ML and OU polices respectively. When the high inventory holding cost, this chapter proposed algorithms have 2.15% and 2.35% better for all sets of instances than those obtained by CWS heuristic under both ML/OU replenishment policies in time period is equal to 3, respectively. In the time period is equal to 6, also the % improvement between two approaches are on average 1.35% and 1.38% for all sets of instances and under ML or OU policies, respectively.

From the %improvement between the IRP-based CWS and the IRP-based Randomised CWS heuristics can be seen the potential of the proposed randomisation, which is using the probability distribution - it is an efficient way to select the next edge from the priority list. Consequently, these results show that biased randomisation is able to improve CWS heuristic for providing good solutions of IRP. Also, it is very easy to apply the randomisation technique to a classical heuristic.

4.4.1.2. Comparison of the results of IRP-based Randomised CWS with the best results in the literature

Table 4.3 presents the comparison of average results between IRP-based Randomised CWS with the best solutions in ABLS (Archetti et al., 2007) and ALNS (Leandro et al., 2012). Each result for all sets of instances show the average of five instances in a set which has ten instances sets and six instances sets in each time period $p = 3\ and\ p = 6$, respectively. The average total costs are provided by IRP-based Randomised CWS and the average percentage gap between IRP-based Randomised CWS heuristic with

respect to the best-known solution in literature ABLS (Archetti et al., 2007) and ALNS (Leandro et al., 2012) are presented in the bold numbers.

It can be seen from Table 4.3 below, the IRP-based Randomised CWS heuristic is able to find feasible solutions for all instances with up to 50 clients, but they have higher than the solutions are obtained by ABLS. For low inventory holding cost and $p = 3$, this chapter proposed algorithm are on average 2.97% under ML policy and 3.39% under OU policy higher than those obtained by ABLS with the average computation time less than 15 minutes. In the time period $p = 3$ with high inventory holding cost, the solutions from Archetti et al. (2007) proposed method have 3.70% and 3.76% better than the solutions solved by Randomised CWS under ML/OU policies, respectively. The IRP-based Randomised CWS heuristic under both ML/OU inventory replenishment policies required 45 minutes for solving the instances with up to 50 customers.

When the time horizon is equal to 6, all instances with up to 30 customers can be solved in less than one hour for low/high inventory holding costs. The average results obtained by ABLS are on average 4.55%, 4.33%, 4.04% and 3.36% better than those obtained by Randomised CWS heuristic for the IRP_ML and the IRP_OU with both the low and high holding costs, respectively. This is sensible that ABLS is the exact method in which it always provides the optimal solution. Nonetheless, Randomised CWS algorithm is heuristic, which can find good solution but it cannot obtain optimal solution. However, Randomised CWS heuristic is able to find good solution with a less computation time when the size of instances is become larger.

Additionally, this chapter proposed algorithm is able to give good solutions but these results are slightly worse than those obtained by ALNS (Leandro et al., 2012). It is only reasonable that Leandro's work adopted meta-heuristics to solve the problem, whereas heuristics are used in this work. The current best knowledge shows that meta-heuristics are more effective in solving NP-hard problem than heuristics. On the other hand, the IRP-based Randomised CWS has taken less run time to obtain the solutions. It was very fast, taking on average less than 1,100s for all instances of the both time periods. Beside that the computation time of the solution generated by ALNS (Leandro et al., 2012) with the maximum was about 2,400s.

Low inventory holding cost and Time period = 3								High inventory holding cost and Time period = 3							
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based RandomisedCWS		Gap (%)	Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based RandomisedCWS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A		(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	1275.86	1275.99	6.08	0.01	1279.20	3.10	0.26	absn05	2187.30	2201.20	5.82	0.64	2241.04	3.97	2.46
absn10	1910.93	1911.08	18.59	0.01	1986.81	14.65	3.97	absn10	4337.97	4339.35	17.13	0.03	4456.98	15.55	2.74
absn15	2207.77	2208.57	44.69	0.04	2214.29	38.09	0.30	absn15	5435.80	5438.80	46.34	0.06	5459.13	41.67	0.43
absn20	2665.58	2675.58	98.05	0.38	2882.35	79.08	8.13	absn20	7225.70	7262.24	93.08	0.51	7270.10	87.88	0.61
absn25	2987.90	2996.77	171.38	0.30	2998.79	201.55	0.36	absn25	7501.07	9007.11	164.36	20.08	9045.38	240.23	20.59
absn30	3292.93	3330.77	331.45	1.15	3447.70	490.98	4.70	absn30	10918.31	10941.55	315.76	0.21	11002.28	559.34	0.77
absn35	3448.84	3495.04	495.46	1.34	3525.52	1023.57	2.22	absn35	11411.67	11472.14	575.54	0.53	11518.85	1123.33	0.94
absn40	3703.82	3736.05	793.99	0.87	3910.74	1334.44	5.59	absn40	12541.06	12632.72	801.64	0.73	12644.99	1436.99	0.83
absn45	3867.48	3886.26	1405.72	0.49	3965.62	1662.77	2.54	absn45	13865.34	13928.10	1197.02	0.45	14504.25	1802.22	4.61
absn50	4327.16	4366.43	1719.41	0.91	4396.36	2202.22	1.60	absn50	15410.82	15512.49	2109.81	0.66	15878.17	2466.45	3.03
Average	**2968.83**	**2988.25**	**508.48**	**0.55**	**3060.74**	**705.05**	**2.97**	**Average**	**9083.50**	**9273.57**	**532.65**	**2.39**	**9402.12**	**777.76**	**3.70**
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_OU-based RandomisedCWS		Gap (%)	Name of insnatce	ABLS	ALNS		Gap (%)	IRP_OU-based RandomisedCWS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A		(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	1418.76	1418.75	10.71	0.00	1493.04	3.93	5.24	absn05	2354.18	2354.17	9.45	0.00	2441.04	4.89	3.69
absn10	2228.67	2228.66	35.28	0.00	2256.81	16.01	1.26	absn10	4690.46	4691.02	34.62	0.01	4696.98	18.77	0.14
absn15	2493.47	2493.47	99.32	0.00	2494.29	48.88	0.03	absn15	5736.91	5740.66	97.09	0.07	5979.13	56.44	4.22
absn20	3053.02	3055.58	239.76	0.08	3082.35	89.99	0.96	absn20	7619.91	7626.94	224.24	0.09	7930.10	93.06	4.07
absn25	3451.15	3451.86	572.28	0.02	3598.79	237.81	4.28	absn25	9460.75	9476.04	446.47	0.16	10025.38	296.05	5.97
absn30	3643.22	3645.70	1072.47	0.07	3847.70	521.11	5.61	absn30	11320.65	11354.66	890.16	0.30	11356.28	601.66	0.31
absn35	3846.87	3850.83	1439.28	0.10	4025.52	1177.33	4.64	absn35	11828.82	11848.90	1600.56	0.17	12109.85	1205.55	2.38
absn40	4125.70	4140.16	2755.72	0.35	4290.74	1409.50	4.00	absn40	13011.46	13043.95	2767.76	0.25	13494.99	1579.98	3.72
absn45	4270.61	4283.33	3417.87	0.30	4325.62	1708.12	1.29	absn45	14317.82	14392.04	3010.08	0.52	15204.25	1880.44	6.19
absn50	4810.87	4841.26	2675.47	0.63	5126.36	2300.27	6.56	absn50	15948.78	16077.86	2987.26	0.81	17058.17	2503.23	6.96
Average	**3334.23**	**3340.96**	**1231.82**	**0.16**	**3454.12**	**751.30**	**3.39**	**Average**	**9628.97**	**9660.62**	**1206.77**	**0.24**	**10029.62**	**824.01**	**3.76**
Low inventory holding cost and Time period = 6								High inventory holding cost and Time period = 6							
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based RandomisedCWS		Gap (%)	Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based RandomisedCWS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A		(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	3136.90	3288.30	12.90	4.83	3352.54	10.56	6.87	absn05	5354.20	5514.49	12.20	2.99	5531.58	13.66	3.31
absn10	4612.50	4751.96	54.26	3.02	4764.47	58.07	3.29	absn10	8601.92	8757.00	56.27	1.80	8827.21	98.05	2.62
absn15	5418.55	5546.96	160.81	2.37	5565.34	188.03	2.71	absn15	11543.04	11681.58	154.74	1.20	12136.19	279.45	5.14
absn20	6625.35	6762.36	381.15	2.07	7179.09	665.02	8.36	absn20	14602.14	14750.66	348.11	1.02	15598.48	803.65	6.82
absn25	7261.77	7443.39	599.87	2.50	7458.28	1390.50	2.71	absn25	16913.97	17113.84	703.10	1.18	17892.42	1496.34	5.78
absn30	7710.01	7835.02	1415.08	1.62	7967.26	3001.60	3.34	absn30	20410.65	20547.02	1597.86	0.67	20882.21	3205.66	2.31
Average	**5794.18**	**5938.00**	**437.35**	**2.74**	**6047.83**	**885.63**	**4.55**	**Average**	**12904.32**	**13060.77**	**478.71**	**1.48**	**13478.02**	**982.80**	**4.33**
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_OU-based RandomisedCWS		Gap (%)	Name of insnatce	ABLS	ALNS		Gap (%)	IRP_OU-based RandomisedCWS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A		(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	3299.98	3299.97	20.92	0.00	3452.54	13.33	4.62	absn05	5538.02	5538.91	22.82	0.02	5631.58	14.78	1.69
absn10	4832.89	4832.87	95.88	0.00	4964.47	68.01	2.72	absn10	8872.41	8872.41	106.84	0.00	9027.21	106.77	1.74
absn15	5566.39	5582.80	337.70	0.29	5885.34	220.45	5.73	absn15	11721.82	11738.50	370.67	0.14	12236.19	311.15	4.39
absn20	6833.29	6857.90	797.63	0.36	7279.09	699.12	6.52	absn20	14863.86	14883.49	1021.13	0.13	15598.48	898.88	4.94
absn25	7454.15	7487.80	1610.54	0.45	7498.28	1405.50	0.59	absn25	17170.82	17223.47	2221.46	0.31	18092.42	1540.50	5.37
absn30	7847.39	7888.56	3031.66	0.52	8167.26	3109.20	4.08	absn30	20657.28	20752.32	3399.49	0.46	21072.21	3409.22	2.01
Average	**5972.35**	**5991.65**	**982.39**	**0.27**	**6207.83**	**919.27**	**4.04**	**Average**	**13137.37**	**13168.18**	**1190.40**	**0.18**	**13609.68**	**1046.88**	**3.36**

Table 4.3 : Comparison of average results between the IRP-based Randomised CWS heuristic and ABLS, ALNS

4.4.2. IRPT-based Randomised CWS heuristic

As from our knowledge, transhipment is a service between the two points (source and the demand) in which each supply and destination can act as an intermediate point through which products can be transhipped. From one point (product is not sent directly from the depot) to the final destination of demand. This later transhipment policy is an alternative method, used for minimising total cost. Therefore, this thesis applied transhipment for the IRP-based Randomised CWS heuristic. To evaluate the performance of this method, the proposed algorithm generates and solves the instances from the benchmark problem from Leandro et al. (2012b). In the case of lateral transhipment, there are only a few papers published with reported solutions since the problem was introduced by Leandro et al. (2012b). For this reason, this study has taken data from Leandro et al. (2012b) and compared the solution of IRPT-based Randomised CWS heuristic with their work.

Table 4.4 gives the average costs breakdown of the total costs for the IRPT under ML and OU inventory replenishment policies with low/high inventory holding costs in the time horizon equal to three and six as follows: the inventory holding (HC), transhipment (TrC), routing (RC) and total costs (TC). The average total costs are provided by IRPT-based Randomised CWS heuristic under ML replenishment policy are slightly less than those costs obtained when using the OU replenishment policy. Using OU policy, the quantity of products supplied does not reach the maximum inventory level, so if extra demand is encountered during the same time period, penalty and transhipment costs are charge.

Low inventory holding cost and Time period = 3									
Name of Instance		IRPT_ML-based Randomised CWS heuristic				IRPT_OU-based Randomised CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	87.38	0.00	667	754.38	91.38	0.00	663	754.38
2	absn10	210.70	100.90	1278	1589.60	226.90	106.05	1338	1670.95
3	absn15	341.30	145.89	1415	1902.19	349.60	152.68	1499	2001.28
4	absn20	482.55	181.14	1722	2385.69	496.52	230.34	1660	2386.86
5	absn25	565.70	203.13	1999	2767.83	578.95	260.01	1933	2771.96
6	absn30	667.40	231.15	2100	2998.55	687.80	315.20	2390	3393.00
7	absn35	728.10	274.43	2209	3211.53	738.66	359.30	2483	3580.96
8	absn40	809.13	300.99	2301	3411.12	826.40	398.50	2555	3779.90
9	absn45	889.99	334.28	2414	3638.27	957.10	416.38	2708	4081.48
10	absn50	950.15	373.10	2771	4094.25	1000.27	490.39	2995	4485.66
	Average	**573.24**	**214.50**	**1887.60**	**2675.34**	**595.36**	**272.89**	**2022.40**	**2890.64**
High inventory holding cost and Time period = 3									
Name of Instance		IRPT_ML-based Randomised CWS heuristic				IRPT_OU-basedRandomised CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	866.90	0.00	809	1675.90	899.90	0.00	890	1789.90
2	absn10	2228.12	111.10	1651	3990.22	2321.11	133.50	1653	4107.61
3	absn15	2796.40	160.50	2181	5137.90	2899.30	190.70	2190	5280.00
4	absn20	3696.34	217.15	2798	6711.49	3691.26	273.20	3078	7042.46
5	absn25	4913.44	289.05	3421	8623.49	4999.96	396.76	3596	8992.72
6	absn30	5998.07	363.33	4204	10565.40	6214.56	432.29	4225	10871.85
7	absn35	6663.23	390.43	4335	11388.66	6609.69	466.54	4305	11381.23
8	absn40	7881.22	417.77	4663	12961.99	7757.89	532.50	4891	13181.39
9	absn45	8299.90	479.09	5164	13942.99	8232.20	565.80	5614	14412.00
10	absn50	9376.55	499.79	5415	15291.34	9445.25	709.20	6491	16645.45
	Average	**5272.02**	**292.82**	**3464.10**	**9028.94**	**5307.11**	**370.05**	**3693.30**	**9370.46**
Low inventory holding cost and Time period = 6									
Name of Instance		IRPT_ML-based Randomised CWS heuristic				IRPT_OU-based Randomised CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	380.90	0.00	2177	2557.90	399.99	0.00	2167	2566.99
2	absn10	598.80	129.50	3369	4097.30	600.10	134.50	3230	3964.60
3	absn15	629.11	234.56	4022	4885.67	637.44	247.79	4005	4890.23
4	absn20	701.09	301.10	5058	6060.19	731.21	359.99	5278	6369.20
5	absn25	913.02	377.66	5553	6843.68	916.76	420.50	5501	6838.26
6	absn30	995.00	403.23	6245	7643.23	1033.20	495.95	6460	7989.15
	Average	**702.99**	**241.01**	**4404.00**	**5348.00**	**719.78**	**276.46**	**4440.17**	**5436.41**
High inventory holding cost and Time period = 6									
Name of Instance		IRPT_ML-based Randomised CWS heuristic				IRPT_OU-based Randomised CWS heuristic			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	2398.90	0.00	2382	4780.90	2399.54	0.00	2395	4794.54
2	absn10	3932.19	103.85	3908	7944.04	3939.39	113.88	4029	8082.27
3	absn15	5504.50	201.01	5288	10993.51	5656.98	204.40	5256	11117.38
4	absn20	7544.40	289.88	6289	14123.28	6973.20	389.11	6969	14331.31
5	absn25	8096.55	329.45	8390	16816.00	8101.01	445.07	8419	16965.08
6	absn30	10318.90	544.45	10033	20896.35	10394.66	587.35	9999	20981.01
	Average	**6299.24**	**244.77**	**6048.33**	**12592.35**	**6244.13**	**289.97**	**6177.83**	**12711.93**

Table 4.4 : Average result of cost breakdown of the IRPT-based Randomised CWS under ML/OU, p=3 and p= 6

It can be seen in Table 4.4, the average HC for the set with low inventory holding costs of IRPT-based Randomised CWS heuristic under both ML and OU replenishment policies are lower than the average RC. Whereas, the average HC for the set with high inventory holding costs are slightly more than the average RC. When the time horizon is 3, all sets of instances with up to 50 customers give lower the average RC than all

sets of instances with up to 30 customers in six time periods. The average TrC for all sets of instances in three time and six time periods are not significantly different. However, adopting the OU policy will result in higher transhipment cost than that using the ML policy for all instances. The reason is that, when using the OU replenishment policy, supplier is not fully served the customer inventory that can lead to extra demand during the period, in turn, transhipment cost may be occurred.

4.4.2.1. Comparison of the results of IRPT-based Randomised CWS with the best results in the IRPT-based CWS

To evaluate the performance of this chapter proposed heuristic, this experimental has used the instances of the IRP generated and solved to optimality by Randomised CWS. These are adapted to account for transhipments, as described in Section 4.3.3. For the cases with transhipment, there are no pervious presented results solved by Randomised CWS heuristic.

Low inventory holding cost and Time period = 3						
Name of insnatce	**IRPT_ML-based CWS (A)**	**IRPT_ML-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**	**IRPT_OU-based CWS (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**
absn05	776.38	754.38	2.92	776.38	754.38	2.92
absn10	1605.71	1588.46	1.09	1736.78	1700.73	2.12
absn15	1962.67	1900.75	3.26	2051.57	1975.93	3.83
absn20	2498.99	2381.66	4.93	2498.99	2386.86	4.70
absn25	2796.79	2735.93	2.22	2796.79	2749.03	1.74
absn30	3082.55	2990.72	3.07	3382.55	3370.70	0.35
absn35	3296.03	3211.45	2.63	3596.03	3548.75	1.33
absn40	3499.58	3400.20	2.92	3799.58	3778.48	0.56
absn45	3682.37	3562.32	3.37	4082.37	4081.47	0.02
absn50	4077.12	4059.30	0.44	4597.12	4582.29	0.32
Average	**2727.82**	**2658.52**	**2.68**	**2931.81**	**2892.86**	**1.79**
High inventory holding cost and Time period = 3						
Name of insnatce	**IRPT_ML-based CWS (A)**	**IRPT_ML-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**	**IRPT_OU-based CWS (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**
absn05	1667.44	1660.54	0.42	1785.91	1782.61	0.18
absn10	3999.05	3998.84	0.01	4132.11	4106.76	0.62
absn15	5061.54	5061.19	0.01	5362.62	5260.77	1.94
absn20	6868.82	6854.22	0.21	7407.42	7041.41	5.20
absn25	8618.38	8568.38	0.58	9103.38	8979.99	1.37
absn30	10591.63	10555.20	0.35	10903.66	10877.14	0.24
absn35	11397.84	11359.07	0.34	11409.36	11380.55	0.25
absn40	13185.09	12968.37	1.67	13458.98	13136.81	2.45
absn45	14214.48	13913.98	2.16	14595.90	14400.25	1.36
absn50	15826.23	15281.54	3.56	16727.93	16672.17	0.33
Average	**9143.05**	**9022.13**	**0.93**	**9488.73**	**9363.85**	**1.40**
Low inventory holding cost and Time period = 6						
Name of insnatce	**IRPT_ML-based CWS (A)**	**IRPT_ML-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**	**IRPT_OU-based CWS (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**
absn05	2554.46	2554.40	0.00	2564.46	2564.40	0.00
absn10	4099.42	4095.17	0.10	3979.42	3975.17	0.11
absn15	4817.97	4810.23	0.16	4881.02	4876.96	0.08
absn20	6089.20	6059.01	0.50	6395.20	6359.01	0.57
absn25	6812.54	6809.26	0.05	6842.54	6811.26	0.46
absn30	7758.82	7629.87	1.69	7992.72	7987.81	0.06
Average	**5355.40**	**5326.32**	**0.42**	**5442.56**	**5429.10**	**0.21**
High inventory holding cost and Time period = 6						
Name of insnatce	**IRPT_ML-based CWS (A)**	**IRPT_ML-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**	**IRPT_OU-based CWS (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**
absn05	4749.01	4748.99	0.00	4766.69	4762.79	0.08
absn10	8015.25	7940.74	0.94	8038.54	8034.79	0.05
absn15	10983.76	10954.53	0.27	11106.95	11100.44	0.06
absn20	14205.09	14155.34	0.35	14326.66	14302.05	0.17
absn25	16991.51	16780.26	1.26	16975.62	16966.48	0.05
absn30	20838.57	20737.41	0.49	21097.34	20994.48	0.49
Average	**12630.53**	**12552.88**	**0.55**	**12718.63**	**12693.51**	**0.15**

Table 4.5 : Comparison between the IRP-based CWS and the IRP-based Randomised CWS heuristic under ML/OU, p=3 and p= 6

Table 4.5 reports the average results obtained from the different proposed approaches; the IRP-based CWS and the IRPT-based Randomised CWS heuristics. The low/high inventory holding costs and the three time and six time periods are considered for all sets of instances. These are categorised as: the low/high inventory holding costs for the IRPT_ML- based Randomised CWS and the IRPT_OU-based Randomised CWS with time horizon is equal to 3 and 6. The proposed Randomised CWS algorithm is able to find better results than CWS heuristic for all of instances on all sets. The solutions of the IRPT-based Randomised CWS under ML and OU policies are on average 2.68% and 1.79% for low holding costs, and 0.93% and 1.40% for high inventory holding costs, respectively in the time period is equal to 3. When the time horizon is 6, Randomised CWS have slightly better average total costs than CWS. They are on average 0.42% for IRPT_ML and 0.21% for IRPT_OU with low inventory holding, while when the high inventory holding costs, they have 0.55% and 0.15% for IRPT under ML and OU policies, respectively. Which can be seen from the average improvement percentage, the IRPT-based Randomised CWS can lead to improve the solution from the IRPT-based CWS, which means that it is able to show lower average total costs compared to CWS heuristic. Thus, applied biased randomisation with CWS heuristic can lead to an increase performance of this algorithm especially when instance size is larger.

4.4.2.2. Comparison of the results of IRPT-based Randomised CWS with the lower bound and the best results in the literature

Table 4.6 reports the average results of IRPT-based Randomised CWS heuristic compared to the lower bound (LB) and upper bound (UB) solutions truncated by CPLEX (Leandro et al., 2012), The average solutions solved by Randomised CWS under both ML/OU policies with the Low/High inventory holding costs and in the time periods are equal 3 and 6 are also, presented in Table 4.6. These solutions represent the average total costs of IRPT for all sets of instances. This chapter results are compared to those of LB, UB and ALNS on these IRPT under ML and OU replenishment policies.

The solutions obtained by this chapter proposed method show higher total costs than those obtained by LB while it required less running time. However, the IRPT-based Randomised CWS heuristic is able to find good solution in which most of the instances on all sets are better than UB (see red numbers). These solutions are on average 11.28% and 18.09% better than those percentage gaps of UB solution truncated by CPLEX for the IRPT under ML and OU replenishment policies with the low inventory holding

costs in three time periods, respectively. For the high inventory holding costs and the three time periods, they have better on all seven sets of instances handle by ML policy and up to 0.20% slightly better than those UB %gaps for the instances set under the OU inventory replenishment policy.

For the IRPT_ML, the solutions solved by this chapter proposed method are better on up to five sets of instances in six time periods. They are on average 28.83% better than those %gap of UB solved by CPLEX with the low inventory holding costs, and up to 10.55% better for the same set with high holding costs. While the IRPT_OU in the time period is equal to 6, the results have also, better on up to five sets of instances. For low inventory holding costs, they are on average 28.92% better than those %gaps of UB, and up to 11.39% better for the same set with high inventory holding costs. It can be seen from percentage gaps the performance of this chapter proposed algorithm is better when instances become a large size and more complex.

Table 4.6 also shows that the IRPT-based Randomised CWS heuristic is required less computation time than the method used in the literature. When the time period is equal to 3 and low/high inventory holding costs, all instances for the IRPT_ML with up to 50 customers can be solved in less than 15 minutes, while all instances for the IRPT_OU with the same customers cloud be solved in about 45 minutes. For all instances with up to 30 customers with low/high inventory holding costs and the six time periods, the IRPT under ML policy is required less than 15 minutes for solving, whereas the IRPT_OU is required about 25 minutes using the same instances sets.

According to the average total costs report in the table, most instances obtained by Randomised CWS under both ML or OU policies are slightly higher than those total costs solved by ALNS (Leandro et al., 2012). The solutions costs obtained by ALNS are better than the solutions presented by this chapter proposed algorithm, they are on average 16.85% and 4.08% lower for the IRPT_OU, low/high inventory holding costs in time period is equal to 3. For the IRPT_ML with high inventory holding cost and three time periods, ALNS has found better results than those solved by Randomised CWS with they are on average 2.78% better on all sets of instances. Whereas, the IRPT_ML-based Randomised CWS with low inventory holding cost in time period = 3, they are better on average total costs than those obtained by ALNS for all instances sets.

In the time horizon is equal to 6 and low/high inventory costs, the solution costs for the IRPT_ML-based Randomised CWS are higher than the average solution costs solved by the proposed method by Leandro et al. (2012), the percentage gaps are on average 28.83% and 28.92%, respectively on all sets of instances. While, the IRPT_OU-based Randomised CWS provided better results than those found by ALNS, they are on average 28.92% and 11.39% lower on all sets of instances for low/high inventory holding costs, respectively. Additionally, some results show lower total costs than those results by ALNS (see green number) and it used less run time to find the solutions.

Low inventory holding cost and Time period = 3												High inventory holding cost and Time period = 3									
	LB	UB	Time (s)	Gap (%)	ALNS		Gap (%)	IRPT_ML-based RandomisedCWS		Gap (%)		LB	UB	Time (s)	Gap (%)	ALNS		Gap (%)	IRPT_ML-based RandomisedCWS		Gap (%)
	(A)	(C)		(C-A)/A	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A		(A)	(C)		(C-A)/A	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	741.76	741.76	0.29	0.00	744.80	5.28	0.41	754.38	3.77	1.70		1660.26	1660.26	0.72	0.00	1660.26	5.98	0.00	1660.54	4.66	0.02
absn10	1577.23	1577.30	2.64	0.00	1586.91	14.52	0.61	1588.46	12.65	0.71		3998.8	3999.02	3.17	0.01	**4011.81**	15.36	0.33	**3998.84**	11.34	0.00
absn15	1839.88	1840.06	58.46	0.01	1849.83	30.64	0.54	1900.75	28.88	3.31		5054.12	5054.58	35.86	0.01	**5061.92**	33.76	0.15	5061.19	29.67	0.14
absn20	2232.38	2278.04	1143.92	2.05	2290.55	56.74	2.61	2381.66	48.20	6.69		6752.82	6822.95	1134.22	1.04	**6869.26**	57.73	1.72	6854.22	51.32	1.50
absn25	2367.55	2657.38	1920.44	12.24	2579.18	92.04	8.94	2735.93	83.23	15.56		8342.03	8714.21	1496.09	4.46	8562.59	91.26	2.64	**8568.38**	85.11	2.71
absn30	2735.52	3116.05	2006.29	13.91	2985.99	165.20	9.16	**2990.72**	122.66	9.33		10319.56	10716.07	1586.59	3.84	**10557.63**	159.13	2.31	**10555.20**	140.98	2.28
absn35	2742.38	4034.69	1200.07	47.12	**3448.17**	248.74	25.74	**3211.45**	222.55	17.10		10689.39	11511.05	1191.40	7.69	11309.46	282.47	5.80	**11359.07**	243.87	6.26
absn40	2799.85	4480.11	1992.43	60.01	3361.80	348.93	20.07	**3400.20**	301.09	21.44		11573.32	14329.19	1643.60	23.81	12165.28	360.83	5.11	**12968.37**	341.22	12.05
absn45	3034.49	5110.99	1613.87	68.43	**3697.61**	461.71	21.85	**3562.32**	381.78	17.39		12979.04	15822.42	2143.09	21.91	13699.96	627.96	5.55	**13913.98**	501.44	7.20
absn50	3396.46	9172.72	2301.51	170.07	**4071.09**	760.30	19.86	**4059.30**	604.65	19.52		14406.63	18144.64	2461.43	25.95	15004.18	939.87	4.15	**15281.54**	737.40	6.07
Average	**2346.75**	**3500.91**	**1223.99**	**37.38**	**2661.59**	**218.41**	**10.98**	**2658.52**	**180.95**	**11.28**		**8577.60**	**9677.44**	**1169.62**	**8.87**	**8890.24**	**257.44**	**2.78**	**9022.13**	**214.70**	**3.83**
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based RandomisedCWS**		**Gap (%)**		**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based RandomisedCWS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**		**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	745.39	745.39	0.06	0.00	745.39	6.56	0.00	**754.38**	3.97	1.21		1664.38	1664.38	0.07	0.00	1664.38	8.06	0.00	1782.61	5.19	7.10
absn10	1616.08	1616.11	2.29	0.00	1617.51	29.55	0.09	1700.73	16.56	5.24		4043.83	4043.94	2.40	0.00	4043.94	30.12	0.00	4106.76	17.09	1.56
absn15	1851.02	1851.13	19.58	0.01	1864.70	82.24	0.74	1975.93	59.89	6.75		5069.04	5069.51	20.64	0.01	5116.21	74.63	0.93	5260.77	66.49	3.78
absn20	2274.24	2339.63	557.58	2.88	**2442.44**	183.28	7.40	2386.86	111.30	4.95		6818.37	6877.20	684.51	0.86	6927.36	151.21	1.60	7041.41	123.19	3.27
absn25	2421.25	2710.08	1887.00	11.93	2724.67	389.10	12.53	2749.03	192.91	13.54		8422.54	8611.29	2100.71	2.24	8754.03	350.23	3.94	8979.99	256.77	6.62
absn30	2812.98	3123.22	2037.20	11.03	3341.49	635.79	18.79	3370.70	433.75	19.83		10332.37	10765.01	1362.58	4.19	10867.17	521.52	5.18	10877.14	521.01	5.27
absn35	2772.64	4316.84	1339.51	55.69	3522.66	895.11	27.05	**3548.75**	781.14	27.99		10695.61	11608.97	1622.33	8.54	11367.04	930.08	6.28	**11380.55**	802.33	6.40
absn40	2828.15	4531.14	1760.32	60.22	**3795.60**	1577.21	34.21	**3778.48**	1155.02	33.60		11588.44	13117.59	1427.80	13.20	12563.16	1577.64	8.41	**13136.81**	1229.50	13.36
absn45	3090.45	4540.60	1511.04	46.92	4078.89	2350.23	31.98	**4081.47**	1697.34	32.07		13069.51	16468.68	1427.33	26.01	13921.34	2244.01	6.52	**14400.25**	1905.95	10.18
absn50	3376.61	6491.95	1610.72	92.26	4581.07	2898.06	35.67	**4582.29**	2195.45	35.71		14418.41	17326.09	2009.60	20.17	15560.06	3375.50	7.92	**16672.17**	2556.71	15.63
Average	**2378.88**	**3226.61**	**1072.53**	**28.09**	**2871.44**	**904.71**	**16.85**	**2892.86**	**664.73**	**18.09**		**8612.25**	**9555.27**	**1065.80**	**7.52**	**9078.47**	**926.30**	**4.08**	**9363.85**	**748.42**	**7.32**
Low inventory holding cost and Time period = 6												**High inventory holding cost and Time period = 6**									
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_ML-based RandomisedCWS**		**Gap (%)**		**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_ML-based RandomisedCWS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**		**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	2554.37	2554.45	2.62	0.00	**2558.37**	10.90	0.16	2554.40	5.80	0.00		4741.97	4742.19	2.23	0.00	4748.31	13.28	0.13	4748.99	9.90	0.15
absn10	3443.84	4056.36	1660.45	17.79	4095.10	36.70	18.91	4095.17	17.71	18.91		7429.95	8015.35	2035.50	7.88	**7961.72**	37.91	7.16	**7940.74**	25.30	6.87
absn15	3619.79	5114.79	976.65	41.30	**4834.73**	80.68	33.56	**4810.23**	59.33	32.89		9933.52	11262.90	813.05	13.38	10949.14	88.45	10.22	**10954.53**	78.28	10.28
absn20	4385.61	9880.24	1367.07	125.29	6020.83	174.24	37.29	**6059.01**	149.23	38.16		12321.4	17437.75	1166.17	41.52	14152.04	179.70	14.86	**14155.34**	166.61	14.88
absn25	4679.71	13786.58	1769.39	194.60	6808.40	295.30	45.49	**6809.26**	241.86	45.51		14333.9	21962.55	1937.04	53.22	16320.18	329.51	13.86	**16780.26**	294.70	17.07
absn30	5547.73	14296.67	2336.32	157.70	7466.89	671.24	34.59	**7629.87**	500.98	37.53		18182.52	29215.44	2064.83	60.68	20235.50	787.47	11.29	**20737.41**	550.55	14.05
Average	**4038.51**	**8281.52**	**1352.08**	**89.45**	**5297.39**	**211.51**	**28.33**	**5326.32**	**162.49**	**28.83**		**11157.21**	**15439.36**	**1336.47**	**29.45**	**12394.48**	**239.39**	**9.59**	**12552.88**	**187.56**	**10.55**
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based RandomisedCWS**		**Gap (%)**		**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based RandomisedCWS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**		**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	2561.77	2561.84	1.35	0.00	**3299.97**	16.23	28.82	2564.40	11.40	0.10		4759.29	4759.54	2.21	0.01	4760.94	14.98	0.03	4762.79	12.22	0.07
absn10	3524.75	4054.39	2221.58	15.03	**4832.87**	70.29	37.11	**3975.17**	47.20	12.78		7495.05	8077.14	1477.52	7.77	**8038.53**	65.29	7.25	8034.79	48.48	7.20
absn15	3880.76	5042.41	1029.67	29.93	**5582.80**	208.08	43.86	**4876.96**	188.10	25.67		10037.35	11225.20	1109.10	11.83	11027.72	156.07	9.87	11100.44	202.45	10.59
absn20	4396.94	7418.80	1382.76	68.73	**6857.90**	491.31	55.97	**6359.01**	390.78	44.62		12332.22	16778.25	1207.34	36.05	14278.32	442.43	15.78	**14302.05**	399.89	15.97
absn25	4641.33	10566.48	1834.25	127.66	**7487.80**	805.16	61.33	**6811.26**	583.55	46.75		14262.97	20807.23	1540.39	45.88	16867.70	906.47	18.26	**16966.48**	606.67	18.95
absn30	5562.96	13917.82	1590.49	150.19	7888.56	1650.37	41.81	**7987.81**	1100.45	43.59		18169.95	26411.34	1951.69	45.36	20492.18	1718.08	12.78	**20994.48**	1330.04	15.55
Average	**4094.75**	**7260.29**	**1343.35**	**65.26**	**5991.65**	**540.24**	**44.82**	**5429.10**	**386.91**	**28.92**		**11176.14**	**14676.45**	**1214.71**	**24.48**	**12577.57**	**550.55**	**10.66**	**12693.51**	**433.29**	**11.39**

Table 4.6 : Comparison of average results for the IRPT-based Randomised CWS under ML/OU, p=3, p=6 and literatures

4.4.2.3. *Comparison of the results of IRPT-based Randomised CWS with IRP-based Randomised CWS Heuristics*

Tables 4.7 shows the comparison of average solutions between IRP-based Randomised CWS and IRPT-based Randomised CWS heuristics under both the OU and ML replenishment policies with the time periods (p) are equal to three and six respectively. This table shows the average results aggregated in four classes of instances. The first class contains the instances in the time period $p = 3$ and low inventory holding cost. The second class contains the instances with time period $p = 3$ and high inventory holding cost, whilst the third class the instances with time period $p = 6$ and low inventory holding cost. The fourth class contains the instances with time period $p = 6$ and high inventory holding cost.

The results solved by Randomised CWS heuristic with the low or high inventory holding costs and $p = 3, p = 6$ for the IRP and IRPT under the ML/OU replenishment policies are represented as IRP_ML-based Randomised CWS, IRP_OU-based Randomised CWS, IRPT_ML-based Randomised CWS and IRPT_OU-based Randomised CWS heuristics. The first column represents the instance name and the second, the third, the fifth and the sixth columns give the IRP_ML-based Randomised CWS and IRPT_ML-based Randomised CWS solutions, respectively. Then the fourth and the seventh column present the percentage gaps of the IRPT-based Randomised CWS compared to the IRP-based Randomised CWS under ML/OU policies and low and high inventory holding cots, respectively.

Low inventory holding cost and Time period = 3				High inventory holding cost and Time period = 3		
Name of insnatce	IRP_ML-based Randomised CWS (A)	IRPT_ML-based Randomised CWS (B)	Improvement (%) (A-B)/B	IRP_ML-based Randomised CWS (A)	IRPT_ML-based Randomised CWS (B)	Improvement (%) (A-B)/B
absn05	1279.20	754.38	69.57	2241.04	1660.54	34.96
absn10	1986.81	1588.46	25.08	4456.98	3998.84	11.46
absn15	2214.29	1900.75	16.50	5459.13	5061.19	7.86
absn20	2882.35	2381.66	21.02	7270.10	6854.22	6.07
absn25	2998.79	2735.93	9.61	9045.38	8568.38	5.57
absn30	3447.70	2990.72	15.28	11002.28	10555.20	4.24
absn35	3525.52	3211.45	9.78	11518.85	11359.07	1.41
absn40	3910.74	3400.20	15.02	12644.99	12968.37	-2.49
absn45	3965.62	3562.32	11.32	14504.25	13913.98	4.24
absn50	4396.36	4059.30	8.30	15878.17	15281.54	3.90
Average	**3060.74**	**2658.52**	**20.15**	**9402.12**	**9022.13**	**7.72**
Name of insnatce	**IRP_OU-based Randomised CWS (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**	**IRP_OU-based Randomised CWS (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**
absn05	1493.04	754.38	97.92	2441.04	1782.61	36.94
absn10	2256.81	1700.73	32.70	4696.98	4106.76	14.37
absn15	2494.29	1975.93	26.23	5979.13	5260.77	13.66
absn20	3082.35	2386.86	29.14	7930.10	7041.41	12.62
absn25	3598.79	2749.03	30.91	10025.38	8979.99	11.64
absn30	3847.70	3370.70	14.15	11356.28	10877.14	4.41
absn35	4025.52	3548.75	13.43	12109.85	11380.55	6.41
absn40	4290.74	3778.48	13.56	13494.99	13136.81	2.73
absn45	4325.62	4081.47	5.98	15204.25	14400.25	5.58
absn50	5126.36	4582.29	11.87	17058.17	16672.17	2.32
Average	**3454.12**	**2892.86**	**27.59**	**10029.62**	**9363.85**	**11.07**
Low inventory holding cost and Time period = 6				**High inventory holding cost and Time period = 6**		
Name of insnatce	**IRP_ML-based Randomised CWS (A)**	**IRPT_ML-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**	**IRP_ML-based Randomised CWS (A)**	**IRPT_ML-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**
absn05	3352.54	2554.40	31.25	5531.58	4748.99	16.48
absn10	4764.47	4095.17	16.34	8827.21	7940.74	11.16
absn15	5565.34	4810.23	15.70	12136.19	10954.53	10.79
absn20	7179.09	6059.01	18.49	15598.48	14155.34	10.20
absn25	7458.28	6809.26	9.53	17892.42	16780.26	6.63
absn30	7967.26	7629.87	4.42	20882.21	20737.41	0.70
Average	**6047.83**	**5326.32**	**15.95**	**13478.02**	**12552.88**	**9.33**
Name of insnatce	**IRP_OU-based CWS heuristic (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**	**IRP_OU-based Randomised CWS (A)**	**IRPT_OU-based Randomised CWS (B)**	**Improvement (%) (A-B)/B**
absn05	3452.54	2564.40	34.63	5631.58	4762.79	18.24
absn10	4964.47	3975.17	24.89	9027.21	8034.79	12.35
absn15	5885.34	4876.96	20.68	12236.19	11100.44	10.23
absn20	7279.09	6359.01	14.47	15598.48	14302.05	9.06
absn25	7498.28	6811.26	10.09	18092.42	16966.48	6.64
absn30	8167.26	7987.81	2.25	21072.21	20994.48	0.37
Average	**6207.83**	**5429.10**	**17.83**	**13609.68**	**12693.51**	**9.48**

Table 4.7 : Comparison average results between the IRP and IRPT –based Randomised CWS under ML/OU, p=3, 6

Table 4.7 presents the average total cost of two different approaches; the IRP without lateral transhipment and another where the lateral transhipment is allowed. The IRPT-based Randomise CWS results show smaller figures than the IRP-based Randomised CWS under both ML/OU inventory policies for most of the instances. The results for the IRPT_ML and IRPT_OU have 20.15% and 7.72% better than those results for the IRP_ML and IRP_OU with low inventory holding costs and $p = 3$, while the solitons with the same class of instances sets have 27.59% and 11.07% better than those the IRP_ML and IRP_OU with high holding costs, respectively. When the time period = 6, the IRPT-based Randomised CWS is able to improve the solutions on average 15.95% and 9.33% better than the IRP-based Randomised CWS under ML and OU policies with the low inventory holding costs. For the same all sets of instances with high inventory holding costs, they are on average 17.83% and 9.48% better than those the average total costs provided by the IRP-based Randomised CWS under ML and OU replenishment policies, respectively. Which can be seen form the percentage improvement, the IRPT show lower averages total costs than those characterised by the IRPs. As the benefit from allowing transhipment is taken into account (IRP).

According to the experiment (Table 4.7), which can be described that the total costs are reduced when transhipment policy is allowed to use with the IRP. Thus, lateral transhipment is an alternative policy, which can be used to solve IRP. This case decides to use a customer to serve another customer rather than paying a penalty, or another surplus cost may happen. The averages total costs from this experiment of all instances show the benefit when the transhipment is used. For this reason, transhipment can be advantageous in the deterministic context in which no shortage occurs as, they may yield both an overall reduced distribution and inventory holding cost. Also, to solve the IRP without transhipment is setting the transhipment cost sufficiently larger than the IRPT. So, the total costs of the IRPs are higher than IRPTs; however, companies decide to arrange a lateral transhipment only in the case of an emergency or in the case of preventing stock-out caused by high demand. With regard to the means of transportation, the companies prefer to use their vehicles for direct delivery. Therefore, the reason for avoiding transhipment is that the companies have to pay more, not only shipping costs but also other operational costs.

4.5. Chapter Conclusion

This chapter focuses on minimising total costs of the IRP and IRPT under OU or ML replenishment policy by applying biased randomisation version classical CWS heuristic. The proposed method's solution is compared by a branch and cut, algorithm (ABLS) (Archetti et al., 2007) for IRP. For IRPT, Randomised CWS heuristic compared with the exact method (LB and UB) produced by CPLEX (Leandro et al., 2012b). Also, CWS heuristics approach described in Chapter 3 and Randomised CWS heuristic are compared. This chapter has described how to implement a biased randomisation technique into the CWS heuristic. The benefit of integrating biased randomisation can so far be seen by the improvement in performance the implemented Randomised CWS heuristic. In addition, this chapter suggests that applying transhipment to the IRP results in cheaper costs compared to IRP costs - by paying transhipment costs it is more likely to reduce total costs than the potential penalty costs incurred as a result of stock-outs. Also, the IRPT increases supplier reliability. On average, the cost resulting from using transhipment are lower than the approach without transhipment. This shows that incurring transhipment cost is more beneficial for solving the IRP than other recourse actions (penalty costs).

Regarding the IRP_OU and IRP_ML, it is important to state that the results obtained by using Randomised CWS heuristic are better than the results of CWS heuristic.

However, the average results of this proposed method are worse than the average results obtained by ABLS and ALNS. This is because ABLS is the exact method, which always gives optimal solution. Also, IRPT_ML and IRPT_OU –based Randomised CWS heuristic provides better results than the CWS heuristic for all instances. Also, ALNS is meta-heuristic in which it can produce a better solution than heuristics. From best our knowledge that, meta-heuristics are the most effective method to solve the complex problems. However, the IRPT-based Randomised CWS heuristics find better solution than UB (Leandro et al., 2012b) for most of instance even though they worse than LB. This chapter proposed algorithm can be found some better results than those solved by ALNS for the IRP with transhipment case. Therefore, Randomised CWS heuristic able to find better when the problem size is very large.

Additionally, the effectiveness of the Randomised CWS heuristic was required time for solving less than 45 minutes with the time horizon is equal to 3 and up to 50 customers. When the time period is equal to 6, this chapter proposed algorithm solutions were obtained in approximate 25 minutes for the IRP and IRPT under ML/OU policies with up to 30 customers. For randomised CWS heuristic optimisation, to be able to solve more complex problems and find better solutions for all instances sizes. Choosing IG algorithm with local search to be combined with this algorithm. Since the IG algorithm is simply easy to apply together with other algorithms. Therefore, this proposed method can be implemented with IG and local search algorithm, in which an IG algorithm builds a solution by going one step at a time though the feasible solution, applying a heuristic to determine the best choice. Hence, a heuristic applied to IG an insight to solving the IRP and IRPT, such as always choose the largest, smallest, etc., Also, the application of Randomised CWS heuristic and IG with local search to solve IRP and IRPT deterministic demand has not been published yet. So, the next chapter present an extension of propose method then tests and compare with the literature.

Chapter 5: Randomised CWS Heuristic and Iterated Greedy (IG) with local search for Deterministic IRP and IRPT

5.1. Introduction

Iterated Greedy (IG) algorithm is a heuristic, which has not been proposed to solve the deterministic demand IRP and IRPT. This method has been introduced to solve the flowshop scheduling problem in order to minimise the makespan (Ruiz and Stutzle, 2007). Their work showed that IG is very effective to provide new best solution for Permutation Flowshop scheduling Problem (PFSP). Also, IG algorithm, despite its simplicity, is very competitive as is the case of most efficient heuristic. Heuristics and meta-heuristics have been successfully applied for large instances of IRP (Roldan et al., 2016) such as genetic algorithm, tabu search and local search. Aghezzaf (2008) was the first person who proposed the combination of saving based heuristic with an insertion move for solving the Single Vehicle Cycle IRP (SV-CIRP). Also, local search can be stated as a heuristic method to solve combinatorial optimisation problems (COPs). Local search techniques were proposed a few years ago and have been used for solving IRP and shown good performance in experimental studies (Qin et al., 2014). There are many literature reviews (e.g. Benoist et al. (2002), Benoist et al. (2009) and Qin et al. (2014)), which highlight examples of successful local search implementations for solving IRP. They stated that local search algorithms are considered to be one of the available methods, which can produce good solutions with large problem size. Also, Ruiz and Stutzle (2007) showed excellent results that were obtained with an Iterated Greedy (IG) method for the Permutation Flowshop Scheduling Problem (PFSP), where it yielded results that were superior to those from several more complex algorithms. IG method is easily applied to other FSP, it is very simple to code and parameter free (Ruiz & Stutzle, (2007). The result produced by IG is very effective and provided new best-known solutions for PSFP, as shown by Ruiz and Stutzle (2007).

As IG is an effective algorithm and has been adapted successfully for many COPs, it is therefore reasonable to apply IG to combine with other state-of-the-art techniques. The IG algorithm considers a constructive greedy approach with two phases: destruction and construction. Thus, it is reasonable to apply the local search into the

IG algorithm to move from one solution to another in the space of candidate solutions (with applied local changes), until a solution is deemed optimal, or stopping condition. Therefore, the combination of the LS and IG concepts are able to make the parallels between those algorithms become more obvious. The aim of using a local search algorithm is to move from one solution to another in the space of candidate solutions by applying local changes, until a solution is deemed optimal, is found or a time bound is elapsed. Therefore, two common choices can be used to terminate the local search steps: it can be based on a time bound, or based on the best solution found by the algorithm that has not been improved in a given number of steps.

For this reason, this chapter proposes the implementation of combined Randomised CWS heuristic and IG with a local search algorithm to perform good solution of deterministic IRP and IRPT, in order to improve the proposed method in Chapter 4. This method presents as IRP and IRPT –based Randomised CWS heuristic and IG algorithm with local search. The chapter is structured as follows: section 5.2 presents this chapter contributions and section 5.3 describes the proposed approach to solve IRP and IRPT with deterministic demand. The computational experiments of this chapter are shown in section 5.4 and the conclusion of this chapter is given in section 5.5.

5.2. Contribution

An implementation of an IG algorithm to solve the deterministic IRP and IRPT has not yet been proposed in the literature. Therefore, IG with local search with Randomised CWS heuristic for the deterministic IRP and IRPT is proposed here. This is a major contribution of this chapter.

The basic IG method has two phases: the destruction and the construction phases. In the destruction phase, the current solution is partially destroyed. Here, some customers are selected from the current solution. Then the construction phase is started. The selected customers are re-inserted into the partially destroyed solution until the stopping condition is met. The construction phase uses a rather straightforward local search method that is based on the insertion neighbourhood. In literature, this local search is able to improve a solution, which can choose neighbourhood in iterative improvement. The proposal for combining LS and IG concepts is able to be made because the similarities between those algorithms become more pronounced. In practice, the IG algorithm with a local search is repeated until the solution cannot be improved upon any more.

In addition, applying the IG algorithm with the local search to Randomised CWS heuristic can contribute to improved solutions for deterministic IRP and IRPT.

5.3. Proposed Randomised CWS heuristic and IG algorithm with local search for solving Deterministic IRP and IRPT

This chapter proposes the Iterated greedy (IG) algorithm with local search, which generates solutions by iterating greedy constructive heuristics, by applying two main phases: these are named destruction and construction. The IG algorithm using local search in the construction phase absolutely, has been applied for improving the final solutions with the aim to minimise the total costs. Also, a local search is straightforward to add into the procedure of the IG method, which is generally considered as being a very good option for the complex problems (Ruiz & Stutzle, 2007).

5.3.1. **Applying local search into IG procedure**

There are two main procedures in IG algorithm including as followed. In the destruction phase, some elements are removed from a current solution, thus obtaining a partial solution. During the construction phase, the procedure applies a greedy constructive heuristic which inserts elements into a partial solution until a complete candidate solution is reached. Once the candidate solution has been completed, an acceptance criterion is applied to the reconstructed completed candidate solution, to decide whether it will replace the incumbent solution. The process is iterated over these steps until some stopping criterion is met (Ruiz & Stutzle, 2007). Improving each solution during the IG algorithm is procedure in construction phase, which is straightforward by adding a local search inserting process. Following this concept of having a simple and easily implementable algorithm, this chapter has taken a rather straightforward local search which is based on the insertion neighbourhood. Because LS starts with some feasible solution to the problem and iterated to improve it progressively. Each stage in the procedure carries out a movement from one solution to another one with a better value; when there is no other available solution to improve the process is stopped. Thus, if it found a new solution, which is better than the current solution then it will make a change to the current solution.

Algorithm 5.1: Procedure for DESTRUCTION & CONSTRUCTION

1: **Procedure Destruction & Construction** (r', r^D, r^R, d)
▷ r': *Initail solution*
▷ r^D: Partial sequecne to reconstruct
▷ r^R: Customers to reinsert
▷ d: Random chosen number of customer
2: r': $GenerateInitialSolution()$; ▷ *Randomised CWS*
3: Set r^D empty ▷ *Destruction step*
4: **for** $i \leftarrow 1 to\ d$ **do**
5: $r^D \leftarrow$ the set of d selected customer randomly from r'
6: **end for**
7: $r^R \leftarrow$ the remaining set of customers
8: **for j** $\leftarrow 1 to\ d$ **do** ▷ *Construction step*
9: r_{best} = best cost obtained after inserting customer from r^R in all possible position of r^D
10: **end for**
11: **end procedure**

There are two main procedure in any IG algorithm: the procedures for destruction and construction as shown in Algorithm 5.1.

- Destruction phase: The original solution containing a sequence of nodes is partially destroyed into two sub-sequences of a trial list and the remaining list. The trial list contains randomly chosen nodes from the original solution while the remaining list contains the remaining nodes kept in their original sequence.

- Construction phase: This phase starts with the sub-sequence in the trial list and performs a neighbourhood search insertion. The nodes contained in the trial list are re-inserted into the original subsequence of nodes contained in the remaining list to form a new solution. Another important step-in the IG procedure is to decide whether the new solution will be accepted or not. This step uses an acceptance criterion to decide the solution for the next iteration.

- Local search and acceptance criterion: this chapter added a local search procedure to IG; it is done to improve each solution that is generated in the construction phase. There are many methods that can be considered. Having a simple and easily implementable local search that is based on the insertion neighbourhood (Algorithm 5.2) is commonly regarded as being a very good choice for the IRP (Santos et al., 2016).

Algorithm 5.2: Procedure for ITERATIVE_IMPROVEMENT_INSERTION

1: **Procedure Iterative_Improvement_Insertion** (r)
▷ r : *Cost before iimprovement insertion*
▷ r^n: Best cost obtained after inserting
2: improve = true
3: **while** (improve = true) **do**
4: improve = false;
5: **for** $i = 1$ *to* n **do** ▷ *randomly remove one customer from* r
6: r^n := best cost obtained by inserting customer in all possible positions of r;
7: **if** $C(r^n) < C(r)$ **then**
8: $r := r^n$;
9: improve = true;
10: **end if**
11: **end for**
12: **end while**
13: **return** r
14: **end procedure**

Figure 5.1 shows an example of the application of one iteration of IG. This example uses only a single vehicle to deliver to five customers. It assumes that for the initial solution the vehicle begins at the depot to serve customers 5, 1, 2, 4 and 3 then customers 1 and 4 are selected by random selection from the initial solution. Next is inserting the customer in different positions based on insertion neighbourhood; the cost is calculated and compared with the initial cost (previous cost).

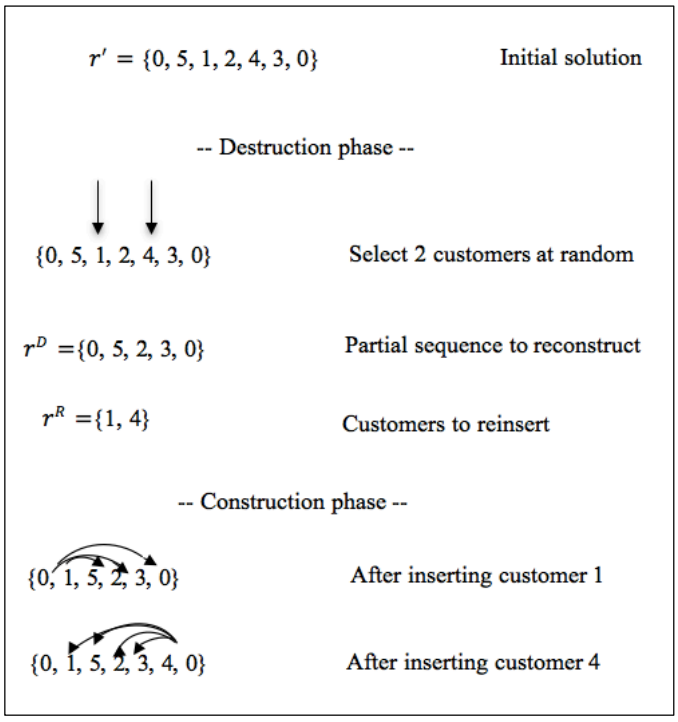

Figure 5.1 : An example of the application of one iteration of IG with local search

5.3.2. IG local search Randomised CWS heuristic for deterministic IRP

This chapter proposes an extension by applying local search into the IG algorithm in order to improve the solution of the deterministic IRP under both inventory replenishment policies from previous chapter. This is one of the best choices for implementing the Randomised CWS heuristic - applying an IG algorithm with local search to solve all instances of these problems. The process of implementation is described as follows: at the beginning of the algorithm initialisation uses the Randomised CWS heuristic, which was proposed in Chapter 4. This heuristic obtains the routes and gives the initial solution. The Randomised CWS solution is processed by an iterative greedy procedure (IG), which uses the best results from the current solution to improve on it. This sub-section explains in detail, Randomised CWS heuristic and IG algorithm with local search, for solving the IRP with deterministic demand under OU/ML policies.

The main procedure is as follows: (a) Randomised CWS heuristic, which uses a multi-start solution generation approach, (b) IG, which improves the solution by using an iterative greedy heuristic; using two main phases (destruction and construction), (c) the use of local search methods, to improve the resulting solution (Algorithm 5.3).

Algorithm 5.3: Pseudocode for IRP RANDOMISE CWS and IG with local search

1: **Procedure IRP RandomisedCWSIteratedGreedyLS** *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, b_n, q_n, \ l_n, w_n, n, \beta, r', r^D, r^R, d)$
- ▷ d_n: $Demand$
- ▷ I_n: Current inventory level
- ▷ q_n: Quantity delivery
- ▷ n: Customer node
- ▷ β: parameter for biased randomisation
- ▷ r': $Initail\ solution$
- ▷ r^D: Partial sequecne to reconstruct
- ▷ r^R: Customers to reinsert
- ▷ d: Random chosen number

2: **for** *each customer* n **do**
3: $h_n \leftarrow$inventory holding cost of each Node n ▷ $Inventory\ Procedure$
4: *nodeList* ← getNodeList (inventoryPolicy)
5: $inventoryCosts \leftarrow h_n * I_n + p_n * l_n$
6: $Depot_{ML/OU} \leftarrow Distribution\ (routing) cost\ associated\ with\ r$
7: currentSol ← initialSol ▷ *Randomised CWS*
8: innitialSol ← IterativeImprovement_Insertion(initailSol) ▷*IG with local search*
9: $r' \leftarrow$ currentSol
10: **while** stopping criteria not satisfied **do** ▷*Destruction_ Construction*
11: $r^{'} \leftarrow$ currentSol
12: **for** $n = 1\ to\ d$ **do** ▷ $d = 3$
13: $r^{'} \leftarrow$ remove one node at random from $r^{'}$ and insert it in r^R;
14: **end for**
15: **for** $n = 1\ to\ d$ **do**
16: $r^{'} \leftarrow$ best permutation obtained by inserting node r^R in all **possible positions** of r^D;
17: **end for**
18: $r^{''} \leftarrow$ IterativeImprovement_Insertion ($r^{'}$); ▷ *Neighbourhood (LS)*
19: **if** $Costs_{max}(r^{''}) < Costs_{max}(r)$**then**
20: $r = r^{''}$
21: **if** $Costs_{max}(r) < Costs_{max}(r^{''})$ **then**
22: $r' = r$
23: **end if**
24: **elseif** $(\boldsymbol{random} \leq \frac{\boldsymbol{exp}\{-\boldsymbol{Costs_{max}}(\boldsymbol{r^{''}})-\boldsymbol{Costs_{max}}(\boldsymbol{r})\}}{\boldsymbol{Temperture}})$ **then**
25: $r = r^{''}$;
26: **end if**
27: **end while**
28: **end for**
29: **return** *bestSolStocSolList*
30**: end procedure**

Algorithm 5.3 Pseudocode for deterministic IRP/IRPT-based Randomised CWS heuristic and IG algorithm with local search. Firstly, the supplier has made a decision on the quantities for serving the demand of the customers by applying inventory replenishment policy (Pseudocode: line 2 to line 5). So, the inventory level for each

customer at the end period is calculated; it depends on the initial stock level and also on the end-customer's demands during that period. As this situation considers deterministic demand, so the amount of the customer's demand is known beforehand. Then, an initial solution is generated by Randomised CWS heuristic (Pseudocode: line 7). After that, an iterative improvement algorithm is started, by using a first improvement type pivoting rule (Pseudocode: line 8 to line 28). Following, the destruction and construction phases and the optional local search phase, then whether the new route is accepted or not as the current solution for next iteration is considered. The reason simplest acceptance criteria is new routes they provide a better solution (lowest total costs), then the supplier will deliver all customer demands. However, an IG algorithm using this acceptance criterion may lead to a stagnation situation of the search, due to insufficient diversification. Therefore, a simple simulated annealing-like acceptance criterion, with a constant temperature is considered (Ruben & Stuzle, 2007). This constant temperature depends on the particular instance and it is computed following the suggestion of Osman and Potts (1989) as

$$Temperature = T \times \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} p_{ij}}{n \times m \times 10} \quad (1)$$

where T is a parameter that needs to be adjusted, p_{ij} is the customer (i) demand, m is number of vehicle and n is number of customer.

5.3.3. **IG local search Randomised CWS heuristic for deterministic IRPT**

This section gives Randomised CWS heuristic and IG algorithm with local search is also applied for solving the deterministic IRPT under both ML and OU policies as shown in Algorithm 5.4. There is no any structure or procedure changed from the procedure for IRP-based Randomised CWS heuristic, it is dissimilar when the lateral transhipments are allowed to take place after the demand is realised. Also, the case of the inventory level of customer becomes negative or stock-out. The process of selecting the customer to be the transhipment point starts by checking the updated inventory level of all customers, then finding the nearest customer which has enough inventory level to be the supply node. In addition, the cost of transhipment will be added to the total costs - the supplier has responsibility for this. This solution will provide an estimate of the total cost under the ML/OU replenishment and transhipment policies with known demand (Pseudocode: line 6 to line 8). In this case, if lateral transhipments are not allowed to take place after the demand is fulfilled then,

the extra demand may occur or customer face stock-out. It is reasonable to say that with the use of transhipment total costs can be reduced, which means the supplier avoids paying the penalty costs.

Algorithm 5.4: Pseudocode for IRPT RANDOMISE CWS and IG with local search

1: **Procedure IRPT RandomisedCWSIteratedGreedyLS** *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, b_n, q_n, \ l_n, w_n, n, \beta, r', r^D, r^R, d)$
- ▷ d_n: $Demand$
- ▷ I_n: Current inventory level
- ▷ q_n: Quantity delivery
- ▷ n: Customer node
- ▷ β: parameter for biased randomisation
- ▷ r': $Initail\ solution$
- ▷ r^D: Partial sequecne to reconstruct
- ▷ r^R: Customers to reinsert
- ▷ d: Random chosen number

2: **for** *each customer* n **do**
3: h_n←inventory holding cost of each Node n ▷ $Inventory\ Procedure$
4: *nodeList* ← getNodeList (inventoryPolicy)
5: *inventoryCosts* ← $h_n * I_n + p_n * l_n$
6: **if** policy = TRANSHIPMENT **then**
7: *transhipmentCosts* ← $b_n * w_n$ ▷ *Transhipment Procedure*
8: **end if**
9: $Depot_{ML/OU} \leftarrow Distribution\ (routing) cost\ associated\ with\ r$
10: currentSol ← initialSol ▷ *Randomised CWS*
11: innitialSol ← IterativeImprovement_Insertion(initailSol) ▷*IG with local search*
12: r'← currentSol
13: **while** stopping criteria not satisfied **do** ▷*Destruction_ Construction*
14: $r^{'}$ ← currentSol
15: **for** $n = 1\ to\ d$ **do** ▷ $d = 3$
16: $r^{'}$ ← remove one node at random from $r^{'}$ and insert it in r^R;
17: **end for**
18: **for** $n = 1\ to\ d$ **do**
19: $r^{'}$← best permutation obtained by inserting node r^R in all **possible positions** of r^D;
20: **end for**
21: $r^{''}$ ← IterativeImprovement_Insertion ($r^{'}$); ▷ *Neighbourhood (LS)*
22: **if** $Costs_{max}(r^{''}) < Costs_{max}(r)$**then**
23: $r = r^{''}$
24: **if** $Costs_{max}(r) < Costs_{max}(r^{''})$ **then**
25: $r' = r$
26: **end if**
27: **elseif** $(\boldsymbol{random} \leq \frac{\boldsymbol{exp}\{-\boldsymbol{Costs_{max}}(\boldsymbol{r^{''}})-\boldsymbol{Costs_{max}}(\boldsymbol{r})\}}{\boldsymbol{Temperture}})$ **then**
28: $r = r^{''}$;
29: **end if**
30: **end while**
31: **end for**
32: **return** *bestSolStocSolList*
33**: end procedure**

5.4. Computational experiments

The performance of IRP and IRPT-based Randomised CWS and IG algorithm with local search under ML/OU replenishment policies are examined. In this section, the results regarding the average solution quality over all instances are discussed. The case study in this chapter focuses on the deterministic IRP and IRPT where demand is known, and the same problem instances as the previous chapter are used. The results of the algorithm proposed by this chapter show better than the solutions from Chapter 4 in every data sets. As a result of 20 experiments, the average for all instances sets have been shown. The solutions are applied from this chapter proposed method and, in the literature, both are also compared. The potential of the proposed method for solving IRP and IRPT will be represented as the percentage gaps.

The instances used for experiments are taken from the benchmark (Archetti et al., 2007 and Leandro et al., 2012b). These instances are divided into two classes according to their inventory cost, and in each set of instances they are divided by the number of customers, each of which contains five instances. For example, absn05 is an instance with 5 customers which has been averaged over 5 instances such as abs1n01 to abs5n05. These all sets of instances are divided and classed as described in Chapter 3. The solutions provided by IRP-based Randomised CWS heuristic and IG algorithm with local search under ML/OU inventory replenishment policies present in sub-section 5.4.1. The next sub-section (5.4.2) shows the average results obtained by IRPT-based obtained by Randomised CWS heuristic and IG algorithm with local search under ML/OU inventory replenishment policies. All sets of instances have been generated and solved on the computer with a Core i5, 2.30 GHz processor and 4GB of RAM.

Table 5.1 and 5.5 the solutions of the total costs can be breakdown as follows; the inventory holding (HC), penalty (PC), transhipment (TrC), routing (RC) and total costs (TC) are given. The average costs for the set of instances presents in bold number. These tables clearly show one result calculated from five instances for each combination of the cost element of IRP-based Randomised CWS and IG algorithm with local search and IRPT-based Randomised CWS and IG algorithm with local search under OU or ML replenishment policies and the time periods (p) equal to three and six, respectively. All tables illustrate the results of the low or high inventory cost and the time periods ($p = 3, p = 6$) for the IRP and IRPT under ML/OU policies, which is represented as IRP_ML-based Randomised CWS and IG algorithm with local

search, IRP_OU-based Randomised CWS and IG algorithm with local search, IRPT_ML-based Randomised CWS and IG algorithm with local search and IRPT_OU-based Randomised CWS and IG algorithm with local search. The first column gives the instance name, which includes the number of the customers. The cost breakdown of the IRP_ML and the IRPT_ML solved by this chapter proposed method presents in the second, the third and the fourth, respectively. Then the fifth, the sixth and the seventh column show the cost breakdown of IRP_OU and IRPT_OU, respectively.

5.4.1. **IRP-based Randomised CWS heuristic and IG with local search**

In the performed experiments, this chapter implemented proposed algorithm by applying 'an IG algorithm with local search' to 'Randomised CWS heuristic', and all sets of instances are used. In this section, firstly the results regarding the average solution quality over on all sets of instances. Since, the solutions obtained in this sub-section often improve the previous solutions, the updated results on all sets instances are presented. Next, the comparison of the performance of Randomised CWS heuristic and IG algorithm with local search against the best performing algorithm for the IRP-based Randomised CWS (in sub-section 5.4.1.1), while the best performing method in the literature (in sub-section 5.4.1.2). The experimental of the implementation of deterministic IRP-based Randomised CWS heuristic and IG algorithm with local search for all sets of instances are presented. Table 5.1 shows the average costs and the average costs breakdown, which presents in bold numbers. Table is categorised as: the low/high inventory holding costs on IRP_ML and IRP_OU-based Randomised CWS heuristic and IG algorithm with local search in the time horizon $p = 3$ and $p = 6$.

The average costs provided by IRP_ML-based Randomised CWS heuristic and IG algorithm with local search are slightly lower than those average total cost provided by IRP_OU-based Randomised CWS heuristic and IG algorithm with local search. This reason is that using either the ML or OU policy can lead to a different impact on inventory holding cost. In the former type of policy, the quantity delivered to a customer is such that the level of the inventory at the customer is not greater than the maximum level in a time period. In the latter type of policy, the quantity shipped is such that the level of the inventory at the customer reaches exactly its maximum level at the time period, as long as its holding capacity is respected. Therefore, ML replenishment policy able to avoid routing and inventory holding cost in which it is less likely to deliver more products to the customer when its inventory capacity is fully served.

According to Table 5.1, it is obvious that using ML or OU policies will impact to inventory and transportation costs. In particularly, in the case of low inventory holding cost, utilising ML policy will result in lower cost than that using OU policy. On the other hand, the OU policy will contribute to cost saving when inventory cost is high. When the time period is equal to 6, the average HC, RC and TC are higher than those average costs when $p = 3$ on the same sets of instances for IRP under ML/OU policies and low/high inventory holding costs. Which can be seen that holding costs are much higher if the products store in the warehouse long time period. For the low inventory holding costs provided lower holding costs than the high inventory holding costs, while the routing costs are not much different for all sets of instances of the IRP under ML/OU policies and $p = 3$ and $p = 6$.

Low inventory holding cost and Time period = 3							
Name of Instance		IRP_ML-based Randomised CWS and IG with LS			IRP_OU-based Randomised CWS and IG with LS		
		HC	RC	TC	HC	RC	TC
1	absn05	88.50	798	886.50	93.80	1103	1196.80
2	absn10	125.09	1295	1420.09	210.20	1879	2089.20
3	absn15	278.34	1879	2157.34	289.40	1998	2287.40
4	absn20	491.11	2001	2492.11	377.89	2098	2475.89
5	absn25	501.30	2298	2799.30	487.93	2684	3171.93
6	absn30	589.34	2590	3179.34	578.56	2987	3565.56
7	absn35	656.89	2776	3432.89	703.23	3003	3706.23
8	absn40	779.20	2989	3768.20	798.77	3201	3999.77
9	absn45	829.50	3100	3929.50	800.09	3339	4139.09
10	absn50	986.90	3298	4284.90	943.45	3675	4618.45
	Average	**532.62**	**2302.40**	**2835.02**	**528.33**	**2596.70**	**3125.03**
High inventory holding cost and Time period = 3							
Name of Instance		IRP_ML-based Randomised CWS and IG with LS			IRP_OU-based Randomised CWS and IG with LS		
		HC	RC	TC	HC	RC	TC
1	absn05	670.40	1298	1968.40	778.44	1398	2176.44
2	absn10	1987.60	1769	3756.60	1887.40	2300	4187.40
3	absn15	2001.45	2006	4007.45	3003.55	2570	5573.55
4	absn20	3987.34	2798	6785.34	4098.60	3100	7198.60
5	absn25	4459.46	2998	7457.46	5897.67	3987	9884.67
6	absn30	6003.60	3789	9792.60	6980.70	4302	11282.70
7	absn35	6978.89	4030	11008.89	7409.77	4465	11874.77
8	absn40	7893.33	4564	12457.33	8734.45	4699	13433.45
9	absn45	8989.77	4987	13976.77	9124.30	5055	14179.30
10	absn50	10001.56	5005	15006.56	10089.00	6129	16218.00
	Average	**5297.34**	**3324.40**	**8621.74**	**5800.39**	**3800.50**	**9600.89**
Low inventory holding cost and Time period = 6							
Name of Instance		IRP_ML-based Randomised CWS and IG with LS			IRP_OU-based Randomised CWS and IG with LS		
		HC	RC	TC	HC	RC	TC
1	absn05	399.80	2900	3299.80	399.10	2998	3397.10
2	absn10	580.56	4001	4581.56	498.50	4302	4800.50
3	absn15	607.40	4798	5405.40	700.10	5001	5701.10
4	absn20	767.56	6108	6875.56	866.34	5988	6854.34
5	absn25	897.45	6277	7174.45	898.65	6202	7100.65
6	absn30	956.98	6490	7446.98	1009.50	6981	7990.50
	Average	**701.63**	**5095.67**	**5797.29**	**728.70**	**5245.33**	**5974.03**
High inventory holding cost and Time period = 6							
Name of Instance		IRP_ML-based Randomised CWS and IG with LS			IRP_OU-based Randomised CWS and IG with LS		
		HC	RC	TC	HC	RC	TC
1	absn05	2009.34	3142	5151.34	2699.50	2966	5665.50
2	absn10	3908.23	4093	8001.23	4270.08	4563	8833.08
3	absn15	4789.54	6565	11354.54	5204.45	6500	11704.45
4	absn20	7411.90	7467	14878.90	7156.45	7931	15087.45
5	absn25	8907.88	8003	16910.88	8922.90	8911	17833.90
6	absn30	10720.07	10001	20721.07	10289.21	10132	20421.21
	Average	**6291.16**	**6545.17**	**12836.33**	**6423.77**	**6833.83**	**13257.60**

Table 5.1 : Average result of cost breakdown of the IRP-based Randomised CWS and IG algorithm with local search under ML/OU, p=3 and p= 6

5.4.1.1. *Comparison of the results of IRP-based Randomised CWS and IG with local search, with the best results in the IRP-based Randomised CWS*

This sub-section presents the comparison of the average results between this chapter proposed method and IRP-based Randomised CWS heuristics. Table 5.2 shows the average solutions provided by two different algorithms, the IRP-based Randomised CWS and IG with local search, and the IRP-based Randomised CWS heuristics. These algorithms approach consider the average results of five instances generated for each combination of customer (n) and time horizon (H). This table gives details for each

category as follows: the first column gives the number of customers. The second and fifth columns are the average results solved by IRP_ML-based Randomised CWS and IRP_OU-based Randomised CWS heuristics, respectively. The average total costs obtained by IRP-based Randomised CWS heuristic and IG algorithm with local search under ML and OU replenishment policies are presented in the third and the sixth columns, respectively. The forth and the seventh columns show percentage improvement of these two algorithms under ML and OU policies, respectively.

The Randomised CWS heuristic and IG algorithm with local search able to find better solutions than the Randomised CWS heuristic for all instances on all sets of the IRP under ML/OU replenishment policies with low/high inventory holding costs and the time periods are equal to 3 and 6. Which form the results show that applying the IG algorithm with the local search to Randomised CWS heuristic can contribute to improved solutions for the deterministic IRP. It can be described that the solutions approach presented in this chapter are on average 2.53 % for IRP_ML and 3.27% for IRP_OU better than the solutions approach in Chapter 4 for all sets of all instances with low inventory holding costs and $p = 3$. For the IRP_ML and IRP_OU with high holding costs and $p = 3$, this chapter proposed algorithm performs better results than the algorithms presented in Chapter 4 for all of the instances on all sets. They have 1.48% better than those obtained by Randomised CWS, and up to 3.64% better for the same set.

When the time horizon is equal to 6, the IRP_ML, average total costs have also better on all sets of instances. They are on average 2.70% lower than the average total costs given by Randomised CWS for low inventory holding cost, and 3.07% lower for high inventory holding cost. The average total costs given by Randomised CWS and IG with local search are on average 3.90% and 3.21% lower than those provided by Randomised CWS for the IRP_OU with low and high inventory holding costs, respectively. Which can be seen from the percentage differences between two values obtained by the two different algorithms, it meant that this chapter proposed algorithm has more performance for finding better solution than the method that has been proposed in Chapter 4. It is necessary to notice that using Randomised CWS and IG algorithm with local search it can lead to decrease the total costs on all sets of instances.

Low inventory holding cost and Time period = 3						
Name of insnatce	IRP_ML-based Randomised CWS (A)	IRP_ML-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B	IRP_OU-based Randomised CWS (A)	IRP_OU-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B
absn05	1279.20	1275.99	0.25	1493.04	1418.99	5.22
absn10	1986.81	1911.08	3.96	2256.81	2229.01	1.25
absn15	2214.29	2207.99	0.29	2494.29	2493.48	0.03
absn20	2882.35	2675.82	7.72	3082.35	3054.77	0.90
absn25	2998.79	2998.76	0.00	3598.79	3451.50	4.27
absn30	3447.70	3327.02	3.63	3847.70	3644.27	5.58
absn35	3525.52	3497.52	0.80	4025.52	3848.49	4.60
absn40	3910.74	3710.50	5.40	4290.74	4141.09	3.61
absn45	3965.62	3875.82	2.32	4325.62	4277.67	1.12
absn50	4396.36	4356.53	0.91	5126.36	4830.01	6.14
Average	**3060.74**	**2983.70**	**2.53**	**3454.12**	**3338.93**	**3.27**
High inventory holding cost and Time period = 3						
Name of insnatce	IRP_ML-based Randomised CWS (A)	IRP_ML-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B	IRP_OU-based Randomised CWS (A)	IRP_OU-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B
absn05	2241.04	2201.22	1.81	2441.04	2354.18	3.69
absn10	4456.98	4339.09	2.72	4696.98	4690.76	0.13
absn15	5459.13	5437.01	0.41	5979.13	5737.33	4.21
absn20	7270.10	7240.47	0.41	7930.10	7623.17	4.03
absn25	9045.38	9000.88	0.49	10025.38	9466.04	5.91
absn30	11002.28	10920.32	0.75	11356.28	11331.01	0.22
absn35	11518.85	11435.05	0.73	12109.85	11840.58	2.27
absn40	12644.99	12579.45	0.52	13494.99	13022.43	3.63
absn45	14504.25	13894.99	4.38	15204.25	14389.98	5.66
absn50	15878.17	15486.75	2.53	17058.17	15998.97	6.62
Average	**9402.12**	**9253.52**	**1.48**	**10029.62**	**9645.45**	**3.64**
Low inventory holding cost and Time period = 6						
Name of insnatce	IRP_ML-based Randomised CWS (A)	IRP_ML-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B	IRP_OU-based Randomised CWS (A)	IRP_OU-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B
absn05	3352.54	3288.33	1.95	3452.54	3299.98	4.62
absn10	4764.47	4664.55	2.14	4964.47	4832.91	2.72
absn15	5565.34	5465.48	1.83	5885.34	5581.04	5.45
absn20	7179.09	6699.99	7.15	7279.09	6857.57	6.15
absn25	7458.28	7380.26	1.06	7498.28	7463.01	0.47
absn30	7967.26	7807.13	2.05	8167.26	7854.38	3.98
Average	**6047.83**	**5884.29**	**2.70**	**6207.83**	**5981.48**	**3.90**
High inventory holding cost and Time period = 6						
Name of insnatce	IRP_ML-based Randomised CWS (A)	IRP_ML-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B	IRP_OU-based Randomised CWS (A)	IRP_OU-based Randomised CWS and IG with LS (B)	Improvement (%) (A-B)/B
absn05	5531.58	5514.50	0.31	5631.58	5538.30	1.68
absn10	8827.21	8667.55	1.84	9027.21	8872.42	1.74
absn15	12136.19	11601.89	4.61	12236.19	11731.08	4.31
absn20	15598.48	14770.36	5.61	15598.48	14880.11	4.83
absn25	17892.42	17101.04	4.63	18092.42	17202.22	5.17
absn30	20882.21	20589.89	1.42	21072.21	20760.05	1.50
Average	**13478.02**	**13040.87**	**3.07**	**13609.68**	**13164.03**	**3.21**

Table 5.2 : Comparison average results between the IRP–based Randomised CWS and the IRP-based Randomised CWS and IG algorithm with local search under ML/OU, p=3, 6

5.4.1.2. *Comparison of the results of IRP-based Randomised CWS and IG algorithm with local search, with the best results in the literature*

Table 5.3 presents the comparison of average results between IRP-based Randomised CWS and IG algorithm with local search, and the best solutions in ABLS (Archetti et al., 2007). The solutions obtained by ALNS (Coelho et al., 2012) are also reported in Table 5.3. The result on all sets of instances give the average of five instances generated for each combination of the number of customer (n)and the time period (p). In the experiments conducted, the performance of this chapter proposed Randomised CWS and IG algorithm with local search is tested on the same instances sets, which has been used in the literature. The computational results provided in term of the average solution costs which is the best solution. Table 5.3 shows the comparison between the results achieved through this chapter proposed method for all instances of the difference set in the inventory holding costs and the time periods.

The performance of this chapter approach and in the literature, are compared against the best performing solution for Randomised CWS and IG algorithm with local search. The column 'ABLS' summarised the average solutions obtained by A Branch and Cut algorithm (Archetti et al., 2007). The column 'ALNS' summarised the average results obtained by Adapted Large Neighbourhood Search (Coelho et al., 2012). The column 'IRP_ML-based-Randomised CWS and IG with LS' and IRP_OU-based Randomised CWS and IG with LS' summarised the average results obtained by applying Randomised CWS and IG algorithm with local search for solving IRP under ML and OU inventory replenishment policies. The table below consists of the following: the bottom row divides into two columns, which present the Low/High inventory holding cost and the time horizon $p = 3$ ($n = 50$) and $p = 6$ ($n = 30$). The average total costs provided by IRP-based Randomised CWS and IG algorithm with local search, and the average %gap between this chapter proposed algorithm against the best-known solution in literature ABLS (Archetti et al., 2007) and the average %gaps of those results of ABLS are compared by ALNS (Leandro et al., 2012) are presented in the bold numbers. Whereas, the good solutions are found by this chapter proposed approaches, which have better than those best-known solution obtained by ALNS are presented in red numbers.

This chapter proposed approach is able to find good solutions, but these average total costs are slightly higher than the average total costs obtained by ABLS. Although, the results of this chapter worse than those of Archetti et al. (2007) but the running time of this chapter proposed method is less than those algorithms. The solutions obtained

by IRP-based Randomised CWS and IG algorithm with local search are on average 0.43 % and 0.11% higher than those obtained by ABLS for the IRP_ML and the IRP_OU, respectively with low inventory holding costs and the time horizon is equal to 3. They have 2.25% and 0.12% slightly higher than for the set with three time periods and high inventory holding costs of the IRP under ML and OU policies, respectively. For the IRP of the set with six time periods and low inventory holding costs, the solutions obtained by ABLS are on average 1.81% and 0.14% better than those average solutions found by this chapter proposed method with under ML/OU replenishment policies, respectively. While, the set with high inventory holding costs and six time periods, they are on average 1.23% slightly lower than for the IRP_ML and up to 0.15% slightly lower than for the IRP_OU with the same set. The solutions provided by this chapter proposed algorithm are near-optimal solution even for the large instances, but they are not better than those solutions found by ABLS. This is reasonable that ABLS is the exact method, which always provides optimal solution. While, this chapter proposed method is a heuristic algorithm, which is not guaranteed to find an optimal solution but, in practice, it is often able to find good solution, albeit possibly suboptimal, in a relative short time.

In this section, the solution obtained by this chapter proposed method and ALNS (Coelho et al., 2012) are compared as shown in Table 5.3. Regarding to red figure indicated that the Randomised CWS and IG algorithm with local search can lead to improve the best heuristic solution. This chapter proposed algorithm is able to find better results than ALNS for most of the instances on all sets. They have better on all five sets of instances for the IRP_ML of the set with three time periods and low inventory holding costs. For the IRP_OU, they have also, better on all six instances sets for the same set. When the time period is 3 with high inventory holding costs, they have better on seven sets of instances for the IRP under ML policy, while the average total costs for most of instances on all sets lower than the average total costs given by ALNS for the IRP under OU policy with the same set.

When the time horizon is 6 with low inventory holding costs, they have better on all four instances sets for the IRP under ML/OU policies (see red numbers). For the set with six time periods and high inventory holding costs, the average results for the IRP_ML have better on all three sets of instances, and up to four sets of instances for the IRP_OU with the same set. Which can be seen from the red numbers that IRP-based Randomised CWS and IG algorithm with local search has performed good

quality solution when compared to heuristic algorithm. It is remarkable that all instances with up to 50 customers can be solved in short time within less than an hour of the time required when $p = 3$, while when $p = 6$, only instances with not more than 30 customers can be solved in about one and half hours.

Low inventory holding cost and Time period = 3								High inventory holding cost and Time period = 3						
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based Randomised CWS and IG with LS		Gap (%)	ABLS	ALNS		Gap (%)	IRP_ML-based Randomised CWS and IG with LS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	1275.86	1275.99	6.08	0.01	1275.99	4.10	0.01	2187.30	2201.20	5.82	0.64	2201.22	4.20	0.64
absn10	1910.93	1911.08	18.59	0.01	1911.08	17.80	0.01	4337.97	4339.35	17.13	0.03	4339.09	18.90	0.03
absn15	2207.77	2208.57	44.69	0.04	**2207.99**	45.23	0.01	5435.80	5438.80	46.34	0.06	**5437.01**	77.30	0.02
absn20	2665.58	2675.58	98.05	0.38	2675.82	108.30	0.38	7225.70	7262.24	93.08	0.51	**7240.47**	113.20	0.20
absn25	2987.90	2996.77	171.38	0.30	2998.76	351.50	0.36	7501.07	9007.11	164.36	20.08	**9000.88**	289.50	19.99
absn30	3292.93	3330.77	331.45	1.15	**3327.02**	630.19	1.04	10918.31	10941.55	315.76	0.21	**10920.32**	789.30	0.02
absn35	3448.84	3495.04	495.46	1.34	3497.52	1509.20	1.41	11411.67	11472.14	575.54	0.53	11475.05	1505.30	0.56
absn40	3703.82	3736.05	793.99	0.87	**3710.50**	1784.56	0.18	12541.06	12632.72	801.64	0.73	**12579.45**	1880.40	0.31
absn45	3867.48	3886.26	1405.72	0.49	**3875.82**	2456.30	0.22	13865.34	13928.10	1197.02	0.45	**13894.99**	2659.44	0.21
absn50	4327.16	4366.43	1719.41	0.91	**4356.53**	3009.30	0.68	15410.82	15512.49	2109.81	0.66	**15486.75**	3100.30	0.49
Average	**2968.83**	**2988.25**	**508.48**	**0.55**	**2983.70**	**991.65**	**0.43**	**9083.50**	**9273.57**	**532.65**	**2.39**	**9257.52**	**1043.78**	**2.25**
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_OU-based Randomised CWS and IG with LS		Gap (%)	ABLS	ALNS		Gap (%)	IRP_OU-based Randomised CWS and IG with LS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	1418.76	1418.75	10.71	0.00	1418.99	4.90	0.02	2354.18	2354.17	9.45	0.00	2354.18	6.34	0.00
absn10	2228.67	2228.66	35.28	0.00	2229.01	18.10	0.02	4690.46	4691.02	34.62	0.01	**4690.76**	23.73	0.01
absn15	2493.47	2493.47	99.32	0.00	2493.48	67.45	0.00	5736.91	5740.66	97.09	0.07	**5737.33**	99.10	0.01
absn20	3053.02	3055.58	239.76	0.08	**3054.77**	110.65	0.06	7619.91	7626.94	224.24	0.09	**7623.17**	150.30	0.04
absn25	3451.15	3451.86	572.28	0.02	**3451.50**	398.23	0.01	9460.75	9476.04	446.47	0.16	**9466.04**	440.20	0.06
absn30	3643.22	3645.70	1072.47	0.07	**3644.27**	604.40	0.03	11320.65	11354.66	890.16	0.30	**11331.01**	898.20	0.09
absn35	3846.87	3850.83	1439.28	0.10	**3848.49**	1773.88	0.04	11828.82	11848.90	1600.56	0.17	**11840.58**	1505.09	0.10
absn40	4125.70	4140.16	2755.72	0.35	4141.09	2010.20	0.37	13011.46	13043.95	2767.76	0.25	**13022.43**	2005.60	0.08
absn45	4270.61	4283.33	3417.87	0.30	**4277.67**	2244.50	0.17	14317.82	14392.04	3010.08	0.52	**14389.98**	2601.50	0.50
absn50	4810.87	4841.26	2675.47	0.63	**4830.01**	3003.30	0.40	15948.78	16077.86	2987.26	0.81	**15998.97**	3101.70	0.31
Average	**3334.23**	**3340.96**	**1231.82**	**0.16**	**3338.93**	**1023.56**	**0.11**	**9628.97**	**9660.62**	**1206.77**	**0.24**	**9645.45**	**1083.18**	**0.12**
Low inventory holding cost and Time period = 6								High inventory holding cost and Time period = 6						
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_ML-based Randomised CWS and IG with LS		Gap (%)	ABLS	ALNS		Gap (%)	IRP_ML-based Randomised CWS and IG with LS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	3136.90	3288.30	12.90	4.83	3288.33	19.33	4.83	5354.20	5514.49	12.20	2.99	5514.50	20.20	2.99
absn10	4612.50	4751.96	54.26	3.02	**4664.55**	94.30	1.13	8601.92	8757.00	56.27	1.80	**8667.55**	143.60	0.76
absn15	5418.55	5546.96	160.81	2.37	5465.48	239.12	0.87	11543.04	11681.58	154.74	1.20	**11601.89**	404.22	0.51
absn20	6625.35	6762.36	381.15	2.07	**6699.99**	893.65	1.13	14602.14	14750.66	348.11	1.02	14770.36	989.99	1.15
absn25	7261.77	7443.39	599.87	2.50	**7380.26**	1798.40	1.63	16913.97	17113.84	703.10	1.18	**17101.04**	1745.30	1.11
absn30	7710.01	7835.02	1415.08	1.62	**7807.13**	4103.40	1.26	20410.65	20547.02	1597.86	0.67	20589.89	4300.50	0.88
Average	**5794.18**	**5938.00**	**437.35**	**2.74**	**5884.29**	**1191.37**	**1.81**	**12904.32**	**13060.77**	**478.71**	**1.48**	**13040.87**	**1267.30**	**1.23**
Name of insnatce	ABLS	ALNS		Gap (%)	IRP_OU-based Randomised CWS and IG with LS		Gap (%)	ABLS	ALNS		Gap (%)	IRP_OU-based Randomised CWS and IG with LS		Gap (%)
	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A	(A)	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	3299.98	3299.97	20.92	0.00	3299.98	16.77	0.00	5538.02	5538.91	22.82	0.02	**5538.30**	23.34	0.01
absn10	4832.89	4832.87	95.88	0.00	4832.91	87.80	0.00	8872.41	8872.41	106.84	0.00	8872.42	177.60	0.00
absn15	5566.39	5582.80	337.70	0.29	**5581.04**	298.34	0.26	11721.82	11738.50	370.67	0.14	**11731.08**	494.30	0.08
absn20	6833.29	6857.90	797.63	0.36	**6857.57**	870.50	0.36	14863.86	14883.49	1021.13	0.13	**14880.11**	1001.30	0.11
absn25	7454.15	7487.80	1610.54	0.45	**7463.01**	1898.80	0.12	17170.82	17223.47	2221.46	0.31	**17202.22**	1999.40	0.18
absn30	7847.39	7888.56	3031.66	0.52	**7854.38**	4300.20	0.09	20657.28	20752.32	3399.49	0.46	20760.05	4699.80	0.50
Average	**5972.35**	**5991.65**	**982.39**	**0.27**	**5981.48**	**1245.40**	**0.14**	**13137.37**	**13168.18**	**1190.40**	**0.18**	**13164.03**	**1399.29**	**0.15**

Table 5.3 : Comparison average results of ABLS, ALNS and IRP-based Randomised CWS heuristic and IG algorithm with local search under ML/OU, p=3, 6

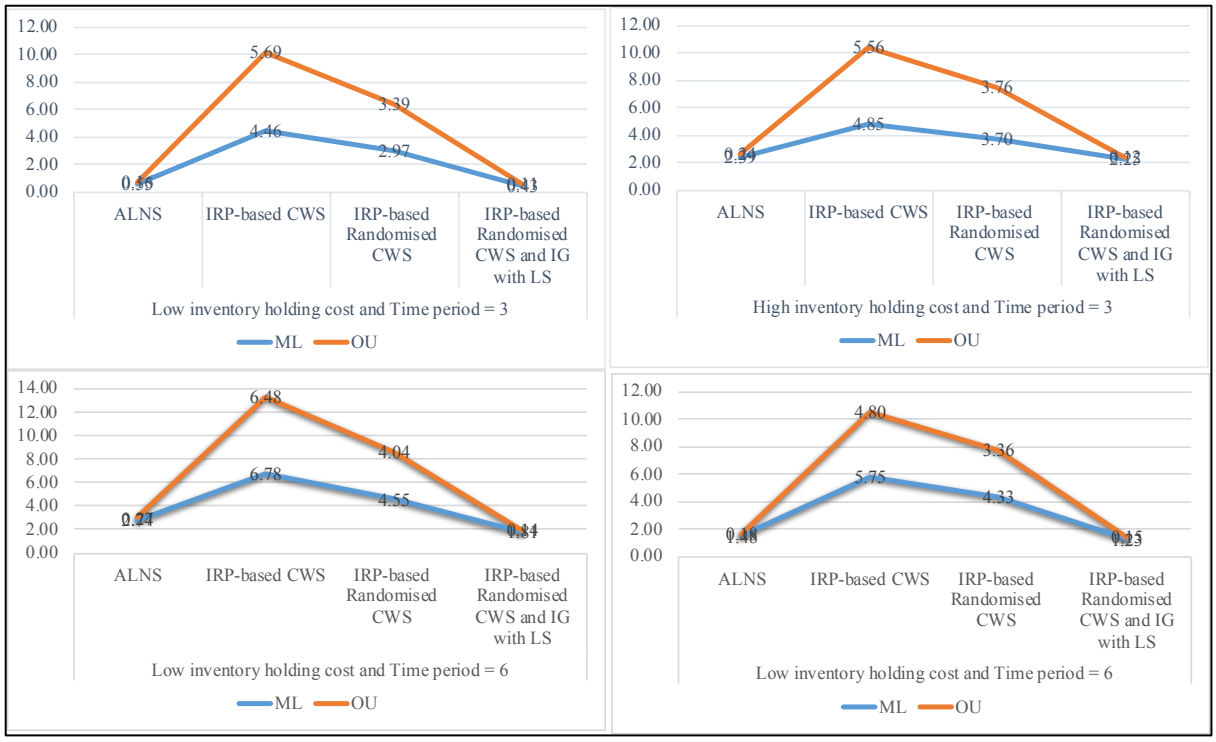

Figure 5.2 : Average percentage gap of algorithms when compare to ABLS

Figure 5.2 shows the average percentage gap of the IRP's results of ALNS and this thesis proposed methods are compared to the results of ABLS. It can be seen that the IRP-based Randomised CWS and IG with local search under ML/OU policies are closely to the results of the ABLS. Its shown the lowest average percentage gaps than those solved by ALNS, CWS, and Randomised CWS.

5.4.2. **IRPT-based Randomised CWS heuristic and IG with local search**

In the performed experiments, this chapter implemented proposed algorithm by applying 'an IG algorithm with local search' to 'Randomised CWS heuristic', for solving IRPT and all sets of instances are used. In this section, the results regarding the average solution quality over on all sets of instances are the first presented. Since, the solutions obtained in this sub-section often improve the previous solutions, the updated results on all sets instances are also shown. After that, the comparison of the performance of Randomised CWS heuristic and IG algorithm with local search for solving the IRPT against the best performing algorithm for Randomised CWS (in sub-section 5.4.2.1), while the best performing method in the literature (in sub-section 5.4.2.2). Finally, the performance of this proposed algorithm for solving the IRP and IRPT on the same all instances sets, and inventory replenishment policies are compared (in sub-section 5.4.2.3). The cost breakdown and total cost obtained by the IRPT-based Randomised CWS heuristic and IG algorithm with local search under

OU/ML inventory replenishment policies are presented in Table 5.4. The average costs provided by the IRPT_ML-based Randomised CWS heuristic and IG algorithm with local search are slightly lower than those costs obtained by using the OU replenishment policy. There are no significant differences in the results between the IRP and the IRPT under ML or OU policies, which has been explained in the section 5.4.1.

For the set, low inventory holding costs and three time and six time periods, for the IRPT under ML/OU replenishment pollicises shows the average HC lower than the average RC, while the average HC are higher than the average RC for the set of high inventory holding costs with the same time period. In fact, holding costs are the costs associated with storing inventory then high inventory holding costs and high level of inventory can lead to higher inventory holding costs than other costs (RC/TrC). When the time period is equal to 6, to hold and stored its inventory with the long-time period can also lead to higher inventory holding costs than short time periods. Therefore, all sets of instances and three time periods with up to 50 customers show the average RC lower than the same sets and six time periods with up to 30 customers. Moreover, on all sets of instances with three time and six time periods are not difference in the transhipment costs. Adopting the OU policy can lead to higher transhipment cost than that using the ML policy for all instances. The reason is that, when using the OU replenishment policy, supplier is not fully served the customer inventory that can lead to extra demand during the period, in turn, transhipment cost may be occurred.

Low inventory holding cost and Time period = 3									
Name of Instance		IRPT_ML-based Randomised CWS and IG with LS				IRPT_OU-based Randomised CWS and IG with LS			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	87.38	0.00	667	754.38	89.90	0.00	550	639.90
2	absn10	210.70	89.45	1278	1578.15	209.40	89.99	1201	1500.39
3	absn15	303.45	123.56	1415	1842.01	310.45	142.12	1302	1754.57
4	absn20	442.12	181.14	1660	2283.26	445.89	208.56	1476	2130.45
5	absn25	509.40	198.77	1933	2641.17	519.14	224.66	1791	2534.80
6	absn30	611.33	202.34	2298	3111.67	687.80	298.55	2299	3285.35
7	absn35	679.39	231.44	2301	3211.83	738.66	359.30	2355	3452.96
8	absn40	798.45	278.77	2483	3560.22	826.40	373.76	2424	3624.16
9	absn45	889.99	334.28	2590	3814.27	829.50	416.38	2708	3953.88
10	absn50	900.50	360.60	2607	3867.87	1086.90	476.70	2900	4463.60
	Average	**543.27**	**200.04**	**1923.18**	**2666.48**	**574.40**	**259.00**	**1900.60**	**2734.01**
High inventory holding cost and Time period = 3									
Name of Instance		IRPT_ML-based Randomised CWS and IG with LS				IRPT_OU-based Randomised CWS and IG with LS			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	866.90	0.00	809	1675.90	899.90	0.00	699	1598.90
2	absn10	2228.12	89.00	1519	3836.23	2321.11	103.30	1505	3929.41
3	absn15	2453.88	130.56	1999	4583.43	2899.30	177.43	2088	5164.73
4	absn20	3696.34	200.50	2540	6436.84	3691.26	240.93	2977	6909.19
5	absn25	4432.22	289.05	3002	7723.27	4999.96	367.88	3596	8963.84
6	absn30	5667.93	363.33	4009	10040.26	6214.56	402.43	4101	10717.99
7	absn35	6001.60	370.17	4115	10486.77	6609.69	444.44	4222	11276.13
8	absn40	7731.28	417.77	4663	12812.05	7757.89	532.50	4891	13181.39
9	absn45	8200.28	454.32	5080	13734.60	8232.20	545.55	5209	13986.75
10	absn50	8990.91	480.60	5200	14671.51	9445.25	609.78	6006	16061.03
	Average	**5026.95**	**279.53**	**3293.61**	**8600.09**	**5307.11**	**342.42**	**3529.40**	**9178.94**
Low inventory holding cost and Time period = 6									
Name of Instance		IRPT_ML-based Randomised CWS and IG with LS				IRPT_OU-based Randomised CWS and IG with LS			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	360.30	0.00	2177	2537.30	399.99	0.00	2067	2466.99
2	absn10	509.45	100.50	3369	3978.95	600.10	123.50	3032	3755.60
3	absn15	600.34	200.40	4022	4822.74	637.44	234.32	3989	4860.76
4	absn20	701.09	301.10	4930	5932.19	731.21	359.99	5045	6136.20
5	absn25	890.00	365.44	5080	6335.44	916.76	409.45	5501	6827.21
6	absn30	909.30	403.23	5922	7234.53	1033.20	490.00	6005	7528.20
	Average	**661.75**	**228.45**	**4250.00**	**5140.19**	**719.78**	**269.54**	**4273.17**	**5262.49**
High inventory holding cost and Time period = 6									
Name of Instance		IRPT_ML-based Randomised CWS and IG with LS				IRPT_OU-based Randomised CWS and IG with LS			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	absn05	2032.34	0.00	2008	4040.34	2399.54	0.00	2300	4699.54
2	absn10	3769.45	98.88	3102	6970.33	3939.39	111.11	3893	7943.50
3	absn15	5504.50	179.80	4892	10576.30	5656.98	199.89	5025	10881.87
4	absn20	7004.89	260.34	5378	12643.23	6973.20	378.56	6212	13563.76
5	absn25	7984.55	329.45	8390	16704.00	8101.01	445.07	8324	16870.08
6	absn30	10034.30	444.44	9093	19571.74	10394.66	555.45	9324	20274.11
	Average	**6055.01**	**218.82**	**5477.17**	**11750.99**	**6244.13**	**281.68**	**5846.33**	**12372.14**

Table 5.4 : Average result of cost breakdown of the IRPT-based Randomised CWS and IG algorithm with local search under ML/OU, p=3 and p= 6

5.4.2.1. *Comparison of the results of IRPT-based Randomised CWS and IG with local search, with the best results in the IRPT-based Randomised CWS*

For the case of the IRP with transhipment, the results provided by Randomised CWS and IG algorithm with local search and the IRPT-based Randomised CWS heuristic are compared. The solutions on all sets of instances in this chapter can be compared with the aforementioned best-known solutions of Chapter 4 as shown in Table 5.5. The comparison of the average results between these proposed methods for solving

IRPT in Chapter 4 and Chapter 5 is consistent with IRP as explained in the section 5.4.1. They are the average results combined in four classes on all sets of instances for IRPT under ML/OU replenishment policies. The first class comprises the instances set with low inventory costs and $p = 3$. The second class comprises the set of instances with high inventory costs and $p = 3$. The third class comprises the instances sets with low inventory costs and $p = 6$. The fourth class comprises the instances sets with high inventory costs and $p = 6$.

Table 5.5 shows the average total costs found by the different proposed algorithms. This chapter proposed method is able to find better solutions than the method proposed by Chapter 4 for on all sets of instances. The IRPT_ML-based Randomised CWS and IG algorithm with local search have the average results 4.36% better than those average results provided by the IRPT_ML-based Randomised CWS for the first class on all sets of instances. For the IRPT_OU, this chapter proposed method obtained better average total costs than Randomised CWS, which is on average 4.29%, for the same class on all sets instances. The second class on all instances sets, the solution obtained by this chapter proposed method have better than those obtained by Chapter 4 proposed method, they are on average 0.46% and 5.16% better for the IRPT under ML/OU policies, respectively. The solutions of the IRPT-based Randomised CWS under ML and OU policies have 1.36% and 0.43% higher for the third class on all sets of instances. While, the forth class on all instances sets, Randomised CWS and IG algorithm with local search obtained the average total costs are on average 1.44% and 1.59% lower than those Randomised CWS heuristic for the IRPT under ML/OU policies, respectively.

Notices that using Randomised CWS and IG algorithm with local search for solving the IRPT under ML/OU policies is able to find good solution. Therefore, applying IG algorithm with local search can lead to increase the performance of Randomised CWS heuristic. This improve algorithm extend the searching to find better solution during the iteration. It is obvious that from the results in the same data set and the class instances, this chapter proposed algorithm can obtain better results especially when instance size is larger as shown in Table 5.5.

Low inventory holding cost and Time period = 3						
Name of insnatce	IRPT_ML-based Randomised CWS	IRPT_ML-based Randomised CWS and IG with LS	Improvement (%)	IRPT_OU-based Randomised CWS	IRPT_OU-based Randomised CWS and IG with LS	Improvement (%)
	(B)	(B)	(A-B)/B	(B)	(B)	(A-B)/B
absn05	754.38	743.79	1.42	754.38	754.38	0.00
absn10	1588.46	1585.61	0.18	1700.73	1616.16	5.23
absn15	1900.75	1844.88	3.03	1975.93	1855.19	6.51
absn20	2381.66	2280.77	4.42	2386.86	2343.99	1.83
absn25	2735.93	2409.38	13.55	2749.03	2709.67	1.45
absn30	2990.72	2804.56	6.64	3370.70	3169.81	6.34
absn35	3211.45	3089.45	3.95	3548.75	3308.54	7.26
absn40	3400.20	3287.32	3.43	3778.48	3595.63	5.09
absn45	3562.32	3377.17	5.48	4081.47	3880.13	5.19
absn50	4059.30	3998.89	1.51	4582.29	4404.41	4.04
Average	**2658.52**	**2542.18**	**4.36**	**2892.86**	**2763.79**	**4.29**
High inventory holding cost and Time period = 3						
Name of insnatce	IRPT_ML-based Randomised CWS	IRPT_ML-based Randomised CWS and IG with LS	Improvement (%)	IRPT_OU-based Randomised CWS	IRPT_OU-based Randomised CWS and IG with LS	Improvement (%)
	(B)	(B)	(A-B)/B	(B)	(B)	(A-B)/B
absn05	1660.54	1660.28	0.02	1782.61	1664.39	7.10
absn10	3998.84	3989.89	0.22	4106.76	4043.99	1.55
absn15	5061.19	5055.49	0.11	5260.77	5069.53	3.77
absn20	6854.22	6777.34	1.13	7041.41	6888.88	2.21
absn25	8568.38	8092.22	5.88	8979.99	8677.11	3.49
absn30	10555.20	10345.91	2.02	10877.14	10702.22	1.63
absn35	11359.07	11200.97	1.41	11380.55	11196.26	1.65
absn40	12968.37	11067.43	17.18	13136.81	12009.63	9.39
absn45	13913.98	12768.54	8.97	14400.25	13130.33	9.67
absn50	15281.54	14747.88	3.62	16672.17	15001.63	11.14
Average	**9022.13**	**8570.60**	**4.06**	**9363.85**	**8838.40**	**5.16**
Low inventory holding cost and Time period = 6						
Name of insnatce	IRPT_ML-based Randomised CWS	IRPT_ML-based Randomised CWS and IG with LS	Improvement (%)	IRPT_OU-based Randomised CWS	IRPT_OU-based Randomised CWS and IG with LS	Improvement (%)
	(B)	(B)	(A-B)/B	(B)	(B)	(A-B)/B
absn05	2554.40	2554.40	0.00	2564.40	2561.84	0.10
absn10	4095.17	4095.17	0.00	3975.17	3889.75	2.20
absn15	4810.23	4801.09	0.19	4876.96	4858.90	0.37
absn20	6059.01	6021.56	0.62	6359.01	6201.11	2.55
absn25	6809.26	6809.26	0.00	6811.26	7276.88	-6.40
absn30	7629.87	7108.11	7.34	7987.81	7698.44	3.76
Average	**5326.32**	**5231.60**	**1.36**	**5429.10**	**5414.49**	**0.43**
High inventory holding cost and Time period = 6						
Name of insnatce	IRPT_ML-based Randomised CWS	IRPT_ML-based Randomised CWS and IG with LS	Improvement (%)	IRPT_OU-based Randomised CWS	IRPT_OU-based Randomised CWS and IG with LS	Improvement (%)
	(B)	(B)	(A-B)/B	(B)	(B)	(A-B)/B
absn05	4748.99	4748.99	0.00	4762.79	4761.03	0.04
absn10	7940.74	7940.74	0.00	8034.79	8033.79	0.01
absn15	10954.53	10944.85	0.09	11100.44	11028.24	0.65
absn20	14155.34	14055.84	0.71	14302.05	14014.51	2.05
absn25	16780.26	16080.48	4.35	16966.48	16629.99	2.02
absn30	20737.41	20037.13	3.49	20994.48	20043.12	4.75
Average	**12552.88**	**12301.34**	**1.44**	**12693.51**	**12418.45**	**1.59**

Table 5.5 : Comparison average results between the IRP–based Randomised CWS heuristic and IRP-based Randomised CWS heuristic and IG algorithm with local search under ML/OU, p=3, 6

5.4.2.2. *Comparison of the results of IRPT-based Randomised CWS and IG with local search, with the lower bound and the best results in the literature*

Table 5.6 shows the comparison among the results obtained by the IRPT-based Randomised CWS heuristic and IG with local search, the results (UB) truncated by CPLEX (Leandro et al., 2012). The results obtained by ALNS (Leandro et al. 2012) are also presented in this table. These results on all sets of instances give the average of five instances generated for each combination of the number of customer (n)and the time period (p). In the performed experiments, this chapter implemented the proposed Randomised CWS and IG algorithm with local search by using on all sets of instances, which has been used in the literature. The computational results provided in term of the average solution costs which is the best solution.

The performance of Randomised CWS heuristic and IG with local search and the best performing algorithm in the literature are compared as shown in Table 5.6. This table gives, the column 'LB' and 'UB' summarised the average solutions obtained by CPLEX (Leandro et al., 2012). The column 'ALNS' summarised the average results obtained by Adapted Large Neighbourhood Search (Coelho et al., 2012). The column 'IRPT_ML-based-Randomised CWS and IG with LS' and IRPT_OU-based Randomised CWS and IG with LS' summarised the average results obtained by applying Randomised CWS and IG algorithm with local search for solving IRP under ML and OU inventory replenishment policies. The table below contains of the following: the bottom row divides into two columns, which present the Low/High inventory holding cost and the time horizon $p = 3$ ($n = 50$) and $p = 6$ ($n = 30$). The average total costs provided by IRPT-based Randomised CWS and IG algorithm with local search, and the average %gap between this chapter proposed algorithm against the best-known LB solutions and the results of LB are compared with UB and ALNS (Leandro et al., 2012) are presented in the bold numbers. While, the good solutions are found by this chapter proposed approaches are worse than those LB but they have better than those best-known solutions obtained by UB and ALNS are presented in red and green numbers, respectively.

Using Randomised CWS and IG algorithm with local search is able to find better solutions than the results obtained by UB for most of the instances on all sets. The solutions represent the average total costs of the IRPT under ML/OU policies. For the instances sets with low inventory holding costs and the time horizon is equal to 3, the Randomised CWS and IG algorithm with local search is able to find better solution than UB, they are better on six sets of instances for the IRPT under ML and OU policies (see red numbers in Table 5.6). The solutions obtained by this chapter proposed algorithm are on eight instances sets better than those obtained by CPLEX and on average 0.94% for the IRPT_ML. They have five instances sets better for the IRPT_OU with three time periods and high inventory holding costs. For the IRPT_ML, this chapter solutions have five instances sets better than those in six time periods and low holding costs and on average 27.08%. They are on average 28.63% and have five sets of instances better than those of UB results for the IRPT under OU policy with the same class instances. When p=6 and high holding costs, this chapter proposed method provided the average total costs on average 8.94 % and they have five sets of the instances better than those UB for the IRPT under ML and policies, while the IRPT_OU, they are on average 9.61% for all instances sets.

The following table shows the running time of these algorithms for solving the IRPT. Here, computation time is taken for finding the solutions on all instances sets. Randomised CWS and IG algorithm with local search performed within 15 minutes for low/high inventory holing costs of the IRPT under ML policy and on all instances with up to 50 customers when $p = 3$, up to 30 customers when $p = 6$. For low/high holding costs with three time periods, all instances with up to 50 clients can be performed in less than one hour, while when the time period is equal to 6, only instances with up to 30 clients can be performed in less than 30 minutes. Therefore, this chapter proposed algorithm requires less computation time than those algorithms in literature.

It should be noticed that when the IG algorithm with local search is applied after Randomised CWS heuristic, the results on the benchmark problem are improved. This is reasonable that this chapter proposed algorithm can provide better quality solutions on average than the approach, which has been presented in Chapter 4.

The IG algorithm with local search approach clearly outperformance Randomised CWS heuristic for the IRPT under ML/OU policies in most of the instances on all sets.

Table 5.6 is also clearly shows comparison of the two average results obtained by this chapter proposed method and ALNS (Coelho et al., 2012). Experiments are reported which show that this chapter proposed algorithm often finds the solutions more quickly than the proposed algorithm by Coelho et al. (2012) and it is significantly more efficient than ALNS when the instances become large (see green numbers in Table 5.6). Which can be seen the green numbers in Table 5.6 present the higher total costs than those obtained by the Randomised CWS and IG algorithm with local search thus the Randomised CWS and IG algorithm with local search can be performed the best heuristic solution. This chapter proposed algorithm performed much better solutions than those obtained by ALNS for most of the instances on all sets. For the IRPT under ML /OU policies on the set with low inventory holding costs and $p = 3$, they provide on average 6.66% and 13.16% better results on all sets of instances, respectively. When $p = 3$ and high inventory holding costs, they are on average 0.94% better on all instances and nine instances sets have lower total costs for the IRPT_ML. While for the IRPT_OU, they have eight sets of instances have lower total costs for the same class set.

For the class instances set on low inventory holding costs with the time period = 6, this chapter approach is performed better average total costs than those ALNS on three instances sets for the IRPT under ML policy, while the IRPT under OU replenishment policies, they have better on all sets of instances. Moreover, $p = 6$ and high inventory holding costs, they are five sets of instances better than those results solved by ALNS for the IRPT_ML, whereas the IRPT_OU, they have four sets of instances (see green numbers) better than for the same class instances set.

From the results, it can be concluded that the Randomised CWS and IG algorithm with local search can also have better results when compared to ALNS. It is remarkable that this chapter proposed algorithm requires less time than those algorithms to solve the IRPT on all instances with up to 50 customers. Additionally, only a few average total costs obtained by ALNS are lower than the

average total costs obtained by IRPT-based Randomised CWS heuristic and IG with local search. Therefore, among heuristic algorithms, it is obvious that this chapter proposed method shows a better performance in terms of solution quality. Thus, this approach has the potential in providing good quality solutions. With regard to computer runtime, this proposed method also gives a shorter computer run time on average in both UB and ANLS. Consequently, the implement of the Randomised CWS heuristic by applying IG algorithm with local search can contribute to better solutions with larger instance size and more complicated problem.

Moreover, Fig 5.3 shows the average percentage gaps when the results of the IRPT under ML/OU policies obtained by ALNS and this thesis proposed methods are compared to the results of LB achieved though CPLEX. It can be seen that the IRP-based Randomised CWS and IG with local search under ML/OU policies can provide the lowest average percentage gaps than those solution costs found by UB, ALNS, CWS, and Randomised CWS.

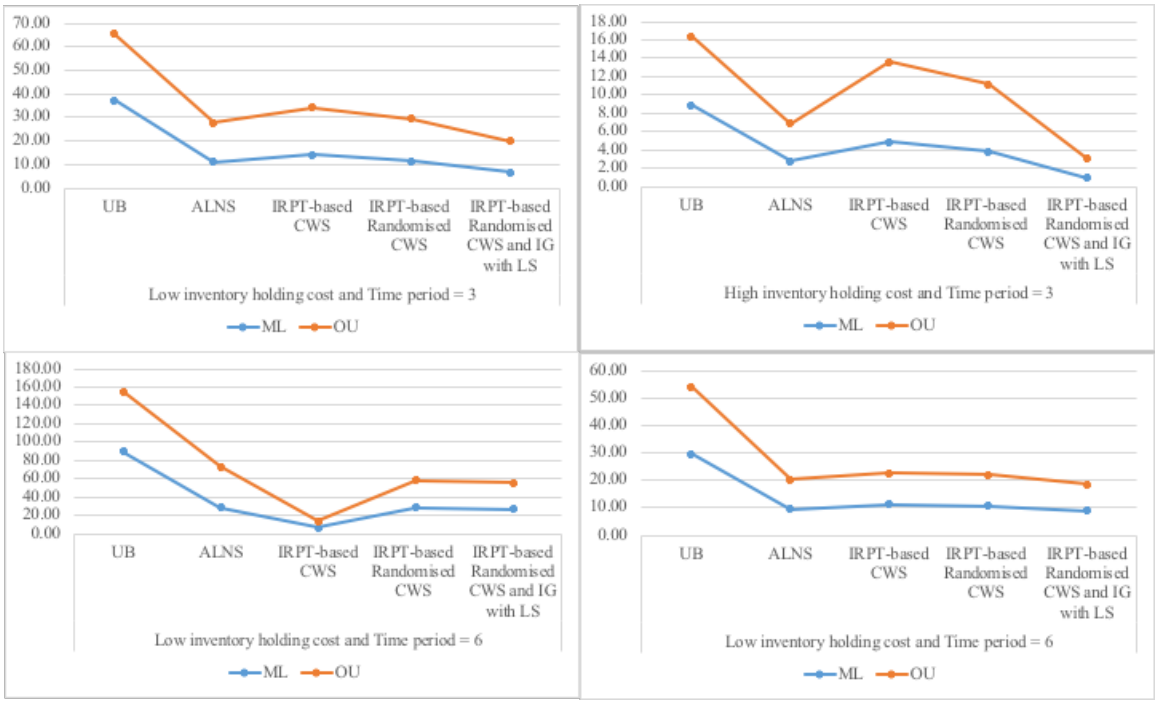

Figure 5.3 : Average percentage gap of algorithms when compare to LB

Low inventory holding cost and Time period = 3											High inventory holding cost and Time period = 3									
	LB	UB	Time (s)	Gap (%)	ALNS		Gap (%)	IRPT_ML-based Randomised CWS and IG with LS		Gap (%)	LB	UB	Time (s)	Gap (%)	ALNS		Gap (%)	IRPT_ML-based Randomised CWS and IG with LS		Gap (%)
	(A)	(C)		(C-A)/A	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A	(A)	(C)		(C-A)/A	(D)	Time (s)	(D-A)/A	(B)	Time (s)	(B-A)/A
absn05	741.76	741.76	0.29	0.00	744.80	5.28	0.41	743.79	5.65	0.27	1660.26	1660.26	0.72	0.00	1660.26	5.98	0.00	1660.28	5.89	0.00
absn10	1577.23	1577.30	2.64	0.00	1586.91	14.52	0.61	1585.61	14.98	0.53	3998.82	3999.02	3.17	0.01	4011.81	15.36	0.32	3999.89	14.87	0.03
absn15	1839.88	1840.06	58.46	0.01	1849.83	30.64	0.54	1844.88	29.65	0.27	5054.12	5054.58	35.86	0.01	5061.92	33.76	0.15	5055.49	30.03	0.03
absn20	2232.38	2278.04	1143.92	2.05	2290.55	56.74	2.61	2280.77	55.03	2.17	6752.82	6822.95	1134.22	1.04	6869.26	57.73	1.72	6777.34	56.19	0.36
absn25	2367.55	2657.38	1920.44	12.24	2579.18	92.04	8.94	2409.38	89.34	1.77	8342.03	8714.21	1496.09	4.46	8562.59	91.26	2.64	8392.22	88.89	0.60
absn30	2735.52	3116.05	2006.29	13.91	2985.99	165.20	9.16	2804.56	150.76	2.52	10319.56	10716.07	1586.59	3.84	10557.63	159.13	2.31	10345.91	159.01	0.26
absn35	2742.38	4034.69	1200.07	47.12	3448.17	248.74	25.74	3089.45	232.44	12.66	10689.39	11511.05	1191.40	7.69	11309.46	282.47	5.80	11200.97	280.45	4.79
absn40	2799.85	4480.11	1992.43	60.01	3361.80	348.93	20.07	3287.32	350.30	17.41	11573.32	14329.19	1643.60	23.81	12165.28	360.83	5.11	11667.43	360.50	0.81
absn45	3034.49	5110.99	1613.87	68.43	3697.61	461.71	21.85	3377.17	416.56	11.29	12979.04	15822.42	2143.09	21.91	13699.96	627.96	5.55	12998.54	580.60	0.15
absn50	3396.46	9172.72	2301.51	170.07	4071.09	760.30	19.86	3998.89	689.34	17.74	14406.63	18144.64	2461.43	25.95	15004.18	939.87	4.15	14747.88	892.34	2.37
Average	**2346.75**	**3500.91**	**1223.99**	**37.38**	**2661.59**	**218.41**	**10.98**	**2542.18**	**203.41**	**6.66**	**8577.60**	**9677.44**	**1169.62**	**8.87**	**8890.24**	**257.44**	**2.78**	**8684.60**	**246.88**	**0.94**
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based Randomised CWS and IG with LS**		**Gap (%)**	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based Randomised CWS and IG with LS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**	**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	745.39	745.39	0.06	0.00	745.39	6.56	0.00	754.38	5.94	1.21	1664.38	1664.38	0.07	0.00	1664.38	8.06	0.00	1664.39	6.03	0.00
absn10	1616.08	1616.11	2.29	0.00	1617.51	29.55	0.09	1616.16	21.40	0.00	4043.83	4043.94	2.40	0.00	4043.94	30.12	0.00	4043.99	23.65	0.00
absn15	1851.02	1851.13	19.58	0.01	1864.70	82.24	0.74	1855.19	76.80	0.23	5069.04	5069.51	20.64	0.01	5116.21	74.63	0.93	5069.53	70.44	0.01
absn20	2274.24	2339.63	557.58	2.88	2442.44	183.28	7.40	2343.99	176.70	3.07	6818.37	6877.20	684.51	0.86	6927.36	151.21	1.60	6888.88	154.50	1.03
absn25	2421.25	2710.08	1887.00	11.93	2724.67	389.10	12.53	2709.67	254.90	11.91	8422.54	8611.29	2100.71	2.24	8754.03	350.23	3.94	8677.11	333.66	3.02
absn30	2812.98	3123.22	2037.20	11.03	3341.49	635.79	18.79	3169.81	598.20	12.69	10332.37	10765.01	1362.58	4.19	10867.17	521.52	5.18	10702.22	521.01	3.58
absn35	2772.64	4316.84	1339.51	55.69	3522.66	895.11	27.05	3308.54	895.02	19.33	10695.61	11608.97	1622.33	8.54	11367.04	930.08	6.28	11196.26	909.17	4.68
absn40	2828.15	4531.14	1760.32	60.22	3795.60	1577.21	34.21	3595.63	1593.30	27.14	11588.44	13117.59	1427.80	13.20	12563.16	1577.64	8.41	12009.63	1405.60	3.63
absn45	3090.45	4540.60	1511.04	46.92	4078.89	2350.23	31.98	3880.13	2004.45	25.55	13069.51	16468.68	1427.33	26.01	13921.34	2244.01	6.52	13130.33	3240.83	0.47
absn50	3376.61	6491.95	1610.72	92.26	4581.07	2898.06	35.67	4404.41	2784.32	30.44	14418.41	17326.09	2009.60	20.17	15560.06	3375.50	7.92	15001.63	2556.71	4.04
Average	**2378.88**	**3226.61**	**1072.53**	**28.09**	**2871.44**	**904.71**	**16.85**	**2763.79**	**841.10**	**13.16**	**8612.25**	**9555.27**	**1065.80**	**7.52**	**9078.47**	**926.30**	**4.08**	**8838.40**	**922.16**	**2.05**
Low inventory holding cost and Time period = 6											**High inventory holding cost and Time period = 6**									
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_ML-based Randomised CWS and IG with LS**		**Gap (%)**	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_ML-based Randomised CWS and IG with LS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**	**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	2554.37	2554.45	2.62	0.00	2558.37	10.90	0.16	2554.40	10.89	0.00	4741.97	4742.19	2.23	0.00	4748.31	13.28	0.13	4748.99	13.09	0.15
absn10	3443.84	4056.36	1660.45	17.79	4095.10	36.70	18.91	4095.17	26.70	18.91	7429.95	8015.35	2035.50	7.88	7961.72	37.91	7.16	7940.74	30.59	6.87
absn15	3619.79	5114.79	976.65	41.30	4834.73	80.68	33.56	4801.09	71.88	32.63	9933.52	11262.90	813.05	13.38	10949.14	88.45	10.22	10944.85	84.98	10.18
absn20	4385.61	9880.24	1367.07	125.29	6020.83	174.24	37.29	6021.56	159.65	37.30	12321.4	17437.75	1166.17	41.52	14152.04	179.70	14.86	14055.84	179.01	14.08
absn25	4679.71	13786.58	1769.39	194.60	6808.40	295.30	45.49	6809.26	294.90	45.51	14333.9	21962.55	1937.04	53.22	16320.18	329.51	13.86	16080.48	302.34	12.18
absn30	5547.73	14296.67	2336.32	157.70	7466.89	671.24	34.59	7108.11	670.50	28.13	18182.52	29215.44	2064.83	60.68	20235.50	787.47	11.29	20037.13	667.87	10.20
Average	**4038.51**	**8281.52**	**1352.08**	**89.45**	**5297.39**	**211.51**	**28.33**	**5231.60**	**205.75**	**27.08**	**11157.21**	**15439.36**	**1336.47**	**29.45**	**12394.48**	**239.39**	**9.59**	**12301.34**	**212.98**	**8.94**
Name of insnatce	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based Randomised CWS and IG with LS**		**Gap (%)**	**LB**	**UB**	**Time (s)**	**Gap (%)**	**ALNS**		**Gap (%)**	**IRPT_OU-based Randomised CWS and IG with LS**		**Gap (%)**
	(A)	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**	**(A)**	**(C)**		**(C-A)/A**	**(D)**	**Time (s)**	**(D-A)/A**	**(B)**	**Time (s)**	**(B-A)/A**
absn05	2561.77	2561.84	1.35	0.00	3299.97	16.23	28.82	2561.84	16.03	0.00	4759.29	4759.54	2.21	0.01	4760.94	14.98	0.03	4761.03	13.69	0.04
absn10	3524.75	4054.39	2221.58	15.03	4832.87	70.29	37.11	3889.75	63.32	10.36	7495.05	8077.14	1477.52	7.77	8038.53	65.29	7.25	8033.79	59.96	7.19
absn15	3880.76	5042.41	1029.67	29.93	5582.80	208.08	43.86	4858.90	187.75	25.20	10037.35	11225.20	1109.10	11.83	11027.72	156.07	9.87	11028.24	201.56	9.87
absn20	4396.94	7418.80	1382.76	68.73	6857.90	491.31	55.97	6201.11	442.42	41.03	12332.22	16778.25	1207.34	36.05	14278.32	442.43	15.78	14014.51	430.44	13.64
absn25	4641.33	10566.48	1834.25	127.66	7487.80	805.16	61.33	7276.88	775.14	56.78	14262.97	20807.23	1540.39	45.88	16867.70	906.47	18.26	16629.99	886.59	16.60
absn30	5562.96	13917.82	1590.49	150.19	7888.56	1650.37	41.81	7698.44	1693.80	38.39	18169.95	26411.34	1951.69	45.36	20492.18	1718.08	12.78	20043.12	1608.17	10.31
Average	**4094.75**	**7260.29**	**1343.35**	**65.26**	**5991.65**	**540.24**	**44.82**	**5414.49**	**529.74**	**28.63**	**11176.14**	**14676.45**	**1214.71**	**24.48**	**12577.57**	**550.55**	**10.66**	**12418.45**	**533.40**	**9.61**

Table 5.6 : Comparison average results of LB, UB, ALNS and IRPT-based Randomised CWS heuristic and IG algorithm with local search under ML/OU, p=3, 6

5.4.2.3. *Comparison of the results of the IRPT-based Randomised CWS and IG with local search, and the IRP-based Randomised CWS and IG with local search*

Table 5.7 shows the comparison of experimental, IRP and IRPT are obtained by Randomised CWS heuristic and IG algorithms with local search under ML/OU replenishment policies on four classes' instances sets. This table clarifies the results of the low or high inventory cost in each time period ($p = 3, p = 6$) for the IRP and IRPT under ML/OU replenishment policies, which is represented as IRP_ML-based Randomised CWS and IG with local search, IRP_OU-based Randomised CWS and IG with local search, IRPT_ML-based Randomised CWS and IG with local search, and IRPT_OU-based Randomised CWS and IG with local search. The first column represents the instance name and the second, the third, the fifth and the sixth columns give the average results of IRP_ML and IRPT_ML, respectively. Then the fourth and the seventh columns present the average percentage gaps.

The average total costs of two different approaches; the IRP without transhipment and with transhipment are presented in Table 5.7. The IRPT-based Randomise CWS and IG with local search performed better results than the IRP-based Randomised CWS and IG with local search on all sets of instances under both ML/OU inventory policies. For the IRPT_ML and IRPT_OU, they have 22.20% and 28.44% lower than those the average total costs of the IRP_ML and IRP_OU for low inventory holding costs and $p = 3$. While at the same time period and high inventory holding costs, the IRPT_ML provides 9.01% better than those of the IRP_ML and the IRPT_OU is 12.17% better than those solitons for the IRP_OU on the same all sets of instances. When the time period is equal to 6, the IRP with transhipment is also able to perform better those solutions of the IRP, in which these are solved by Randomised CWS and IG algorithm with local search on the same all sets of instances. They are 14.33% and 13.82% better than those the IRP for low holding costs with under ML and OU policies, respectively. For high inventory holding costs on the same all sets of instances, they are on average 7.58% for IRPT_ML and 7.67% for IRPT_OU better than those the average total costs provided by the IRP-based Randomised CWS and IG algorithm with local search under ML and OU replenishment policies, respectively. Which can be seen form the percentage improvement, the IRPT show lower averages total costs than those characterised by the IRPs. As the benefit from allowing transhipment is taken into account (IRP).

It is remarkable that the IRPT_ML/OU-based Randomised CWS heuristic and IG algorithms with local search performs better results than those obtained by IRP_ML/OU-based Randomised CWS heuristic and IG algorithms with local search. This is because the transhipment can be advantageous in a deterministic context in which no shortage occurs as they may yield both an overall reduced distribution and inventory holding cost. Also, solving the IRP without transhipment ($b_{ij} = 1.00c_{ij}$) has set the transhipment cost sufficiently larger than the IRPT ($b_{ij} = 0.01c_{ij}$). Using the transhipment can lead to reduce the transportation costs (routing and transhipment) then the total costs of the IRPs are higher than IRPTs. However, companies decide to arrange a lateral transhipment only in the case of an emergency or in the case of preventing stock-out caused by high demand. With regard to the means of transportation, the companies prefer to use their vehicles for direct delivery. Therefore, the reason for avoiding transhipment is that the companies have to pay more, not only in shipping costs, but also other operational costs.

Low inventory holding cost and Time period = 3				High inventory holding cost and Time period = 3		
Name of insnatce	IRP_ML-based Randomised CWS and IG with LS	IRPT_ML-based Randomised CWS and IG with LS	Improvement (%)	IRP_ML-based Randomised CWS and IG with LS	IRPT_ML-based Randomised CWS and IG with LS	Improvement (%)
	(A)	(B)	(A-B)/B	(A)	(B)	(A-B)/B
absn05	1275.99	743.79	71.55	2201.22	1660.28	32.58
absn10	1911.08	1585.61	20.53	4339.09	3999.89	8.48
absn15	2207.99	1844.88	19.68	5437.01	5055.49	7.55
absn20	2675.82	2280.77	17.32	7240.47	6777.34	6.83
absn25	2998.76	2409.38	24.46	9000.88	8392.22	7.25
absn30	3327.02	2804.56	18.63	10920.32	10345.91	5.55
absn35	3497.52	3089.45	13.21	11435.05	11200.97	2.09
absn40	3710.50	3287.32	12.87	12579.45	11667.43	7.82
absn45	3875.82	3377.17	14.77	13894.99	12998.54	6.90
absn50	4356.53	3998.89	8.94	15486.75	14747.88	5.01
Average	**2983.70**	**2542.18**	**22.20**	**9253.52**	**8684.60**	**9.01**
Name of insnatce	**IRP_OU-based Randomised CWS and IG with LS**	**IRPT_OU-based Randomised CWS and IG with LS**	**Improvement (%)**	**IRP_OU-based Randomised CWS and IG with LS**	**IRPT_OU-based Randomised CWS and IG with LS**	**Improvement (%)**
	(A)	**(B)**	**(A-B)/B**	**(A)**	**(B)**	**(A-B)/B**
absn05	1418.99	754.38	88.10	2354.18	1664.39	41.44
absn10	2229.01	1616.16	37.92	4490.76	4043.99	11.05
absn15	2493.48	1855.19	34.41	5737.33	5069.53	13.17
absn20	3054.77	2343.99	30.32	7623.17	6888.88	10.66
absn25	3451.50	2709.67	27.38	9466.04	8677.11	9.09
absn30	3644.27	3169.81	14.97	11331.01	10702.22	5.88
absn35	3848.49	3308.54	16.32	11840.58	11196.26	5.75
absn40	4139.09	3595.63	15.11	13022.43	12009.63	8.43
absn45	4277.67	3880.13	10.25	14389.98	13130.33	9.59
absn50	4830.01	4404.41	9.66	15998.97	15001.63	6.65
Average	**3338.73**	**2763.79**	**28.44**	**9625.45**	**8838.40**	**12.17**
Low inventory holding cost and Time period = 6				**High inventory holding cost and Time period = 6**		
Name of insnatce	**IRP_ML-based Randomised CWS and IG with LS**	**IRPT_ML-based Randomised CWS and IG with LS**	**Improvement (%)**	**IRP_ML-based Randomised CWS and IG with LS**	**IRPT_ML-based Randomised CWS and IG with LS**	**Improvement (%)**
	(A)	**(B)**	**(A-B)/B**	**(A)**	**(B)**	**(A-B)/B**
absn05	3288.33	2554.40	28.73	5514.50	4748.99	16.12
absn10	4664.55	4095.17	13.90	8667.55	7940.74	9.15
absn15	5465.48	4801.09	13.84	11601.89	10944.85	6.00
absn20	6699.99	6021.56	11.27	14770.36	14055.84	5.08
absn25	7380.26	6809.26	8.39	17101.04	16080.48	6.35
absn30	7807.13	7108.11	9.83	20589.89	20037.13	2.76
Average	**5884.29**	**5231.60**	**14.33**	**13040.87**	**12301.34**	**7.58**
Name of insnatce	**IRP_OU-based Randomised CWS and IG with LS**	**IRPT_OU-based Randomised CWS and IG with LS**	**Improvement (%)**	**IRP_OU-based Randomised CWS and IG with LS**	**IRPT_OU-based Randomised CWS and IG with LS**	**Improvement (%)**
	(A)	**(B)**	**(A-B)/B**	**(A)**	**(B)**	**(A-B)/B**
absn05	3299.98	2561.84	28.81	5538.30	4761.03	16.33
absn10	4832.91	3889.75	24.25	8872.42	8033.79	10.44
absn15	5581.04	4858.90	14.86	11731.08	11028.24	6.37
absn20	6847.57	6201.11	10.42	14880.11	14014.51	6.18
absn25	7463.01	7276.88	2.56	17202.22	16629.99	3.44
absn30	7854.38	7698.44	2.03	20700.05	20043.12	3.28
Average	**5979.82**	**5414.49**	**13.82**	**13154.03**	**12418.45**	**7.67**

Table 5.7 : Comparison average results between the IRP and IRPT –based Randomised CWS heuristic and IG algorithm with local search under ML/OU, p=3, 6

5.5. Chapter Conclusion

This chapter focuses on minimising total costs of the deterministic IRP and IRPT under OU or ML replenishment policies by using Randomised CWS heuristic and IG algorithm with local search. The combination of the two methods seem to have performed an essential role in developing a better routing to minimise the transportation costs in the IRP and the IRPT. This chapter has described the way of implementing IG algorithm with local search to Randomised CWS heuristic. The IG algorithm, despite its simplicity, is very competitive and is a very efficient heuristic. It is reasonable to select the IG algorithm, which has been adapted successfully for many COPs. Moreover, applying the local search into the IG algorithm is the concepts, which are able to make the parallels between those algorithms become more obvious. Owing to the IG algorithm with local search adapts the lowest cost found by changing it to give a better cost. Which can be describes as the current cost solution is replaced by the better cost solution. Otherwise, if there is no lower cost than the current cost then the current cost is considered to be the best solution cost. The search procedure is repeated until a pre-specified stopping condition is satisfied. Therefore, this chapter proposes the implementation of combined Randomised CWS heuristic and IG algorithm with local search. With using the IG algorithm with local search after applying Randomised CWS heuristic to obtain high quality solution especially for the IRP and the IRPT. This is the basic concept of the contribution is to apply the IG algorithm with the local search in order to improve solutions for the deterministic IRP and IRPT.

For the performed experiments, by using well-known benchmark instances, the implementation of these approaches enhanced solutions. Firstly, the results obtained by Randomised CWS heuristic, which has proposed in Chapter 4 and Randomised CWS heuristic and IG algorithm with local search are compared. Secondly, for the IRP, the proposed method's solution is compared by a branch and cut, algorithm (ABLS) (Archetti et al., 2007), while for the IRPT, the upper bound (LB) and the upper bound (UB) solution produced by CPLEX and the solution obtained by ALNS (Leandro et al., 2012) are compared. Finally, the solutions of the two different approaches are compared; the results provided by the IRP-based Randomised CWS heuristic and IG algorithm with local search and the results obtained by the IRPT-based Randomised CWS heuristic and IG algorithm with local search under ML/OU replenishment policies.

From the experiments solutions that the IG algorithm with local search can improve the heuristic to get high quality solution for the deterministic IRP and IRPT, which can be concluded as following:

- This chapter proposed approach is able to improve the solutions on all sets of the IRP and the IRPT under ML/OU replenishment policies with low/high inventory holding costs and the time periods are equal to 3 and 6. They are on average 3.02% and 3.20 % better than those solution obtained by Chapter 4 proposed approach for on the same instances sets of the IRP and the IRPT, respectively.
- The algorithm proposed by this chapter can provided near-optimal solutions, especially on sets of large instances for both the IRP and the IRPT. For all four classes' instances of the IRP, they are on average 0.26% slightly higher than those costs solutions solved by ABLS. This is reasonable that ABLS is the exact method, which always provides optimal solution. While, this chapter proposed method is a heuristic algorithm, which is not guaranteed to find an optimal solution but, in practice, it is often able to find good solution, which short time is taken. However, for the IRPT on the same set, Randomised CWS and IG algorithm with local search is able to find better solutions than the results obtained by UB for most of the instances on all sets. Even through these results are worse than LB but it has taken less computation time for solving the IRPT.
- For both the IRP and IRPT that Randomised CWS and IG algorithm with local search can lead to improve the best heuristic solution. This chapter proposed algorithm is able to find better results than ALNS for most of the instances on all sets of the four classes instances with under ML/OU policies. Hence, among heuristic algorithms, it is obvious that this chapter proposed method shows a better performance in terms of solution quality. Thus, this approach has the potential in providing good quality solutions. With regard to computer runtime, this proposed method also gives a shorter computer run time on average in both UB and ANLS. Consequently, the implement of the Randomised CWS heuristic by applying IG algorithm with local search can contribute to better solutions with larger instance size and more complicated problem.
- Furthermore, this chapter suggests that applying transhipment to the IRP results in cheaper costs compared to IRP costs - by paying transhipment costs

it is more likely to reduce total costs than the potential penalty costs incurred as a result of stock-outs. Also, the IRPT increases supplier reliability.

Additionally, those problems can be considered for stochastic demand, which extend the problems and implement another algorithm to this approach. Furthermore, the Sim-heuristic is found to be an efficient way since this is able to combine the strength of simulation, which engages with stochastic problems, and the strength of meta-heuristics. A novel decoding method is, thus, applied in order to interpret the Randomized CWS with MCS solution to IRPs. Additionally, the next chapter intends to extend the benchmark, not only to generate data, but also make of the data available for future experiments and for comparison and test.

Chapter 6: Sim-Randomised CWS Heuristic for Stochastic IRP and IRPT

6.1. Introduction

The stochastic IRP has received significant attention in recent years, with important variants of the problem such as time, demand, customer and location. For example, stochastic IRP with a single-depot and multi-retailers was first introduced by Barnes and Bassok (1997). Bard et al. (1998) focused on the customer demand, which was not known with certainty and routing decisions taken over the short run in satellite facilities. Recently, the IRP with a single-period, stochastic demand, allowing stock-out, is proposed by Juan et al. (2014). Coelho et al. (2014) studied a dynamic version of IRP whereby lateral transhipment is allowed. This chapter focuses on stochastic demand for IRP with and without transhipment, where demand is not known until the vehicle arrives at customer location.

This chapter presents a Sim-Randomised CWS heuristic algorithm for solving stochastic demand IRP (SIRP) where customer demands are regularly exposed over time and planning must be made at the beginning of each time period. This problem is similar to the problem proposed by Coelho et al. (2014), which proposes an algorithm that takes advantage of historical information in order to take into account stochastic demand. Inventory was handled by using different replenishment policies (ML and OU). Also, the use of transhipment between customers is allowed in order to avoid the customer facing stock-out when demand is high. The combined simulation with randomised CWS heuristic is proposed to optimise system performance and solved the SIRP and SIRPT under replenishment policies (ML and OU). The structure of this chapter is: section 5.2 presents this chapter contribution, the SIRP and SIRPT modelling is proposed in section 5.3. Section 5.4 presents the proposed approach to solve IRP and IRPT with stochastic demand. All the computational experiments of this chapter are shown in section 5.5. Finally, the conclusion of this chapter is given in section 5.6.

6.2. Contribution

Most of the literature on IRP focused on deterministic IRP where demand is known in advance, which is less likely to be realistic in real world problems. Therefore, the aims of this chapter are: Firstly, to solve the IRP and IRPT to deal with stochastic elements,

which is called IRP and IRPT with stochastic demand (SIRP and SIRP). The complexity of this problem makes the choice of approximate methods a preferred tool for solving these types of problems. Secondly, to develop a simulation-optimisation technique (combined biased-randomised heuristics with Monte-Carlo Simulation), called Sim-heuristic (Juan et al., 2014). Consequently, this chapter extends IRP with lateral transhipment and implement a (Sim-heuristic).

This chapter considers stochastic demand on IRP and IRPT (SIRP and IRPT), similar to (Coelho et al., 2014). The difference however is, the algorithm proposed to solve the problem. Also, these customer demands are stochastic in nature where the demand is unknown until the vehicle arrives at the inventory location of customer. In this approach, we will assume that for each customer, it has been possible to use historical data to model the customer demand. Thus, the vehicle delivers to the customer's inventory when the demand is generated by using simulation technique (uniform distribution). Also, our proposed approach uses the two different replenishment policies (OU and ML) and the allowance of lateral transhipment to minimise inventory holding cost and avoiding stock-out when high demand occurs. In this chapter, SIRP and SIRPT are studied by combining simulation techniques and the randomised CWS heuristic (Sim-Randomised CWS heuristic).

6.3. SIRPT and SIRP optimisation models

The proposed SIRPT and SIRP optimisation model are defined as a graph $G = (V, E)$ where $V = \{0, \dots, n\}$ is vertex set and $E\{(i,j): i,j \in V, i \neq j\}$ is the arc set. Vertex 0 is the depot where the supplier is located and the vertices of $V' = V \setminus \{0\}$ represent customers. Both the supplier and customers incur unit inventory holding costs h_i per period (i, V) and each customer has an inventory holding capacity C_i. This case assumes the supplier has enough inventory to meet all the demand during the planning horizon. If the demand of customer i is higher than its inventory level, it is then lost and a unit shortage penalty p_i is incurred. At the beginning of the planning horizon the decision maker knows the inventory level I_0^0 and I_i^0 of the supplier and of each customer i, respectively. The duration of the planning horizon is p and, at each time period $t \in \mathrm{T} = \{1, \dots, p\}$.

The problem is defined by the quantity of product made available at the depot and quantity absorbed by the customer i at each discrete time instant of the planning time

horizon are z_0^t and z_i^t respectively. The demand d_i^t of customer i is a random variable D_i^t. Considering that the supplier's customers are often retailers, which are themselves facing an external demand from the end customers, this value may be interpreted as the total orders received by a retailer in a given time period. In practice, the demand is not known by the decision maker (usually the supplier) who has to estimate it on the basis of historical data. Also, the decision maker can use any kind of forecast and input this information into the algorithmic framework. The decision maker becomes aware of the actual values of d_i^t at the end of each period t. A single vehicle of capacity Q is available at the depot and the vehicle is able to perform one route per time period, from the supplier to a subset of customers. A routing cost c_{ij} is associated with arc $(i, j) \in A$ and these costs satisfy the triangle inequality. Two inventory policies are considered. The first one is called ML inventory replenishment policy which allows the supplier to freely choose the quantity to deliver to the customers and is limited only by the inventory capacity at the customers. The second is called OU inventory replenishment policy which has been widely used in IRPs (Bertazzi et al., 2002; Savelsbergh et al., 2012; Archetti et al., 2012 and Coelho et al. 2012b) and ensures that whenever a customer is visited, the quantity delivered is that needed to fill its inventory capacity. It is important to ensure the feasibility of such a policy, given that there is only one vehicle available. So, one assumes direct deliveries can take place from the supplier to any customer, by subcontracting to a carrier, to allow for planning deliveries to meet the OU requirements.

In addition, after the demand is realised, if a customer faces a shortage, it can arrange a lateral emergency transhipment from another customer. Both types of outsourced deliveries (direct deliveries and lateral emergency transhipments) are only made by direct shipping and the unit cost associated with direct deliveries or transhipments from i to j is bc_{ij}, where $b > 0$. As is standard in vehicle routing, travel costs are distance-dependent and are unrelated to the vehicle load. However, direct delivery and transhipment costs are distance and volume dependent because this is often how outsourced carriers define the terms of their contracts.

Regarding temporal issues, this case considers that the decision maker first decides which customers to replenish in each period as well as the associated vehicle route and the direct shipments, if any. After demand is revealed, lateral transhipments may be arranged if any customer faces a shortage. The SIRPS and SIRPT can be solved under ML or OU replenishment policies for each policy, whether emergency lateral

transhipments can be allowed or not. Formally, under the ML policy, replenishment is implemented at the end of the period regardless of remaining inventories. Besides, the OU policy triggers a replenishment order to bring the inventory position up to level S whenever the inventory reaches the reorder point s. The reorder point s should consider the delivery lead time and the stock-out risk resulting from the stochasticity of the demand. In systems, demand can be measured by the unit, with very small-time intervals. This is formally defined as a (s, S) replenishment system (Kenneth et al. 2010 and Phillips et al. 2016).

The first decision made under OU policy is regard to the level of the inventory at the reorder point s_i of customer i is set. It is equal to an estimate of the expected demand during the lead time L, plus a safety stock dependent on demand variability, lead time and target service level. This case denotes the estimate of the expected demand μ_i of customer i per period by $\widehat{\mu_i}$ and that of its standard deviation σ_i by $\widehat{\sigma_i}$. These values, as well as the resulting threshold, can be updated at every period. Following classical inventory management policies (Silver, Pyke, and Peterson, 1998) and assuming independent and normally distributed demands, s_i can be computed as;

$$s_i = L\widehat{\mu_i} + z_\alpha \sqrt{\widehat{\sigma_i^2} L} \tag{1}$$

where α is the probability of a stock-out and z_α is the α-order quantile of the demand distribution. The quantity $1 - \alpha$ is usually referred to as the service level. However, this problem is defined as follows, assuming OU policy applies and given that the lead time is equal to one (all deliveries are performed in the next period), the standard deviation is zero, the value of s_i used in the equation is then

$$s_i = \widehat{\mu_i} + z_\alpha \widehat{\sigma_i} \tag{2}$$

One assumes that the lead time is equal to zero (there is no demand taking place between the moment one decides to deliver and the time of the delivery, i.e., the delivery taking place in period t can be used to satisfy the demand of period t), and that the order interval R_i is equal to one (deliveries can take place every period), the value of s_i used in equation (2). The parameters for optimisation model are as follow.

6.3.1. **SIRPT optimisation model**

This problem allows lateral transhipments between customers once the decisions have been made and demand is revealed. If a customer runs out of inventory when its demand is realised, lateral transhipments can take place whenever they are possible. Note that the IRPT optimisation model, the parameter I_i^0 represents the initial inventory of customer i at the beginning of each time slice of the rolling horizon. Let I_i^t be the inventory level at customer i at the end of period t, q_i^t the quantity delivered to customer i in period t using the supplier's vehicle, w_{ij}^t the quantity carried by the outsourced carrier from customer i to customer j in period t, and l_i^t the lost demand at customer i in period t due to insufficient inventory. It is solving once per period, after demand is realised,

This thesis, one also assumes that the target level meets the customer inventory capacity in order to ensure that this rule is always met and to avoid infeasibilities due to insufficient vehicle capacity. Also, direct deliveries are allowed to take place from the depot and this ensures that all customers i whose inventory level is below the threshold s_i will have their inventories filled to their capacity in the next period. Thus, c_{ij} is the routing cost and the routing variables $x_{ij}(i < j)$ are equal to the number of times an edge (i, j) is traversed. The binary variables y_i are equal to one, if and only if vertex i (the supplier or a customer) is visited by the supplier's vehicle. Giving q_i means the quantity delivered by the supplier's vehicle and w_{ij}^tmeans the quantity transhipment delivered from $i\ to\ j$ in time period t. The variables and constraints of the model are as follows:

$$Minimise \sum_{t\in T} \mathrm{h}_0 I_0 + \sum_{i\in V'}\sum_{t\in T} h_i I_i^t + \sum_{i\in V}\sum_{j\in V, i<j}\sum_{t\in T} c_{ij}x_{ij}^t + \sum_{i\in R\{0\}}\sum_{j\in V'}\sum_{t\in T} b_{ij}w_{ij}^t + \sum_{i\in V'} p_i l_i \quad (3)$$

Subject to

$$I_i^t = I_i^{t-1} + q_i^t + \sum_{j\in V} w_{ij}^t - \sum_{j\in V'} w_{ij}^t - d_i^t - l_i^t \quad i \in v' \ \ t \in T' \quad (4)$$

$$I_0^t \geq 0, \ t \in T \quad (5)$$

$$I_i^t = I_i^{t-1} + q_i^t - d_i^t, \qquad i \in V', \qquad t \in T \quad (6)$$

$$I_i^t \geq 0, \quad i \in V', \quad t \in T \quad (7)$$

$$q_i^t \geq C_i \sum_{j \in V'} x_{ij}^t - I_i^{t-1}, \quad i \in V', \quad t \in T \tag{8}$$

$$q_i^t \geq C_i - I_i^{t-1}, \quad i \in V', \quad t \in T \tag{9}$$

$$q_i^t \leq C_i \sum_{j \in V} x_{ij}^t, \quad i \in V', \quad t \in T \tag{10}$$

$$\sum_{i \in v'} q_i^t \leq \mathrm{Q}, \quad t \in T \tag{11}$$

$$\sum_{i \in V} x_{ij}^t = \sum_{i \in V} x_{ji}^t, \quad j \in V \quad , \quad t \in T \tag{12}$$

$$\sum_{i \in v} x_{i0}^t \leq 1, \quad t \in T \tag{13}$$

$$V_i^t, q_i^t, w_{ji}^t \geq 0, \quad i \in V', j \in R\{0\}, t \in T \tag{14}$$

$$x_{ij}^t \in \{0,1\} \quad i, j \in V, i \neq j \ , \quad t \in T \tag{15}$$

- The objective function (3) minimises the total costs, inventory holding cost, routing cost, the transhipment and penalty cost.
- Constraints (4) is the inventory level at the customer (I_i) at the end of period t.
- Constraints (5) avoid stock-outs at the supplier in which the inventory of the supplier cannot be negative.
- Constraints (6) and (7) are similar to constraint (4) and (5) and apply to customers.
- Constraints (8), (9) and 10) define that the quantity delivered by a supplier's vehicle to each customer $i \in V'$ will fill the customer's inventory capacity if the customer is served and will be zero otherwise. These set of constraints impose to the Order-Up-To (OU) policy. If customer i is not visited throughout the period t, then constraint (10) means that the amount delivered to it will be zero (while constraints (8) and (9) are still respected). Otherwise, if customer i is visited during the period t, constraint (10) limits the quantity delivered to the customer's inventory holding capacity, and this bound is tightened by constraint (9), making it impossible to deliver more than what would fill this capacity. Constraint (8) models of OU inventory replenishment policy, ensuring that the quantity transported will be exactly within the bound

provided by constraint (9). Therefore, to solve the IRP under the ML replenishment policy, it significant does to drop constraints (8) and (10).

- Constraints (11) state that the vehicle's capacity is not exceeded.
- Constants (12) and (13) guarantee that a feasible route is designed to visit all customers served in the period t. Constraints (12) enforce the number of arcs entering and leaving a vertex to be the same. Constraints (13) show where a single vehicle is available.
- Constraints (14) and (15) are integrally and non-negativity conditions on the variables.

6.3.2. **SIRP optimisation model**

In this problem, the customer demand is delivered by the supplier's vehicles and no emergency lateral transhipment is allowed when a customer runs out of inventory. Routing decisions are based solely on a customer-dependent threshold s_i and on its inventory level. Additionally, the decision should be taken only once for every period, and if the inventory level at customer i is below s_i when the actual demand is realised at the end of period t, then customer i is selected to be served in period $t+1$. The threshold can be updated after each period. Replenishment level S_i usually depends on ordering and holding costs and is set to bring the inventory level up to a target value. This problem is solved by using heuristics and simulation under both ML and OU replenishment policies.

$$Minimize \sum_{t\in T} \mathrm{h}_0 I_0 + \sum_{i\in V'} \sum_{t\in T} h_i I_i^t + \sum_{i\in V} \sum_{j\in V, i<j} \sum_{t\in T} c_{ij} x_{ij}^t + \sum_{i\in V'} p_i l_i \quad (16)$$

The objective function (16) defines the minimisation of holding cost, routing and penalty costs if the stock-out is occurred. A set of constraints in the IRP mathematical model is the same as used in the SIRPT model; however, wij is set as zero in the IRP model.

6.4. Proposed Sim-Randomised CWS heuristic for solving SIRP and SIRPT

The proposed approach focuses on solving the SIRP and SIRPT with stochastic demands and allows lateral transhipments between customers. This case assumes that updated information on current inventory levels is obtained at the end of each

period. Notice that these end-of-period inventory levels might be very difficult to forecast, especially when the probability distribution model the random demands are characterised by high variances. Thus, one believes that under these realistic conditions it might make sense to follow a plan-one-step-ahead policy, i.e., plan just one period ahead and then update the current inventory levels before planning again.

Simulation is an applied methodology in that one describes the behaviour of complex problems using mathematical models. The simulation technique based a mathematical modelling is a basic method for analysis of complex life support problem states and for forecast of problem evolution. Therefore, the performances of this concept have been widely proven (Carson and Maria, 1997). Besides, the simulation processes with stochastic variables are related to the basic mechanisms **(**Grasman, et al., 2014). The stochastic problem includes the following steps: firstly, the random behaviour selection of a specific variable, which can follow a uniform or non-uniform distribution**.** Secondly, the probability distribution is defined; several parameters need to be settled such as the expected value and the standard deviation.

Monte-Carlo Simulation technique (MCS) is used to model the probability of different outcomes in a process that cannot easily be predicted due to the intervention of random variables. Hence, the combination of MCS with Randomise CWS heuristic has not been proposed to solve IRPT with Stochastic demand in the literature. Subsequently, this chapter proposes a Sim**-**heuristic, which consists of the integration of biased randomised heuristics and Monte**-**Carlo simulation for solving stochastic IRP and IRPT**.** Therefore, some biased random behaviour with the CWS heuristic has shown good performance in the searching process, inside the scope of feasible solutions, as presented in the work of Juan et al. (2011). Each of the feasible solutions consists of a set of round trip routes from the depot that was visited, and which served all nodes exactly once to satisfy all demands. The selection probability for each edge in the saving list uses a geometric statistical distribution during the

The stochastic distribution is analysed using simulation to generate the demand. Firstly, the stochastic distribution is analysed in order to determine the model type, and then to assign the distribution as the previous demand of data. Therefore, the uniform distribution can simulate constant demand. This work takes the impacts of stochastic demands into account for the supplier and the customer. A simulation optimisation technique (Randomised CWS heuristic with MCS) is included in the decision routing process - (Caceres et al., 2013) insists that integrating a simulation

optimisation model is more advantageous for decision makers, than the traditional techniques.

6.4.1. **Sim-Randomised CWS heuristic**

This section describes the concept of the combined simulation technique and Randomised CWS heuristic for solving the SIRP and the SIRPT for under both ML and OU replenishment policies (SIRP_ML, SIRP_OU, SIRPT_ML and SIRPT_OU). Monte-Carlo Simulation (MCS) can be described as a set of techniques that make use of random numbers and statistical distributions for solving stochastic and deterministic problem (Law, 2007). When properly combined with heuristic techniques, MSC has proven to be extremely useful to solve stochastic vehicle routing problems (Juan et al., 2011)

The procedure is as follows: firstly, the historical data is tested to determine what type of probability distribution should be used to generate customer demand and calculate the delivery quantity in which the ML/OU replenishment policy is used. While, supplier must decide the route for delivery to customers by applying Randomised CWS heuristic algorithm. Then, when the vehicle arrives at the customer, demand is generated (Uniform Distribution). When the customers face stock-out or high demand occurs, then the lateral transhipment is allowed. In the randomised version of this algorithm, a probability distribution is used to assign a selection probability to each single edge in the savings list. The savings list with routing cost is then calculated. In addition, this probability should be proportional with the savings value associated with each edge, which means edges with larger savings will more likely be selected from the list than those with smaller savings. The exact probabilities assigned are variable and they depend upon the concrete distribution selected at each step. This method is randomly iterated by beginning with the efficient search process. Nonetheless, the random selection process biased distribution is a geometric distribution which is used instead of a symmetric distribution. Therefore, using a biased distribution aims to give a greater probability of being selected. During the multi-start process, the best solution found so far is recorded.

Finally, as shown on Algorithm 6.1 (Pseudo-code for MSC to obtain variable costs and reliability estimates) at the end of each first-level iteration, the resulting solution goes through a Monte-Carlo simulation process which provides estimates of the demands which are expected values costs that follow a Log Normal distribution with $E[c_i] =$

$D_i, (\forall\, i \in 1, \dots, n)$. The Log Normal distribution has been chosen because it is necessary for modelling positive demands. Then the variance is calculated, defined as $V[c_i] = 0.5E[c_i]$; $V[c_i] = k[c_i], (0 \leq k \leq 1)$ for each value of customer. Next, the expected cost is calculated by long simulation such as 10000 times. These estimates are obtained by iteratively sampling the random variables characterising customer demands in each route, so that total expected value of total costs is calculated for final solution by $\frac{\sum_{i=1}^{10000} E[c_i]}{10000}$. Hence, one changed d_i, the deterministic demand of customer i to stochastic demands D_i with $E[D_i] = d_i$. In other words, the demand of each customer is considered as a random variable following a well-known probability distribution with given mean and variance.

Algorithm 6.1: Procedure for MSC

```
1: Procedure calculated Expected Costs (nodes, Sol)
                                          ▷ node: coordinateds and demand
                                          ▷ Sol: a solution for deterministic IRP
                                          ▷ r: route
                                          ▷ i: customer
2:   solExpectedCosts ← 0                ▷ Reset solution expected costs
3:   for each route r in Sol do          ▷ For each route r in the given solution
4:       rExpectedCosts ← 0
5:       for iter = 1 to (iter − nIter) do   ▷ Generate nIter simulated for r
6:                rCosts ← getCosts(r)       ▷ Fixed costs for r
7:                rAccumDemand ← 0
8:                for each customer i in r do
9:                    newDemand ← generateRandomDemand(i)
10:                   rAccumDemand ← newDemand
11:                   if rAccumDemand > vehicleCapacity then
12:                       rCosts ← rCosts + roundTripCosts(i, depot)
13:                       rAccumDemand ← newDemand
14:                   end if
15:               end for
16:               rExpectedCosts ← rExpectedCosts + rCosts
17:      end for
18:  rExpectedCosts ← rExpectedCosts/nIter   ▷ Estimate expected costs for r and accumulate them
19:  solExpectedCosts ← solExpectedCosts + rExpectedCosts
20:  end for
21:  return solExpectedCosts                ▷ Return expected costs for the given solution
22: end procedure
```

Additionally, the SIRP and SIRP under ML/OU policies-based Sim-Randomised CWS heuristic is related to routing and vehicle load. For that reason, inventory replenishment policies are used to decide the quantity of product to satisfy the demand of each customer. Also, the holding cost is calculated after the policy is applied to estimate inventory levels. Algorithm 6.2 (Pseudocode for inventory) shows procedure for inventory cost-replenishment policy combination. In addition, the transhipment and penalty costs are calculated and added to the total costs. These costs occur when extra demand is realised or if a customer faces inventory shortage. However, if the supplier takes an option to use the transhipment policy, then the transhipment cost will be added to the total costs. On the other hand, a penalty is applied when a supplier cannot deliver the product to meet the demand and lateral transhipment is not available. So, penalty costs will be added by the product of penalty weight and the loss of inventory level.

Algorithm 6.2: Procedure for INVENTORY

1**: Procedure** Inventory *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, b_n, q_n, l_n, w_n, n$*)*

▷ d_n: $Demand$
▷ I_n: Current inventory level
▷ p_n : $Penalty\ costs$
▷ h_n: Inventory holding cost
▷ b_n: Transhipment cost Value
▷ q_n: Quantity delivery
▷ l_n: Lost demand
▷ w_n: Quantity transhipment
▷ n: Customer node

2: **for** each customer n **do**
3: *nodeList* ← getNodeList (inventoryList)
4: OverSuppliedList (OSL) ← declare an empty list to store nodes which have stocks in the inventory
5: UnderSuppliedList (USL)← declare an empty list to store nodes which have not enough stocks in the inventory
6: **for** each node in the list of available nodes n **do**
7: d_n ← demand of Node n
8: I_n ← I_n is equal to I_{n-1} - d_n
9: **if** $I_n > 0$ **then**
10: Adding *n* into *OSL*
11: **else if** $I_n < 0$ **then**
12: Adding *n* into *USL*
13: **end if**
14: **end for**
15: **if** the policy is NO_TRANSHIPMENT **then**
16: **for** each undersupplied node in undersupplied list n **do**
17: $penaltyCosts \leftarrow p_n * l_n$
18: **end for**
19: **else if** the policy is TRANSHIPMENT **then**
20: *transhipmentCosts* ← $b_n * w_n$
21: **end if**
22: **for** each oversupplied node in oversupplied list n **do**
23: *holdingCosts* ← adding *holdingCosts* with h_n * I_n
24: **end for**
25: *nodeToSupply* ← declare an empty list of nodes to record the nodes to be filled in this round
26: **for** each node in *nodeList n* **do**
27: **if** the inventory level I_n is less than the level specified by the policy **then**
28: add *nodeToSupply* with Node *n*
29: **end if**
30: **end for**
31: **end for**
32: **end procedure**

6.4.2. **SIRP-based Sim-Randomised CWS heuristic**

The sub-section explains the procedure of the Sim-Randomised CWS heuristic for solving SIRP_ML and SIRP_OU as shown on Algorithm 6.3 (Pseudocode for SIRP-based Sim-Randomised CWS heuristic). Once the demand is generated (statistic distribution, MSC), the Randomised CWS heuristic generates the routes. The inventory replenishment policy handles the inventory capacity of each customer as follows (Algorithm 6.2): for the ML replenishment policy, the supplier decides a quantity of product for supply each customer and the vehicle delivers from the depot with up to Q units. At the same time, the OU replenishment policy is applied by fixing the quantity of product to deliver to customers in different successive time periods. Algorithm 6.3 presents Pseudocode for SIRP-based Sim-Randomised CWS heuristic and this procedure is explained below:

- Consider a SIRP with an available depot, a set of customers, inventory capacity of each customer, given time period, vehicle capacity, customer location and ML/OU policy. Assume that each customer has a positive stochastic demand characterised by a specific statistic distribution which is used from historical data.

- ML/OU replenishment policy takes into account an estimate of inventory costs, the demand of each node is then simulated (uniform distribution) and the quantity for each customer in time period is calculated as shown on Algorithm 6.3 (Pseudocode: line 4 to line 20).

- Generate routes using the Randomised CWS heuristic algorithm (Pseudocode: line 22 and line 27). The obtained solution (c), will be also a feasible solution for the original SIRP as long as the total route demand calculated during the actual delivery stage, does not exceed the surplus capacity (i.e. the safety stock).

- Simulating the solution (c) to MCS in order to estimate the expected cost with a specific value for the parameter k $(0 \leq k \leq 1)$. This has been explained in the sub-section above and shown on Algorithm 6.1and 6.3 (Pseudocode: line 28 to line 30).

- Repeat the process with a new value of the parameter k to explore.

- Finally, return the solution with the lowest expected total costs found so far.

Algorithm 6.3: Pseudocode for SIRP based SimRandomised CWS Heuristic

1: **Procedure SIRP based SimRandomised CWS** (*inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, l_n, q_n, n$)

▷ d_n: $Demand$
▷ I_n: Current inventory level
▷ p_n : $Penalty\ costs$
▷ h_n: Inventory holding cost
▷ l_n: Lost demand
▷ q_n: Quantity delivery
▷ n: Customer node

2: **for** each customer n **do**
3: h_n←inventory holding cost of each Node n
4: *nodeList* ← getNodeList (inventoryPolicy)
5: oversuppliedList (*OSL)* ← declare an empty list to store nodes which have stocks in the inventory
6: undersuppliedList *(USL)*← declare an empty list to store nodes which have not enough stocks in the inventory
7: **for** each node in the list of available nodes n **do**
8: d_n ← MCS ▷ *Uniform distribution*
9: I_n ← I_{n-1} - d_n
10: $q_{n,ML/OU} \leftarrow \widehat{I_i} - I_i$ ▷ *ML/OU policy*
11: $q \leftarrow add\ q_{n,ML/OU}$ *to the list of demands* q
12: **if** $I_n > 0$ **then**
13: **add** *n* into *OSL*
14: **else if** $I_n < 0$ **then**
15: **add** *n* into *USL*
16: **end if**
17: **end for**
18: **for** each undersupplied node in undersupplied list n **do**
19: $penaltyCosts \leftarrow p_n * l_n$
20: **end for**
21: **end for**
22: *SavingList* ← createSavingList (*nodeList*) ▷ *Randomised CWS*
23: *solution* ← createInitialSol (*nodeList*)
24: **while** the saving list is not empty **do**
25: *edge* ← randomSample(*savingList*) ▷ *Geometric distribution*
26: RCWS*Sol* ← mergeRoute(*route1, route2, edge*)
27: **end while**
28: solExpectedCosts(nodes, Sol) ▷ *MCS*
29: **for** *each sol in best StochSolList do*
30: statistics(*sol)* ← *MCS* (irpSol;bestSol;nSols, β)
31: **end for**
32: **return** *bestSolStocSolList*
33: **end procedure**

6.4.3. **SIRPT-based Sim-Randomised CWS heuristic**

Sim-Randomised CWS heuristic is applied to solve the SIRPT_ML and SIRPT_OU. The lateral transhipment is utilised to avoid infeasible solutions that may arise in the

routing network. For example, the vehicle capacity could be exceeded or a stock-out could occur at a customer as a result of it not being served as often as required. So, all transhipments are kept from both the supplier and customer to deliver to other customers with large associated costs (feasible solutions can always be reached but may incur high costs if transhipment is used). Those costs are penalties in objective function when the customer faces a demand shortage or when the master level heuristic does not add all customers to the current solution. The Sim-Randomised CWS heuristic applied to solve the SIRPT is similar to SIRP except when extra demand occurs, and transhipment is used. Once again, the supplier calculates the quantity of product to deliver from the depot and from all transhipment arcs.

Algorithm 6.4 (Pseudocode: line 2 to line 15) shows the demand at each node and is simulated by sampling the specified stochastic distribution (uniform distribution). The quantity of product to deliver is calculated by the supplier under ML or OU policies. The transhipment is calculated which can be undersupplied or oversupplied (Pseudocode: line 18 to line 22). Transhipment has occurred when the customer faces a shortage or there is higher demand. The current inventory for each node is updated. Then the nearest node is found which has enough product to serve another customer (oversupplied list) - to be the supplier (transhipment). Finally, the round-trip cost between the supplier node and the demand node is calculated. The round-trip costs between them is the transhipment cost which is added to the total cost. After that the initial dummy solution is generated by using CWS heuristic. Then bias randomisation is used to randomly select an edge from the saving list. The new solution is updated in which the set of routes is contained by merging them. Furthermore, the demand is generated once again when the vehicle has arrived, which uses a uniform distribution. In this case, whenever demand is high, the stock-out occurs, then the transhipment is taken, and subsequent cost is added. After iterating this process thousands of times, a random sample of costs is obtained, from which an average value can be estimated. Then, the expected total costs can be calculated by adding these variable costs, due to stock-out or transhipment, and the penalty costs given by p.

Algorithm 6.4: Pseudocode for SIRPT based SimRandomised CWS Heuristic

1: **Procedure SIRPT based SimRandomised CWS** *(inventoryPolicy, nodeList,* $d_n, I_n, p_n, h_n, b_n, q_n, l_n, w_n, n, \beta$*)*
▷ d_n: $Demand$
▷ I_n: Current inventory level
▷ p_n : $Penalty\ costs$
▷ h_n: Inventory holding cost
▷ b_n: Transhipment cost Value
▷ q_n: Quantity delivery
▷ l_n: Lost demand
▷ w_n: Quantity transhipment
▷ n: Customer node
▷ β: parameter for biased randomisation
2: **for** each customer n **do**
3: h_n←inventory holding cost of each Node n ▷ *ML/OU policy*
4: *nodeList* ← getNodeList (inventoryPolicy)
5: oversuppliedList (*OSL)* ← declare an empty list to store nodes which have stocks in the inventory
6: undersuppliedList *(USL)*← declare an empty list to store nodes which have not enough stocks in the inventory
7: **for** each node in the list of available nodes n **do**
8: d_n ← MCS ▷ *Uniform distribution*
9: I_n ← I_{n-1} - d_n
10: $q_{n,ML/OU} \leftarrow \widehat{I_i} - I_i$ ▷ *ML/OU policy*
11: $q \leftarrow add\ q_{n,ML/OU}$ *to the list of demands* q
12: **if** $I_n > 0$ **then**
13: **add** n into *OSL*
14: **else if** $I_n < 0$ **then**
15: **add** n into *USL*
16: **end if**
17: **end for**
18: **if** policy = TRANSHIPMENT **then**
19: *transhipmentCosts* ← $b_n * w_n$ ▷ *Transhipment*
20: **end if**
21: **for** each n in *OSL* **do**
22: *inventoryCosts* ← $h_n * I_n + p_n * l_n$
23: **end for**
24: **end for**
25: *SavingList* ← createSavingList (*nodeList*) ▷ *Randomised CWS*
26: *solution* ← createInitialSol (*nodeList*)
27: **while** the saving list is not empty **do**
28: *edge* ← randomSample(*savingList*) ▷ *Geometric distribution*
29: RCWS*Sol* ← mergeRoute *(route1, route2, edge*)
30: **end while**
31: solExpectedCosts(nodes, Sol) ▷ *MCS*
32: **for** *each sol in best StochSolList do*
33: statistics(*sol)* ← *MCS* (irpSol;bestSol;nSols, β)
34: **end for**
35: **return** *bestSolStocSolList*
36: **end procedure**

6.5. Computational experiments

This section presents the computational experiments, which have been implemented by using well known benchmark problems from the literature. Coelho et al. (2012) extended deterministic problem into stochastic by using generalisation and some historical data of IRP and IRPT instances, to generate the random demand. All computations are performed on a network with 630 nodes and the results report on the performance of our proposed methodology. All instances have used in this chapter are taken from the benchmark introduced by Coelho, Laporte and Cordeau (2012). The instances generated for the IRP by Archetti et al. (2007), namely the mean customer demand, initial inventories, vehicle capacity and geographical location of vertices, are the same as in Archetti et al. (2007). It is used in the literature to evaluate algorithms for the IRP, and instances are generated with 50 past periods of demand information before the future p periods, so that it can be used as historical data.

This thesis study has used the same benchmarks in order to compare the results of this algorithm and those proposed within the literature. This approach has at least three advantages: Firstly, all data details, including customer coordinates, the mean customer demand, initial inventories and vehicle capacity, are given. Therefore, other study works can use the same data set for verifying and benchmarking purposes. Secondly, this experiment is using a well-known set of instances which includes the time period, the number of each instance in a set of five instances and three instances are grouped by their size. Thirdly, the solutions for each instances group based on the SIRP and the SIRPT cases and the inventory replenishment policies are compare with the best-known solution in the literature.

Therefore, the experiments from the proposed method must be compared to the results calculated from the literature to determine the performance of the algorithm. Thus, the instances for the experiment must be the same in order to be comparable. The instances have been generated on the basis of the following data.

The instances are based on the following:

- Number of retailers n: $5k\ where\ k = 1,2,3,5,10,15,20,25,30,40$;
- Time horizon p: $equal\ to\ 5,10\ or\ 20\ periods$;
- Demand distributions: mean demand μ_i is generated as an integer random number, following a discrete uniform distribution in the interval [10,100] and

standard deviation σ_i as integer random number, following a discrete uniform distribution in the interval [2,10]. The demands are generated following a normal distribution with these parameters. If a negative demand value is generated, it is substituted by zero;

- Product availability at the supplier: mean production $\bar{z}$ is generated as an integer random number following a discrete uniform distribution in the interval [100n, 140n] and σ_0 as an integer random number following a discrete uniform distribution in the interval [2,10]. The production is generated following a normal distribution with these parameters. They are used only to account for inventory costs at the supplier, as in Archetti et al. (2007);
- Product quantity z_{0t} made available at the supplier at time t: $\sum_{i \in M} z_i$;
- Maximum inventory level $U_i (i > 0)$: $z_i g_i$, where g_i is randomly selected from the set {2, 3,4} and represents the number of time units needed in order to consume the quantity U_i;
- starting inventory level I_0 at the supplier: $\sum_{i \in V'} U_i$;
- starting inventory level $I_{i0} (i > 0)$ at retailer i: $U_i - z$;
- inventory cost at retailer $i \in V$, h_i: randomly generated in the intervals $[0.02, 0.10]$;
- inventory cost at the supplier h_0: 0.01
- shortage penalty: $p_i = 200 h_i$;
- transportation capacity C: $\frac{3}{2} \sum_{i \in V'} \mu_i$;
- transportation cost c_{ij}: $\left[\sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} + 0.5 \right]$, where the points (X_i, Y_i) are the coordinates of vertex i and are obtained by randomly generating each coordinate as an integer number in the interval [0,500].

In all cases, random selections are performed in order to ensure uniform distribution. The approach described in the previous section has been implemented; Java programming language with eclipse software is used in all in these experiments, with a standard personal computer used to perform all tests: Core i5, 2.30 GHz processor and 4GB of RAM. In the experiments, the set of instances will be called the stationary data set since the mean of demand distribution is stationary. For this data sets, each of the 30 combinations of n and p, five instances have been generated, for total 150 instances. Their categorisation follows the rule $dirp - n - p - 1$ $through$ $dirp - n -$

$p-5$ representing the name of instances in which n stands for number of customers, p indicates the time period and the number of each instance in a set of five instances. Instances are grouped by their size: those with less than 50 customers are labelled small($5 \leq n \leq 25$), those containing between 50 and 100 customers are called medium($50 \leq n \leq 100$) and those with more than 100 customers are called large instances ($125 \leq n \leq 200$). In order to generalize these datasets for the SIRP and SIRPT, the simulation technique is used to generate the future demand, by random demands with known probability distributions, and model is given for the historical data demands. Since this approach uses simulation, random demands can be modelled by any probability distribution using historical information. In this case, uniform distribution is selected for generated expected demands. Notice that historical data would be used in a real-life scenario to model each customer's demand for a different probability distribution; the one that best fits the existing observations (Coelho et al., 2012 & Juan et al., 2014).

In this section, the results regarding the average solution quality over all instances are discussed. The solutions provided by the SIRP-based Sim-Randomised CWS heuristic under ML/OU inventory replenishment policies presents in sub-section 6.5.1. The next sub-section (6.5.2) shows the average results obtained by the SIRPT-based obtained by Randomised CWS heuristic under ML/OU inventory replenishment policies. Finally, the results based on the cases which the SIRP without and the SIRP with transhipment are compared.

The cost breakdown of the total costs such as the inventory holding (HC), penalty (PC), transhipment (TrC), routing (RC) and total costs (TC) are shown in Tables 6.2.and 6.4. The average costs of the total cost for all set of instance are shown in bold number. For the SIRP and SIRPT under the ML/OU replenishment policies solved by Sim-Randomised CWS heuristic are represented as SIRP_ML-based Sim-Randomised CWS, SIRP_OU-based Sim-Randomised CWS, SIRPT_ML-based Sim-Randomised CWS and SIRPT_OU-based Sim-Randomised CWS. Tables 6.2 and 6.4 illustrate one result calculated from five instances for each combination of the cost element of the SIRP-based Sim-Randomised CWS and the SIRPT-based Sim-Randomised CWS heuristics under OU or ML replenishment policies, respectively. The first column represents the instance number and the second column represents instance name, which includes the number of the customers. The cost breakdown of the SIRP_ML and the SIRPT_ML solved by this chapter proposed method presents in the second, the

third the fourth and fifth, respectively. Then the sixth, the seventh, the eight and the ninth column show the cost breakdown of the SIRP_OU and the SIRPT_OU, respectively.

6.5.1. **SIRP-based Sim-Randomised CWS heuristic**

This section provides the implementation of the SIRP-based Sim-Randomised CWS heuristic for all set of instances, firstly the results regarding the average solution quality over on all sets of instances are presented. Then in sub-section 6.5.1.1, the comparison of the performance of Randomised CWS heuristic against the best performing algorithm for the best performing method in the literature. Table 6.1 presents the cost breakdown and the total costs, in which bold numbers are the average costs of the total cost for all set of instances. This table is categorised by under ML or OU replenishment policies as: the SIRP_ML-based Randomised CWS and the SIRP_OU-based Randomised CWS.

For the SIRP case, this chapter proposed approach has also implemented the ML and OU inventory replenishment policies, while using the ML inventory policy, the OU rule is relaxed Under ML/OU policies, and the Sim-randomised CWS heuristic optimise the quantities delivered while respecting the vehicle and the customer inventory capacities. The summary solution costs on all set of instances is given in Table 6.1. Therefore, applying the OU inventory policy can lead to reduce in solution costs. Which can be seen the averages total costs provided by the SIRP based Sim-Randomised CWS heuristic under OU replenishment policy are slightly lower than those costs analysed by using the ML replenishment policy. It is reasonable that this may happen when the demand is unknown before fulfilling the inventory of customer, and the extra demand could happen anytime during the time period. Due to the inventory decision being handled by OU policy, the inventory holding cost is lower than ML policy. While using ML policy, the inventory capacity might be enough for the extra demand, but it will incur extra charge upon inventory holding cost. On the other hand, using OU policy when there is an inventory shortage, penalty cost is charged, and this can lead to increase the penalty costs as shown in the tables below. Moreover, the inventory holding cost plays an important role in changing the balance between making more frequent deliveries and holding higher average the inventories. It can be seen under OU policy, the average holding costs are lower than those using ML policy, but routing costs significantly reduce under ML policy. This due to the fact that when fill up to maximum level of inventory capacities, the algorithm can avoid

and costly visits to the same geographical area. High levels of inventory can be a disadvantage in which carrying too many good on hand can also be increased inventory holding cost.

Name of Instance		SIRP_ML-based Sim-Randomised CWS				SIRP_OU-based Sim-Randomised CWS			
		HC	PC	RC	TC	HC	PC	RC	TC
1	dirp-5-5-1	188.09	506.80	1911.01	2605.90	110.30	512.11	1910.02	2532.43
2	dirp-5-10-1	324.39	1876.00	2001.33	4201.72	103.58	1320.73	2380.08	3804.39
3	dirp-5-20-1	500.98	4142.00	2990.65	7633.63	169.99	3882.34	3099.10	7151.43
4	dirp-10-5-1	200.48	1976.00	3981.85	6158.33	153.58	788.15	4668.62	5610.35
5	dirp-10-10-1	454.77	2838.00	5899.12	9191.89	334.37	2640.43	6149.20	9124.00
6	dirp-10-20-1	899.11	3262.12	5001.11	9162.34	850.66	3952.09	5385.01	10187.76
7	dirp-15-5-1	389.89	1126.55	3988.78	5505.22	362.79	2438.11	4092.27	6893.16
8	dirp-15-10-1	687.44	3220.09	5961.33	9868.86	575.92	3865.45	6151.98	10593.35
9	dirp-15-20-1	1656.88	4226.34	8322.35	14205.57	657.02	4688.85	8664.45	14010.32
10	dirp-25-5-1	607.12	4808.54	4765.09	10180.75	307.32	3906.07	5391.86	9605.25
11	dirp-25-10-1	1036.13	5840.12	8522.66	15398.91	964.05	6913.00	9091.80	16968.85
12	dirp-25-20-1	1921.31	6358.09	10985.28	19264.68	1585.10	8229.91	11661.80	21476.81
13	dirp-50-5-1	1295.72	3962.11	8359.51	13617.34	607.65	3559.00	8461.23	12627.88
14	dirp-50-10-1	2961.61	5372.15	18962.19	27295.95	1630.50	5324.77	18953.03	25908.30
15	dirp-50-20-1	4483.33	9822.00	25109.15	39414.48	4173.52	8939.20	27107.53	40220.25
16	dirp-75-5-1	1602.50	7304.00	8743.55	17650.05	856.32	6490.73	9669.47	17016.52
17	dirp-75-10-1	3362.64	8262.00	13710.20	25334.84	1590.40	8939.90	14418.76	24949.06
18	dirp-75-20-1	5620.75	10314.78	19414.60	35350.13	2658.63	11027.42	22025.75	35711.80
19	dirp-100-5-1	2167.78	10476.00	11329.25	23973.03	1227.18	8276.40	12138.11	21641.70
20	dirp-100-10-1	4208.36	13820.98	26551.89	44581.23	3577.90	11211.90	28439.73	43229.53
21	dirp-100-20-1	8564.80	18622.00	31882.93	59069.73	6997.69	15922.17	33976.71	56896.57
22	dirp-125-5-1	6590.54	9164.60	10668.70	26423.84	4377.04	9022.17	12363.97	25763.18
23	dirp-125-10-1	7067.88	10600.05	24432.89	42100.82	5570.36	10277.83	28812.46	44660.65
24	dirp-125-20-1	9557.80	13360.90	47252.45	70171.15	7022.03	12059.33	70127.19	89208.55
25	dirp-150-5-1	6017.85	15062.00	11007.23	32087.08	5440.81	13758.84	13895.49	33095.14
26	dirp-150-10-1	12532.34	22114.00	28264.78	62911.12	9051.71	18146.53	34410.83	61609.07
27	dirp-150-20-1	13812.62	32550.12	56783.15	103145.89	10599.60	22651.00	6500.77	39751.37
28	dirp-200-5-1	8852.90	20968.00	13757.52	43578.42	6110.50	20099.52	15068.00	41278.02
29	dirp-200-10-1	10301.24	22378.11	38105.35	70784.70	9887.99	23561.00	45915.81	79364.80
30	dirp-200-20-1	17624.30	38166.00	50872.65	106662.95	13831.31	37948.11	67816.55	119595.97
Average		**4516.38**	**10416.68**	**16984.62**	**31917.68**	**3379.53**	**9678.44**	**17958.25**	**31016.21**

Table 6.1 : Comparison of average results of the cost breakdown of the SIRP- based Sim-Randomised CWS under ML/OU policies

6.5.1.1. Comparison of the results of SIRP-based Randomised CWS with the best results in the literature

This sub-section provides results for the stochastic IRP without transhipment solved by Sim-Randomised CWS. Table 6.2 presents solutions cost, the average running time obtained by this chapter proposed method and ALNS proposed by Coelho et al. (2012) under ML/OU policies. The ML/OU policies are used to allow fair comparisons. The results under policies are defined with the instances set for SIRP. This table illustrates two policies, each policy four columns are given; the first column gives the instances

size, the second shows the solution obtained by ALNS, the third is the solution solved by SIRP-based Sim-Randomised CWS heuristic, and the fourth is the % gap between the solutions are proposed by this chapter proposed with ALNS. All solution costs represent the average total costs as the same set, which has been reported in the previous sub-section.

Table 6.2 shows the solution costs for the average of each group instance sizes, the running time and the % gap of the SIRP-based Randomised CWS heuristic compare with best known results of the ALNS under both ML/OU policies. The Sim-Randomised CWS heuristic is able to find good solutions, but they are on average 6.48% and 5.63% higher than the ALNS solutions under ML and OU policies, respectively. Under ML inventory policy, the average total costs obtained by SIRP-based Randomised CWS are slightly higher than those solution costs obtained by ALNS. They are on average 0.05% slightly different for the small instances size. Whereas, they have 5.00% higher than those solved by ALNS for OU policy on the same set. For the medium instances size, the results solved by this chapter proposed algorithm are on average 10.11% different from the results solved by ALNS for under ML policy, and on the same set by using OU replenishment policy, they have 3.30% close to the results found by ALNS. This chapter proposed algorithm is able to provide good solution for the large instances size. They are on average 9.26 % different from those average total costs provided by ALNS for the ML policy. Similarly, they have 8.57% close to the solution costs solved by ALNS on the same instances set handle by OU policy.

Furthermore, this chapter implemented the simulation with Randomised CWS heuristic, which is able to solve small, medium and large instances size problems with a good computation time. All instances with up to 200 customers can be solved in less than one hour for the ML replenishment policy. When using OU inventory policy, the same instances set with up to 200 clients can be solved less than two hours. Additionally, in medium and large instances size, the proposed approach also has the potential to provide good solutions when combined with other techniques because a simulation technique is usually used with other heuristic/meta-heuristic.

Size of instances	ML replenishment policy					OU replenishment policy				
	ALNS		SIRP-based Sim-Randomised CWS		Gap (%)	ALNS		SIRP-based Sim-Randomised CWS		Gap (%)
	Cost	Time (s)	Cost	Time (s)		Cost	Time (s)	Cost	Time (s)	
Small (5≤n≤25)	10,225.93	46.30	10,231.54	20.30	0.05	9131.45	67.10	9588.32	38.60	5.00
Medium (50≤n≤100)	30,360.66	452.70	33,430.23	389.70	10.11	30,137.81	888.30	31,133.45	638.90	3.30
Large (125≤n≤200)	61,250.17	3860.10	66,921.89	2945.1	9.26	60,051.36	9248.90	65,197.88	5765.50	8.57
Average	**33,945.59**	**1453.03**	**36,861.22**	**1118.37**	**6.48**	**33,106.87**	**3401.43**	**35,306.55**	**2147.67**	**5.63**

Table 6.2 : Comparison of average results of the SIRP_ML and SIRP_OU based Sim-Randomised CWS heuristic and ALNS

6.5.2. **SIRPT-based Sim-Randomised CWS heuristic**

This section reports the solution costs of this chapter computational experiments. In the experiments conducted, this thesis tested the performance of Sim-Randomised CWS heuristic with the transhipment case on the same instances of the SIRP and the % gap equation is also used for the SIRPT. More interestingly, the transhipment can effectively improve the system performances and it is also advantageous in terms of logistics costs and supply chain performance as seen in the literature. Therefore, this chapter applied lateral transhipment with the SIRP under ML/OU policies, it can take place at any time to respond to stock-outs or potential stock-outs. For this experimentation, the transhipment cost (b) is set to 0.01 (Coelho et al., 2012).

The inventory replenishment policies are implemented, the OU rule is relaxed when using the ML policy. Under the ML policy, the Sim-Randomised CWS heuristic optimised the quantities delivered while concerning the vehicle and customer inventory capacities. The cost breakdown and total cost of the SIRPT-based Sim-Randomised CWS heuristic under OU/ML inventory replenishment policy are presented in Table 6.3. As a result, applying the ML inventory policy yields reductions in solution costs. The average costs provided by the SIRPT-based Sim-Randomised CWS heuristic under ML replenishment policy are slightly lower than those total costs when applying the OU replenishment policy. Regarding the inventory costs, applying the OU policy yields saving in the inventory holding cost, these but be of a similar order of magnitude as the routing costs, although the transhipment cost is sometimes higher. Therefore, using OU policy, the quantity of products supplied does not attain the maximum inventory level. Thus, in case where extra demand is encountered during the time period, transhipment cost is charge. Attempting to avoid incurring transhipment costs may lead to stock-out on customer side in the event that extra demand randomly occurs. Hence the transhipment helps in covering the extra demand when it occurs. It's is thus reasonable to incorporate the transhipment cost in

the contingency plan rather than paying a penalty cost (loss of good will) when a customer request cannot be filled.

Name of Instance		SIRPT_ML-based Sim-Randomised CWS				SIRPT_OU-based Sim-Randomised CWS			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	dirp-5-5-1	187.16	241.57	999.00	1427.73	135.50	318.90	990.90	1445.30
2	dirp-5-10-1	304.90	317.40	1404.44	2026.74	129.10	493.05	1428.90	2051.05
3	dirp-5-20-1	573.68	474.36	2763.69	3811.73	218.23	613.44	3195.92	4027.60
4	dirp-10-5-1	211.23	455.40	4514.48	5181.11	117.11	795.45	4100.85	5013.41
5	dirp-10-10-1	416.69	512.17	5627.60	6556.46	424.62	872.50	5556.21	6853.33
6	dirp-10-20-1	829.75	802.97	6964.46	8597.18	355.02	1335.60	7001.50	8692.12
7	dirp-15-5-1	367.77	876.33	4435.48	5679.58	203.30	1064.73	4604.57	5872.60
8	dirp-15-10-1	1475.38	1070.88	6260.47	8806.73	330.86	3888.32	4693.56	8912.74
9	dirp-15-20-1	1513.59	2447.04	7171.55	11132.18	636.29	4240.30	6941.17	11817.76
10	dirp-25-5-1	523.13	1677.34	5407.98	7608.44	369.51	1509.68	7104.62	8983.81
11	dirp-25-10-1	980.46	2080.32	8705.22	11766.00	520.86	2932.91	8613.28	12067.06
12	dirp-25-20-1	2227.44	2349.14	11266.20	15842.78	934.56	3436.81	10958.11	15329.49
13	dirp-50-5-1	961.29	1424.44	8570.22	10955.95	555.93	2528.88	6961.66	10046.47
14	dirp-50-10-1	2246.91	3527.55	11882.88	17657.34	1116.57	4242.22	11821.38	17180.17
15	dirp-50-20-1	4004.49	4783.95	16870.21	25658.65	1750.24	6743.08	17001.80	25495.12
16	dirp-75-5-1	1570.98	2020.83	10270.63	13862.44	924.83	2603.09	9977.07	13504.99
17	dirp-75-10-1	3268.59	3629.01	16779.10	23676.69	1548.23	6749.93	15622.11	23920.27
18	dirp-75-20-1	5590.65	4232.06	20315.67	30138.38	3817.80	6358.53	21983.89	32160.22
19	dirp-100-5-1	2065.25	2568.00	11187.16	15820.41	1179.88	4541.64	10002.90	15724.42
20	dirp-100-10-1	4018.00	3873.92	18220.10	26112.02	1994.41	5674.77	18889.90	26559.08
21	dirp-100-20-1	7368.05	5757.94	39153.89	52279.88	3360.93	9285.46	41230.88	53877.27
22	dirp-125-5-1	2847.56	2623.59	12897.10	18368.25	1307.84	7397.80	13333.15	22038.79
23	dirp-125-10-1	5027.15	5572.02	26798.44	37397.60	3389.90	10088.12	25290.56	38768.58
24	dirp-125-20-1	9596.85	11425.20	54255.70	75277.75	4306.31	12088.60	55055.11	71450.02
25	dirp-150-5-1	2845.48	3093.22	14622.96	20561.66	2919.89	4071.37	14901.70	21892.96
26	dirp-150-10-1	6208.23	6047.51	28239.49	40495.23	4062.45	8266.68	29775.71	42104.84
27	dirp-150-20-1	12346.89	13047.92	59408.96	84803.77	8113.11	15566.78	59778.10	83457.99
28	dirp-200-5-1	7488.42	6287.12	2007.89	15783.43	5941.56	9006.38	3330.33	18278.27
29	dirp-200-10-1	8608.75	7665.53	32911.67	49185.94	8046.67	10988.80	33090.80	52126.27
30	dirp-200-20-1	14479.62	15074.07	69181.99	98735.68	10067.39	19052.90	70922.85	100043.14
Average		**3671.81**	**3865.29**	**17303.15**	**24840.26**	**2292.63**	**5558.56**	**17471.98**	**25323.17**

Table 6.3 : Average results of the costs breakdown of SIRPT- based Sim-Randomised CWS heuristic under ML/OU policies

6.5.2.1. Comparison of the results of SIRPT-based Randomised CWS with the best results in the literature

Table 6.5 reports the comparison of the solution costs solved by the proposed Sim-Randomised CWS against the best-known solution from the benchmark. The solution costs obtained by ALNS (Leandro et al. 2012) and the computation time are also presented in this table. These results on all sets of instances give the average of five instances generated for each combination of the number of customer (n)and the time period (p). In the performed experiments, this chapter implemented the proposed Simulation technique and Randomised CWS heuristic by using on all sets of instances,

which has been used in the literature. The computational results provided in term of the average solution costs which is the best solution. The computation times and % improvement is also reported

This chapter proposed approach is able to provide good solutions, but they are on average 6.48% and 5.63% worse than those obtained by the ALNS under ML and OU policies, respectively. Applying OU policy, the average solution costs generated by SIRPT-based Sim-Randomised CWS algorithm are 0.18 % better than the results generated by ALNS on all sets of a small instances size (see red number and -0.18). Whereas, when applying ML policy, the average total costs obtained by SIRPT-based Randomised CWS are on average 9.26% worse than those solution costs obtained by ALNS on the same set. For the set of medium instances size, the average total costs obtained by Sim-Randomised CWS are on average 0.70% and 0.06 % slightly higher than the best solution costs in the literature for the SIRPT_ML and SIRPT_OU, respectively. This chapter proposed method can be found good solution costs for the large instances size but, they are higher than those average costs of the ALNS proposed by Coelho et al. (2012). Under ML policy, they have 3.36% higher than those obtained by the proposed algorithm in the literature, and up to 0.86% slightly worse for the same set with under OU policy. There is a significant difference between the two methods. This is reasonable, Leandro's work adopted a meta-heuristic to solve the problem, whereas a heuristic is used in this work.

The comparison of computer runtimes between the two cases are shown in Table 6.4. First observation is that the average running time is less than in the literature. Since an algorithm's running time may vary with different input of the instances size, which is the maximum amount of time taken on inputs of the large instances size for SIRPT_OU. It can be solved one and half hours, but this algorithm requires less time than algorithm in the literature. Whereas, for the SIRPT_ML on all instances sets with up to 200 customers, this chapter proposed algorithm requires less than one hour. Furthermore, Sim-Randomised CWS heuristic can also solve small and medium instances size problems in which it requires less time than ALNS for the SIPRT under ML/OU inventory policies with a good computation time.

Size of instances	ML replenishment policy					OU replenishment policy				
	ALNS		SIRPT-based Sim-Randomised CWS		Gap (%)	ALNS		SIRPT-based Sim-Randomised CWS		Gap (%)
	Cost	Time (s)	Cost	Time (s)		Cost	Time (s)	Cost	Time (s)	
Small (5≤n≤25)	7926.71	44.60	8193.11	19.10	3.36	8355.69	67.40	**8340.33**	**39.10**	-0.18
Medium (50≤n≤100)	26,527.05	444.10	26,712.80	397.70	0.70	26,891.26	880.50	26,907.30	649.30	0.06
Large (125≤n≤200)	54,292.38	4100.10	56,117.99	2979.30	3.36	55,530.30	9196.00	56,007.20	5407.30	0.86
Average	**29,582.05**	**1529.60**	**30341.30**	**1132.03**	**2.47**	**30,259.08**	**3381.30**	**30,418.28**	**2031.90**	**0.24**

Table 6.4 : Comparison of average results of the SIRPT_ML and SIRPT_OU based Sim-Randomised CWS heuristic with ALNS

6.5.2.2. *Comparison of the results of the SIRPT-based Sim-Randomised CWS with the SIRP-based Sim-Randomised CWS*

To further demonstrate the performance of the chapter proposed approach, a comparison is made between the results for the case without and with lateral transhipments for all instances sets with different customer size and the inventory replenishment policies. This sub-section presents the percentage improvement performance of the case with lateral transhipment, which is reported in Table 6.5. The comparison presents in Table 6.5 that the solution costs obtained in the two cases study are shown with the average total costs for each approach. The last column gives the percentage improvement between these approaches.

Table below clearly shows the computational solution costs obtained by two approaches with under ML/OU policies namely: SIRP-based Sim-Randomised CWS and SIRPT-based Sim-Randomised CWS, in term of the average total costs. The SIRPT-based Sim-Randomised CWS results are better than the SIRP-based Sim-Randomised CWS under ML/OU policies for all instances sizes. Under the ML inventory policy, the SIRPT-based Sim-Randomised CWS is able to provide better performance than the SIRP-based Sim-Randomised CWS for all groups of instances size, they are on average 23.09% better than those solution costs of the SIRP. When using OU policy, the SIRPT shows less average total costs than those characterised by the SIRP, they are on average 15.69% improvement for all three groups of instances sizes. Which is evident in Table 6.5, applying the transhipment can lead to minimise the total costs, in such a way that would decrease shipping costs. It also able to improve the services to satisfy the customer demand. However, the use of transhipment is not considered when there are relatively few stock-outs, allowing them further reduces stock-out as well as increasing the total cost. It is remarkable that, allowing lateral transhipment enables all customers to share stock-out risk and their respective inventories, which means the inventory being delivered to the customers can decrease inventory holding costs.

Size of instances	ML replenishment policy		
	SIRP-based Sim-Randomised CWS	**SIRPT-based Sim-Randomised CWS**	**Improvement (%)**
Small (5≤n≤25)	10,231.54	8193.11	24.88
Medium (50≤n≤100)	33,430.23	26,712.80	25.15
Large (125≤n≤200)	66,921.89	56,117.99	19.25
Average	**36,861.22**	**30,341.30**	**23.09**
Size of instances	**OU replenishment policy**		
	SIRP-based Sim-Randomised CWS	**SIRPT-based Sim-Randomised CWS**	**Improvement (%)**
Small (5≤n≤25)	9588.32	8340.33	14.96
Medium (50≤n≤100)	31,133.45	26,907.30	15.71
Large (125≤n≤200)	65,197.88	56,007.20	16.41
Average	**35,306.55**	**30,418.28**	**15.69**

Table 6.5 : Comparison of average results of the SIRP and SIRPT based Sim-Randomised CWS heuristic under ML/OU policies

6.6. Chapter Conclusion

This chapter focuses on IRP and IRPT with stochastic demand (SIRP and SIRPT) which is an extension of the problem studied in the previous chapter. From the literature point of view, stochastic and a more realistic version of IRP and IRPT have been considered. VMI is introduced in the problem where the supplier is responsible for managing customers' inventories under OU and ML policies. Therefore, in the case of SIRP, a penalty is considered when the supplier is not able to satisfy the customer demand and stock-out. While, in the case of SIRPT, if customer faces stock-out or inventory is not able to serve demand, then it allows transhipment to be taken into account instead of a penalty being charged. Transhipment can be either supplier-to-customer, or customer-to-customer. To solve these stochastic problems, Sim-Randomised CWS heuristics, which combines a Randomised Clark and Wright saving heuristic and Simulation technique (generated demand and Monte-Carlo) is implemented. By applying the Monte-Carlo simulation, the method is able to deal with some real-life situations or stochastic problem effectively. The results show that the proposed Sim-Randomised CWS heuristics can able to solve the SIRP and SIRPT. However, Sim-Randomised CWS heuristics does not provide good solutions as ALNS.

Moreover, implementing transhipment in the problem shows better results than that using SIRP without transhipment, which means the transhipment can easily be integrated in the routing problem and it has proved successful as it acts as a recourse function for the SIRP. To our knowledge, this is the first time that Randomised CWS algorithms with Simulation methodologies have been used to solve SIRPT and SIRP. In addition, the Sim-heuristic is found to be an efficient method since they are able to

apply the strength of simulation, which engages with the stochastic problem. SIRPT-based Sim-Randomise CWS heuristic proved that for small size instances, it has an improved performance, shown above where the average total costs are less than that of the ALNS for medium and large size instances. The next chapter considers SIRP and SIRPT using different algorithm (an Iterated Greedy algorithm) to improve the Sim-Randomised CWS heuristic approach. Computational experiments will be conducted and results will be compared with ALNS (Coelho et al., 2012).

Chapter 7: Sim-Randomised CWS Heuristic and Iterated Greedy (IG) with local search for Stochastic IRP and IRPT

7.1. Introduction

Heuristic and meta-heuristics have been successfully applied for large VRP, Flowshop Scheduling Problem (FSP), IRP instances (Blanton & Wainwright, 1993; Roldan et al., 2016). Meta-heuristics method such as simulated annealing, iterated greedy, Tabu search and local search have been used to solve the IRP (Qin et al., 2014). Local search techniques were proposed a few years ago and have shown good performance in experimental studies. Recently, many research have proposed local search to solve IRP (Benoist, Jeanjean, & Estellon, 2002; Benoist, Estellon, Gardi, & Jeanjean, 2009; Qin, Miao, Ruan, & Zhang, 2014). Their results showed examples of successful local search implementations and high performance for solving IRP. They stated that local search algorithms are considered to be one of the available methods, which can provide good solution to large problem size. Also, Ruiz and Stutzle (2007) showed excellent results that were obtained with an Iterated Greedy (IG) method for the Permutation Flowshop Scheduling Problem (PFSP), where it yielded results that were superior to those from several more complex algorithms. IG method is easily applied to other FSP, it is very simple to code and parameter free (Ruiz & Stutzle, (2007). The result produced by IG is very effective and provided new best-known solutions for PSFP, as shown by Ruiz and Stutzle (2007).

For this reason, this chapter proposes the implementation of combined Sim-heuristic and IG with a local search algorithm to perform good solution of SIRP and SIRPT, in order to improve the previous results, which presented in Chapter 6. This method will present as SIRP and SIRPT – based Sim-Randomised CWS heuristic and IG with local search algorithm under both ML and OU inventory replenishment policies. The IG algorithm, despite its simplicity, is very competitive, and is a very efficient heuristic. It is sensible to choose the IG, which has been adapted successfully for many COPs. This algorithm has not previously been applied for solving IRP and IRPT in both deterministic and stochastic versions.

The chapter is structured as follows: section 7.2 presents this chapter contributions, and section 7.3 presents the proposed approach to solve SIRP and SIRPT. All of the

computational experiments of this chapter are shown in section 7.4. Finally, the conclusion of this chapter is in section 7.5.

7.2. Contribution

There are many published works that have used heuristic/meta-heuristic to address IRP with stochastic demand. However, implementing of an IG with local search algorithm to solve the SIRP and SIRPT has not yet been proposed in the literature. Therefore, this chapter proposes the combined IG with local search and Sim-Randomised CWS heuristic for SIRPT. This is a major contribution of this chapter.

The basic IG method has two phases: the destruction and the construction phases. In the destruction phase, the current solution is partially destroyed. Here, some customers are selected from the current solution. Then the construction phase is started. The selected customers are re-inserted into the partially destroyed solution until the stopping condition is met. The construction phase uses a rather straightforward local search method that is based on the insertion neighbourhood. In literature, this local search is able to improve a solution, which can choose neighbourhood in iterative improvement.

7.3. Proposed SIRP and SIRPT-based Sim-Randomised CWS heuristic and IG with local search algorithm

This chapter proposes the Iterated greedy (IG) method with local search which generates solutions by iterating greedy constructive heuristics, by applying two main phases: these are named destruction and construction. Then, the IG algorithm using local search in the construction phase absolutely, has been applied for improving the final solutions with the aim to minimise the expected total costs. Also, a local search is straightforward to add into the procedure of the IG method, which is generally considered as being a very good option for the complex problems (Ruiz & Stutzle, 2007).

This approach has three main stages. In the first stage, the supplier has made a decision on the quantities for serving the demand of the customers and an initial route is generated by applying inventory replenishment policies and Sim-Randomised CWS heuristic. In the second stage, the supplier has delivered all demand of the customers by implementing IG with local search. In the final stage, MSC is used to generate random variables to estimate the expected total cost in order to minimise total costs. In the case of SIRPT where stock-out has occurred, the lateral transhipments are

allowed to take place after demand is realised (Chapter 5: section 5.3 and Chapter 6: section 6.3).

The processes of implanting IG with local search to the proposed methods have been explained in Chapter 5. Thus, the same processes are also used for stochastic demand in the next section.

7.3.1. SIRP-based Sim-Randomised CWS heuristic and IG algorithm with local search

This chapter proposes an extension by applying local search into the IG algorithm in order to improve the solution of SIRP under both inventory replenishment policies from Chapter 5. This is one of the best choices for implementing Sim-Randomised CWS heuristic - applying an IG algorithm with local search to solve all instances of these problems. The process of implementation is described as follows: at the beginning of the algorithm initialisation uses Sim-Randomised CWS heuristic, which was proposed in Chapter 6. This heuristic obtains the routes and gives the initial solution. Randomised CWS solution is processed by an iterative greedy procedure (IG), which uses the best results from the current solution to improve on it. This sub-section explains in detail, the Sim-Randomised CWS heuristic and IG algorithm with local search, for solving the IRP and IRPT with stochastic demand under OU/ML policies. The description of the main procedure is as follows: (a) the Randomised CWS heuristic, which uses a multi-start solution generation approach, (b) a simulation of random demand, which follows a uniform distribution, (c) IG, which improves the solution by using an iterative greedy heuristics; using two main phases (destruction and construction), (d) the use of local search methods, to improve the resulting solution and (e) Applying multi times of Monte Carlo simulation procedure, to search for the better the results as show in Algorithm 7.1 (Pseudocode for SIRP–based Sim-Randomised CWS heuristic and IG with local search algorithm).

Algorithm 7.1: Pseudocode for SIRP-based SimRandomised CWS Heuristic and IG with local search

1: **Procedure IRP SimRandomisedCWSIteratedGreedyLS**
(inventoryPolicy, nodeList, $d_n, I_n, p_n, h_n, b_n, q_n, l_n, w_n, n, \beta, r', r^D, r^R, d$*)*
▷ d_n: *Demand*
▷ I_n: Current inventory level
▷ q_n: Quantity delivery
▷ n: Customer node
▷ β: parameter for biased randomisation
▷ r': *Initail solution*
▷ r^D: Partial sequecne to reconstruct
▷ r^R: Customers to reinsert
▷ d: Random chosen number

2: **for** *each customer* n **do** ▷ *Inventory Procedure*
3: $q_{n,ML/OU} \leftarrow \widehat{I_n} - I_n$
4: $q \leftarrow add\ q_{n,ML/OU}$ *to the list of demands* d_n
5: **end for**
6: $Depot_{ML/OU} \leftarrow Distribution\ (routing) cost\ associated\ with\ r$
7: currentSol ← initialSol ▷ *RandomisedCWS*
8: innitialSol ← IterativeImprovement_Insertion(initailSol) ▷*IG with local search*
9: $r' \leftarrow$ currentSol
10: **while** stopping criteria not satisfied **do**
▷Destruction_ Construction
11: $r^{'} \leftarrow$ currentSol
12: **for** $n = 1\ to\ d$ **do** ▷ $d = 3$
13: $r^{'} \leftarrow$ remove one node at random from $r^{'}$ and insert it in r^R;
14: **end for**
15: **for** $n = 1\ to\ d$ **do**
16: $r^{'} \leftarrow$ best permutation obtained by inserting node r^R in all **possible positions** of r^D;
17: **end for**
18: $r^{''} \leftarrow$ IterativeImprovement_Insertion ($r^{'}$); ▷ Neighbourhood (LS)
19: **if** $Costs_{max}(r^{''}) < Costs_{max}(r)$**then**
20: $r = r^{''}$
21: **if** $Costs_{max}(r) < Costs_{max}(r^{''})$ **then**
22: $r' = r$
23: **end if**
24: **elseif** $(\boldsymbol{random} \leq \frac{\boldsymbol{exp}\{-\boldsymbol{Costs_{max}}(\boldsymbol{r^{''}})-\boldsymbol{Costs_{max}}(\boldsymbol{r})\}}{\boldsymbol{Temperture}})$ **then**
25: $r = r^{''}$;
26: **end if**
27: **end while**
28: solExpectedCosts (nodes, Sol) ▷ MCS
31**:** **for** *each sol in best StochSolList do*
32: statistics(*sol)* ← *MCS* (irpSol;bestSol;nSols, β)
33**:** **end for**
34: **return** *bestSolStocSolList*
35**: end procedure**

Algorithm 7.1 shows the procedure of the proposed SIRP–based Sim-Randomised CWS heuristic and IG algorithm with local search. Firstly (Pseudocode: line 2 to line 4), the inventory level for each customer at the end period is calculated; it depends on the initial stock level and also on the end-customer's demands during that period. These end-customer's demands are stochastic in nature, one assumes that for each customer it has been possible to use historical data to model an end-customer's demand through a theoretical or empirical probability distribution. Therefore, a natural generalisation has carried out by using random instead of deterministic demands, then in this work, the uniform distribution is used to generate the demand of each customer. At the end of each period, inventory levels are registered by the customer and updated in the central depot. So, new routing strategy is defined for a new period taking account the new data. Then, an initial dummy solution is generated (Pseudocode: line 6 and line 7) by Randomised CWS heuristic. After that, an iterative improvement algorithm is started, by using a first improvement type pivoting rule (Pseudocode: line 8 to line 16). Following, the destruction and construction phases and the optional local search phase, then whether the new route is accepted or not as the current solution for next iteration is considered. The reason simplest acceptance criteria is new route, which provide a better solution (lowest total costs).

The completed Iterated Greedy algorithm with the aforementioned iterative improvement algorithm and acceptance criterion, is summarised (Pseudocode: line 18 to line 25). This procedure has only two parameters, T andd, and the application of IG iteration to this instance is shown (Chapter 5 section 5.3.1). This instance has $m = 1$ and n classified to small($5 \leq n \leq 25$), medium ($50 \leq n \leq 100$) and large ($125 \leq n \leq 200$) and the local search phase is using $d = 3$. A natural generalisation has been carried out by using randomisation. A natural generalisation has been carried out by using randomisation. At the end of each first level iteration, the resulting solution goes through Monte-Carlo simulation (MCS) procedure which provide estimates of associated expected variable costs (Pseudocode: line 28 to line 34). Since MCS has been used, these random demands can follow any probability distribution as far as it has a mean.

7.3.2. SIRPT-based Sim-Randomised CWS heuristic and IG algorithm with local search

Sim-Randomised CWS heuristic and IG algorithm with local search is also applied for solving the SIRPT_ML and SIRPT_OU without any structure or procedure changed.

In this case, if lateral transhipments are not allowed to take place after the demand is fulfilled then, the customer may face stock-out. In another case, where high demand has occurred, demand once again is generated after the vehicle has arrived at the customer location. It is an alternative technique to use transhipment later than the supplier pays the penalty costs.

Algorithm 7.2 (Pseudocode: line 2 to line 4) shows the demand at each node and is simulated by sampling the specified stochastic distribution (uniform distribution). The inventory level for each customer at the end period is calculated. Then the quantity of product to deliver is calculated by the supplier under ML or OU policies (see Inventory Procedure 6.2). The transhipment is calculated which can be undersupplied or oversupplied (Pseudocode: line 5 to line 7). Transhipment has occurred when the customer faces a shortage or there is higher demand. The current inventory for each node is updated. Then the nearest node is found which has enough product to serve another customer (oversupplied list) - to be the supplier (transhipment). Finally, the round-trip cost between the supplier node and the demand node is calculated. The round-trip costs between them is the transhipment cost which is added to the total cost. After that the initial dummy solution is generated by using Randomised CWS heuristic, the initial solution is improved by using an iterative Greedy algorithm (see Algorithm 5.1. and 5.2.). For the SIRPT case, the demand is generated once again when the vehicle has arrived, which uses a uniform distribution. Whenever demand is high, the stock-out occurs, then the transhipment is taken and subsequent cost is added (see Algorithm 3.2: Transhipment procedure). After iterating this process thousands of times, a random sample of costs is obtained, from which an average value can be estimated. After the final iteration, the Monte-Carlo simulation (MCS) procedure is used to simulate the solutions providing estimates of associated expected variable costs (Pseudocode: line 30 to line 33).

Algorithm 7.2: Pseudocode for SIRPT-based SimRandomised CWS Heuristic and IG with local search

1: **Procedure IRPT SimRandomisedCWSIteratedGreedyLS**
(inventoryPolicy, nodeList, $d_n, I_n, p_n, h_n, b_n, q_n, l_n, w_n, n, \beta, r', r^D, r^R, d$*)*
▷ d_n: *Demand*
▷ I_n: Current inventory level
▷ q_n: Quantity delivery
▷ n: Customer node
▷ β: parameter for biased randomisation
▷ r': *Initail solution*
▷ r^D: Partial sequecne to reconstruct
▷ r^R: Customers to reinsert
▷ d: Random chosen number

2: **for** *each customer* n **do** ▷ *Inventory Procedure*
3: $q_{n,ML/OU} \leftarrow \widehat{I_n} - I_n$
4: $q \leftarrow add\ q_{n,ML/OU}$ *to the list of demands* d_n
5: **if** policy = TRANSHIPMENT **then**
6: *transhipmentCosts* $\leftarrow b_n * w_n$ ▷ *Transhipment*
7: **end if**
8: **for** each n in *OSL* **do**
9: *inventoryCosts* $\leftarrow h_n * I_n + p_n * l_n$
10: **end for**
11: **end for**
12: $Depot_{ML/OU} \leftarrow Distribution\ (routing) cost\ associated\ with\ r$
13: currentSol ← initialSol ▷ *RandomisedCWS*
14: innitialSol ← IterativeImprovement_Insertion(initailSol) ▷*IG with local search*
15: $r' \leftarrow$ currentSol
16: **while** stopping criteria not satisfied **do** ▷Destruction_ Construction
17: **for** $n = 1\ to\ d$ **do** ▷ $d = 3$
18: **end for**
19: $r'' \leftarrow$ IterativeImprovement_Insertion (r'); ▷ Neighbourhood (LS)
20: **if** $Costs_{max}(r'') < Costs_{max}(r)$**then**
21: $r = r''$
22: **if** $Costs_{max}(r) < Costs_{max}(r'')$ **then**
23: $r' = r$
24: **end if**
25: **elseif** $(\boldsymbol{random} \leq \frac{\boldsymbol{exp}\{-\boldsymbol{Costs_{max}}(\boldsymbol{r''})-\boldsymbol{Costs_{max}}(\boldsymbol{r})\}}{\boldsymbol{Temperture}})$ **then**
26: $r = r''$;
27: **end if**
28: **end while**
29: solExpectedCosts (nodes, Sol) ▷ MCS
30: **for** *each sol in best StochSolList do*
31: statistics(*sol)* ← *MCS* (irpSol;bestSol;nSols, β)
32**:** **end for**
33: **return** *bestSolStocSolList*
34**: end procedure**

7.4. Computational experiments

The algorithm described in this chapter has been implemented as a Java application. The algorithms are coded in Java programming language with eclipse software, and all the tests are run on a computer with Intel(R) Core(TM) i5-4590, CPU 3.30 GHz processor and 8.00 GB of RAM. In the case of the Java implementation that it used some class from SSJ library by Juan et al. (2012). Therefore, to evaluate the performance of the proposed algorithm, it was tested in a data set introduced by Coelho et al. (2012), composed by 150 instances in each set for total of 450 instances in which these details have been presented in Chapter 6. There are three data sets and each of the 30 combinations n and p generated five instances. Their nomenclature follows the rule $dirp - n - p - 1$ through $dirp - n - p - 5$. Data set provides summaries of aggregating instances by three sizes; those with less than 50 customers are labelled small ($5 \leq n \leq 25$), those containing between 50 and 100 customers are called medium ($50 \leq n \leq 100$) and those with more than 100 customers are large ($125 \leq n \leq 200$). Number of customers (n) is equal $5k$ where $k =$ 1,2,3,5,10,15,20,25,30,40 and the different planning horizons are equal to 5, 10 or 20 periods. This set of instances will be called the stationary data set since the mean of demand distribution is stationary. The reorder point of the stationary data set is computed by $s_i = \hat{\mu}_i + z_\alpha \hat{\sigma}_i$ which assumes that demand of consecutive periods is independent. From the equation, using the last known demand as an expectation of future demand and given the lead time is equal to one (all deliveries are performed in the next period) and standard deviation is zero. However, (Coelho et al., 2012) believes that computing the reorder point in this approximate way does not have a major impact on the results.

Subsequently, this chapter used instances set the same as Chapter 6, which compares the results after implementing the Sim-Randomised CWS heuristic by combining IG algorithm with local search for improving the performances of our proposed methodology. This is in order to achieve the aims, which have been presented in Chapter 1and 5; also, to improve this algorithm to find good quality solutions for the SIRP and SIRPT under ML/OU replenishment policy.

Nevertheless, the results are solved by under the OU and ML inventory replenishment policies allow for fair comparison with the results proposed by Coelho et al. (2012) with the same policies. Under these policies, Sim-Randomised CWS heuristic and IG

algorithm with local search reduces the supply quantities while respecting the vehicle and the customer capacities. The transhipment cost $b_i$was set to 0.01 as in previous chapter and Coelho et al. (2012). Most of the findings will be giving explanations and presented through tables.

The experimental results of cost breakdown of the total costs for the SIRP_ML and the SIRP_OU, and the SIRPT_ML and the SIRPT_OU are shown in Table 7.1 and 7.4, respectively. In following tables: the first column is the name of instance, which are set of instances as explained above. Next column is HC representing holding cost, the penalty cost is denoted by PC, TrC stands for transhipment cost, fifth column is the routing cost (RC) and the last column is TC, denoting total costs. The cost breakdown of total costs chooses one result from results of each five instances in each data set.

7.4.1. SIRP-based Sim-Randomised CWS heuristic and IG algorithm with local search

Table 7.1 shows the result of cost breakdown of SIRP_ML/OU-based Sim-Randomised CWS heuristic and IG algorithm with local search in which the average costs are presented in bold number. The average total cost of SIRP_ML-based Sim-Randomised CWS heuristic and IG algorithm with local search, are slightly higher than those costs resulting from SIRP_OU-based Sim-Randomised CWS heuristic and IG algorithm with local search. This is the same reason, which has been given and described in Chapter 6. In the case of demand that is not known in the future, an increased demand could happen, or demand is lower than expected if these cases occur then the quantity for serving the customer by using OU or ML policy affects the total cost. The use of ML inventory replenishment policy may have a high inventory holding cost, but it could be held the high demand, which reduces paying a penalty cost. On the other hand, the OU replenishment policy can help to reduce inventory holding cost but the supplier could not serve customer as required, therefore penalty costs will be charged into total costs. Hence, SIRP without transhipment, which do not give the transhipment cost, but a penalty cost is charged when the customer faces the stock-out or the supplier could not serve customer demand when extra demand is occurred (Table 7.1). According to the experimental results, the average penalty and routing costs obtained by SIRP_OU-based Sim-Randomised CWS heuristic and IG algorithm with local search have consistently higher than those average costs provided by the SIRP_ML-based Sim-Randomised CWS heuristic, whereas holding costs are reduced when using the OU policy. This is reasonable that using the ML

policy, the inventory capacity fills up to maximum level. Therefore, the supplier can avoid and costly visits to the same location, which can lead to reduce routing costs. However, carrying high levels of inventory can be disadvantage in which it can lead to increase inventory holding cost.

Name of Instance		SIRP_ML-based Sim-Randomised CWS and IG with local search				SIRP_OU-based Sim-Randomised CWS and IG with local search			
		HC	PC	RC	TC	HC	PC	RC	TC
1	dirp-5-5-1	125.12	75.76	1911.01	2111.89	105.99	340.73	1465.91	1912.62
2	dirp-5-10-1	324.39	1076.34	2418.65	3819.38	62.66	1274.60	2184.69	3521.95
3	dirp-5-20-1	535.28	1512.10	3616.20	5663.58	178.07	2377.31	3002.17	5557.56
4	dirp-10-5-1	228.65	1220.00	4374.94	5823.59	144.89	1308.29	4053.14	5506.32
5	dirp-10-10-1	457.97	2254.05	5187.29	7899.31	232.46	2931.40	5861.35	9025.21
6	dirp-10-20-1	875.34	2726.55	6303.16	9905.05	425.33	2432.89	6407.47	9265.69
7	dirp-15-5-1	283.66	1254.65	4502.99	6041.30	181.39	1420.29	5046.13	6647.82
8	dirp-15-10-1	621.42	2068.00	7300.85	9990.27	276.30	1717.60	7308.91	9302.81
9	dirp-15-20-1	1239.80	3456.89	8633.98	13330.67	663.40	2404.17	10637.57	13705.14
10	dirp-25-5-1	543.22	3234.09	6042.95	9820.26	309.92	1041.71	8009.91	9361.54
11	dirp-25-10-1	846.00	3609.50	8907.90	13363.39	482.02	4548.57	11394.00	16424.59
12	dirp-25-20-1	1816.85	5958.56	12305.73	20081.14	885.10	5616.97	13629.55	20131.61
13	dirp-50-5-1	1022.51	2172.77	9525.75	12721.03	603.72	1572.36	9761.31	11937.39
14	dirp-50-10-1	2283.67	4882.88	19249.53	26416.08	1147.16	5529.27	18553.89	25230.32
15	dirp-50-20-1	4533.43	8930.50	26162.31	39626.24	1644.66	10962.46	27565.23	40172.35
16	dirp-75-5-1	1464.68	6772.55	9056.10	17293.33	846.63	5958.15	10052.14	16856.92
17	dirp-75-10-1	3381.80	9766.01	10486.50	23634.31	1625.21	10092.97	12141.29	23859.46
18	dirp-75-20-1	5628.33	10964.01	16529.44	33121.78	2673.21	11230.40	22000.87	35904.48
19	dirp-100-5-1	2130.44	7886.00	10067.10	20083.54	1222.19	8415.13	11202.29	20839.61
20	dirp-100-10-1	4104.18	13208.33	20275.94	37588.45	1976.32	10420.10	26758.12	39154.55
21	dirp-100-20-1	7670.73	10766.44	30988.39	49425.56	3428.81	12656.03	38740.70	54825.54
22	dirp-125-5-1	2290.91	11088.99	11375.83	24755.73	1400.44	9001.31	11981.57	22383.32
23	dirp-125-10-1	5344.32	14346.11	19827.27	39517.70	2785.18	10917.33	29479.51	43182.02
24	dirp-125-20-1	9641.64	19816.18	39144.32	68602.14	4224.17	20092.31	40068.16	64384.65
25	dirp-150-5-1	2880.06	11744.00	14144.49	28768.55	1720.40	11091.93	12795.49	25607.82
26	dirp-150-10-1	6266.17	20554.21	31632.39	58452.77	3056.98	22708.13	31966.82	57731.94
27	dirp-150-20-1	13919.33	28858.21	57120.65	99898.19	6110.97	34263.69	58122.06	98496.72
28	dirp-200-5-1	3874.99	15254.21	16030.49	35159.69	2279.80	12091.09	16957.90	31328.79
29	dirp-200-10-1	8312.15	27208.23	33052.67	68573.05	4131.17	20524.33	33118.05	57773.55
30	dirp-200-20-1	16710.18	37684.76	68216.36	122611.30	6815.69	44716.23	67747.61	119279.53
Average		**3645.24**	**9678.36**	**17146.37**	**30469.98**	**1721.34**	**9655.26**	**18600.46**	**29977.06**

Table 7.1 : Average results of the cost breakdown of SIRP under ML and OU policies

7.4.1.1. *Comparison of the results of SIRP-based Sim-Randomised CWS and IG with local search, with the best results in the SIRP-based Sim-Randomised CWS*

Table 7.2 presents the percentage improvement of performance after implementing a Sim-Randomised CWS heuristic by applying the IG algorithm with local search for solving SIRP_ML and SIRP_OU inventory replenishment policy respectively. Table bellows clearly show the average solution costs for SIRP with difference methods being compared, which are reported in 4 columns by each policy as following. The first column gives the name of the instances size, the second shows the average solution costs obtained by Sim-Randomised CWS heuristic (Table 7.2) the third column is the average total costs solved by SIRP-based Sim-Randomised CWS heuristic and IG algorithm with local search. The percentage change between two values obtained by this chapter proposed method and the Sim-Randomised CWS heuristic (Chapter 6) are shown in the last column.

Size of instances	**ML inventory replenishment policy**		
	SIRP-based Sim-Randomised CWS	**SIRP-based Sim-Randomised CWS and IG with local search**	**Improvement (%)**
Small (5≤n≤25)	10,231.54	**8858.29**	15.50
Medium (50≤n≤100)	33,430.23	**28,919.69**	15.60
Large (125≤n≤200)	66,921.89	**59,877.88**	11.76
Average	36,861.22	**32,551.95**	**14.29**
Size of instances	**OU inventory replenishment policy**		
	SIRP-based Sim-Randomised CWS	**SIRP-based Sim-Randomised CWS and IG with local search**	**Improvement (%)**
Small (5≤n≤25)	9588.32	**8827.98**	8.61
Medium (50≤n≤100)	31,133.45	**26,236.66**	18.66
Large (125≤n≤200)	65,197.88	**56,037.85**	16.35
Average	35,306.55	**30,367.50**	**14.54**

Table 7.2 : Comparison of average results between SIRP based Sim-Randomised CWS heuristic and IG algorithm with local search and SIRP based Sim-Randomised CWS under ML and OU policies

The computation experiment results show an improvement of the proposed method when implementing IG algorithm with local search to Sim-Randomised CWS heuristic. The proposed algorithm can provide good quality solutions for all instances of SIRP_ML and SIRP_OU-based Sim-Randomised CWS heuristic and IG algorithm with local search. They are giving better average total costs than those algorithms which were introduced by Coelho et al. (2012) and the proposed method in Chapter 6. The percentage improvements of solutions after combination are; 14.29 % for SIRP_ML and 15.54 % for SIRP_OU, respectively as shown in Table 7.2.

7.4.1.2. *Comparison of the results of SIRP-based Sim-Randomised CWS and IG with local search, with the best results in the literature*

Table 7.3 presents the comparison between the proposed algorithm (Sim-Randomised CWS heuristic and IG algorithm with local search) and ANLS (Coelho et al., 2012). SIRP -based Sim-Randomised CWS heuristic and IG algorithm with local search find better results than ANLS for all average total costs, small, medium and large instances size. The average percentage gaps are approximately 6.79% and 7.65% with respect to ML and OU policies. Also, this chapter proposed method which uses less computer runtime than that ALNS. For using ML policy, the average total costs obtained by SIRP-based Randomised CWS and IG algorithm with local search are better than those average costs obtained by ALNS. They are on average 13.37% better for the small instances size. Whereas, under OU policy, they have 3.32% lower than those solved by ALNS for the same data set. For the medium group instances size and under ML policy, the solutions solved by this chapter proposed algorithm are on average 4.75% better than those results solved by ALNS and on the same set by using OU replenishment policy, they have 12.94% lower than the results found by ALNS. This chapter proposed algorithm is able to provide best new solution for the large instances size. They are on average 2.24 % better than those average total costs provided by ALNS under the ML policy. Similarly, they have 6.68% better on the same instances set with handle by OU policy.

Furthermore, this chapter implemented method is able to be solved small, medium and large group instances size problems with a good computation time. All instances with up to 200 customers can be solved an about one hour under the ML policy. When using OU policy for the same group instances set with up to 200 customers can solve less than two and half hours. From all results, and the comparison of solutions with the benchmark and previously proposed approach in this study, it can be seen that the solution obtained by this proposed approach perform the potential to solve SIRP, which is shown in the table below. Consequently, applying IG algorithm with local search is able to provide good solutions. It means that the IG algorithm with local search has the ability to improve the Sim-Randomised CWS heuristic. This proposed approach has met this chapter's aim, which is to improve the solution generated by the Sim-Randomised CWS heuristic.

Size of instances	ML inventory replenishment policy				
	ALNS		SIRP-based Sim-Randomised CWS and IG with local search		Improvememt (%)
	Cost	Time (s)	Cost	Time (s)	
Small (5≤n≤25)	10,225.93	46.3	**8858.29**	35.8	13.37
Medium (50≤n≤100)	30,360.66	452.7	**28,919.69**	403.4	4.75
Large (125≤n≤200)	61,250.17	3860.1	**59,877.88**	3709.1	2.24
Average	33,945.59	**1453.0**	**32,551.95**	**1382.8**	**6.79**
Size of instances	**OU inventory replenishment policy**				
	ALNS		**SIRP-based Sim-Randomised CWS and IG with local search**		**Improvememt (%)**
	Cost	**Time (s)**	**Cost**	**Time (s)**	
Small (5≤n≤25)	9131.45	67.1	**8827.98**	49.2	3.32
Medium (50≤n≤100)	30,137.81	888.3	**26,236.66**	750.6	12.94
Large (125≤n≤200)	60,051.36	9248.9	**56,037.85**	7802.9	6.68
Average	33,106.87	**3401.4**	**30,367.50**	**2867.6**	**7.65**

Table 7.3 : Comparison of average results between ALNS and SIRP based Sim-Randomised CWS heuristic and IG algorithm with local search under ML and OU policies

7.4.2. SIRPT-based Sim-Randomised CWS heuristic and IG algorithm with local search

Table 7.4 gives costs breakdown for SIRPT_ML and SIRPT_OU- based Sim-Randomised CWS heuristic and IG algorithm with local search, which have been mentioned and explained in the sub-section above. In this case transhipments are considered in order to further reduce total costs and stock-out, as well as allowing all customers to share stock-out risks and their respective inventories. SIRPT, the lateral transhipment is an option for the supplier to use to serve the extra demand. Also, the supplier is responsible for the transhipment cost but it can avoid the stock-out and the penalty cost. Therefore, applying ML inventory replenishment policy yields reductions in solution costs and in lost demand. It can be seen that SIRPT_ML –based Sim-Randomised CWS heuristic and IG algorithm with local search is able to reduce total costs more than SIRPT_OU. The ML policy has an inventory holding cost, but it could reduce penalty costs (penalty costs are higher than transhipment and holding costs). According to Table 7.4, SIRPT_OU has lower inventory holding cost; however, transhipment and routing costs are higher than SIRPT under ML policy.

Name of Instance		SIRPT_ML-based Sim-Randomised CWS and IG with local search				SIRPT_OU-based Sim-Randomised CWS and IG with local search			
		HC	TrC	RC	TC	HC	TrC	RC	TC
1	dirp-5-5-1	194.95	129.52	729.98	1054.46	108.76	215.38	728.90	1053.05
2	dirp-5-10-1	313.43	385.42	1136.57	1835.42	129.10	405.87	1206.94	1741.91
3	dirp-5-20-1	537.59	1307.82	1784.51	3629.91	218.63	1527.09	1833.44	3579.17
4	dirp-10-5-1	213.10	560.05	3479.06	4252.21	119.56	704.90	3316.63	4141.09
5	dirp-10-10-1	433.13	1057.07	4527.09	6017.29	212.31	1737.70	4294.01	6244.02
6	dirp-10-20-1	808.12	1932.48	5444.29	8184.89	356.20	2488.26	5207.72	8052.18
7	dirp-15-5-1	379.39	712.51	3085.42	4177.31	209.31	1016.19	4001.59	5227.09
8	dirp-15-10-1	737.69	1447.04	6418.37	8603.10	320.23	4183.83	4320.61	8824.67
9	dirp-15-20-1	1009.40	2374.42	7044.39	10428.21	638.28	3093.82	5734.84	9466.94
10	dirp-25-5-1	490.00	630.78	6268.25	7389.03	373.37	1963.33	6221.84	8558.55
11	dirp-25-10-1	1024.65	2049.14	8466.20	11539.99	516.35	4468.99	8702.71	13688.05
12	dirp-25-20-1	2208.25	3214.37	10414.77	15837.39	927.32	7006.47	10025.35	17959.14
13	dirp-50-5-1	961.29	1324.44	7520.22	9805.95	560.60	3504.40	5218.37	9283.37
14	dirp-50-10-1	2246.91	4187.89	11019.65	17454.45	1097.09	8019.48	9081.68	18198.25
15	dirp-50-20-1	4052.26	5115.99	16144.34	25312.59	1781.26	10029.23	16244.52	28055.00
16	dirp-75-5-1	1570.98	2020.83	9270.63	12862.44	919.98	2290.05	10080.99	13291.03
17	dirp-75-10-1	3277.26	4262.73	12063.55	19603.54	1557.31	3532.34	15004.36	20094.00
18	dirp-75-20-1	5592.82	6628.38	17204.97	29426.18	2817.35	10177.48	19980.36	32975.18
19	dirp-100-5-1	2054.32	2131.82	10067.43	14253.58	1173.79	3966.53	10494.49	15634.81
20	dirp-100-10-1	4077.00	5579.30	17681.69	27338.00	1975.35	9670.30	20817.27	32462.93
21	dirp-100-20-1	7431.08	10245.42	31960.81	49637.31	3351.46	17957.56	36599.73	57908.74
22	dirp-125-5-1	2765.20	3145.99	11912.95	17824.14	1303.61	5840.33	12056.26	19200.20
23	dirp-125-10-1	5091.53	4834.78	26766.13	36692.44	2382.20	12765.16	24063.62	39210.98
24	dirp-125-20-1	9668.10	10991.10	54207.79	74866.99	4283.96	14728.60	54198.45	73211.01
25	dirp-150-5-1	2827.72	2825.98	14892.48	20546.18	1648.69	4956.16	15054.95	21659.81
26	dirp-150-10-1	6102.67	5751.12	28200.54	40054.33	2882.88	7057.45	30044.51	39984.83
27	dirp-150-20-1	12437.84	12808.29	58719.68	83965.81	6007.64	12853.16	57961.28	76822.08
28	dirp-200-5-1	3744.21	3143.56	16101.24	22989.01	2153.19	6325.93	17304.04	25783.16
29	dirp-200-10-1	8532.00	7282.20	32167.60	47981.80	4145.10	11019.85	33296.65	48461.60
30	dirp-200-20-1	14542.92	14344.31	68246.82	97134.05	6645.67	16062.66	69506.33	92214.67
Average		**3510.86**	**4080.83**	**16764.91**	**24356.60**	**1693.89**	**6318.95**	**17086.75**	**25099.58**

Table 7.4 : Average results of the cost breakdown of SIRPT under ML and OU policies

7.4.2.1. *Comparison of the results of SIRPT-based Sim-Randomised CWS and IG with local search, with the best results in the SIRPT-based Sim-Randomised CWS*

The experimental results on all set of instances for the SIRPT by using Sim-Randomised CWS heuristic and IG algorithm with local search are able to compare with the solution costs obtained by Chapter 6 proposed method, which present in Table 7.5. This chapter implementation method is able to provide better solutions costs than the algorithm proposed by Chapter 6 for on all sets of instances. Under ML replenishment policy, the SIRPT-based Sim-Randomised CWS and IG algorithm with local search can be found the average results 10.78% better than those average results obtained by the SIRPT -based Sim-Randomised CWS on all sets of instances. For the small group of instances size, the SIRPT-based Sim-Randomised CWS and IG algorithm with local search have the average results 15.50% better than those average results provided by the SIRPT-based Sim-Randomised CWS, and they are on average 15.60% better for the medium group of instances. When the instances size is large, they are on average 11.76% better than those obtained by Sim-Randomised CWS heuristic. For the SIRPT under OU replenishment policy, this chapter proposed method is able to provide better average total costs than those obtained by Chapter 6 proposed method, they are on average 14.54% lower on all instances sets. On the small, medium and large group of instances sizes, they are 8.61%, 18.66% and 16.35% better than those solutions of the SIRPT solved by Sim-Randomised CWS, respectively.

Which can be seen from the percentage gap that this chapter proposed an implementation algorithm has potential to improve the previous solutions costs of the SIRPT under ML/OU policies. Thus, the implementation of Sim-Randomised with an IG algorithm with local search can lead to improve the performance for solving the SIRPT, which is able to find best new solutions. This implementation method also extends the searching to provide better result during the iteration. As can be seen, the best new solutions for the same data set and the group of instances, this chapter implemented algorithm can obtain best new solutions especially when instance size is larger.

Size of instances	ML inventory replenishment policy		
	SIRPT-based Sim-Randomised CWS	SIRPT-based Sim-Randomised CWS and IG with local search	Improvement (%)
Small (5≤n≤25)	8193.11	**6965.29**	17.63
Medium (50≤n≤100	26,712.80	**22,975.59**	16.27
Large (125≤n≤200)	56,117.99	**51,268.87**	9.46
Average	30,341.30	**27,069.92**	**14.45**
Size of instances	**OU inventory replenishment policy**		
	SIRPT-based Sim-Randomised CWS	**SIRPT-based Sim-Randomised CWS and IG with local search**	**Improvement (%)**
Small (5≤n≤25)	8340.33	**7537.50**	10.65
Medium (50≤n≤100	26,907.30	**25,323.87**	6.25
Large (125≤n≤200)	56,007.20	**52,129.26**	7.44
Average	30,418.28	**28,330.21**	**8.11**

Table 7.5 : Comparison of average results between SIRP based Sim-Randomised CWS heuristic and IG algorithm with local search and SIRP based Sim-Randomised CWS under ML and OU policies

7.4.2.2. Comparison of the results of SIRPT-based Sim-Randomised CWS and IG with local search, with the best results in the literature

The comparison of the average solution between ANLS (Coelho et al., 2012) and the SIRPT-based Sim-Randomised CWS heuristic and IG algorithm with local search presents in Table 7.6. The computation time of this chapter implementation algorithm and ALNS are also shown. The last column in this table presents the average percentage gaps. This chapter implemented algorithm shows the potential of applying an IG algorithm with local search that can be improved the Sim-Randomised CWS heuristic to find best solutions. The solution costs obtained by this chapter implementation method is better than those obtained by ALNS, they are on average 10.36% better for SIRPT_ML and 7.25% for SIRPT_OU, respectively (Table 7.6). Applying ML policy, the average solutions costs obtained by SIRPT-based Randomised CWS and IG algorithm with local search are on average 12.13% lower than those solution costs obtained by ALNS for all sets of a small instances size. When using the OU policy, the solution costs found by SIRPT-based Sim-Randomised CWS and IG algorithm with local search are on average 9.79 % better than those generated by ALNS for the same set. For the medium instances size, the average total costs obtained by this chapter implemented algorithm are on average 13.39% and 5.83% better than the best known in the literature for the SIRPT_ML and the SIRPT_OU, respectively. This chapter proposed method is able to find best new solution costs for the large instances size, they are on average 5.57% better than those found by ALNS for the SIRPT under ML policy, and up to 6.12% better for the SIRPT under OU policy.

Table 7.6 also reports the computation time of the two different algorithms. This chapter proposed method requires less time than ALNS for solving the SIRPT under ML/OU policies. Since this proposed algorithm's computation time can vary with

different input of the instances size, which is the maximum total of time required on inputs of the large instances size for SIRPT_OU. A possible explanation is that this chapter proposed algorithm developed for the OU policy solves the SIRPT in large group instances with up to 200 customers, it can be solved in less than two and half hours, while ALNS requires much time than this chapter proposed algorithm. When the one proposed for the ML policy on all instances sets with up to 200 customers, this chapter proposed method requires in one hour.

As it has been discussed in the sub-section above, the implementation of combined the IG algorithm with local search and Sim-Randomised CWS heuristic shows good performance for solving all sets of instances sizes in which it has the ability to find best new solution. In addition, this chapter proposed and implemented a fast and efficient meta-heuristic which is able to achieve the problem objective.

Size of instances	**ML inventory replenishment policy**				
	ALNS		**SIRPT-based Sim-Randomised CWS and IG with local search**		**Improvememt (%)**
	Cost	**Time (s)**	**Cost**	**Time (s)**	
Small (5≤n≤25)	7926.71	44.6	**6965.29**	29.6	12.13
Medium (50≤n≤100)	26,527.05	444.1	**22,975.59**	403.1	13.39
Large (125≤n≤200)	54,292.38	4100.1	**51,268.87**	3819.5	5.57
Average	29,582.05	**1529.6**	**27,069.92**	**1417.4**	**10.36**
Size of instances	**OU inventory replenishment policy**				
	ALNS		**SIRPT-based Sim-Randomised CWS and IG with local search**		**Improvememt (%)**
	Cost	**Time (s)**	**Cost**	**Time (s)**	
Small (5≤n≤25)	8355.69	67.4	**7537.50**	48.9	9.79
Medium (50≤n≤100)	26,891.26	880.5	**25,323.87**	749.4	5.83
Large (125≤n≤200)	55,530.30	9196.0	**52,129.26**	8290.4	6.12
Average	30,259.08	**3381.3**	**28,330.21**	**3029.6**	**7.25**

Table 7.6 : Comparison of average results between ALNS and SIRP based Sim-Randomised CWS heuristic and IG algorithm with local search under ML and OU policies

7.4.2.3. Comparison of the results of the SIRPT and the SIRP solved by Sim-Randomised CWS and IG with local search

Table 7.7 presents the values of total costs for comparing the performance of each approach, whereby better results were achieved by Sim-Randomised CWS and IG algorithm with local search. This table illustrates that the solution values obtained by using the SIRP and the SIRPT –based Sim-Randomised CWS heuristic and IG with local search under OU replenishment policy give better results than those generated under ML replenishment policy. Thus, under OU policy for the SIRP and the SIRPT give reductions total costs and lost demand. The percentage gap when the lateral

transhipments are allowed is given in Table 7.7 (last column). The lateral transhipment is able to reduce the total costs 23.28 % for SIRPT_ML and 9.41 % for SIRPT_OU, which meant that transhipment able to improve the potential of the organisation and reduce the overall logistics costs.

Size of instances	ML invnetory replenishment policy		
	SIRP-based Sim-Randomised CWS and IG with local search	SIRPT-based Sim-Randomised CWS and IG with local search	Improvement (%)
Small (5≤n≤25)	8858.29	6965.29	27.18
Medium (50≤n≤100	28,919.69	22,975.59	25.87
Large (125≤n≤200)	59,877.88	51,268.87	16.79
Average	**32,551.95**	**27,069.92**	**23.28**
Size of instances	**OU inventory replenishment policy**		
	SIRP-based Sim-Randomised CWS and IG with local search	SIRPT-based Sim-Randomised CWS and IG with local search	Improvement (%)
Small (5≤n≤25)	8827.98	7537.50	17.12
Medium (50≤n≤100	26,236.66	25,323.87	3.60
Large (125≤n≤200)	56,037.85	52,129.26	7.50
Average	**30,367.50**	**28,330.21**	**9.41**

Table 7.7 : Comparison of average results between SIRP and IRPT-based Sim-Randomised CWS heuristic and IG algorithm with local search under ML and OU policies

Hence, the implementation of Sim-Randomised CWS when applied with IG and local search algorithms provide quite a competitive solution for most instances, and is able to solve each periodic problem. Additionally, this chapter proposed algorithm has proved to be very efficient and flexible in the sense that it able to be solve the problem under two inventory replenishment policies. Also, when considering the use of the lateral transhipment, Sim-Randomised CWS IG with local search was also able to solve the problem and provide good solutions to further reduce stock-outs as well as the total cost. Nevertheless, it requires the use of an optimisation algorithm that sometimes can take very long to run if high quality solutions are expected.

7.5. Chapter Conclusions

The application of the combination of Sim-heuristic with an iterated greedy algorithm is described and proposed. This chapter presents a challenge for research in this area because this is the first time the combination of random behaviour (local search based on insertion neighbourhood) and IG algorithm has been applied into Sim-Randomised CWS heuristic to solve the problem in which the two components of supply chain management problem have been integrated - inventory management and vehicle routing. The IG algorithm with local search is previously shown to reach excellent performance on the VRP and PFSP. As discussed in the literature section, the combination of IG algorithm and local search method have performed well and are important for improving better solution costs. It can be seen that, adapting local search based on insertion neighbourhood with IG's procedures, which the best

solution can be found so that the best solution is changed when it gives new better solution costs.

This is the first time that Sim-Randomised CWS IG heuristic with local search, which has been successfully applied to solve the SIRP and the SIRPT under ML/OU replenishment inventory policies without and with transhipment. For the lateral transhipment is able to reduce lost demand and decrease total cost. The replenishment inventory policies have been implemented with a rolling time-horizon, and it has shown that increasing the length of the rolling horizon does not have a positive impact on the overall solution quality. The effectiveness of the SIRP-based Sim-Randomised CWS IG heuristic with local search and SIRPT-based Sim-Randomised CWS IG heuristic with local search are tested though a set of computational experiments. For small and medium instances sizes with up to 100 customers require less than 8 minutes for solving. While the largest instances size with up to 200 customers can be solved in two and half hours.

Experiments have shown that the proposed method is able to find best new quality solution when compared to ALNS algorithm approached in Coelho et al. (2012). Moreover, the computational time is lower than Coelho et al. (2012). Therefore, this approach can be adapted quickly to solve other complex combinatorial optimisation problems such as those arising in manufacturing and logistics applications. This study has helped unify the body of knowledge on the IRP and will stimulate future research to pursue the study of this attractive field.

Chapter 8: Conclusions and future work

This chapter gives a discussion regarding the research conclusions and its opportunity to apply the contributions in the real-world industries in the future. The research conclusions encompass the key summary of the research problems, research aims and objectives, research methodologies, then express the specific contributions. Finally, the future research is addressed by applying the contributions in the logistics field.

8.1. Conclusion

In the real-world supply chain, many issues of delivery uncertainty inherently occur during the products movement starting from suppliers to customers, e.g. when to supply products to customers, how much to deliver, and how to combine customers into routes in order to optimisation the delivery costs. This thesis dealt with several approaches for the IRP and IRPT. The IRP and IRPT can be classified as deterministic demand and stochastic demand such as the SIRP and SIRPT. There are a number of operational researches found in the literature, focused on both types of the IRPs and the IRPTs and they have proposed methods to solve these problems, including exact algorithms, heuristics and sim-heuristics.

To tackle the problems stated in the supply chain, this thesis proposes two optimisation models encompassing order-up-to (OU) and maximum level (ML) inventory replenishment policies. The four algorithms are proposed in this thesis: classical Clark and Wright saving (CWS) heuristic, biased randomisation version Clark and Wright saving heuristics (Randomised CWS), and Sim-Randomised CWS, and Iterated Greedy (IG) algorithm with local search (based on insertion neighbourhood). Although, many studies have addressed IRP with deterministic and stochastic demands, not many researches are focussed on IRPT. Consequently, the wide-ranging literature review was carried out, addressing the describing the evolution of main contributions of previous works. Through an extensive number of publications on the IRP, this optimisation model is definitely an area of focus and continuous research. This is reasonable that this thesis proposed an implementation of the improved method for solving optimisation model of the IRP and IRPT for both deterministic and stochastic demands in order to minimise the total costs.

The lateral transhipment policy is an option considered for avoiding stock-out and penalty charge and it also applied when the extra demand is encountered. The results show that employing the policy can lead to cost reduction which, in turn, contributes

to decreasing in total transport costs. Utilising transhipment policy into the IRP can be an option for solving IRP with better solutions compared to the problem without any policy. Moreover, to eliminate a stock-out situation, the transhipment policy is adopted in the case of the existing of excessive demand. Applying transhipment in the problem shows a decrease in the overall costs; nevertheless, there are usually extra costs hidden in the real-world logistics that this research does not take into consideration. While, the OU and ML inventory replenishment policies are applied to determine the quantity for deliveries product to each customer, which in turn can reduce the inventory holding cost from the supplier side. Regarding to the experiments, applying the ML replenishment policy shows lower inventory holding cost during period than those when using the OU replenishment policy. Also, the ML replenishment policy may reduce routing and inventory holding cost because it is less likely to deliver more products to the customer when its inventory capacity is fully served. Whereas, adopting OU replenishment policy may lead to multi-trip deliveries and penalty cost due to customer stock out.

The optimisation models for the deterministic and stochastic demand IRP and IRPT under ML or OU inventory replenishment policies are developed. The improved algorithms for solving these optimisation models are proposed as following:

- Deterministic demand: the IRP and IRPT are solved with CWS, Randomised CWS, and Randomised CWS and IG with local search. Firstly, the CWS heuristic was proposed to solve the IRP and IRPT under ML/OU policies. Secondly, the biased randomised is implemented for improving the CWS heuristic. These heuristics are able to find good solution but not as good as the ones in the literature. However, these heuristics showed the shorter computational time than the algorithms in the literature. The Randomised CWS heuristic outperformed CWS heuristic, which can improve the solution. Finally, The IG algorithm is an improve heuristic and simply easy apply to other methods. Thus, the IG algorithm with local search is chosen for improving the solution. The integration of IG algorithm with local search and Randomised CWS heuristic used for solving these problems. The IG algorithm with local search outperformed both Randomised CWS heuristic and the best solutions in the literature. It obtained a remarkable improvement on the solution which showed the lowest total costs. To the best of our knowledge, theses heuristics are

adapted to solve the deterministic demand IRP and IRPT under ML/OU replenishment policies.

- Stochastic demand: the SIRP_ML, the SIRP_OU, the SIRPT_ML and the SIRPT_OU are addressed with the combination of Monte-Carlo simulation and biased randomised version CWS heuristics then called Sim-Randomised CWS heuristic. Once routing solutions are created under special assumptions of the random behaviour of the demand and the quantity is based on the inventory replenishment policies. The demand for each customer in each time period is generated by using simulation techniques. This algorithm can find solution, but it does not provide good solution as in the literature and is effective for small size instances. However, the average running time showed less time than that in the literature. For the large size instances and improving the solutions of the SIRP and the SIRPT, the IG algorithm with local search and Sim-Randomised CWS heuristic are interested. The IG algorithm and local search outperformed the solution of both Sim-Randomised CWS heuristics and the best solutions in the literature. This proposed algorithm shows a shorter computer time than that in the literature. To the best of our knowledge, Sim-Randomised CWS and IG algorithm with local search are implemented and successfully solved these problems.

Moreover, the randomisation of the CWS heuristic process has proven to be effective especially if combined with other solution improvement approaches or simulation techniques for stochastic problems. Also, the randomised heuristics can easily integrate with other methods which can be applied to other similar optimisation problems. Additionally, the IG with local search is proved that it can produce the best results that, in turn, can integrate to other algorithms in order to more improve the problems. It has been proved that applying IG and local search to solving both deterministic and stochastic for the IRP and IRPT. This integrated algorithm shows the better solutions than that used UB and ALNS, which reflects the efficiency and effectiveness of the combination between the IG and local search, and meta-heuristics. This can be interesting and more beneficial for Logistics Companies for adapting to real-life problems.

8.2. Limitation and Future Work

8.2.1. **Limitation of this thesis**

Although this research has completed its aims, there are some unavoidable limitations. This proposed the improved methods are developed for the deterministic and stochastic demand IRP and IRPT under ML/OU replenishment policies.

- This proposed optimisation models and algorithms are use as design classification for a single-depot and a vehicle of these problems.
- The data used is only from the benchmark problem. Therefore, the difficultly in encouragement the companies to provide the data for testing and comparing.
- The ML and OU replenishment polices are applied to determine the quantity for deliveries to each customer and help to reduce the total costs. Both inventory replenishment policies are compared.

8.2.2. **Future work**

This thesis could be extended in include:

- Adapting these algorithms proposed to rea-life problem by investigating with the logistics companies.

- Applying the Randomised CWS and IG with local search or Sim-Randomised CWS and IG with local search to more intricate model in order to gain the best problem solutions. It is suggested that a future research can adopt IG with either other local search algorithms or different meta-heuristics for more intricate models in order to gain the best problem solutions.

- Consider the new IRP application such as a single-period, multi-depots, multi-products, multi-vehicle and different vehicle fleet. All these together with the optimisation methods proposed in this research, will offer great insights in order to reduce more operational costs, including logistics costs. Future research could also consider new IRP application such as single-period, multi-depots, multi-product and different vehicle fleet. Proposed approaches can be tested on IRPs with more realistic model that can be applied to real case studies.

- Establishing good relationships between suppliers and retailers involve sharing information and resources of the partners. Thus, further research is needed to build trust and collaborative relationships. Trust and collaboration

is also very important in retailer-supplier relationships. This can be seen from the Nash equilibrium game theory of a prisoner's dilemma, which is a puzzle in decision analysis in which two players act in their own self-interest that does not result in the area outcome. On the one hand, collaboration can lead to the best situation, a win-win outcome for retailer and supplier. And on the other hand, it can also increase customer satisfaction and trust. The collaborative relationship between retailers and suppliers is gaining more importance as it is seen as one of the business elements that can contribute to a successful relationship among supply chain partners.

- Exploiting lateral transhipment can be taken into account in the IRP as a key concept to solve more complex problems. While utilising transhipment policy into the IRP can be an option for solving IRP with better solutions compared to the problem without any policy. The results show that employing the policy can lead to cost reduction which, in turn, contributes to decreasing in total transport costs.

- Strategies for selecting (transhipment) a retailer to serve another customer is important. For example, for some products, the supplier has to consider the product's cost price and the performance of the partnership before deciding to select the retailer for transhipment.

References

Aarts, E. H. L., & Lenstra, J. K. (1997). *Local search in combinatorial optimization*. Princeton University Press.

Abdelmaguid, T. F., Dessouky, M. M., & Ordonez, F. (2009). Heuristic approaches for the inventory-routing problem with backlogging. *Computers and Industrial Engineering*, *56*(4), 1519–1534.

Achabal, D. D., McIntyre, S. H., Smith, S. A., & Kalyanam, K. (2000). A Decision Support System for Vendor Managed Inventory. *Journal of Retailing*, *76*(4), 430–454.

Adelman, D. (2004). A Price-Directed Approach to Stochastic Inventory/Routing. *Operations Research*, *52*(4), 499–514.

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2014). The production routing problem: A review of formulations and solution algorithms. *Computers and Operations Research*, *55*, 141–152.

Aghezzaf, E. (2008). Robust distribution planning for supplier-managed inventory agreements when demand rates and travel times are stationary, *59*(8), 1055–1065.

Ahmad, H., Hamzah, P., Yasin, Z. A. M. M., & Shariff, S. S. R. (2014). Location Routing Inventory Problem with Transshipment (LRIP-T). *Proceedings of the 2014 International Conference on Industrial Engineering and Operations Management*, 1595–1605.

Al-ameri, T. A. (2008). Optimization of vendor-managed inventory systems in a rolling horizon framework, *54*, 1019–1047.

Alizadeh, Mohammadmahdi, H. E. A. S. M. S. (2011). Analyzing A Stochastic Inventory System For Deteriorating Items With Stochastic Lead time Using Simulation Modeling (pp. 1645–1657).

Allen, S. C. (1958). Redistribution of total stock over several user locations. *Naval Research Logistics Quarterly*, *5*(4), 337–345.

Angelidis, E., Bohn, D., & Rose, O. (2012). A Simulation-Based Optimisation Heursitic Using Self-Organisation For Complex Assembly Lines. In *Proceedings of the 2012 Winter Simulation Conference* (pp. 1219–1229).

Archetti, C., Bertazzi, L., Hertz, A., & Speranza, M. G. (2012). A Hybrid Heuristic for an Inventory Routing Problem. *Inform Jornour Computer*, *24*(1), 101–116.

Archetti, C., Bertazzi, L., Paletta, G., & Speranza, M. G. (2011). Analysis of the Maximum Level Policy in a Production – Distribution System, (1), 1–2.

Archetti, C., Bianchessi, N., & Speranza, M. G. (2014). Branch-and-cut algorithms for the split delivery vehicle routing problem. *European Journal of Operational Research*, *238*(3), 685–698.

Archetti, C., & Laporte, G. (2007). A Branch-and-Cut Algorithm for a Vendor Managed Inventory Routing Problem. *Transportation Science*, *41*(3), 382–391.

Arisha, A. (2010). Simulation Optimisation Methods in Supply Chain Applications : a Review Simulation – Optimisation Methods in Supply.

Baita, F., Ukovich, W., & Pesenti, R. (1998). Dynamic Routing-And-Invenotry Problems : A REVIEW, *32*(8).

Barceló, J., Grzybowska, H., & Pardo, S. (2007). Vehicle Routing And Scheduling Models, Simulation And City Logistics. In V. Zeimpekis, C. D. Tarantilis, G. M. Giaglis, & I. Minis (Eds.), *Dynamic Fleet Management: Concepts, Systems, Algorithms {&} Case Studies* (pp. 163–195). Boston, MA: Springer US.

Bard, J. F., Huang, L., Jaillet, P., & Dror, M. (1998). A Decomposition Approach to the Inventory Routing Problem with Satellite Facilities. *Transportation Science*, *32*(2), 189–203.

Barnes-Schuster, D., & Bassok, Y. (1997). Direct shipping and the dynamic single-depot/multi-retailer inventory system. *European Journal of Operational Research*, *101*(3), 509–518.

Barnes, J. G. (1994). Close to the customer: But is it really a relationship? *Journal of Marketing Management*, *10*(7), 561–570.

Bastian, M. (1986). Joint Replenishment in Multi-Item Inventory Systems. *Operations Research Society of America*, *37*(12), pp 1113–1120.

Battarra, A. subramanian and M. (2013). An iterated local search algorithm for the Travelling Salesman Problem with Pickups and Deliveries. *Operations Research Society*, *64*(3), 402–409.

Bell, Walter J.; Dalberto, Louis M.; Fisher, Marshall L.; Greenfield, Arnold J.; Jaikumar, R.; Kedia, Pradeep; Mack, Robert G.; Prutzman, P. J. (1983). Improving the Distribution of Industrial Gases with an On-line Computerized Routing and Scheduling Optimizer. *Academic Journal*, *13*(6), 4–23. Retrieved from INFORMS

Benoist, T., Jeanjean, A., & Estellon, B. (2002). Randomized local search for real-life inventory routing. *Transportation Science*, *0*(0), 1–43.

Benoist, T., Estellon, B., Gardi, F., & Jeanjean, A. (2009). High-performance local search for solving real-life inventory routing problems. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *5752 LNCS*, 105–109.

Bertazzi, L., Bosco, A., Guerriero, F., & Laganà, D. (2013). A stochastic inventory routing problem with stock-out. *Transportation Research Part C: Emerging Technologies*, *27*, 89–107.

Bertazzi, L., Paletta, G., & Speranza, M. G. (2002). Deterministic Order-Up-To Level Policies in an Inventory Routing Problem. *Transportation Science*, *36*(1), 119–132.

Bertazzi, L., & Speranza, M. G. (2013). Inventory routing problems with multiple customers. *EURO Journal on Transportation and Logistics*, *2*(3), 255–275.

Blanton Jr, J. L., & Wainwright, R. L. (1993). Multiple vehicle routing with time and capacity constraint using genetic algorithms. *Proceedings of the Fifth International Conference on Genetic Algorithms*.

Boesel, J., Glover, F., Bowden, R. O., & Kelly, J. P. (2001). Future Od Simulation

Optimization. In *Winter Simulation Conference* (pp. 1466–1469).

Boudia, M., & Prins, C. (2009). A memetic algorithm with dynamic population management for an integrated production-distribution problem. *European Journal of Operational Research*, *195*(3), 703–715.

Bresina, L. (1996). Heuristic-Biased Stochastic Sampling. In *the Thirteenth National Conference on Artificial Intelligence* (pp. 271–278).

Buffa, F. P., & Munn, J. R. (1989). A Recursive Algorithm for Order Cycle-Time that Minimizes Logistics Cost, *40*(4), 367–377.

Cabrera, V., & Ros, F. J. (2009). Simulation-based Study of Common Issues in VANET Routing Protocols, 1–5.

Caccetta, L., & Abdul-niby, M. (2013). An Improved Clarke and Wright Algorithm to Solve the Capacitated Vehicle Routing Problem. *Engineering Technology & Applied Science Research*, *3*, 413–415.

Caceres-Cruz, J., Grasas, A., Ramalhinho, H., Juan, A. a, & Roca, M. (2014). A Savings-based Randomized Heuristic for the Heterogeneous Fixed Fleet Vehicle Routing Problem with Multi-trips. *Journal of Applied Operational Resaerch*, *accepted f*, 69–81.

Cáceres-Cruz, J., Juan, A. a, Grasman, S. E., Bektas, T., & Faulin, J. (2012). Combining Monte Carlo Simulation With Heuristics for Solving the Inventory Routing Problem With Stochastic Demand. In *Proceedings of the 2012 Winter Simulation Conference* (pp. 3114–3122).

Cáceres Cruz, J. D. J. (2013). *Randomized Algorithms for Rich Vehicle Routing Problems : From a Specialized Approach to a Generic Methodology*.

Cai, G. G. (2010). Channel Selection and Coordination in Dual-Channel Supply Chains. *Journal of Retailing*, *86*(1), 22–36.

Campbell, A., Clarke, L., Kleywegt, A. J., & Savelsbergh, M. (1998). The Inventory Routing Problem. *Fleet Management and Logistics*, (Ddm), 95–113.

Campbell, A. M., & Savelsbergh, M. W. P. (2004). A Decomposition Approach for the Inventory-Routing Problem. *Transportation Science*, *38*(4), 488–502.

Carr, A. S., & Smeltzer, L. R. (1999). The relationship of strategic purchasing to supply chain management, *5*.

Carson, Y., & Maria, A. (1997). Simulation optimization methods and applications. *Proceedings of the 1997 Winter Simulation Conference*, (Figure 1), 118–126.

Cetinkaya, S., & Lee, C.-Y. (2000). Stock Replenishment and Shipment Scheduling for Vendor-Managed Inventory Systems. *Management Science*, *46*(2), 217–232.

Chakravarty, A. K. . (1985). An Optimal Heuristic for Coordinated Multi-Item Inventory Replenishments. *Operations Research Society of America*, *36*(11), 1027–1039.

Chandra, C. (1993). A dynamic distribution model with warehouse and customer replenishment requirements. *Journal of the Operational Research Society*, *44*(7), 681–692.

Chaouch, B. a. (2001). Stock levels and delivery rates in vendor-managed inventory programs. *Production and Operations Management*, *10*(1), 31–44.

Chien, T. W., Balakrishnan, A., & WONG, R. T. (1989). An Integrated Inventory Allocation and Vehicle Routing Problem. *Transportation Science*, *23*(2), 67–76.

Chrysochoou, E. C., & Ziliaskopoulos, P. A. K. (2012). Stochastic inventory routing problem with transshipment recourse action. In *Stochastic Models of Manufacturing and Service Operations* (pp. 1–2).

Clarke and Wright. (1964). Scheduling of Vehicles from a Central Depot to a Number of Delivery Points. *Operational Reserch*, *12*(4), 568–581.

Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2013). Thirty Years of Inventory Routing. *Transportation Science*, *48*(1), 1–19.

Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2014). Heuristics for dynamic and stochastic inventory-routing. *Computers & Operations Research*, *52*, 55–67.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2012a). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, *24*, 270–287.

Coelho, L. C., Cordeau, J. F., & Laporte, G. (2012b). The inventory-routing problem with transshipment. *Computers and Operations Research*, *39*(11), 2537–2548.

Coelho, L. C., & Laporte, G. (2013). The exact solution of several classes of inventory-routing problems. *Computers and Operations Research*, *40*(2), 558–565.

Coelho, L., Laporte, G., & Cordeau, J. (2012). *Dynamic and stochastic inventory-routing*.

Cordeau, J.-F., Gendreau, M., Laporte, G., Potvin, J.-Y., & Semet, F. (2002). A guide to vehicle routing heuristics. *Journal of the Operational Research Society*, *53*(5), 512–522.

Cordeau, J. F., Laganà, D., Musmanno, R., & Vocaturo, F. (2014). A decomposition-based heuristic for the multiple-product inventory-routing problem. *Computers and Operations Research*, *55*, 153–166.

Custódio, A. L., & Oliveira, R. C. (2006). Redesigning distribution operations: a case study on integrating inventory management and vehicle routes design. *International Journal of Logistics*, *9*(2), 169–187.

Darwish, M. A., & Odah, O. M. (2010). Vendor managed inventory model for single-vendor multi-retailer supply chains. *European Journal of Operational Research*, *204*(3), 473–484.

Deng, G. (2007). Simulation-based optimization.

Ding, H., & Schumacher, J. (2004). One A New Tool For Supply Cahin Network Optimisation and Simulation.

Disney, S. M., & Towill, D. R. (2003). The effect of vendor managed inventory (VMI) dynamics on the Bullwhip Effect in supply chains. *International Journal of Production Economics*, *85*(2), 199–215.

Dror, M., & Ball, M. (1987). Reduction from an annual to a short-period problem. *Naval Research Logistics (NRL)*, *34*, 891–905.

Dror, M., Ball, M., & Golden, B. (1985). A computational comparison of algorithms for the inventory routing problem. *Operational Reserch*, *4*(1), 1–23.

Dunham, B., Fridshal, R., & North, J. H. (1961). Division Of Mathematics: Symbolic Logic And Computiong Machines: Some General Reflections and Prejudices. *Transactions of the New York Academy of Sciences*, *24*(1 Series II), 58–63.

Engineer, F. G., Furman, K. C., Nemhauser, G. L., Savelsbergh, M. W. P., & Song, J.-H. (2012). A Branch-Price-and-Cut Algorithm for Single-Product Maritime Inventory Routing. *Operational Reserch*, *60*(1), 106–122.

Fanjul-peyro, L., & Ruiz, R. (2010). Size-reduction heuristics for the unrelated parallel machines scheduling problem, 1–21.

Faulin, J., & Juan, A. a. (2008). A Simulation_Based Algorithm For the Capacitated Vehicle Routing Problem (pp. 2708–2716).

Federgruen, A., & Zipkin, P. (1983). A Combined Vehicle Routing and Inventory Allocation Problem. *Operations Research*, *32*(5), 1019–1037.

Festa, P., & Resende, M. G. C. (2009). GRASP – Part I : Algorithms. *International Transactions in Operational Research*, *16*, 1–24.

García Villalba, L. J., Sandoval Orozco, A. L., Triviño Cabrera, A., & Barenco Abbas, C. J. (2009). Routing Protocols in Wireless Sensor Networks. *Sensors*, *9*(11), 8399.

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

Gaspero, L. Di, & Schaerf, A. (2003). An object-oriented framework for the flexible design of local-search algorithms. *Software: Practice and Experience*, *33*(8), 733–765.

Gaur, V., & Fisher, M. L. (2004). A Periodic Inventory Routing Problem at a Supermarket Chain. *Operations Research*, *52*(6), 813–822.

Gelderman, C. J., & Weele, A. J. Van. (2002). Handling measurement issues and strategic directions in Kraljic ' s purchasing portfolio model, *9*, 207–216.

Glover, F., & Kelly, J. P. (1996). New Advance And Applicationa Of Combining Simulation And Optimisation. In *Einter Simulation Conference* (pp. 144–152).

Glover, F., & Kelly, J. P. (1999). New Advances For Wedding Optimization And Simulation.

Glover, F., Laguna, M., & Marti, R. (2003). Scatter Search and Path Relinking: Advances and Applications BT - Handbook of Metaheuristics. In F. Glover & G. A. Kochenberger (Eds.), (pp. 1–35). Boston, MA: Springer US.

Gonzalez-martin, S., & Juan, A. A. (2014). On the use of Biased Randomisation and Simheuristics to solve vehicle and arc routing problem. In *Proceedings of the 2012 Winter Simulation Conference* (pp. 1875–1884).

Guerrero, W. J., Prodhon, C., Velasco, N., & Amaya, C. A. (2013). Hybrid heuristic for the inventory location-routing problem with deterministic demand. *International Journal of Production Economics*, *146*(1), 359–370.

Guimarans, D., Herrero, R., Riera, D., Juan, A. a., & Ramos, J. J. (2011). Combining probabilistic algorithms, Constraint Programming and Lagrangian Relaxation to solve the Vehicle Routing Problem. *Annals of Mathematics and Artificial Intelligence*, *62*(3–4), 299–315.

Han, W., Yu, Y., Altan, N., & Pick, L. (1993). Multiple proteins interact with the fushi tarazu proximal enhancer. *Molecular and Cellular Biology*, *13*(9), 5549–5559.

Hassanvand, H., & Salehsohrabi, M. (2013). A genetic algorithm method for the inventory routing and optimal pricing in a two-echelon supply chain with demand function. *World Applied Sciences Journal*, *23*(9), 1269–1273.

Hemmelmayr, V., Doerner, K. F., Hartl, R. F., & Savelsbergh, M. W. P. (2009). Delivery strategies for blood products supplies. *OR Spectrum*, *31*(4), 707–725.

Henrik Andersson, ArildHoff, MarielleChristiansen, GeirHasle, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers and Operations Research*, *37*(9), 1515–1536.

Huang, S.-H., & Lin, P.-C. (2010). A modified ant colony optimization algorithm for multi-item inventory routing problems with demand uncertainty. *Transportation Research Part E: Logistics and Transportation Review*, *46*(5), 598–611.

Hvattum, L. M., Lokketangen, A., & Laporte, G. (2009). Scenario Tree-Based Heuristics for Stochastic Inventory-Routing Problems. *Informs Journal on Computing*, *21*(2), 268–285.

Jaillet, P., Bard, J. F., Huang, L., & Dror, M. (2001). Delivery Cost Approximations for Inventory Routing Problems in a Rolling Horizon Framework. *Transportation Science*, *36*(January 2015), 292–300.

Jaillet p., Jonathan F. Bard, L. H. & M. D. (2002). Delivery Cost Approximations for Inventory Routing Problems in a Rolling Horizon Framework. *Transportation Science*, *36*(3), 292–300.

Jarugumilli, S., & Grasman, S. E. (2006). RFID-enabled inventory routing problems. *International Journal of Manufacturing Technology and Management*, *10*(1), 92–105.

Jemai, Z., Rekik, Y., & Kalaï, R. (2012). Inventory routing problems in a context of vendor-managed inventory system with consignment stock and transshipment. *Production Planning & Control*, *7287*(December 2013), 1–13.

Juan, A. A., Cáceres-Cruz, J., González-Martín, S., Riera, D., & Barrios, B. B. (2014). Biased Randomization of Classical Heuristics. In *Encyclopedia of Business Analytics and Optimization* (pp. 304–314). IGI Global.

Juan, A. A., Faulin, J., Ferrer, A., Lourenço, H. R., & Barrios, B. (2013). MIRHA: multi-start biased randomization of heuristics with adaptive local search for solving non-smooth routing problems. *TOP*, *21*(1), 109–132.

Juan, A. A., Faulin, J., Grasman, S. E., Rabe, M., & Figueira, G. (2015). A review of simheuristics: Extending metaheuristics to deal with stochastic combinatorial optimization problems. *Operations Research Perspectives*, *2*, 62–72.

Juan, A. A., Faulin, J., Ruiz, R., Barrios, B., & Caballé, S. (2010). The SR-GCWS hybrid algorithm for solving the capacitated vehicle routing problem. *Applied Soft*

Computing, *10*(1), 215–224.

Juan, A. A., Grasman, S. E., Caceres-cruz, J., & Bektas, T. (2014). A simheuristic algorithm for the Single-Period Stochastic Inventory-Routing Problem with stock-outs. *Simulation Modelling Practice and Theory*, *46*, 40–52.

Juan, Faulin, J., Jorba, J., Riera, D., Masip, D., & Barrios, B. (2011). On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright savings heuristics. *Journal of the Operational Research Society*, *62*(6), 1085–1097.

Karp, R. M. (1991). An introduction algorithms to randomized. *Discrete Applied Mathematics*, *34*, 165–201.

Kenneth J . Arrow , Harris, Jacob MarschakKenneth J . Arrow , Theodore Harris, J. M. (2010). Optimal Inventory Policy Author (s): Kenneth J . Arrow , Theodore Harris , Jacob Marschak Published by : The Econometric Society Stable Optimal Invemtory Policy. *Society*, *19*(3), 250–272.

Kheljani, J. G., Ghodsypour, S. H., & Brien, C. O. (2009). Optimizing whole supply chain benefit versus buyer ' s benefit through supplier selection. *Intern. Journal of Production Economics*, *121*(2), 482–493.

Kleywegt, A. J., Nori, V. S., & Savelsbergh, M. W. P. (2002). The Stochastic Inventory Routing Problem with Direct Deliveries. *Transportation Science*, *36*(1), 94–118.

Kleywegt, A. J., Nori, V. S., & Savelsbergh, M. W. P. (2004). Dynamic Programming Approximations for a Stochastic Inventory Routing Problem. *Transportation Science*, *38*(1), 42–70.

L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.Wang, W. (2006). A Scor Based Supply Chain Transportation Platfrom through Simulatiom and Optimisation Techniques. In *Winter Simulation Conference* (pp. 650–659).

Laporte, G. (2008). Waiting and Buffering Strategies for the Dynamic Pickup and Delivery Problem with Time Windows. *Operational Reserch*, *46(3)*(August), 165–176.

Laporte, G., Coelho, L. C., & Laporte, G. (2013). A Branch-and-Cut Algorithm for the Routing Problem A Branch-and-Cut Algorithm for the Multi-Product Multi-Vehicle Inventory-Routing Problem. *Production Research*, *51*(23–24), 7156–7169.

Larry C. Giunipero, R. R. B. (1996). Purchasing's Role in Supply Chain Management. *International Journal of Logistics*, *7*(1), 29–38.

Lau, H. C., Liu, Q., & Ono, H. (2002). Integrating Local Search and Network Flow to Solve the Inventory Routing Problem. *American Association for Artificial Intelligence*, 9–14.

Law, A. M. (2008). How to Build Valid And Credible Simulation Models. In *Winter Simulation Conference* (pp. 39–47).

Lenstra, E. A. and J. K. (1997). *Local Search in Combinatorial Optimisation* (1 st ed.). New York: Jonh Wiley & ammp; Sons, Inc.

Li, K., Chen, B., Sivakumar, A. I., & Wu, Y. (2014). An inventory–routing problem with the objective of travel time minimization. *European Journal of Operational Research*, *236*(3), 936–945.

Lourenco, R. R. and H. R. (n.d.). Inventory-routing model, for a multi-period problem with stochastic and deterministic demand.

Luca Bertazzi and M. Grazia Speranza. (2012). Inventory routing problems: an introduction. *EURO Journal on Transportation and Logistics*, *1*(4), 307–326.

Lysgaard, J. (1997). *Clarke & Wright ' s Savings Algorithm*.

Marquès, G., Thierry, C., Lamothe, J., Gourc, D., Marquès, G., Thierry, C., … Marque, G. (2017). A review of Vendor Managed Inventory (VMI): from concept to processes, *7287*(June).

Mercer et al. (1996). Alternative Inventory and Distribution Policies of a Food Manufacturer. *Journal of the Operational Research Society*, *47*(6), 755–765.

Mirzaei, S., & Seifi, A. (2015). Considering lost sale in inventory routing problems for perishable goods. *Computers and Industrial Engineering*, *87*.

Mirzapour Al-e-Hashem, S. M. J., & Rekik, Y. (2013). Multi-product multi-period Inventory Routing Problem with a transshipment option: A green approach. *International Journal of Production Economics*, *157*(1), 80–88.

Mjirda, A., Jarboui, B., Macedo, R., & Hanafi, S. (2012). A variable neighborhood search for the multi-product inventory routing problem. *Electronic Notes in Discrete Mathematics*, *39*, 91–98.

Moin, N. H., Salhi, S., & Aziz, N. A. B. (2011). An efficient hybrid genetic algorithm for the multi-product multi-period inventory routing problem. *International Journal of Production Economics*, *133*(1), 334–343.

Montoya-torres, J. R., & Herazo-padilla, N. (2014). Mathematical Programming Modeling and Resolution of the Location-Routing Problem in Urban Logistics 1 Modelación en programación matemática y resolución del, *18*(2), 271–289.

Montoya-torres, J. R., López, J., Nieto, S., Felizzola, H., & Herazo-padilla, N. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering*, *79*, 115–129.

Olorunniwo, F., Hartfield, T., & Olorunniwo, F. (2001). Strategic partnering when the supply base is limited : a case study.

Olsen, R. F., & Ellram, L. M. (1997). A portfolio approach to supplier relationships. *Industrial Marketing Management*, *26*(2), 101–113.

Osman, I., & Potts, C. (1989). Simulated annealing for permutation flow-shop scheduling. *Omega*, *17*(6), 551–557.

Ouelhadj, D., Petrovic, S., & Scheduling, A. (2009). Survey of Dynamic Scheduling, *44*(0), 1–27.

Papageorgiou, D. J., Nemhauser, G. L., Sokol, J., Cheon, M. S., & Keha, A. B. (2014). A library of maritime inventory routing problem instances: Survey, core model, and benchmark results. *European Journal of Operational Research*, *235*(2).

Peter Kraljic. (1983). *Purchasing must become supply management*.

Phillips, P. C. B., Ploberger, W., Phillips, B. Y. P. C. B., & Ploberger, W. (2016). The variability of aggregate demand with (S,s) inventory policies, *64*(2), 381–412.

Pichpibul, T., & Kawtummachai, R. (2012). An improved Clarke and Wright savings algorithm for the capacitated vehicle routing problem. *ScienceAsia*, *38*(3), 307–318.

Pirlot, M. (1996). General local search methods. *European Journal of Operational Research*, *92*(3), 493–511.

Poorbagheri, T., Taghi, S., Niaki, A., & Branch, Q. (2014). Vendor Managed Inventory of a Single-vendor Multiple-retailer Single-warehouse Supply Chain under Stochastic Demands, *1*(3), 297–313.

Popović, D., Bjelić, N., & Radivojević, G. (2011). Simulation approach to analyse deterministic IRP solution of the stochastic fuel delivery problem. *Procedia - Social and Behavioral Sciences*, *20*, 273–282.

Popović, D., Vidović, M., & Radivojević, G. (2012). Variable Neighborhood Search heuristic for the Inventory Routing Problem in fuel delivery. *Expert Systems with Applications*, *39*(18), 13390–13398.

Qin, L., Miao, L., Ruan, Q., & Zhang, Y. (2014). A local search method for periodic inventory routing problem. *Expert Systems with Applications*, *41*(2), 765–778.

Qu, W. W., Bookbinder, J. H., & Iyogun, P. (1999). An integrated inventory–transportation system with modified periodic policy for multiple products. *European Journal of Operational Research*, *115*(2), 254–269.

Rand, G. (2009). The life and times of the savings method for vehicle routing problems. *The Operations Research Sociiety of South Africa*, *25*(2), 125–145.

Razavi, M. K., & Nik, E. R. (2013). Meta Heuristic for Multi Depot Inventory Routing Problem Backlogging. *Scientific Research*, *3*, 273–280.

Reeves, C. R. (1996). Modern heuristic techniques. *Modern Heuristic Search Methods*, 1–25.

Renaud, J., Laporte, G., & Boctor, F. F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research*, *23*(3), 229–235.

Ribeiro, R., & Ramalhinho-Lourenço, H. (2003, June 3). Inventory-routing model, for a multi-period problem with stochastic and deterministic demand. Retrieved from

Ritter, T., & Gemünden, H. G. (2001). Value Creation in Buyer – Seller Relationships, *377*, 365–377.

Roldán, R. F., Basagoiti, R., & Coelho, L. C. (2016). A survey on the inventory-routing problem with stochastic lead times and demands. *Journal of Applied Logic*, *1*, 1–10.

Ruiz, R., & Stützle, T. (2005). An iterated greedy algorithm for the flowshop problem with sequence dependent setup times. *Proceedings of the 6th Metaheuristics ...*, (JANUARY), 817–823.

Ruiz, R., & Stützle, T. (2007). A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, *177*(3), 2033–2049.

Sajadifar, S. M. (2012). A Simulation Approach for an (R,Q) Inventory Model with A Deteriorating Item, Poisson Demand and Stochastic lead time. In *Winter Simulation Conference*.

Santos, E., Ochi, L. S., Simonetti, L., & González, P. H. (2016). A Hybrid Heuristic based on Iterated Local Search for Multivehicle Inventory Routing Problem. *Electronic Notes in Discrete Mathematics*, *52*, 197–204.

Sarkar, N. I., Member, S., & Halim, S. A. (2011). A Review of Simulation of Telecommunication Networks : Simulators , Classification , Comparison , Methodologies , and Recommendations.

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

Shen, Q., Chu, F., & Chen, H. (2011). A Lagrangian relaxation approach for a multi-mode inventory routing problem with transshipment in crude oil transportation. *Computers and Chemical Engineering*, *35*(10), 2113–2123.

Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory Management and Production Planning and Scheduling*.

Singh, T., Arbogast, J. E., & Neagu, N. (2015). An incremental approach using local-search heuristic for inventory routing problem in industrial gases. *Computers and Chemical Engineering*, *80*.

Sofianopoulou, S. (2014). A heuristic approach for an inventory routing problem with backorder decisions. In *Operational Reserch* (Vol. 6, pp. 49–57).

Solyali, H. (2011). A Branch-and-Cut Algorithm Using a Strong Formulation and an A Priori Tour-Based Heuristic for an Inventory-Routing Problem. *Transportation Science*, *45*(3), 335–345.

Straka, M., & Besta, P. (2015). Clarke and wright saving algorithm as a means of distribution streamlining in the environment of a concrete company, 5–8.

Sutcliffe, K. M., & Zaheer, A. (1998). Uncertainty in the Transaction Environment: An Empirical Test. *Strategic Management Journal*, *19*(1), 1–23.

Talbi, E. G. (2009). *Metaheuristics: From Design to Implementation*. *Metaheuristics: From Design to Implementation*.

Tan, V. R. K. and K. C. (2006). Buyer-supplier relationships The impact of supplier selection and. *International Journal of Physical Distribution & Logistics Management*, *20*(258).

Taniguchi, E., Thompson, R. G., & Yamada, T. (2012). Emerging Techniques for Enhancing the Practical Application of City Logistics Models. *Procedia - Social and Behavioral Sciences*, *39*, 3–18.

Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, E. (2014). Simulation Base Optimization For Multi-Echelon Inventory Systems Under

Uncertainty. In *Winter Simulation Conference* (pp. 385–394).

Tolujevs, J. (2012). Simulation-Based Metaheuristic Optimisation of Logistics Systems (pp. 221–226).

Toptal, A. (2009). Replenishment decisions under an all-units discount schedule and stepwise freight costs. *European Journal of Operational Research*, *198*(2), 504–510.

Tu, J. H., Griffin, J., Hart, A., Rowley, C. W., Cattafesta, L. N., & Ukeiley, L. S. (2013). Integration of non-time-resolved PIV and time-resolved velocity point sensors for dynamic estimation of velocity fields. *Experiments in Fluids*, *54*(2), 1429.

Vansteenwegen, P., & Mateo, M. (2014). An iterated local search algorithm for the single-vehicle cyclic inventory routing problem. *European Journal of Operational Research*, *237*(3), 802–813.

Vansteenwegen, P., Souffriau, W., Vanden, G., & Oudheusden, D. Van. (2009). Computers & Operations Research Iterated local search for the team orienteering problem with time windows. *Computers &OperationsResearch36*, *36*.

Waller, M., Johnson, M. E., & Davis, T. (1999). Vendor-managed inventory int he retail supply chain. *Journal of Business Logistics*, *20*(1), 183–203.

Wensing, T. (2011). *Periodic Review Inventory Systems*. *Journal of Policy Analysis and Management* (Vol. 5).

Xiangpei Hu, Yongxian Li,Jianwen Guo, L. S. and A. Z. Z. (2008). A Simulation Optimization Algorithm With Heuristic Transportation and Its Application To Vehicle Routing Problem. *International Journal of Innovative Computing*, *4*(5), 4198.

Yang, F., & Xiao, H. (2007). Models and Algorithms for Vehicle Routing Problem with Transshipment Centers. *Systems Engineering - Theory & Practice*, *27*(3), 28–35.

Yao, Y., Evers, P. T., & Dresner, M. E. (2007). Supply chain integration in vendor-managed inventory. *Decision Support Systems*, *43*(2), 663–674.

Yellow, P. C. (1970). A Computational Modification to the Savings Method of Vehicle Scheduling. *Operational Research Quarterly (1970-1977)*, *21*(2), 281–283.

Zachariadis, E. E., Tarantilis, C. D., & Kiranoudis, C. T. (2009). An integrated local search method for inventory and routing decisions. *Expert Systems with Applications*, *36*(7), 10239–10248.

Zavanella, L., & Zanoni, S. (2009). A one-vendor multi-buyer integrated production-inventory model: The "Consignment Stock" case. *International Journal of Production Economics*, *118*(1), 225–232.

Zhang, Q., Vonderembse, M. A., & Lim, J. (2003). Manufacturing flexibility : defining and analyzing relationships among competence , capability , and customer satisfaction, *21*, 173–191.

Zhao, Q. H., Chen, S., & Zang, C. X. (2008). Model and algorithm for inventory/routing decision in a three-echelon logistics system. *European Journal of Operational Research*, *191*(3), 623–635.

Zhong, Y., & Aghezzaf, E. H. (2011). Combining DC-programming and steepest-descent to solve the single-vehicle inventory routing problem. *Computers and*

Industrial Engineering, *61*(2), 313–321.

Appendix

Java language code is used in this thesis with eclipse software and all the tests are run on a computer with a Core i5, 2.30 GHz processor and 4 GB of RAM.

Iterated Greedy with local search

```
package algorithm;
import java.util.LinkedList;
import algorithm.object.EdgeBean;
import algorithm.object.NodeBean;
import algorithm.object.RouteBean;
import algorithm.object.SIMUniformDistribution;
import sim.object.SolutionBean;
/**
 * @author Chanicha moryadee
*/
public class GreedyIteration {
	public static SolutionBean solve(SolutionBean sol) {
		System.out.println("Begin Greedy Iteration...");
		System.out.println(sol.printRouteList());
		/* Get the original route from CWS */
		LinkedList<RouteBean> routeList = new LinkedList<RouteBean>();
		for (RouteBean r : sol.getRoutes()) {
			routeList.add(r);
		}
		LinkedList<RouteBean> newRouteList = new
LinkedList<RouteBean>();
		/* Run through all existing routes */
		for (RouteBean r : routeList) {
			/* Get the list of nodes in the route */
			LinkedList<NodeBean> nodeList = r.getNodeList();
			/* Save and remove depot from the list */
			NodeBean depot = nodeList.get(0);
```

```
                    nodeList.remove(depot);
                    int originalSize = nodeList.size();
                    /* Show the list */
                    System.out.println(r.printNodeList());
                    /* Get the original cost of CWS route */
                    double originalCosts = r.getCosts();
                    System.out.println("Original route costs: " + originalCosts);
                    if (originalSize < 3) {
                            System.out.println("Simple route cannot be modified!!");
                            newRouteList.add(r);
                    } else {
                            /* Randomly divide the node list in two groups */
                            LinkedList<NodeBean> trialNodeList = new
LinkedList<NodeBean>();
                            LinkedList<NodeBean> remainNodeList = new
LinkedList<NodeBean>();
                            divideNodeList(nodeList, trialNodeList,
remainNodeList);
                            /*
                             * Randomly reconstruct the route and compare with the
original
                             * costs.
                             */
                            RouteBean newRoute = greedyIterate(depot,
originalCosts, originalSize, trialNodeList, remainNodeList);
                            /* Remove r from the route list and add the new route */
                            if (newRoute.getCosts() == 0) {
                                    newRoute = r;
                            }
                            newRouteList.add(newRoute);
                    }
            }
            /*
```

```
		 * Compare old routes and new routes, set the new route list in the
		 * solution.
		 */
		showImprovedResult(sol, newRouteList);
		return sol;
	}
	private static void showImprovedResult(SolutionBean sol,
LinkedList<RouteBean> newRouteList) {
		System.out.println("Solution Costs before GI: " + sol.getCosts());
		sol.setRoutes(newRouteList);
		System.out.println("Solution Costs After GI: " + sol.getCosts());
	}
	private static RouteBean greedyIterate(NodeBean depot, double
originalCosts, int originalSize,
			LinkedList<NodeBean> trialNodeList, LinkedList<NodeBean>
remainNodeList) {
		/* Construct the new route */
		LinkedList<NodeBean> newOrder = new LinkedList<NodeBean>();
		/* Copy the remained list to use as a template */
		for (NodeBean n : remainNodeList) {
			newOrder.add(n);
		}
		/*
		 * Take out a trial node from the trial list one at a time and fill it
		 * randomly in the order
		 */
		double newCosts = originalCosts;
		RouteBean newRoute = new RouteBean();
		/* The condition can only be implemented by a recursive function */
		recursiveReconstruct(newRoute, newOrder, trialNodeList, depot,
newCosts);
		/* Compare the route with the old one */
		return newRoute;
```

```
	}

	private static void recursiveReconstruct(RouteBean newRoute,
LinkedList<NodeBean> newOrder,
			LinkedList<NodeBean> trialNodeList, NodeBean depot, double
newCosts) {
		/* Copy the reconstructed order so far in a new placeholder */
		LinkedList<NodeBean> currentOrder = new LinkedList<NodeBean>();
		for (NodeBean n : newOrder) {
			currentOrder.add(n);
		}
//		System.out.println("\r\nBegin the recursive calling.. Checking
variables.. Current costs: " + newCosts);
		/* Get the first element from the trial list and remove it */
		NodeBean n = trialNodeList.get(0);
		/*
		 * The given node can be placed in different positions 3 times in the
		 * current order.
		 */
		LinkedList<Integer> indexList = new LinkedList<Integer>();
		for (int i = 0; i < currentOrder.size(); i++) {
			indexList.add(i);
		}
		for (int i = 0; i < 3; i++) {
			if (indexList.size() > 0) {
				/* Get a sample of a random index */
				SIMUniformDistribution s = new
SIMUniformDistribution(0, 0, 0, (indexList.size() - 1));
				int randInt = s.getSample();
				int insertIndex = indexList.get(randInt);
				/* Insert the node at the given index */
				currentOrder.add(insertIndex, n);
//				System.out.println("...Insert N" + n.getId() + " at Position
" + insertIndex);
```

```
                        trialNodeList.remove(n);
                        /* Showing the current progress */
//                      String s1 = "Node Order: ";
//                      for (NodeBean nO : currentOrder) {
//                              s1 = s1.concat("N" + nO.getId() + " ");
//                      }
//                      System.out.println(s1);
                        if (trialNodeList.isEmpty()) {
                                /*
                                 * All elements in the trial is used up, so the
current
                                 * order is completed. The current order will be
used to
                                 * generate a new route for comparison.
                                 */
//                              System.out.println("Route completed... Comparing
the costs");
//                              System.out.println("Most efficient costs: " +
newCosts);
                                /* Build the trial route */
                                RouteBean trialRoute = new RouteBean();
                                NodeBean currentPoint = depot;
                                /* Run through all nodes */
                                for (NodeBean nC : currentOrder) {
                                        /* Construct an edge */
                                        EdgeBean e = new EdgeBean(currentPoint,
nC);
                                        trialRoute.getEdges().add(e);
                                        /* Save the current point */
                                        currentPoint = nC;
                                }
                                /* Adding the last trip to Depot */
                                EdgeBean e = new EdgeBean(currentPoint, depot);
```

```
                                        trialRoute.getEdges().add(e);
                                        /* Calculate the costs of the new route */
                                        trialRoute.calCosts();
                                        /* Comparing the costs */
                                        if (trialRoute.getCosts() < newCosts) {
                                                System.out.println("New Route Costs: " +
trialRoute.getCosts());
                                                newCosts = trialRoute.getCosts();
                                        }
                                } else {
                                        /*
                                         * This means the order is not full the function will
call
                                         * itself
                                         */
                                        recursiveReconstruct(newRoute, currentOrder,
trialNodeList, depot, newCosts);
                                }
                                if (indexList.contains(insertIndex)) {
                                        indexList.remove((Integer)insertIndex);
                                }
                                currentOrder.remove(n);
                                trialNodeList.add(n);
                        }
                }
                return;
        }
        private static void divideNodeList(LinkedList<NodeBean> nodeList,
LinkedList<NodeBean> trialNodeList,
                        LinkedList<NodeBean> remainNodeList) {
                /** Conditional division of the trial list based on the number of the
original List
                 * To reduce the simulation time, the trial list should never get too big.
```

```
                * Client request: 0-25 nodes / 3 trial nodes; 25-75 / 5 nodes; 75-125 / 7;
125-200 / 10
                */
                int maxTrialSize = 0;
                int originalSize = nodeList.size();
                if (originalSize < 25) {
                        maxTrialSize = 10;
                } else if ((originalSize >= 25) && (originalSize < 75)) {
                        maxTrialSize = 10;
                } else if ((originalSize >= 75) && (originalSize < 125)) {
                        maxTrialSize = 10;
                } else if (originalSize >= 125) {
                        maxTrialSize = 10;
                }
                if (maxTrialSize > originalSize) {
                        maxTrialSize = originalSize;
                }
                /* Choose random integer less than the size of the node list */
                SIMUniformDistribution s = new SIMUniformDistribution(0, 0, 3,
maxTrialSize);
                /* This will be the number of the node in the shuffling process */
                int nTrialNode = s.getSample();
                System.out.println("Number of the trial nodes: " + nTrialNode);

                for (NodeBean n : nodeList) {
                        remainNodeList.add(n);
                }
                for (int j = 0; j < nTrialNode; j++) {
                        /* Sample a random index */
                        s = new SIMUniformDistribution(0, 0, 0, (remainNodeList.size()
- 1));
                        int randIndex = s.getSample();
```

```
                    /* Get the corresponding node from the node list */
                    trialNodeList.add(remainNodeList.get(randIndex));
                    /*
                     * Remove the corresponding node from the remaining list
                     */
                    remainNodeList.remove(randIndex);
                }
                /* Both sets of nodes are completed, show them */
                String sTrial = "Trial Set: ";
                for (NodeBean n : trialNodeList) {
                    sTrial = sTrial.concat("N" + n.getId() + " ");
                }
//              sTrial = sTrial.concat("\r\n");
                String sRemain = "Remained Set: ";
                for (NodeBean n : remainNodeList) {
                    sRemain = sRemain.concat("N" + n.getId() + " ");
                }
                sRemain = sRemain.concat("\r\n");
                System.out.println(sTrial);
                System.out.println(sRemain);
        }
}
```

Sim-Randomised CWS heuristic and IG algorithm with local search

```
package algorithm;

import java.util.Arrays;
import java.util.LinkedList;
import java.util.List;
import java.util.Random;
import algorithm.object.EdgeBean;
import algorithm.object.NodeBean;
```

```
import algorithm.object.RouteBean;
import sim.object.InputsBean;
import sim.object.InstanceBean;
import sim.object.SimOperationMode;
import sim.object.SolutionBean;
/**
 * @version 150723
 * @author Chanicha Moryadee - chichamoryadee@gmail.com
*/
public class CWSAlgorithm {
	public static SolutionBean solve(InstanceBean aInstance, InputsBean inputs,
			SimOperationMode mode, SolutionBean currentSol) {
		generateDummySol(inputs, currentSol);
		NodeBean depot = inputs.getNodes()[0];
		int index;
		List<EdgeBean> savings = new LinkedList<EdgeBean>();
		/*Adding every edge from the saving list one by one*/
		for (EdgeBean e : inputs.getSavings())
			savings.add(0, e);
		/*Every iteration will remove one edge until nothing left
		 * to consider*/
		while (savings.isEmpty() == false) {
			/*In RANDOM mode, the saving will be taken out randomly
			 * by setting a random index*/
			if (mode.equals(SimOperationMode.RANDOM_CWS)) {
				double beta = 0.1;
				Random rng = new Random();
				index = getRandomPosition(beta, rng, savings.size());
			} else {
				index = 0;
			}
```

```
                    EdgeBean ijEdge = savings.get(index);
                    savings.remove(ijEdge);
                    NodeBean iNode = ijEdge.getOrigin();
                    NodeBean jNode = ijEdge.getEnd();
                    RouteBean iR = iNode.getInRoute();
                    RouteBean jR = jNode.getInRoute();
                    boolean isMergePossible = false;
                    /**/
                    isMergePossible = checkMergingConditions(aInstance, inputs,
iR, jR,
                                    ijEdge);
                    if (isMergePossible == true) {
                            /*Begin the merging process*/
                            EdgeBean iE = getEdge(iR, iNode, depot);
                            /*Remove the edge from Depot to Node I from the
solution*/
                            iR.getEdges().remove(iE);
//                          iR.setCosts(iR.getCosts() - iE.getCosts());
                            /*The origin of the edge is interior*/
                            if (iR.getEdges().size() > 1)
                                    iNode.setIsInterior(true);
                            /*Force the solution to begin from Depot*/
                            if (iR.getEdges().get(0).getOrigin() != depot)
                                    iR.reverse();

                            EdgeBean jE = getEdge(jR, jNode, depot);
                            /*Remove the edge from Depot to Node J*/
                            jR.getEdges().remove(jE);
//                          jR.setCosts(jR.getCosts() - jE.getCosts());
                            /*Set that Node J already in the mid of the route*/
                            if (jR.getEdges().size() > 1)
                                    jNode.setIsInterior(true);
```

```
                                /*Force the solution to end at Depot*/
                                if (jR.getEdges().get(0).getOrigin() == depot)
                                        jR.reverse();
                                /*Adding Edge I to J to iR*/
                                iR.getEdges().add(ijEdge);
//                              iR.setCosts(iR.getCosts() + ijEdge.getCosts());
                                iR.setDemand(iR.getDemand() +
ijEdge.getEnd().getRequestFill());
                                /*Now Node J is transferred to Route I*/
                                jNode.setInRoute(iR);
                                /*Transferring all nodes in Route I to Route J*/
                                for (EdgeBean e : jR.getEdges()) {
                                        iR.getEdges().add(e);
                                        iR.setDemand(iR.getDemand() +
e.getEnd().getRequestFill());
//                                      iR.setCosts(iR.getCosts() + e.getCosts());
                                        e.getEnd().setInRoute(iR);
                                }
                                /*Set new cost and record the new solution*/
                                iR.calCosts();
                                jR.calCosts();
//                              currentSol
//                                              .setCosts(currentSol.getCosts() -
ijEdge.getSavings());
                                /*Remove Route J from the solution*/
                                currentSol.getRoutes().remove(jR);
                        }
                }
                return currentSol;
        }
        /**
         * Constructs an initial dummy feasible solution as described in the CWS
```

```
         * heuristic: dummySol = { (0,i,0) / i in vrpNodesList } During this
         * process, inRoute and isInterior values are assigned.
         */
        private static void generateDummySol(InputsBean inputs, SolutionBean sol) {
                for (int i = 1; i < inputs.getNodes().length; i++) {
                        NodeBean iNode = inputs.getNodes()[i];
                        EdgeBean diEdge = iNode.getDiEdge();
                        EdgeBean idEdge = iNode.getIdEdge();
                        RouteBean didRoute = new RouteBean();
                        didRoute.getEdges().add(diEdge);
                        didRoute.setDemand(
                                        didRoute.getDemand() +
diEdge.getEnd().getRequestFill());
//                      didRoute.setCosts(didRoute.getCosts() + diEdge.getCosts());
                        didRoute.getEdges().add(idEdge);
//                      didRoute.setCosts(didRoute.getCosts() + idEdge.getCosts());
                        didRoute.calCosts();
                        iNode.setInRoute(didRoute);
                        iNode.setIsInterior(false);
                        sol.getRoutes().add(didRoute);
//                      sol.setCosts(sol.getCosts() + didRoute.getCosts());
                        sol.setDemand(sol.getDemand() + didRoute.getDemand());
                }
        }
        private static boolean checkMergingConditions(InstanceBean aInstance,
                        InputsBean inputs, RouteBean iR, RouteBean jR, EdgeBean
ijEdge) {
                if (iR == jR)
                        return false;
                NodeBean iNode = ijEdge.getOrigin();
                NodeBean jNode = ijEdge.getEnd();
                if (iNode.getIsInterior() == true || jNode.getIsInterior() == true)
```

```
                return false;
        if (inputs.getVehCap() < iR.getDemand() + jR.getDemand())
                return false;
//      float maxRoute = 1;
//      float serviceCosts = 9999;
//      int nodesInIR = iR.getEdges().size();
//      int nodesInJR = jR.getEdges().size();
//      double newCost = iR.getCosts() + jR.getCosts() - ijEdge.getSavings();
//      if (newCost > maxRoute - serviceCosts * (nodesInIR + nodesInJR - 2))
//              return false;
        return true;
    }
    /**
     * Given aRoute, iNode and depot, returns the edge in aRoute which contains
     * iNode and depot (it will be the first of the last edge)
     */
    private static EdgeBean getEdge(RouteBean aRoute, NodeBean iNode,
                NodeBean depot) {
        EdgeBean firstEdge = aRoute.getEdges().get(0);
        NodeBean origin = firstEdge.getOrigin();
        NodeBean end = firstEdge.getEnd();
        if ((origin == iNode && end == depot)
                        || (origin == depot && end == iNode))
                return firstEdge;
        else {
                int lastIndex = aRoute.getEdges().size() - 1;
                EdgeBean lastEdge = aRoute.getEdges().get(lastIndex);
                return lastEdge;
        }
    }
    private static int getRandomPosition(double beta, Random r, int size) {
```

```
            int index = (int) (Math.log(r.nextDouble()) / Math.log(1 - beta));
            index = index % size;
            return index;
        }
        @SuppressWarnings("unchecked")
        public static void generateSavingsList(InputsBean inputs) {
            int nNodes = inputs.getNodes().length;
            /* Build Array to keep all possible edges */
            EdgeBean[] savingsArray = new EdgeBean[(nNodes - 1) * (nNodes - 2)
/ 2];
            NodeBean depot = inputs.getNodes()[0];
            int k = 0;
            /*Looping from the first node to the last node*/
            for (int i = 1; i < nNodes - 1; i++) {
                /*Looping from the next node to the last node*/
                for (int j = i + 1; j < nNodes; j++) {
                    /*Set Node I as the departure*/
                    /*Set Node J node as the destination*/
                    NodeBean iNode = inputs.getNodes()[i];
                    NodeBean jNode = inputs.getNodes()[j];
                    /*Create the edge between I and J*/
                    EdgeBean ijEdge = new EdgeBean(iNode, jNode);
                    /*Find the costs (distance) between I and J*/
//                  ijEdge.setCosts(ijEdge.calcCosts(iNode, jNode));
                    /* In every possible pair of nodes, calculating the saving
by
                     * comparing round trip from depot to the direct trip
                     */
                    /*Calculate the saving costs if the connection can be
jointed*/
                    ijEdge.setSavings(EdgeBean.calcSavings(iNode, jNode,
depot));
                    EdgeBean jiEdge = new EdgeBean(jNode, iNode);
```

```
//                              jiEdge.setCosts(jiEdge.calcCosts(jNode, iNode));
                                jiEdge.setSavings(EdgeBean.calcSavings(jNode, iNode,
depot));
                                ijEdge.setInverse(jiEdge);
                                jiEdge.setInverse(ijEdge);
                                savingsArray[k] = ijEdge;
                                k++;
                        }
                }
                /*Now sorting the saving array in the respect to the saving cost*/
                Arrays.sort(savingsArray);
                @SuppressWarnings("rawtypes")
                List sList = Arrays.asList(savingsArray);
                @SuppressWarnings("rawtypes")
                /*Transforming the saving array into a linked list*/
                LinkedList savingsList = new LinkedList(sList);
                inputs.setList(savingsList);
        }
        public static void generateDepotEdges(InputsBean inputs)
        /*
         * Extract the node list and calculate the edge and reverse edge between the
         * respect node and the depot
         */
        {
        }
}
```