DEVELOPMENT OF TWO MATHEURISTICS FOR PRODUCTION-INVENTORY-DISTRIBUTION ROUTING PROBLEM

DICKY LIM TEIK KYEE

FACULTY OF SCIENCE
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

DEVELOPMENT OF TWO MATHEURISTICS FOR PRODUCTION-INVENTORY-DISTRIBUTION ROUTING PROBLEM

DICKY LIM TEIK KYEE

DISSERTATION SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE

INSTITUTE OF MATHEMATICAL SCIENCES
FACULTY OF SCIENCE
UNIVERSITY OF MALAYA
KUALA LUMPUR

2018

UNIVERSITY OF MALAYA

ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: **DICKY LIM TEIK KYEE**

Matric No: **SGP130001**

Name of Degree: **DEGREE OF MASTER OF SCIENCE**

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"):

DEVELOPMENT OF TWO MATHEURISTICS FOR PRODUCTION-INVENOTRY-DISTRIBUTION ROUTING PROBLEM

Field of Study: **OPERATIONS RESEARCH**

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

DEVELOPMENT OF TWO MATHEURISTICS FOR PRODUCTION-INVENTORY-DISTRIBUTION ROUTING PROBLEM

ABSTRACT

Production, inventory and distribution are amongst the most important components of a supply chain network. Integrating production, inventory and distribution decisions is a challenging problem for manufacturers trying to optimize their supply chain. In general, the problem of optimally coordinating production, inventory and transportation is called the production-inventory-distribution routing problem (PIDRP). The PIDRP model considered in this study includes a finite planning horizon, a single production facility, a set of customers with deterministic and time-varying demand, and a fleet of homogeneous capacitated vehicles. The aim of solving the model is to construct a production plan and delivery schedule which minimizes the overall costs while fulfilling customers' demand over the planning horizon. We propose an optimization algorithm designed by the interpolation of metaheuristics and mathematical programming techniques, known as MatHeuristics algorithm, to solve the model. In this study, we develop two different two-phase approaches within a MatHeuristic framework, namely MatHeuristic1 and MatHeuristic2. In MatHeuristic1, Phase 1 solves a mixed integer programming (MIP) model which includes all the constraints in the original model except the routing aspects. The routing constraints are replaced by an approximated routing cost in the objective function. A variable neighborhood search (VNS) is proposed in Phase 2. The VNS is employed to improve the solution and it incorporates some aspects of Tabu search to escape from local optima. Whilst in MatHeuristic2, we adopt a slightly different approach. The algorithm starts with the pre-processing stage where the routes are determined to act as input to the MIP in Phase 1. Phase 2 uses VNS and we utilize both phases in an interactive manner. The models in both approaches are solved by using Concert Technology of CPLEX 12.5 Optimizers with Microsoft Visual

C++ 2010. Both algorithms are tested on a set of 90 benchmark instances with 50, 100 and 200 customers and 20 periods and the results are competitive when compared to the Memetic Algorithm with Population Management (MA|PM), Reactive Tabu Search (RTS) and Scatter Search (SS). MatHeuristic1 performs well in large instances when compared to other metaheuristics but MatHeuristic2 performs well in smaller cases.

Keywords: Production-Inventory-Distribution Routing Problem, MatHeuristics, Variable Neighbourhood Search, Mixed Integer Programming, Concert Technology

PEMBANGUNAN DUA MATHEURISTICS UNTUK MASALAH LALUAN PENGELUARAN, INVENTORI DAN PENGAGIHAN

ABSTRAK

Pengeluaran, inventori dan pengagihan ialah di antara komponen yang terpenting di dalam rantaian bekalan. Mengintegrasi keputusan pengeluaran, inventori dan pengagihan merupakan satu cabaran bagi pengilang yang cuba mengoptimumkan rantaian bekalan mereka. Secara umum, masalah menyelaraskan pengeluaran, inventori dan pengagihan secara optimum dipanggil masalah laluan pengeluaran, inventori dan pengagihan (*Production-Inventory-Distribution Routing Problem (PIDRP)*). Model PIDRP yang dipertimbangkan dalam kajian ini merangkumi tempoh perancangan yang terhad, satu kilang pengeluaran, pelanggan yang mempunyai permintaan deterministik dan berbeza mengikut tempoh masa dan juga kenderaan homogen berkapasiti. Objektif bagi menyelesaikan model ini adalah untuk membina rancangan pengeluaran dan jadual penghantaran yang dapat mengurangkan kos keseluruhan di samping memenuhi kesemua permintaan pelanggan di sepanjang tempoh perancangan. Kami mencadangkan satu algoritma pengoptimuman yang direka oleh interpolasi metaheuristik dengan teknik pengaturcaraan matematik yang dikenali sebagai algoritma *MatHeuristics* bagi menyelesaikan model tersebut. Di dalam kajian ini, kami bangunkan dua pendekatan dua fasa yang berbeza di dalam struktur kerja *MatHeuristic*, iaitu *MatHeuristic1* dan *MatHeuristic2*. Dalam *MatHeuristic1*, fasa 1 menyelesaikan model pengaturcaraan integer campuran (*Mixed Integer Programming (MIP)*) yang merangkumi semua kekangan dalam model asal kecuali kekangan laluan. Kekangan laluan digantikan dengan kos hampiran laluan di dalam fungsi objektif. *Variable Neighborhood Search (VNS)* dicadangkan dalam Fasa 2. *VNS* digunakan untuk menambahbaikkan penyelesaian dan ia menggabungkan beberapa aspek *Tabu Search* untuk melepaskan

diri dari optima tempatan. Manakala di dalam MatHeuristic2, kami menggunakan pendekatan yang sedikit berbeza. Algoritma bermula dari peringkat pra-pemprosesan di mana laluan yang telah ditentukan adalah input bagi MIP dalam fasa 1. Fasa 2 menggunakan VNS dan kedua-dua fasa digunakan secara interaktif. Kedua-dua pendekatan diselesaikan dengan menggunakan kaedah teknologi *Concert* (*Concert Technology*) daripada Pengoptimal *CPLEX 12.5* dengan *Microsoft Visual C++ 2010*. Kedua-dua algoritma diuji pada 90 set data merangkumi 50, 100 dan 200 pelanggan dengan 20 tempoh masa dan keputusan yang didapati adalah kompetitif jika dibandingkan dengan *Memetic Algorithm with Population Management (MA|PM), Reactive Tabu Search (RTS)* dan *Scatter Search (SS)*. *MatHeuristic1* menghasilkan keputusan yang lebih baik berbanding dengan metaheuristik lain untuk set data besar manakala *MatHeuristic2* menghasilkan keputusan yang lebih baik dalam set data yang lebih kecil.

Kata Kunci: Masalah Pengeluaran, Inventori dan Pengagihan (PIDRP), *MatHeuristic, Variable Neighbourhood Search,* pengaturcaraan integer campuran, teknologi *Concert*

ACKNOWLEDGEMENTS

First and foremost, I would like to thank to my supervisors, Associate Prof. Dr. Noor Hasnah Moin and Prof. Dr. Nor Aishah binti Hamzah, for giving me the opportunity to do research and providing invaluable guidance, continuous encouragement and constant support throughout this research. I also sincerely thank them for the time they spent proofreading and commenting my thesis.

Secondly, my sincere thanks go to the financial support of the Skim Biasiswazah under University of Malaya and MyMaster programme from Ministry of Higher Education (KPT).

Thirdly, I am extremely grateful to my family, parents, brothers and sisters for being there when I needed help and guidance and supporting me throughout my studies.

Besides, I would like to say thanks to my friends and research colleagues for their constant encouragement. Lastly, I would like to thank all the people who have supported me to complete the research work directly or indirectly.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

ALNS	:	Adaptive Large Neighborhood Search
B&B	:	Branch-and-Bound
B&P	:	Branch-and-Price
CVRP	:	Capacitated Vehicle Routing Problem
D-W	:	Dantzig-Wolfe
GRASP	:	Greedy Randomized Adaptive Search Procedure
GT	:	Giant Tour
IRP	:	Inventory Routing Problem
LSP	:	Lot-Sizing Problem
LTL	:	Less-than Transporter Load
MA|PM	:	Memetic Algorithm with Population Management
MCF	:	Minimum Cost Flow
MILP	:	Mixed Integer Linear Programming
MIP	:	Mixed Integer Programming
ML	:	Maximum Level
MP	:	Master Problem
OU	:	Order-Up-to Level
PD	:	Production-Distribution
PIDRP	:	Production-Inventory-Distribution Routing Problem
PR	:	Path Relinking
PRP	:	Periodic Routing Problem
PRP-B	:	Production Routing Problem with Backordering
RefSet	:	Reference Set
RTS	:	Reactive Tabu Search

SCM	:	Supply Chain Management
SPRP	:	Stochastic Production Routing Problem
SS	:	Scatter Search
TSPR	:	Tabu Search with Path Relinking
VMI	:	Vendor Managed Inventory
VNS	:	Variable Neighborhood Search
VRP	:	Vehicle Routing Problem

CHAPTER 1 INTRODUCTION

1.1 Overview

In an increasingly competitive business environment, many companies face problems with their inventory and distribution management. After years of focusing on the reduction of costs in production and operation, they are beginning to investigate distribution activities as the last component for cost reduction. One way of achieving this is by identifying and eliminating redundancies (in terms of low capacity utilization) and inefficiencies by coordinating various processes within the whole supply chain network. Supply chain management (SCM) is a set of approaches utilized to efficiently integrate suppliers, manufacturers, warehouses and stores, so that merchandise is produced and distributed in the right quantities, to the right location, and at the right time, in order to minimize system-wide costs while satisfying service level requirements (Simchi-Levi *et al.*, 2003). Despite the extensive literature on SCM, the benefits and challenges of coordinated decisions making within supply chain scheduling models have not been intensively studied.

Production, inventory and distributions are amongst the most critical components of a supply chain network. Integrating production and distribution decisions is a challenging problem for manufacturers trying to optimize their supply chain. At the planning level, the immediate goal is to coordinate production, inventory, and delivery to meet customers' demand so that the corresponding costs are minimized. Achieving this goal provides the foundations for streamlining the logistics network and for integrating other operational and financial components of the organization. In general, the problem of optimally coordinating production, inventory and transportation is called the production-inventory-distribution routing problem (PIDRP) (Lei *et al.*, 2006) or a production routing problem (Adulyasak *et al.*, 2015b). This problem is NP-hard (Bard & Nananukul, 2010).

The PIDRP usually arises in the retail industry where customers or outlets rely on a central supplier or manufacturer to provide them with a given commodity on a regular basis. The integration of production, inventory, and distribution increases the complexity of the problem. It is characterized by a combination of a capacitated lot-sizing problem and a capacitated, multi-period vehicle routing problem (VRP). A manufacturer must develop minimum cost schedules for production and distribution of a single product that are sufficient to meet all customer demands over the planning horizon.

The PIDRP is relevant in Vendor Managed Inventory (VMI), where the supplier (manufacturer or vendor) monitors the inventory at the retailers and coordinates efficient utilization of resources to replenish the retailers. This policy requires the vendor to manage the timing and the amount of deliveries while ensuring that customers will never be out of stock. It also offers the flexibility to choose the most preferred transportation mode.

The PIDRP can be classified according to its characteristics: number of production facilities (single or multiple) with production capacity (capacitated or uncapacitated), production of a single or multiple item(s), distribution to (single or multiple) customer sites by a fleet of (homogeneous, heterogeneous or outsourced) vehicles. In particular, this thesis considers a PIDRP problem with a single production facility with a limited capacity, that produces a single product and distributes to multiple customer sites by a fleet of homogeneous vehicle.

At each time epoch of a finite planning horizon, PIDRP determines the production quantity of the single item at the single facility, the delivery quantity and route of each vehicle. These quantities and routes are to minimize the overall cost over the planning horizon. The cost consists of the production setup cost, the inventory cost of the items stored at the production facility and the routing cost of the vehicles. PIDRP assumes

that no shortage is allowed for any customer; that is, all demand must be satisfied on-time. In summary, three critical decisions have to be made in PIDRP for each period in the planning horizon are the following:

(i) How much product to be manufactured?

(ii) The amount to be delivered to each customer site.

(iii) The route of each vehicle in the fleet. (i.e., which delivery route to use?)

1.2 Research Objective and Contribution

1.2.1 Objective of the Thesis

The main objective of the thesis is to design and evaluate the performance of two MatHeuristic approaches for PIDRP.

First, it proposes a two-phase approach (MatHeuristic1 algorithm) by studying the mathematical model and the allocation model for PIDRP proposed by Bard and Nananukul (2009), developing Variable Neighborhood Search (VNS) algorithm to improve the solution and performing the computational testing and comparing the results from several metaheuristic algorithms.

Second, it develops a two-phase iterative approach in a MatHeuristic framework, known as MatHeuristic2 algorithm, for solving the PIDRP model by developing a modified mixed integer programming (MIP) model which includes a routing parameter, developing different types of VNS and embed simulated annealing in the algorithm and performing the computational testing and comparing the results from several metaheuristic algorithms.

Lastly, it assesses the efficiency of MatHeuristic1 and MatHeuristic2 algorithms by performing statistical analyses.

1.2.2 Contribution to Scientific Knowledge

This research considers the production, inventory and distribution routing problem and develops two different two-phase approaches, called MatHeuristic1 and MatHeuristic2 algorithms, to solve the problem. In this research, the integrated PIDRP is modeled as a one-to-many distribution system, which includes a single production facility and a set of customers with deterministic and time-varying demand. A fleet of homogenous capacitated vehicles for making the deliveries is also considered.

First, this research designs a two-phase algorithm, MatHeuristic1 to solve PIDRP. The first phase exploits the allocation model that solves a mixed integer program (MIP). This MIP simplifies the original model, by using approximated routes. The output of this model determines the optimal allocations, inventories (at both the production facility and customer sites) and the production quantities over the finite planning horizon. Based on the results obtained from phase 1, phase 2 constructs the delivery schedules for each period using the Giant Tour algorithm and improves the solution using a well known metaheuristic algorithm, Variable Neighborhood Search (VNS). The VNS is modified to incorporate the tabu search mechanism where it forbids the move that has recently been made in order to avoid being trapped in local minima. The model is solved by using the Concert Technology of CPLEX 12.5 Optimizers with Microsoft Visual C++ 2010. The algorithm is applied to the data sets proposed by Boudia *et al.* (2009) consisting of 50, 100 and 200 customers, all with 20 periods. The computational results are then compared to the best results obtained from Memetic Algorithm with Population Management (MA|PM) (Boudia & Prins, 2009), Reactive Tabu Search (RTS) (Bard & Nananukul, 2009) and Scatter Search (SS) (Moin & Yuliana, 2015). MatHeuristic1 is also tested on a randomly generated data set where holding cost at the customer sites is set at more than 0 and compared to the best integer solutions from CPLEX. The gap between MatHeuristic1 and the adjusted lower bound

is computed. Some non-parametric procedures such as Friedman test and a post-hoc test are performed to determine whether MatHeuristic1 algorithm is significantly better than the existing metaheuristic algorithms.

Second, this thesis designs an alternative approach, MatHeuristic2. To construct the initial routes, an initial feasible solution is created by using zero-inventory with a VNS that makes use of the three neighborhood structures defined in Imran *et al.* (2009) in the shaking step, and some perturbations such that better initial feasible routes are obtained. Based on the output of the MIP model, the routes are formed again and improved using two types of VNS, namely VNS with best move and VNS with random move. Some properties of simulated annealing are applied with the aim to accept a worse solution according to a probability of acceptance. The process is iterated until a defined maximum of iterations is reached. The algorithm is applied to the same data set as in MatHeuristic1 and also compared the results to MA|PM, RTS and SS as well. The gap between MatHeuristic2 and the adjusted lower bound (Bard & Nananukul, 2009) is also computed. Two non-parametric procedures are performed to detect the significant difference between MatHeuristic1 and MatHeuristic2 algorithms. Its major difference in comparison to MatHeuristic1 is its iterative steps. MatHeuristic2 iterates between the allocation model and VNS whereas in MatHeuristic1, the allocation model is applied only once. In MatHeuristic2, the MIP model is modified and it requires the input of the real routes obtained from a heuristic method as opposed to MatHeuristic1 where the routes used were approximated.

Computational performance of both proposed algorithms MatHeuristic1 and MatHeurstic2 are promising. We are able to solve PIDRP within reasonable computational time. Besides, the proposed algorithms improve some upper bounds. The methodology can also be applied to some real life supply problems, such as

manufacturing companies: Intel (M) Sdn. Bhd., Dell Asia Pacific Sdn. Bhd., etc. and retail companies: Tesco, Parkson, Harvey Norman, etc..

1.3 Organization of the Thesis

This thesis is organized as follows:

A literature review is provided in Chapter 2. The papers of the inventory routing problem (IRP) and its extension as well as the PIDRP that are related to the lot-sizing problem (LSP) and VRP are provided in the introduction part of the chapter. The classification based on the characteristics and types of the algorithms for the PIDRP is also displayed in a table form. The literature is divided according to the type of solution methodology for PIDRP:

a) **Exact algorithms**, such as the branch-and-cut with LSP reformulation (Ruokokoski *et al.*, 2010), the branch-and-cut for the PIDRP with single capacitated vehicle (Archetti *et al.*, 2011), etc.
b) **Heuristic algorithms**, such as a two-phase heuristic approach (Lei *et al.*, 2006), a branch-and-price-based hybrid heuristic (Bard & Nananukul, 2010), etc.
c) **Metaheuristics algorithms**, such as GRASP (Boudia *et al.*, 2007), MA|PM (Boudia & Prins, 2009), RTS (Bard & Nananukul, 2009), etc.

At the end of the chapter, the research direction and the description of data set provided by Boudia *et al.* (2005) is presented.

Chapter 3 focuses on the development of MatHeuristic1. First, the description of the PIDRP model with all assumptions is provided. The notations used in the development of the mathematical formulation are presented and the PIDRP model is formulated. In the next section, the formulation of the mixed integer program (MIP) model with the additional notations and additional constraint are given. MIP is used to obtain a set of

feasible allocations in order to construct an initial feasible solution. Variable Neighborhood Search (VNS), a well known metaheuristics, is proposed to improve the initial solution. The general steps and the flow chart of VNS are provided. In the initialization of VNS, Giant Tour procedure and Dijkstra's algorithm are discussed in detail. For VNS shaking step, we consider three moves: forward transfer, backward transfer and swap as the neighborhood structures. The explanation and examples of the three moves are provided and discussed in detail. We consider 6 local searches, namely 1-insertion inter-route, 2-opt inter-route, 2-opt intra route, swap intra route, 1-insertion intra-route, and 2-insertion intra-route, to improve the solution obtained from shaking. In addition, we also incorporate the concept of tabu search into the algorithm to escape from local optima. The important steps of the MatHeuristic1 algorithm are summarized and elaborated.

MatHeuristic2 is proposed in Chapter 4. The three types of VNS are used in this algorithm: VNS-interchange which adopts the three neighborhood structures proposed by Imran and Salhi (2009), VNS-Bestmove and VNS-Random. They are explained in details. The modified (or enhanced) version of MIP with the additional routing parameter is presented. The explanation of the important steps and the flow chart of MatHeuristic2 algorithm are also provided. MatHeuristic2 incorporates the concept of simulated annealing, in addition to the concept tabu search embedded in MatHeuristic1.

Both algorithms are tested on a benchmark data set given by Boudia *et al.* (2009) consisting of 50, 100 and 200 customers with 20 periods. The results obtained from both algorithms are compared to memetic algorithm with population management, MA|PM (Boudia & Prins, 2009), Reactive Tabu Search, RTS (Bard & Nananukul, 2009) and Scatter Search, SS (Moin & Yuliana, 2015) in terms of total cost and runtime. The gap between our algorithms and the adjusted lower bound obtained from Bard and

Nananukul (2009) is calculated in all cases. For MatHeuristic1, we also tested on a randomly generated data set consisting of 12, 20, 50 and 100 customers with 5, 10, 14 and 21 periods and compared with CPLEX best integer solutions. In addition, we also compare MatHeuristic1 algorithm with MatHeuristic2 approach. In terms of total cost, to test the significant difference between MatHeuristic1 and the existing algorithms from the literature (MA|PM, RTS and SS), we perform three non-parametric procedures, namely the Friedman test, Iman and Davenport and Freidman Aligned Rank Test. To determine whether our algorithms are significantly better, we carry out a post-hoc test using Holm procedure. Whilst in Chapter 4, we perform two non-parametric procedures, namely the Sign test and the Wilcoxon signed ranks test to identify the significant differences of MatHeuristic1 and MatHeuristic2 algorithms.

In Chapter 5, all the work done is concluded and further extensions are outlined.

CHAPTER 2 LITERATURE REVIEW

2.1 Introduction

The production, inventory and distribution routing problem combines the inventory routing problem and the periodic routing problem. For literature reviews on IRP and its extensions, see Abdelmaguid and Dessouky (2006), Moin and Salhi (2007), Savelsbergh and Song (2007), Andersson *et al.* (2010), Aghezzaf *et al.* (2012), and Coelho *et al.* (2013). Whilst for the literature on PRP, see Gaudioso and Paletta (1992), Parthanadee and Logendran (2006) and Mourgaya and Vanderback (2007).

The PIDRP is also be related to the lot-sizing problem and the vehicle routing problem. LSP considers the production and the distribution of the demand to a set of customers over a finite planning horizon. For literature reviews on LSP, see Karimi *et al.* (2003) and Brahimi *et al.* (2006). The VRP constructs the delivery schedules based on the planned delivery quantities. We refer the readers to the book by Toth and Vigo (2014) for a comprehensive review of VRP.

2.2 Solution methodology for PIDRP

The majority of complex real world decision problems, including PIDRP, belong to the class of NP-hard problems. Several solution methods have been developed to solve the PIDRP model. These methodologies can be classified as: exact, heuristic and metaheuristic. The models considered in PIDRP are studied under the following three assumptions. The production facility can be single or multiple, and can produce a single or multiple products. The distribution is carried out by a fleet of homogeneous or heterogeneous vehicles with limited or unlimited capacity. The production and the storage capacity can be limited or unlimited. Table 2.1 illustrates the classification of the characteristics and types of the algorithms (in the literature) for the PIDRP. We also refer the reader to Adulyasak *et al.* (2015b) for a detailed classification.

2.2.1 Exact Algorithm

This section provides a detailed review of the exact algorithms for solving the PIDRP. The four studies are the branch-and-cut with LSP reformulation (Ruokokoski *et al.*, 2010), the branch-and-cut for the PIDRP with single capacitated vehicle (Archetti *et al.*, 2011), the branch-and-cut for the PIDRP with multiple capacitated vehicles (Adulyasak *et al.*, 2013) and lastly the Benders decomposition algorithm that solved PIDRP with demand uncertainty (Adulyasak *et al.*, 2015a).

Ruokokoski *et al.* (2010) was the first to develop an exact algorithm and several LSP reformulations for solving the PIDRP problem with a single uncapacitated plant and a single vehicle with unlimited capacity are solved. The authors proposed a branch-and-cut algorithm as an exact algorithm to solve the LSP reformulations. Next, they introduced 2-matching inequalities proposed by Fischetti *et al.* (1997) and generalized comb algorithm to strengthen the formulations. Then, a heuristic exact separation procedure is applied to the generalized comb inequalities. Lastly, the authors adapted a priori tour based heuristic (Solyalı & Süral, 2008) to obtain high-quality solutions. The algorithm can solve the instances with up to 40 customers and 15 periods or 80 customers and 8 periods to optimality within two hours.

Archetti *et al.* (2011) presented a branch-and-cut algorithm to solve the PIDRP with a single capacitated vehicle in each period. An optimal allocation (delivery quantity), the vehicles used and the distribution sequence of the customers served are determined by the proposed exact algorithm. During the branch-and-cut process, the subtour elimination constraints are removed and the violated cuts are introduced iteratively. The algorithm is tested on a set of instances with 14 customers and 6 periods. Results showed that most of the instances solved to optimality within a few seconds.

Table 2.1: Classification of the characteristics and types of the algorithms for the PIDRP

Author/s (year)	Characteristics/Types							
	Solution method	**Plant**	**Product**	**Fleet**	**Number of Vehicle**	**Prod. Cap.**	**Inv. Cap.**	**Veh. Cap.**
Exact algorithm								
Ruokokoski *et al.* (2010)	Branch-and-cut	Single	Single	Homogeneous	Single	No	No	No
Archetti *et al.* (2011)	Branch-and-cut	Single	Single	Homogeneous	Single	No	Yes	Yes
Adulyasak *et al.* (2013)	Branch-and-cut	Single	Single	Homogeneous	Multiple	Yes	Yes	Yes
Adulyasak *et al.* (2015a)	Benders-based Branch-and-cut	Single	Single	Homogeneous	Multiple	Yes	Yes	Yes
Heuristic algorithm								
Chandra (1993)	Decomposition	Single	Multiple	Homogeneous	Unlimited	Yes	No	Yes
Chandra and Fisher (1994)	Decomposition	Single	Multiple	Homogeneous	Unlimited	Yes	No	Yes
Lei *et al.* (2006)	Two-phase	Multiple	Single	Heterogeneous	Limited	Yes	Yes	Yes
Boudia *et al.* (2008)	Two-phase	Single	Single	Homogeneous	Limited	Yes	Yes	Yes
Bard and Nananukul (2010)	B&P based hybrid	Single	Single	Homogeneous	Limited	Yes	Yes	Yes
Archetti *et al.* (2011)	MIP based hybrid	Single	Single	Homogeneous	Single	No	Yes	Yes
Absi *et al.* (2013)	Iterative MIP	Single	Single	Homogeneous	Multiple	No	Yes	Yes
Brahimi and Aouam (2015)	HRF	Single	Multiple	Homogeneous	Limited	Yes	Yes	Yes
Watanabe *et al.* (2017)	Model-based	Single	Multiple	Homogeneous	Limited	Yes	Yes	Yes
Metaheuristic algorithm								
Boudia et al. (2007)	GRASP	Single	Single	Homogeneous	Limited	Yes	Yes	Yes
Boudia and Prins (2009)	MA|PM	Single	Single	Homogeneous	Limited	Yes	Yes	Yes
Bard and Nananukul (2009)	Tabu search	Single	Single	Homogeneous	Limited	Yes	Yes	Yes
Armentano *et al.* (2011)	Tabu search	Single	Multiple	Homogeneous	Limited	Yes	Yes	Yes
Adulyasak *et al.* (2012)	Op-ALNS	Single	Single	Homogeneous	Multiple	Yes	Yes	Yes
Moin and Yuliana (2015)	Scatter Search	Single	Single	Homogeneous	Limited	Yes	Yes	Yes

Adulyasak *et al.* (2013) extended the branch-and-cut algorithm of Archetti *et al.* (2011) to the PIDRP by including multiple vehicles instead of a single vehicle. The authors considered and compared two formulations of PIDRP: one with a vehicle index and the other without a vehicle index. They solved the first formulation by a branch-and-cut approach similar to the one proposed by Ruokokoski *et al.* (2010). The authors used the separation algorithms of Lysgaard *et al.* (2004) and developed a greedy heuristic in the second formulation. The first formulation can solve instances with up to 35 customers, 3 periods and 3 vehicles to optimality within 12 hours, while the second formulation can produce better lower bounds for the instances that were not solved to optimality within two hours.

Adulyasak *et al.* (2015a) studied the PIDRP under demand uncertainty in two- or multiple-stage decision processes. The first stage decision includes the production and distribution plans, while the subsequent stages determine the production and delivery quantities when the demand is known. The authors proposed an exact algorithm based on Benders decomposition. They proposed two different formulations for two-stage and multi-stage stochastic PIDRP; compared a branch-and-cut algorithm and a Benders decomposition algorithm to solve the problems; enhanced the formulations by improving the lower bound, aggregate Benders cuts (using scenario groups) and Pareto-optimal cuts (Magnati & Wong, 1981); and obtained good feasible solutions by developing an effective rollout heuristic. The computational results showed that the algorithm outperforms the branch-and-cut approach of Adulyasak *et al.* (2013) when solving large instances. Besides, the computational results also showed that the instances can be solved to optimality for both the two-stage and multiple-stage problems.

2.2.2 Heuristic Algorithm

Heuristic algorithms are developed to solve optimization problems with the advantages of simplicity, easy to apply and low time complexity. In this section, a few studies, which proposed a heuristic approach to solve the PIDRP, have been studied and discussed in details.

The first heuristic approach was first proposed by Chandra (1993) and Chandra and Fisher (1994) to solve a PIDRP involving a single plant, multiple products and a fleet of homogeneous vehicles. The authors evaluated two different solution approaches to analyze the value of coordinating production and distribution. The first is based on the decoupled approach where it solved the production and distribution scheduling problems separately. The approach solved the capacitated lot size problem to optimality and a distribution schedule is determined for each period and improved using sweep (Gillette & Miller, 1974), nearest neighbor rule (Rosenkrantz *et al.*, 1974), feasible insertion rule (Chandra, 1991) and followed by 3-opt interchange procedure (Lin & Kernighan, 1973). The solution was then improved by combining the amount delivered to customer in a period with another delivery to the same customer in an earlier period. However, the modification of the production plan is not allowed. In the second approach, they coordinated the production and distribution planning within a single model. The coordinated approach follows the same procedure used in the first approach except that the production shifting is allowed over the periods. Both heuristic approaches were tested on problem instances with up to 50 customers, 10 periods and 10 products. The computational results presented a reduction in the total cost obtained by the second (or coordinated) approach ranged from 3% to 20% when compared to the first (decoupled) approach.

Lei *et al.* (2006) considered a PIDRP with multiple production plants and a heterogeneous fleet of vehicles. The objective is to coordinate the production, inventory and transportation schedules to minimize the total costs. The authors proposed a two-phase heuristic approach to solve the problem. In phase 1, they solved a simplified version of the problem as a mixed integer programming (MIP) model subject to all the constraints in the full model and the routings is restricted to direct shipment between facilities and customer sites. The production schedule and the amount delivered to each customer are obtained from the results from phase 1. In phase 2, a capacitated vehicle routing problem (CVRP) heuristic based on an extended optimal partitioning procedure that consolidated the Less-than Transporter Load (LTL) assignments is applied to produce the delivery schedules. The approach is tested on instances up to 12 customers, 4 periods and 2 vehicles. The computational results showed that the two-phase heuristic can produce better solutions than CPLEX.

An improved version of decomposition-based heuristic approach was proposed by Boudia *et al.* (2008) to solve PIDRP with a single plant and multiple customers with known demand. Similar to Lei *et al.* (2006), the authors compared two different approaches to evaluate the effectiveness of the integration of production and distribution decisions. The first approach is the two-phase decoupled heuristic, where the production schedules are determined using the Wagner and Whitin algorithm (Wagner & Whitin, 1958) and the distribution schedules constructed by a 3-opt procedure. The second one is the coupled approach, where the production quantities are set as large as possible to cover some future periods and the delivery schedule is determined by the saving algorithm proposed by Clarke and Wright (1964). The algorithm then tries to reduce production cost by applying the Wagner and Whitin algorithm. Next, local search consisting of swap, insertion and 3-opt procedure is applied to improve the solution.

Both heuristic approaches are tested on randomly generated instances of Boudia *et al.* (2005) with 50 to 200 customers and 20 periods. The second approach provides a cost savings ranging from 10% to 15% when compared to the first approach.

Bard and Nananukul (2010) proposed a branch-and-price based hybrid heuristic to solve the PIDRP that includes a single production facility, multiple customers, a fleet of homogeneous vehicles and a finite time horizon. They developed a tabu search metaheuristic and an exact method based on a branch-and-price (B&P) that combines a Dantzig-Wolfe (D-W) decomposition (Vanderback, 2000) and a standard branch-and-bound (B&B) (Wolsey, 1998). First, an initial solution is constructed by the following three steps: (i) solves an allocation model proposed by Bard and Nananukul (2009); (ii) solves a capacitated vehicle routing problem (CVRP) based on tabu search (Carlton, 1995); (iii) improves the allocations and the routing solutions obtained from (ii) by applying a full tabu search. Second, they derive a master problem (MP) and uses its solution in pricing subproblem during the column generation process. Third, They use the branching strategy for 0-1 mixed integer formulation to accommodate the unique degeneracy characteristics of the MP. Lastly, they apply a rounding heuristic approach to improve the efficiency of the algorithm. Computational experiments are constructed on a data set of instances with up to 50 customers and 8 periods and the results showed that the B&P heuristic could obtain better solutions when compared to CPLEX given the maximum computational runtime of one hour.

Archetti *et al.* (2011) considered a model with a one-to-many distribution, a finite planning horizon and a fleet of capacitated vehicles. The production, replenishment and delivery plans are to be determined so as the overall cost is minimized. The authors proposed a MIP-based hybrid heuristic for solving the problem with two different policies: maximum level (ML) and order-up to level (OU). The proposed model is

decomposed into two subproblems, namely, uncapacitated production lot-sizing and distribution subproblems. In the first step, the quantity delivered to each customer and delivery routes are determined over the planning horizon by assuming unlimited production capacity. The problem of determining production quantity in each period is optimally solved using the results obtained from the first step. Consequently the solution obtained in the previous two steps is improved iteratively by removing and reinserting two customers. If the total cost is reduced, the production lot-sizing subproblem is solved again. The improvement procedure is repeated as long as there is a reduction in the total cost. Computational results are conducted on a large data set of randomly generated instances and the authors showed that the ML policy obtained better results when compared to the OU policy.

Absi *et al.* (2014) considered a single-item PIDRP with a production facility and multiple retailers. The authors presented a two-phase iterative heuristic for solving the problem with maximum level policy. In phase 1, called the lot-sizing phase, the MIP model is solved to determine the production, inventory and distribution decisions. In phase 2, called the routing phase, a heuristic is applied to determine and improve the routes based on the routing information obtained in the first phase. The cost of the incumbent solution is updated and used for the next iteration of the phase 1. Two diversification mechanisms: A multi-start procedure and a second procedure called "update diversification mechanism" are applied when there is no improvement after a specified number of iterations. The procedures are repeated until a maximum number of iterations is reached. Computational experiments are conducted on the data set with uncapacitated production proposed by Archetti *et al.* (2011) and the results showed that the proposed heuristic outperforms Memetic Algorithm (MA) (Boudia *et al.*, 2009), Reactive Tabu Search (RTS) (Bard & Nananukul, 2009), Tabu Search with Path

Relinking (TSPR) (Armentano *et al.*, 2011) and Adaptive Large Neighborhood Search (ALNS) (Adulyasak *et al*., 2012).

In the following two years, Brahimi and Aouam (2016) developed two mixed integer linear programming (MILP) formulations and a new hybrid heuristic known as HRF heuristic for solving the integrated multi-item PIDRP. The PIDRP considered consists of a single production facility with limited production and inventory capacities and a set of geographically dispersed customers with limited storage capacities. Backordering is allowed by incurring a penalty cost. The HRF heuristic combines a relax-and-fix heuristic for solving a lot sizing problem and a VRP local search procedure for constructing the delivery schedule. The difference between the HRF heuristic and other similar heuristic is that they solve the model without completely removing routing components. The formulation of MILP is based on the aggregated lot-sizing formulation and the two-index vehicle flow model. The numerical results showed that the HRF heuristic outperformed the solution obtained from a commercial solver, in terms of solution quality and computational time.

Watanabe *et al.* (2017) proposed a model-based heuristic for solving the PIDRP with the same features considered by Armentano *et al.* (2011) for a single production facility, multiple products and a fleet of homogeneous vehicles. The aim of solving the problem is to minimize the sum of production, inventory and transportation costs over a finite planning horizon. In their heuristic, an initial feasible solution is generated by a relax-and-fix approach. Next, a fix-and-optimize improvement approach is applied to the initial solution obtained, aimed at finding a better solution. Both approaches are based on a decomposition of the set of all binary and integer decision variables. The authors generated a set of instances with multiple products with up to 15 customers, 14 periods and 5 products to test the proposed heuristic and a set of instances proposed by Archetti

et al. (2011). The computational results showed that the heuristic approach can provide better solutions than solution obtained from CPLEX 12.6.

2.2.3 Metaheuristics Algorithm

Metaheuristics have received more attention because of its ability to solve hard combinatorial optimization problems within a reasonable computational time. Some metaheuristic approaches such as GRASP (Boudia *et al.*, 2007), MA|PM (Boudia & Prins, 2009), RTS (Bard & Nananukul, 2009 and Armentano *et al.*, 2011), Op-ALNS (Adulyasak *et al.*, 2012) and Scatter Search (Moin & Yuliana, 2015) were adapted to solving the PIDRP and they are discussed in details in this section.

Boudia *et al.* (2007) considered a network with a single production facility and a set of customers with time-varying demands. The authors proposed a greedy randomized adaptive search procedure (GRASP) to solve PIDRP with a single item and a limited fleet of homogeneous vehicles over a multi-period planning horizon. The aim is to minimize the total cost (production setup, inventories and transportation) while fulfilling all demand for each period. No backorder is allowed.

Each GRASP iteration produces a feasible solution and has two main phases. The first phase is the construction phase, where a production plan and delivery schedules were determined and a balance between setup and inventory costs is achieved by shifting some production periods. The second phase is a local search. In this phase, the present delivery schedules were improved by 2-opt moves, swap, exchange and insertion without violating capacity constraint. Basic GRASP was also made reactive (proposed for the first time by Prais and Ribeiro (2000)) and it gives better results than GRASP itself.

Besides, the authors proposed a path-relinking (PR) procedure to improve a basic GRASP. Path relinking explores paths between best solutions found and creates a pool of the best solutions by storing a limited number of new best solutions. The resulting pool is then sorted in an ascending order of total costs. Instead of using similarity between the solutions as in genetic algorithms, path relinking focuses on the dissimilarity measure (Resendel & Riberio, 2005) to ensure that the pool of solutions is diversified. The pool is divided into two consisting of initiating and guiding solutions respectively. Each path links an initiating group is then modified to match the path in the guiding group. This is carried out until some specified number of iterations. The authors also suggested two strategies for the combination of PR and GRASP: (1) the GRASP with PR at the end, and (2) perform PR every time GRASP found a new best solution.

The algorithms were tested on a data set given by Boudia *et al.* (2005) with 30 instances of 50, 100 and 200 customers, all with 20 periods. The data in the instances are randomly generated and depends on the number of customers such as the production capacity, storage capacities (in both customer sites and production facility), number of vehicles available and setup cost. On average, when compared to basic GRASP and the GRASP with PR, reactive GRASP obtained more saving of about 1% and 0.3%, respectively, although the computation time is the worst among all the compared algorithms.

Boudia and Prins (2009) presented an alternative and improved algorithm, called the memetic algorithm with population management (MA|PM) to solve the PIDRP. MA|PM is known to be a recent way of tackling the diversity problem (Sörensen & Sevaux, 2006).

In this algorithm, an initial population is constructed by setting the total production quantities equal to the total demand over the planning horizon. Next, the authors constructed the delivery trips by using the Clarke and Wright saving algorithm (Clarke & Wright, 1964). Lastly, the adjustment of the production day is solved using Wagner and Whitin's method (Wagner & Whitin, 1958). After an initial population is created, the parents for a crossover were selected by the binary tournament method. A local search procedure is then performed to improve the generated solutions. The local search component used in this algorithm was similar to the GRASP of Boudia *et al.* (2007). The distance measure was also used to prevent duplicate solutions.

The algorithm is tested on a set of 90 benchmark problems (Boudia *et al.*, 2005) and compared with a two-phase heuristic (Boudia *et al.*, 2005) and GRASP (Boudia *et al.*, 2007). Testing showed that MA|PM can tackle large instances in a reasonable computational time. Besides, MA|PM outperforms the GRASP in all instances except for instance 10 consisting of 200customers-20period.

The PIDRP model considered in Bard and Nananukul (2009) includes a single factory and multiple customers with time-varying and deterministic demand over a finite planning horizon. The authors proposed a two-phase procedure focusing on reactive tabu search (RTS) algorithm for solving the problem.

In the first part of phase 1, an initial feasible solution is created by solving the allocation model in the form of mixed integer programming (MIP) with the absence of the routing component. The routing cost of the full model is approximated by the round trip cost (direct shipping). In the second part of phase 1, the capacitated vehicle routing problem (CVRP) subroutine is called to construct delivery routes based on the results obtained from the allocation model whilst the current solution is improved by a neighborhood search procedure in phase 2. The neighborhood structures in their algorithm consisted of swap and transfer moves. Swap involves two customers in two

consecutive periods and exchanges the maximum portion of delivery quantities between these two customers without violating any of the constraints. Whilst transfer tries to combine the full amount of the delivery quantity of a customer in a period with another delivery in a previous period. The customer is therefore eliminated from the delivery route in that period. The moves that can produce a better cost reduction are considered and infeasible solutions are not permitted.

In their implementation, the authors used a reactive tabu search approach, where the size of the tabu list is dynamic. An aspiration criterion that allows the tabu status to be overridden is also specified in their algorithm. The algorithm is tested on a set of 90 benchmark instances (Boudia *et al.*, 2005) and it increased savings by 10-20% when compared to the results obtained from GRASP (Boudia *et al.*, 2007), but with the greater computational runtime.

Armentano *et al.* (2011) considered a PIDRP with a single plant that produces multiple items delivered by a fleet of homogenous vehicles to a set of customers over a finite planning horizon. The authors developed two approaches of tabu search procedure for solving the problem. The first approach involves the construction phase and a short-term memory, while the second makes use of path relinking (PR) that is integrated with the first approach.

An initial solution created in the construction phase is improved in three steps:

a) Transfer an amount of delivered in a given period to either preceding or succeeding period;
b) Apply Clarke and Wright algorithm (Clarke & Wright, 1964) for the determination of the routes and
c) Apply Wagner and Whitin algorithm (Wagner & Whitin, 1958) to determine the production schedule over the planning horizon.

The neighborhood structure used is similar to the transfer move in Bard and Nananukul (2009) but it also considered the transfer of the maximum possible amount of delivery in a chosen period to be accumulated with another delivered quantity in some succeeding period. Then, a new production plan is determined and adjusted by using Evan's algorithm (Evans, 1985).

The path relinking procedure that is used to diversify the search is applied to the algorithm in the second approach. It is a tabu search with path relinking (TSPR). In this procedure, every tabu search local minimum is set as an initiating solution and linked with the farthest solution of a set of elite solutions found during the search as the guiding solution.

Both approaches, tabu search (TS) and TSPR are tested on generated instances with multiple items. Computational results showed that the two approaches can give the high-quality solutions, with a higher but acceptable computational runtime. Besides, the authors also tested both algorithms on a single item benchmark instances given by Boudia *et al.* (2005) and both TS and TSPR outperformed the MA|PM (Boudia *et al.*, 2009) and the RTS algorithm (Bard & Nananukul, 2009) in all instances.

Adulyasak *et al.* (2012) solve the PIDRP model by proposing the optimization-based adaptive large neighborhood search (Op-ALNS) heuristic. This heuristic was initially proposed by Ropke and Pisinger (2006) for solving the VRP problem.

First, in the initialization phase, initial solutions are created by solving two decomposed problems:

a) The production-distribution (PD) subproblem that is used to obtain a production, inventory and distribution plan, similar to the one in Bard and Nananukul (2009);
b) The routing (R) subproblem works with the determination of the delivery routes based on the results from PD subproblem.

The initial solution found is stored in the initial solution pool and another initial solution is generated with a different production setup by applying the local branching inequalities of Fischetti and Lodi (2003). Secondly, the authors proposed two sets of operators, namely the selection and transformation operators. The selection operator is used to construct a list of node candidates (customer-period combination). Whilst the transformation operator is applied to remove and reinsert node candidates and it is repeated until all candidates have been considered. Lastly, after the considered node candidates are inserted in the routes using cheapest insertion rule, the minimum cost flow (MCF) subproblem is executed to obtain the optimal allocations and inventories. The algorithm is repeated until a maximum number of iterations is reached.

The algorithm is tested on the benchmark instances of Archetti *et al.* (2011) and of Boudia *et al.* (2005). The computational results showed that Op-ALNS outperforms all the former heuristic approaches in all cases.

Moin and Yuliana (2015) developed a scatter search metaheuristic to solve the PIDRP with a single production site that produces a single item and distributes them to multiple customers by a fleet of homogenous vehicles. The aim of solving the problem is to minimize the total costs, without incurring any stockouts at the customer sites.

The algorithm consists of three phases. Phase 1 solves the allocation model, a simplified version of the full model to determine the optimal value of production quantity, inventories and amount delivered to each customer over the planning horizon, similar to the one proposed in Bard and Nananukul (2009). In phase 2, the authors constructed the routes based on the results from phase 1 using Giant Tour procedure (Imran *et al.*, 2009), sweep algorithm (Gillett & Miller, 1974) and saving algorithm (Clarke & Wright, 1964). In phase 3, two pairs of solution of the reference set (RefSet) are generated and the new combined solutions are also generated in solution combination method proposed by Torabi *et al.* (2009). Then, the solutions are improved

using updating the inventory level (forward and backward transfers), following by applying 1-0 exchange, 1-1 interchange and 2 opt*. Next, the reference set is updated and the existing solutions are replaced if the combined solutions are better than the incumbent. Phase 3 is repeated until a stopping criterion is met.

The scatter search algorithm is tested on a set of benchmark instances of Boudia *et al.* (2005). The algorithm is competitive when compared to the MA|PM of Boudia *et al.* (2007) and RTS of Bard and Nananukul (2009) but performs poorly when compared to Armentano *et al.* (2011) and Adulyasak *et al.* (2012).

2.3 Research Directions

With the advancement in technology, decision making processes are increasing complex and more encompassing, in the sense that more decision variables can be used to model complex situations and more input data and parameters are available to capture the complexity of the problems. Therefore the need for a more effective and better efficient algorithm is the focus of the current research.

It is well known that metaheuristics have provided extremely effective algorithmic schemes for solving hard optimization problems as demonstrated by the results reported in the literature. Recently, innovative optimization problems that are combinatorial, stochastic and continuous in nature have been proposed. These tools however focus on the exact solution of these problems. With this new development, contributions to the improvement of the state of art in optimization can be derived from the use of mathematics in metaheuristics. This development has given birth to the field of MatHeuristics. Since variable neighborhood search (VNS) has been proven to an effective metaheuristic, we combine with the mathematical programming to form MatHeuristic1 and we further modified the formulation to form MatHeuristic2.

The main aim of this research is to design and test the strength of the MatHeuristic algorithms. In this thesis, the PIDRP considered is similar to that of Bard and

Nananukul (2009), which involves a single production facility that produces a single item, has multiple customers with time-varying deterministic demand, over a finite planning horizon and delivers production using a fleet of homogeneous vehicles. The objective is to minimize the total costs (production setup, holding and routing costs) without incurring any stockouts at the customer sites.

We design two different types of two-phase approaches, within a MatHeuristic framework, namely MatHeuristic1 and MatHeuristic2, for solving the PIDRP. The major modification (compared to MatHeuristic1 algorithm) is the iterative procedure between MIP and VNS. In MatHeuristic2, the formulation of the MIP model is modified to include the routing component and the shaking step of VNS is embedded to include the additional three neighborhood structures proposed by Imran *et al.* (2009). The model is solved by using Concert Technology of CPLEX 12.5 Optimizers with Microsoft Visual C++ 2010 and tested on a set of benchmark instances (Boudia *et al.*, 2005). A few non-parametric tests are also performed on the results obtained to test the significant differences.

2.4 Description of Data Set

In the computational testing of both MatHeuristic algorithms, we used a data set provided by Boudia *et al.* (2005) consisting of 30 instances, each with 50, 100 and 200 customers problem with a 20-period planning horizon. The holding costs for production facility and customer sites are assigned at 1 and 0, respectively. These instances were randomly generated on a 100×100 Euclidean grid. All of the data sets have demands in every period and for each customer i, demand was uniformly distributed between 0 and the storage capacity $I_{max,i}^{C}$. The vehicle capacities are set at 8000 for instances with 50 and 100 customers and 12000 for instances with 200 customers. The production capacities are fixed at 50000, 120000 and 240000 for instances with 50, 100 and 200

customers, respectively. Every subset of instances has a limited number of vehicles available, with 5, 9 and 13 vehicles, respectively.

CHAPTER 3 INTEGRATED MATHEMATICAL PROGRAMMING AND METAHEURISTIC- MATHEURISTIC1 APPROACH

3.1 Introduction

In this study, a two-phase approach within a MatHeuristic framework, called MatHeuristic1 approach, is designed for the PIDRP model. In phase 1, a mixed integer program (simplified version of the original model, excluding the routing aspects) is solved to determine the production quantities to be delivered, amount delivered to customer sites, and inventories at both the production facility and customer sites. Whilst in phase 2, the delivery routes (or schedules) for each period are constructed based on the results obtained from phase 1 by using Giant Tour and Dijkstra's algorithm. A variable neighborhood search (VNS) is then performed to improve the initial solution. The VNS incorporates some aspects of Tabu Search to escape from local optima.

3.2 Problem Description

We consider a PIDRP model similar to the one proposed by Bard and Nananukul (2009). The problem consists of a single production facility and a set of customers located over a wide geographical region. At the production facility, a manufacturer produces a single item to supply the customers over a planning horizon which subdivided into periods or "days". If the production takes place in period t, then a setup cost is incurred. The number of items produced (daily production) is limited and the product may be either delivered to customer sites or stored at the production facility by incurring a unit holding cost. Each customer has a known deterministic, time varying and non-negative demand over the finite planning horizon. The number of items stored at customer sites is limited and a unit holding cost is incurred. Each customer can be visited at most once per day by a single vehicle. The items are delivered by a fleet

of homogeneous vehicles. Moreover, we consider the problem with the following assumptions:

- The initial inventories at the customer sites are set to be zero.
- All inventories (production facility and customer sites) are required to be zero at the end of the planning horizon.
- All deliveries take place at the beginning of the period and arrive on time to fulfill the demand.
- Shortages are not allowed.
- Transshipments between customers are not permitted. (Herer *et al.*, 2006)

The objective is to develop a coordinated operations plan, in order to minimize the overall cost (production setup, inventories and transportation costs) while fulfilling all customers' demands over the planning horizon.

3.3 Model Formulation

3.3.1 Notations

The following notations are used in the development of the mathematical formulation. Indices, parameters and the decision variables of the model are defined in Table 3.1, 3.2 and 3.3, respectively.

Table 3.1: Index definitions of the model

Symbol/s	Definition
i, j	Indices for customers, where 0 denotes the depot
t	Index for periods
N	Set of customers; $N_0 = N \cup \{0\}$ and $\|N\| = n$
T	Set of periods in the planning horizon; $T_0 = T \cup \{0\}$ and $\|T\| = \tau$

Table 3.2: Parameter definitions of the model

Symbol	Definition
d_{it}	Demand of customer i in period t
θ	Number of vehicles available
Q	Vehicle capacity
c_{ij}	Travel distance between customer i and j, where $c_{ij} = c_{ji}$ and the triangle inequality, $c_{ik} + c_{kj} \geq c_{ij}$, holds for any i, j, k with $i \neq j, k \neq i$ and $k \neq j$
C	Production capacity
f	Fixed production setup cost
h^P	Unit production setup cost
I^P_{max}	Maximum inventory level at the production facility
h^C_i	Unit inventory holding cost at customer site i
$I^C_{max,i}$	Maximum inventory level at customer site i

Table 3.3: Decision variable definitions of the model

Symbol	Definition
x_{ijt}	1 if customer i immediately precedes customer j on a delivery route in period t; 0 otherwise
y_{it}	Load on a vehicle immediately before making a delivery to customer i in period t
w_{it}	Amount delivered to customer i in period t
p_t	Production quantity in period t
I^P_t	Inventory at the production facility at the end of period t
z_t	1 if there is production; 0 otherwise
I^C_{it}	Inventory at customer sites i at the end of period t

3.3.2 Formulation of PIDRP Model

The PIDRP can be formulated as follows (model *P*)

$$\emptyset_{IP} = \min \sum_{t\in T}\sum_{i\in N_0}\sum_{j\in N_0} c_{ij}x_{ijt} + \sum_{t\in T} fz_t + \sum_{t\in T_0\setminus\{\tau\}} h^P I^P_t + \sum_{t\in T\setminus\{\tau\}}\sum_{i\in N} h^C_i I^C_{it} \qquad (1)$$

subject to:

$$I_t^P = I_{t-1}^P + p_t - \sum_{i \in N} w_{it} \quad , \forall t \in T_0 \tag{2}$$

$$I_{it}^C = I_{i,t-1}^C + w_{it} - d_{it} \quad , \forall i \in N, t \in T \tag{3}$$

$$\sum_{i \in N} w_{it} \le I_{t-1}^P \quad , \forall i \in N, t \in T \tag{4}$$

$$p_t \le C z_t \quad , \forall t \in T_0 \setminus \{\tau\} \tag{5}$$

$$p_0 \ge \sum_{i \in N} \left(d_{i1} - I_{i0}^C \right) \tag{6}$$

$$\sum_{\substack{j \in N_0 \\ j \ne i}} x_{ijt} \le 1 \quad , \forall i \in N, t \in T \tag{7}$$

$$\sum_{\substack{i \in N_0 \\ i \ne j}} x_{ijt} = \sum_{\substack{i \in N_0 \\ i \ne j}} x_{jit} \quad , \forall j \in N, t \in T \tag{8}$$

$$\sum_{j \in N} x_{0jt} \le \theta \quad , \forall t \in T \tag{9}$$

$$y_{jt} \le y_{it} - w_{it} + D_t^{\max} \left(1 - x_{ijt} \right) \quad , \forall i \in N, j \in N_0, t \in T \tag{10}$$

$$w_{it} \le D_{it}^{\max} \sum_{j \in N_0} x_{ijt} \quad , \forall i \in N, t \in T \tag{11}$$

$$0 \le I_t^P \le I_{\max}^P, \quad 0 \le I_{it}^C \le I_{\max,i}^C; \quad \forall i \in N, t \in T \setminus \{\tau\}; \quad I_\tau^P = I_{i\tau}^C = 0, \quad \forall i \in N \tag{12}$$

$$x_{ijt} \in \{0,1\}, \quad z_t \in \{0,1\}, \quad 0 \le y_{it} \le Q, \quad p_t \ge 0, \quad w_{it} \ge 0 \text{ and integer}, \ \forall i \ne j \in N_0, t \in T \tag{13}$$

where $D_{it}^{\max} = \min\left\{Q, \sum_{l=t}^{\tau} d_{il}\right\}$ and $D_t^{\max} = \min\left\{Q, \sum_{i \in N} \sum_{l=t}^{\tau} d_{il}\right\}$

The objective function (1) expresses the minimization of the sum of transportation costs, production setup costs, holding costs at the production facility and the holding costs at the customer sites. Constraints (2) and (3) represent the inventory flow balance equations for production facility and customers, respectively. The total amount available for delivery on day t is limited by the amount in inventory at the production facility in

period $t-1$ as formulated in (4). The inequality (5) limits production in period t to the capacity of the production facility, and (6) allows production in period 0. The bound in (7) ensures that if customer i is serviced in period t, then it must have a successor on its route, while the route continuity is enforced by (8). The inequality in (9) limits the number of vehicles that depart from the production facility in period t to the number of vehicles available θ. Constraint (10) keeps track of the load on the vehicles. The value of D_t^{max} is specified to be as small as possible while ensuring that (10) is feasible. The amount delivered to each customer is limited by the parameter D_{it}^{max} in (11). To conclude the formulation, the variable bounds are defined in (12) and (13).

3.4 Initial solution

An initial solution can be obtained by solving the mixed integer program (MIP) to get a set of feasible allocations for each period in the planning horizon. We modify the full model with the absence of routing components, i.e. the routing variable, x_{ijt} and the associated routing constraints (7)-(11). An additional constraint that defines an aggregated vehicle capacity constraint is introduced into the model. An alternative representation is needed for the cost term $\sum_t \sum_i \sum_j c_{ij} x_{ijt}$ because the actual transportation cost cannot be determined without the routing constraints. The formulation of MIP model without the routing components requires the following additional notations:

Table 3.4: Descriptions of the additional notations

Notation	Description
f_{it}^C	The fixed cost of making delivery to customer i on day t
e_{it}^C	The variable cost of delivering one item to customer i on day t
z_{it}^C	1 if delivery is made to customer i; 0 otherwise

As in Nananukul (2008), we divide the problem into two cases, which are problem instances with $n^2\tau > 500$ and $n^2\tau \leq 500$. In small instances where $n^2\tau \leq 500$, the routing costs on any period t are approximated by the cost of a round trip between the depot and customer i (round trip value= $2c_{i0}$), and we use a surrogate cost term $\sum_{it} f_{it}^C z_{it}^C$ with $f_{it}^C = 2c_{i0}$ for all i and t with e_{it}^C is set to zero. For the instances with $n^2\tau > 500$, we set $z_{it}^C = f_{it}^C = 0$ for all i and t, and the variable cost term $\sum_{it} e_{it}^C w_{it}$ is used to replace the routing component, where e_{it}^C is approximated by the ratio of the cost of making a delivery to customer i directly from the depot and the total demand of customer i in period t (i.e., $e_{it}^C = 2c_{0i}/d_{it}$).

Because in our instances, $n^2\tau > 500$, we set the variables, $z_{it}^C = f_{it}^C = 0$ and the allocation model is simplified as follows.

$$\emptyset_{IP} = \min \sum_{t\in T} f z_t + \sum_{t\in T}\sum_{i\in N} e_{it}^C w_{it} + \sum_{t\in T_0\setminus\{\tau\}} h^P I_t^P + \sum_{t\epsilon T\setminus\{\tau\}}\sum_{i\epsilon N} h_i^C I_{it}^C \qquad (1a)$$

subject to: (2), (3), (4), (5), (6), (12), (13) and an additional new constraint (14).

$$\sum_{i\in N} w_{it} \leq 0.8QK\,, \quad \forall t \in T \qquad (14)$$

The difference between the MIP model and the original model is that the routing variable, x_{ijt} and related constraints (7)-(11) in the full model have been eliminated and an additional constraint (14) has been introduced into the model. Constraint (14) limits the total amount that can be delivered in period t to a fixed percentage of the total transportation capacity. By limiting the maximum load to 80% of the total vehicle capacity, the author in Nananukul (2008) showed that the model always produces feasible solutions. Moreover, the actual transportation cost term

$\sum_{t\in T}\sum_{i\in N_0}\sum_{j\in N_0} c_{ij}x_{ijt}$ has been replaced by the approximated transportation cost $\sum_{t\in T}\sum_{i\in N} e_{it}^{c} w_{it}$.

3.5 Variable Neighborhood Search

3.5.1 Introduction

Variable Neighborhood Search (VNS) was initially proposed by Mladenovic and Hansen (1997) for solving combinatorial and global optimization problems. It works on the principle that different neighborhoods generate different search topologies and it is applied within a local search algorithm. Exploration of the search space is carried out by the local search which allows the algorithm to jump from one neighborhood to another. This allows the algorithm to escape from the local optimum.

3.5.2 Steps of VNS algorithm

Let us denote a finite set of pre-selected neighborhood structures by $\mathcal{N}_k$, where $k = 1, \ldots, k_{max}$, with k_{max} referring to the maximum number of neighborhoods used, and $\mathcal{N}_k(x)$ the set of solutions in the kth neighborhood of x. The stopping condition may be the maximum number of iterations, maximum number of iterations without improvement or the maximum CPU time allowed. There are three phases of the main VNS: *Shaking*, *Local Search*, and *Move or Not*. The basic VNS heuristic taken from Hansen and Mladenovic (2001) comprises the steps given in Figure 3.1. The flow chart of the VNS is outlined in Figure 3.2.

Initialization. Step 0: Define a set of neighborhood structures $\mathcal{N}_k$, $k = 1, \dots, k_{max}$, that will be used in the search and a set of local searches $\mathcal{R}_l$, $l = 1, \dots, l_{max}$; generate an initial solution x; choose a stopping condition;

Repeat the following steps until the stopping condition is met:

Step 1: Set $k \leftarrow 1$;

Step 2: Until $k = k_{max}$, repeat the following steps:

(a) **Shaking**. Generate a point x' at random from the k^{th} neighborhood of x ($x' \in N_k(x)$)

(b) **Local Search**. Apply some local search method with x' as initial solution; denote with x'' the so obtained local optimum;

(c) **Move or not**. If this local optimum is better than the incumbent, move there ($x \leftarrow x''$), and go back to (1); otherwise, set $k \leftarrow k + 1$.

Figure 3.1: General outline of VNS

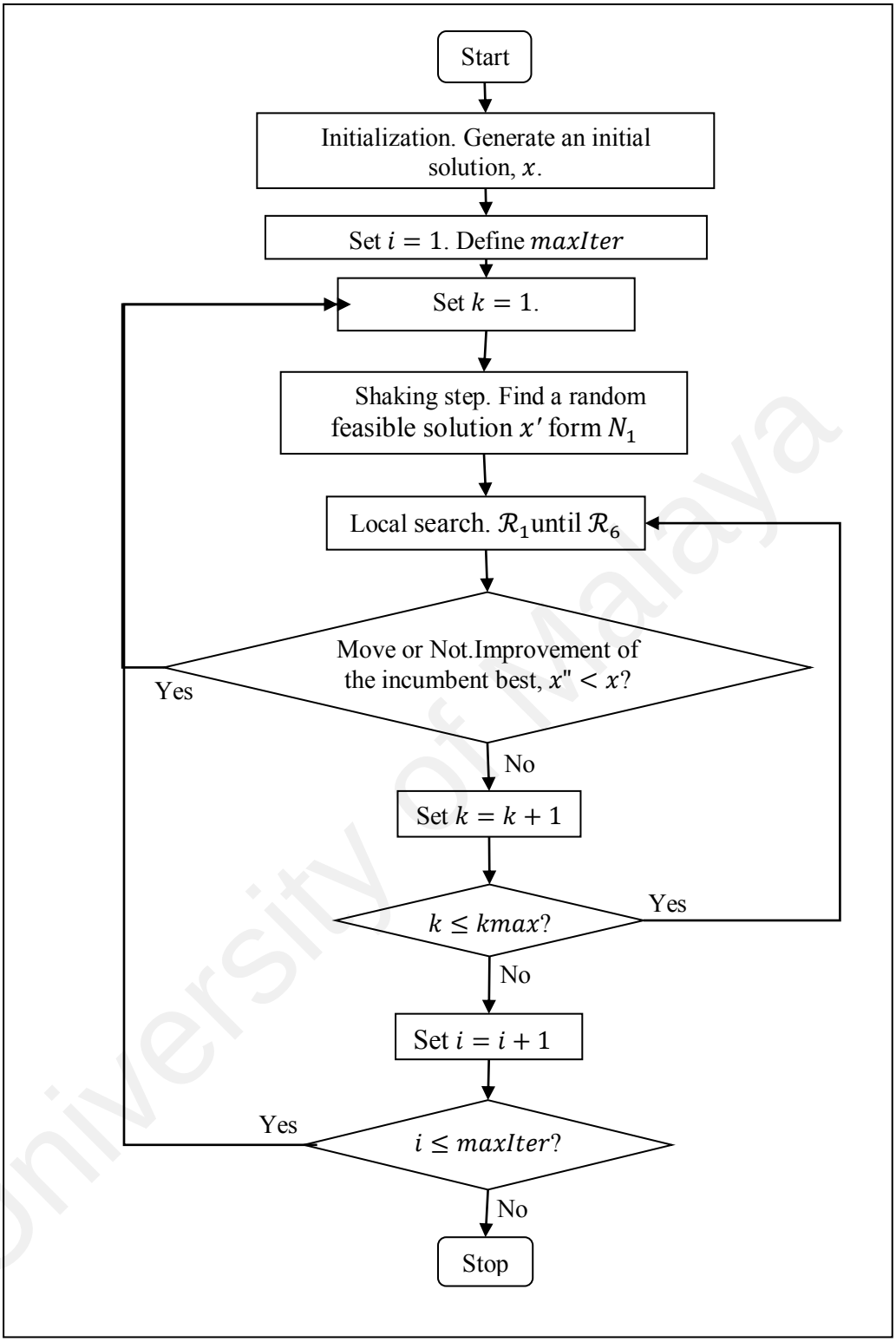

Figure 3.2: Flow chart of the VNS algorithm

3.5.2.1 Step 0: Initialization

The initial solution is obtained in two steps: (a) Construct a Giant Tour using the sweep algorithm as described in Gillett and Miller (1974); (b) Find the corresponding optimal fleet size by constructing a cost network and apply Dijkstra's algorithm, which provides an initial feasible solution that contains the routes.

First, a Giant Tour is constructed considering customers' allocations obtained in phase 1. We define a tour $G = (V, A)$ with $v = \{v_1, v_2, \dots, v_n\}$, the set of nodes representing all the customers' positions in the tour, while $A = \{(v_i, v_j): v_i, v_j \in V\}$ form the arcs which maintain the order of the customers' visitation sequence, together with a distance cost c_{ij}. Next, we define a path starting from the depot to the closest customer, and this step is repeated at each node $v_i \in V$ where $i = 1, 2, 3, \dots, n$ with n represent the total number of customers to be served in the current period, until the last customer is reached.

In order to apply Dijkstra's algorithm, we first construct a cost network considering customer data, capacity and distance constraints, and vehicle unit variable and fixed costs. For example, by illustration, we consider 12 customers making up the following Giant Tour $\sigma = (1,2,3,4,5,6,7,8,9,10,11,12)$ with customer demand d= (4,2,2,2,1,4,2, 2,2,1,4,2). The maximum capacity of a homogeneous vehicle is 10 units. Let c_{ij} be the distance between node i and node j. Let F denote the fixed vehicle cost and v the unit variable cost. Note that in our study, F and v are assumed to be 0 and 1, respectively.

The cost network is constructed by calculating the cost from the depot, denoted by 0, to customer 1 and from this customer to the depot (return journey) as the cost of arcs $0 - 1$ and is expressed as $C_{01} = F + v(2d_{01})$. If the total demand of both customers 1 and 2 does not violate the capacity constraint of the vehicle, we calculate

the cost of the arcs $1-2$ as $C_{02} = F + v(d_{01} + d_{12} + d_{20})$. We continue the construction of cost network until the entire vehicle is full, and then we start using the next vehicle. Figure 3.3 shows that the vehicle can only visit customers 1, 2, 3 and 4. The process is continued until there is no more arc connecting the last customer in the Giant Tour. In general, the cost of arc ij is stated as

$$C_{ij} = F + v\left(d_{0,i+1} + \sum_{k=i+1}^{j-1} d_{k,k+1} + d_{j,0}\right) \tag{15}$$

After creating the cost network, whose origin is depot'0' and the destination is the last node in the Giant Tour, Dijkstra's algorithm is applied to obtain the initial feasible solution. This procedure is repeated for each period considered. After an initial feasible solution is found, set $x_{best} \leftarrow x$ and proceed to Step 1.

Defining the stopping condition can differ from one algorithm to another. Most of them choose the maximum number of iterations as the stopping condition, as in our study. Other criteria can also be defined in the algorithm such as maximum running time or CPU time allowed, or number of iterations between two improvements.

3.5.2.2 Step 2(a): Shaking Step

In this step, a solution x' is picked randomly from the k^{th} neighborhood of the current solution x. This will ensure that the solution is not far from the current best solution x. We consider three moves: Forward transfer, backward transfer and swap for the VNS. The algorithm of the shaking step is shown in Figure 3.4. The detailed explanations of the neighborhood structures used in the shaking step by illustrative examples are given in the Section 3.6.

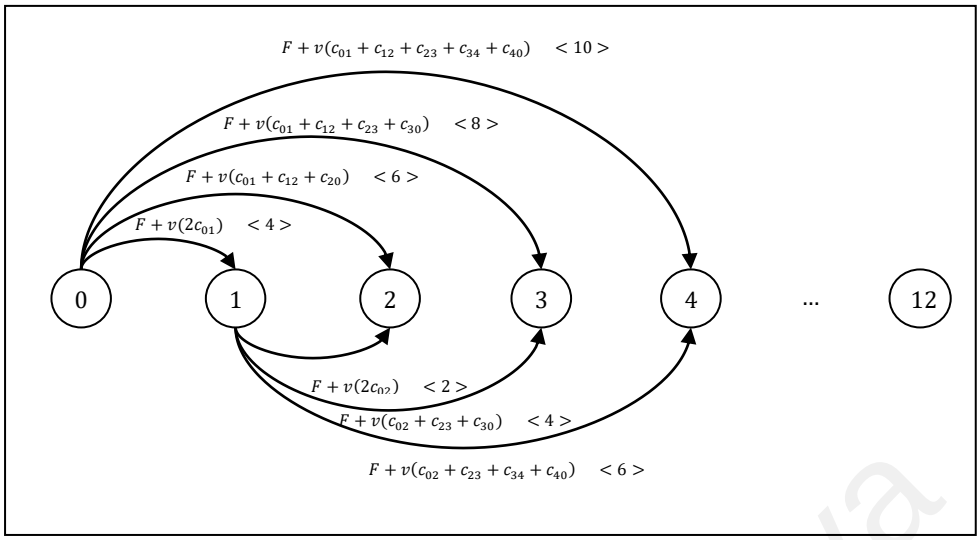

Figure 3.3: Construction of the cost network

```
Set num=1
While (num<=k) do    //the number of changes depends on the value of k
{
        Randomly generate the value of r where 0<r<1. Define p1, p2 and p3
            if (r<=p1)      // the value of p1 represents the chosen probability
                Apply forward transfer
            elseif (p1<r<=p2)
                Apply backward transfer
            elseif(p2<r<=p3)
                Apply swap
            else
                 Apply transfer
            endif
            num=num+1
}
```

Figure 3.4: The algorithm of the shaking step

3.5.2.3 Step 2(b): Local Search

In our study, the local search consists of six refinement procedures adopted from Imran *et al.* (2009). The order of the local searches is as follows: the 1-insertion inter-route as $\mathcal{R}_1$, the 2-opt inter-route as $\mathcal{R}_2$, the 2-opt intra route as $\mathcal{R}_3$, the swap intra route as $\mathcal{R}_4$, the 1-insertion intra-route as $\mathcal{R}_5$, and finally the 2-insertion intra-route as $\mathcal{R}_6$. The process starts by generating a random feasible solution x' from N_1, to be used as a temporary solution. Then, the multi-level approach starts by finding the best solution x"

using $\mathcal{R}_1$. If x" is better than x', then $x' = x$" and the search returns to $\mathcal{R}_1$, otherwise the next refinement procedure, $\mathcal{R}_2$ is applied. This process is repeated until $\mathcal{R}_6$ cannot produce a better solution.

3.5.2.4 Step 2(c): Move or Not

If the solution x" obtained by the multi-level approach is better than the incumbent best solution x, then we set $x = x$" and the search returns to N_1. However, if x" worse or the same as x, we generate x' from the next neighborhood, say $N_{k+1}(x)$, and return to step 2(b). The process is repeated until the search reaches $N_{k_{max}}$.

3.6 Neighborhood Structures

We consider three neighborhood structures for each N_k: Forward transfer, backward transfer and swap. Examples of a forward transfer, backward transfer and a swap are given in section 3.6.1, 3.6.2 and 3.6.3, respectively. In these three examples, the demand d_{it} of the customers in period 1 and 2 are given in Table 3.2. The holding cost, h_i of customers 1 to 5 are 5, 2, 3, 2 and 4, respectively. It is also assumed that the vehicle capacity $Q = 25$. Note that the routings are separated by zeros and the route costs are calculated by summing up the individual link distance cost, c_{ij} where $c_{01} = 16, c_{02} = 25, c_{03} = 17, c_{04} = 35, c_{05} = 22, c_{13} = 5, c_{15} = 18, c_{23} = 8, c_{45} = 9$ and $c_{ij} = c_{ji}$ for all i and j, i.e. $c_{01} = c_{10} = 16$. For example, the total route cost of the first route in period 1 in Figure 3.5 is $c_{02} + c_{23} + c_{31} + c_{10} = 54$.

Table 3.5: Demands of customers in period 1 and 2

Period/Customer	1	2	3	4	5
1	4	9	6	7	10
2	6	12	11	8	7

We note that the feasibility is maintained for all the transfers and the swap.

3.6.1 Forward Transfer

Figure 3.5 depicts an example of a forward transfer. The main aim of the transfer is to reduce the inventory holding cost while achieving a balance between the inventory and the transportation costs.

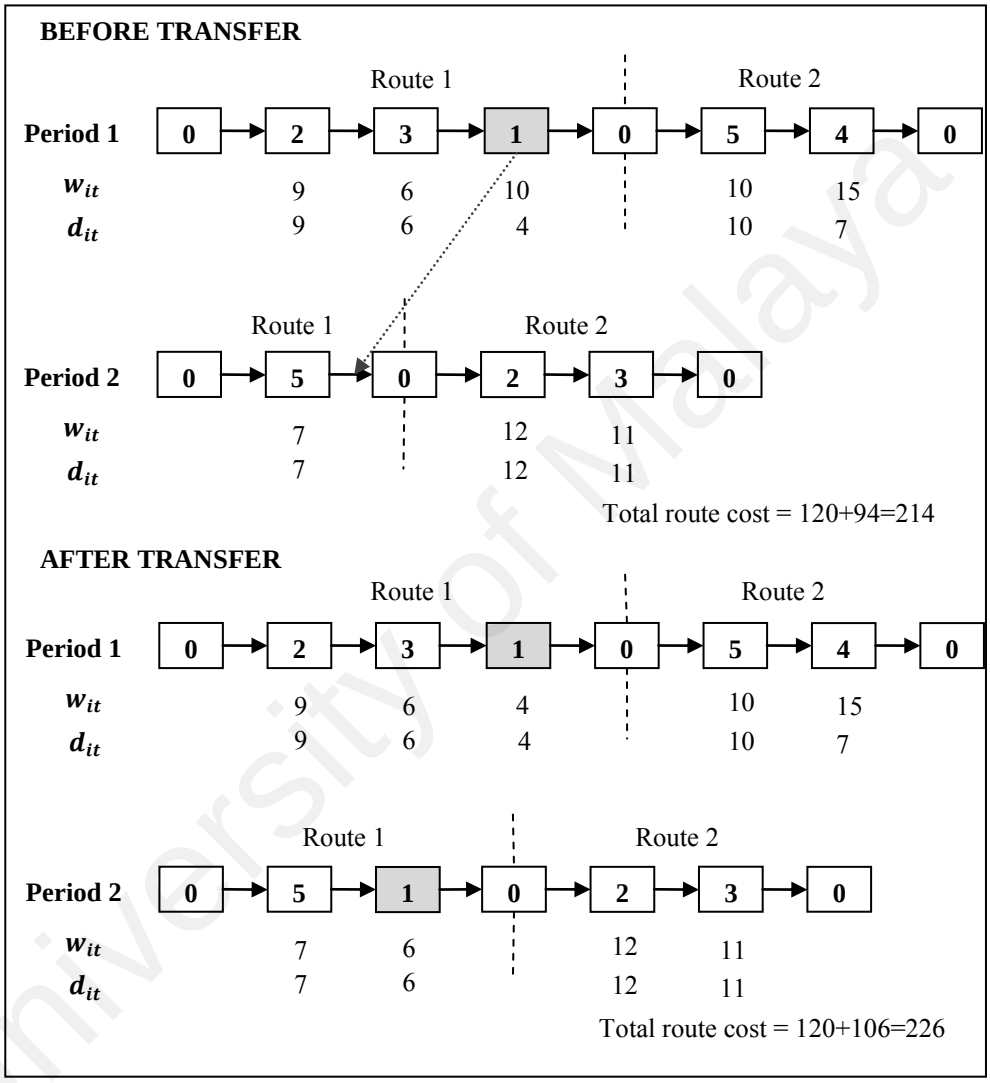

Figure 3.5: Example of a forward transfer

In forward transfer, the selection of period and customer to be transferred is biased towards customers with high holding cost. In this example, we select customer 1 in period 1. Before the transfer, the amount of delivery for customer 1 are 4 and 6 in period 1 and 2 respectively. Because customer 1 is not visited in both routes in period 2,

we apply forward transfer by inserting customer 1 into period 2 based on the best insertion. Note that, in this example, we insert customer 1 into route 1 instead of route 2 because insertion to route 2 causes violation of vehicle capacity constraint. Note that if we insert customer 1 before customer 5 would get the same result, i.e. $0-1-5-0$. The amount to be delivered to customer 1 is the maximum number of items that can be transferred to period 2 without causing a shortage, which is 6 in this case since customer 1 has a demand of 4 in period 1 and the current delivered quantity is 10.

After the forward transfer, the route cost after the insertion of customer 1 is increased from 214 to 226. The holding cost is reduced by 20 (6 units with $h_1 = 5$). The overall savings after the transfer is $30 - 12 = 18$.

3.6.2 Backward Transfer

Figure 3.6 presents an example of a backward transfer. In the backward transfer, preference is given to the customers with lower holding costs in order to determine whether the transportation and the inventory holding cost can be further consolidated. In this example, we select customer 4 in period 2.

Before the transfer, the amount delivered to customer 4 is 7 and 8 units in periods 1 and 2, respectively. The transfer of full amount of 8 items in period 2 to period 1 eliminates the delivery in period 2. The move is feasible since the aggregated amount (amount to be transferred + existing delivery quantity) does not violate the vehicle capacity constraint. The transfer increases the holding cost of customer 4 from 0 to 16. The transportation costs reduced from 236 to 214 gives the overall savings of 22-16=6.

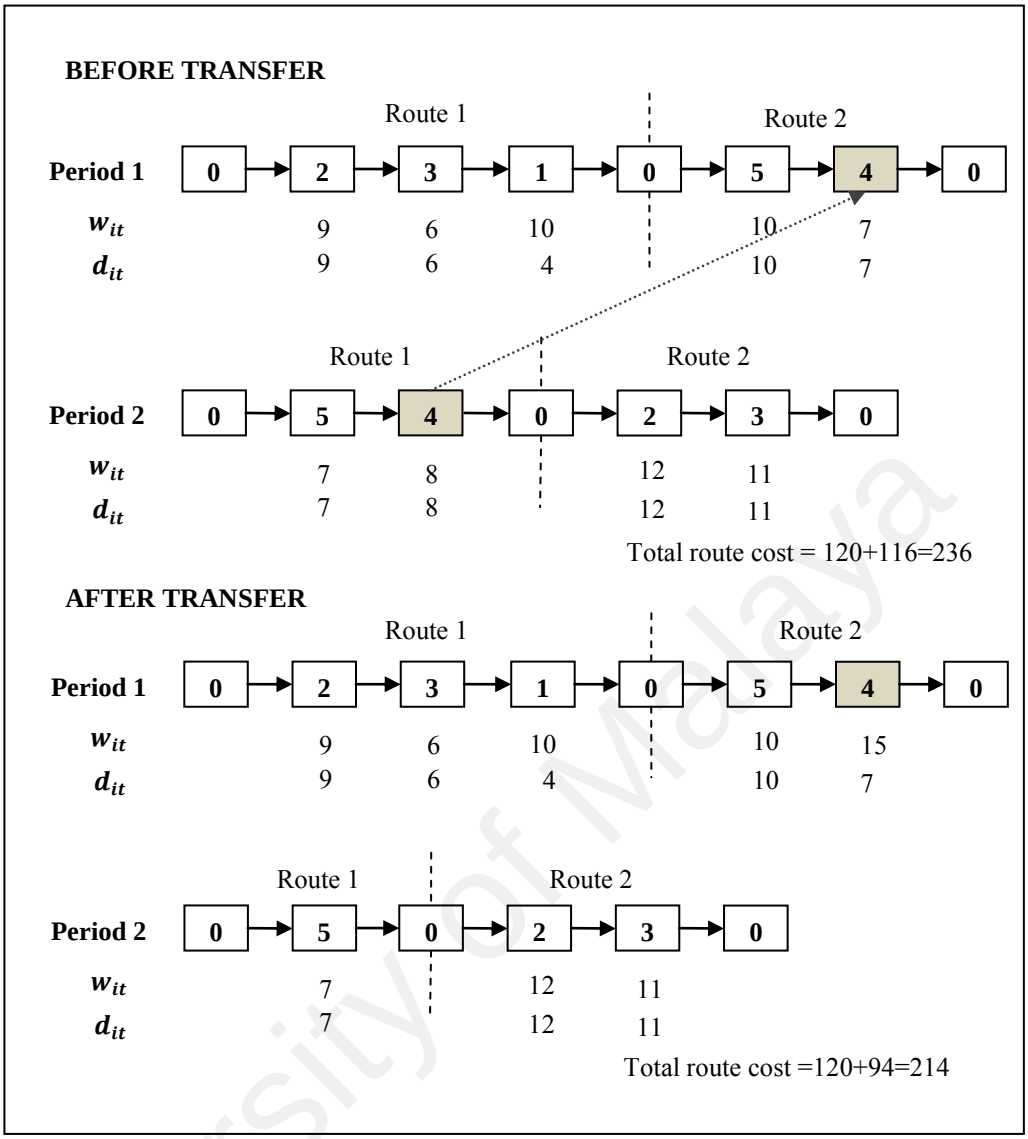

Figure 3.6: Example of a backward transfer

3.6.3 Swap

In swap, the delivery quantity of a customer i_1 in period t_1 ($w_{i_1 t_1}$) is to be exchanged with the delivery quantity of customer i_2 in period t_2 ($w_{i_2 t_2}$), where t_2 is the next period of t_1 and $w_{i_2 t_2} > 0$. Figure 3.7 shows an example of a swap move. In this example, we choose customer 1 as i_1 in period 1 ($t_1 = 1$) and customer 4 as i_2 in period 2 ($t_2 = 2$).

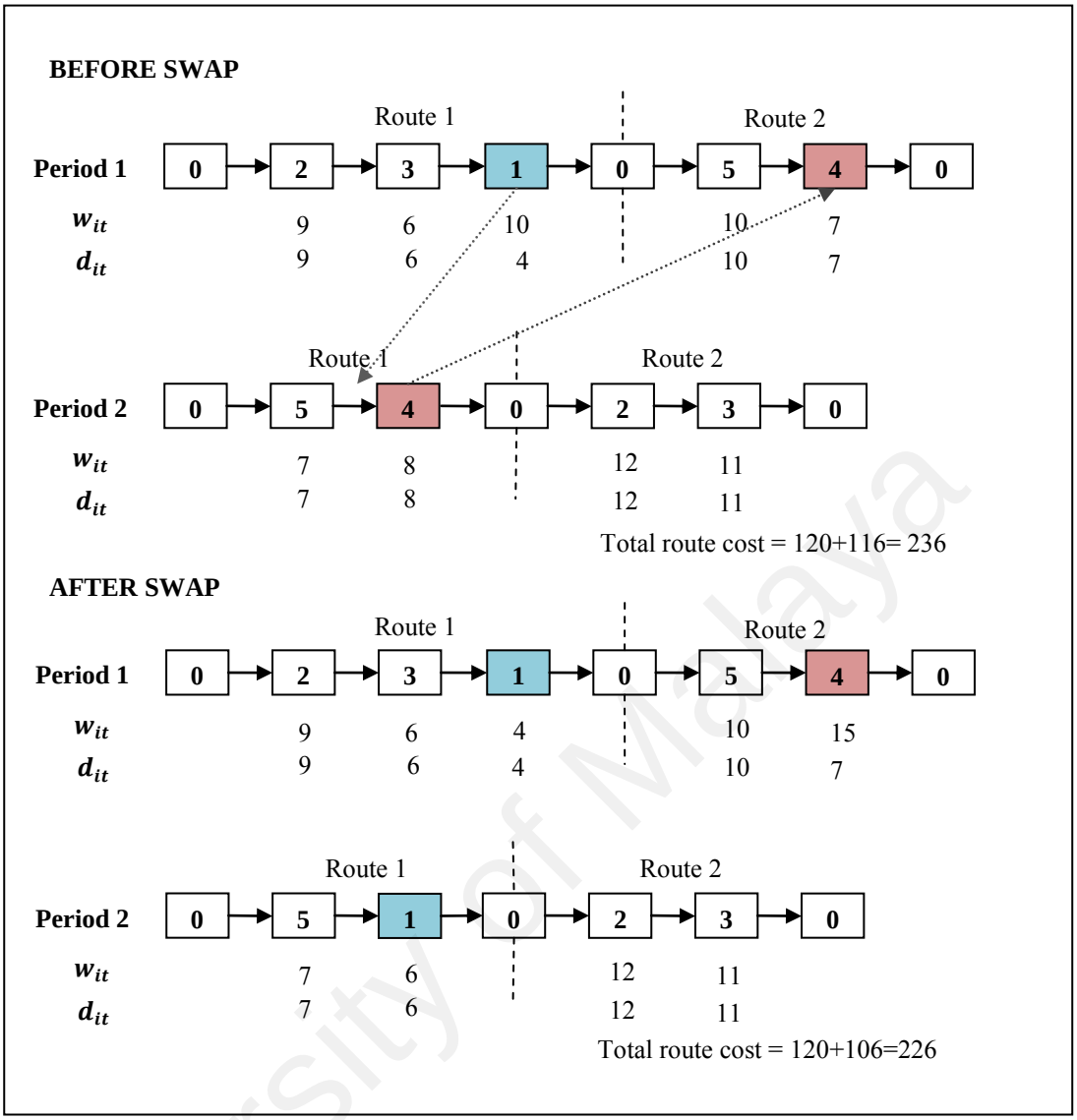

Figure 3.7: Example of a swap

Before the swap, the current quantity delivery of customer 1 is 10 units in period 1, while the existing amount deliveries of customer 4 are 7 and 8 items in period 1 and 2, respectively. For customer 4, the swap considers the maximum amount of delivery quantity, which is 6 in this case, that can be transferred to period 2 without causing a shortage in period 1, whilst for customer 1, the full amount of 8 items are transferred to period 1. Since customer 1 was not visited in period 2, it must be inserted into one of the K routes with the checking of violation of vehicle capacity constraint. In this example, we insert customer 4 into route 1 instead of route 2. The holding costs

decreased by 30 (6 units×5) for customer 1 and increased by 16 (8 units×2). Including the transportation cost, we obtain our overall savings of $30 - 16 + 10 = 24$.

3.6.4 Tabu Search Properties and Feasibility

In this study, we also incorporate the concept of tabu search, which forbids the movement of a customer for a few iterations if the customer is chosen in the transfers or swap. In all the three moves, only moves that produce feasible solutions are allowed, so it is necessary to check for violations of the production constraints and the inventory bounds at the plant and the customer sites, as well as the vehicle capacity constraint.

3.7 Local Searches

In this study, we use six local searches, namely 1-insertion inter-route, 2-opt inter-route, 2-opt intra route, swap intra route, 1-insertion intra-route, and 2-insertion intra-route, to improve the current solution, x' obtained from the shaking step. The descriptions and the corresponding examples of the local searches are presented in the section 3.7.1-3.7.6.

3.7.1 1-Insertion Inter-Route

We select a customer i from the route in the period to be shifted from its current position and insert into another route within the same period. If this insertion does not violate any constraints and reduces the routing cost, then the selected customer is shifted. Figure 3.8 shows an example of 1-insertion inter-route.

3.7.2 2-Opt Inter-Route

In 2-opt intra-route, we select two different edges from two different routes and then delete and reconnect them to one another. If the resulting routes are feasible, i.e. do not violate any constraints, and routing cost is lower than the previous cost, then the move is accepted and performed. Figure 3.9 shows an example of 2-opt inter-route.

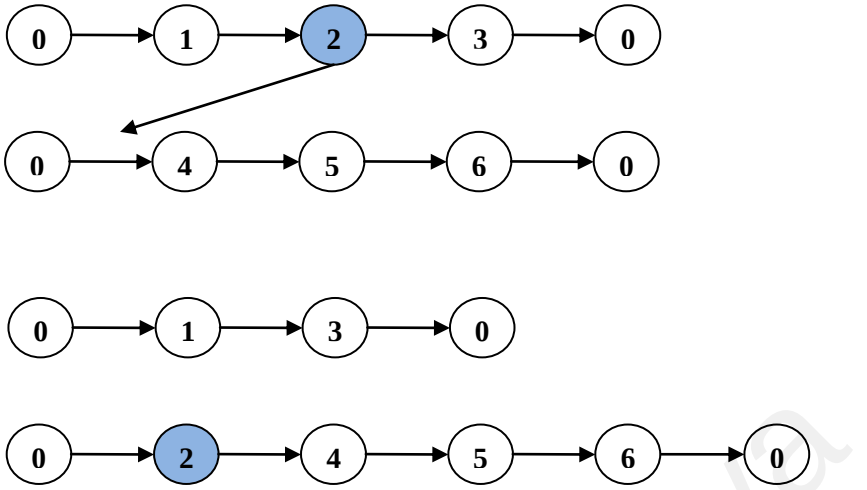

Figure 3.8: Example of a 1-insertion inter-route

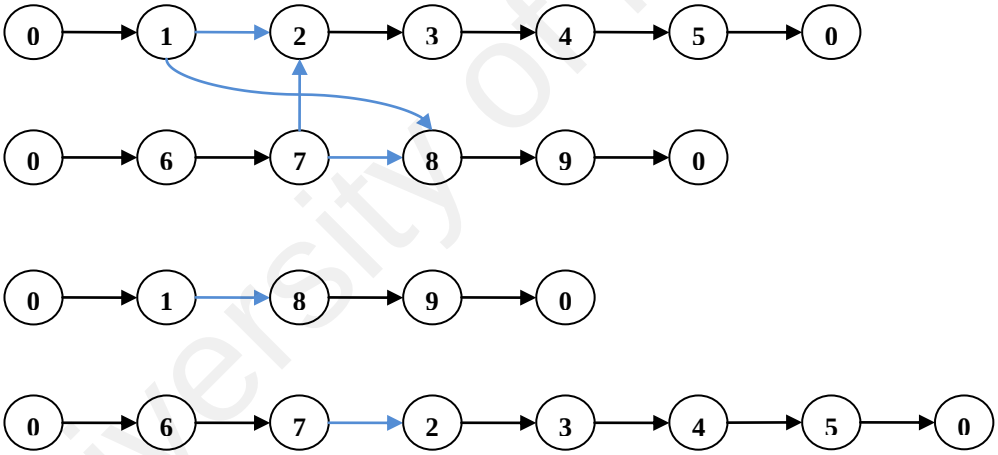

Figure 3.9: Example of a 2-opt inter-route

3.7.3 2-Opt Intra-Route

2-opt intra-route works by removing two non-adjacent arcs and adding two new arcs within the same route while maintaining the tour structure. A move is accepted and applied if the resulting total cost is lower than the previous total cost. The process is continued until no further improvement can be found. Figure 3.10 presents an example of 2-opt intra-route.

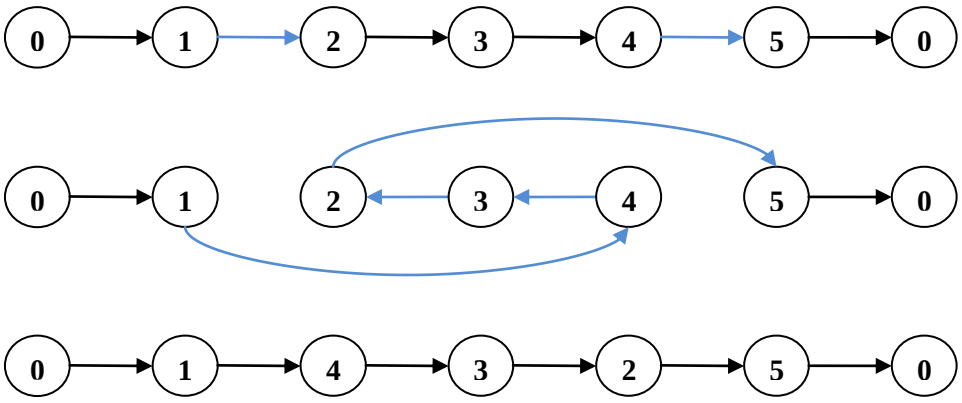

Figure 3.10: Example of a 2-opt intra-route

3.7.4 Swap Intra-Route

Swap intra-route swaps positions of a pair of customers within a route. The move is accepted if the routing cost is reduced. Figure 3.11 shows an example of swap-intra-route.

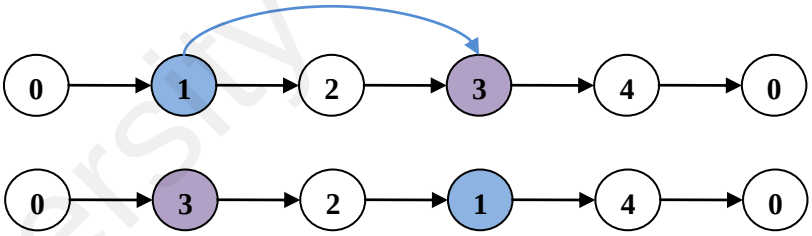

Figure 3.11: Example of a swap intra-route

3.7.5 1-Insertion Intra-Route

1-insertion intra-route removes a customer from its current position in a route and try to insert the customer elsewhere within the route. The move is applied if it produces a better routing cost. Figure 3.12 shows an example of 1-insertion intra-route.

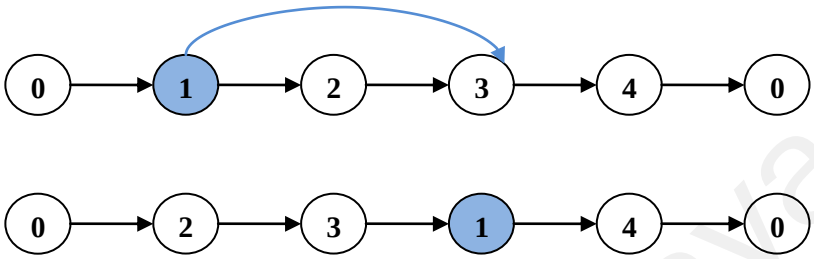

Figure 3.12: Example of a 1-insertion inter-route

3.7.6 2-Insertion Intra-Route

2-insertion intra-route remove two consecutive customers and insert them elsewhere within a route in order to produce a better routing cost. Figure 3.13 presents an example of 2-insertion intra-route.

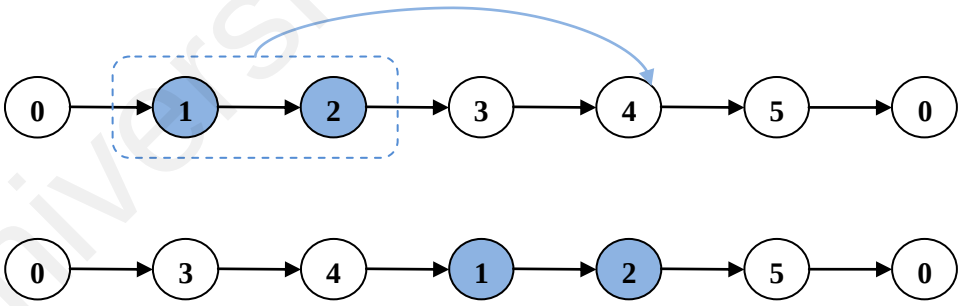

Figure 3.13: Example of a 1-insertion inter-route

3.8 Algorithm Description

Figure 3.14 summarizes the important steps of the MatHeuristic1 algorithm. In phase 1, the optimal allocations (delivery quantities) and inventories are obtained from solving the MIP model (described in Section 3.4.1). An initial feasible solution is created using Giant Tour procedure, Dijkstra's algorithm and the amount delivered to each customer, w_{it} obtained from phase 1, which is the initialization phase in VNS (Phase 2). Then the solution is saved as both the *curr_solution* and the *best_solution*. Then the algorithm iterates until a defined *maximum_iteration* is reached. In each iteration, we input *curr_solution* into shaking step in VNS (described in Section 3.5.2.2) to find a new feasible solution using forward transfer, backward transfer and swap. A move (whether a transfer or a swap) is considered feasible if it does not violate any constraint and it is not in the tabu list. Once the best move is found and production levels, as well as tabu list, are updated, we save it as *curr_solution* and input it into the local search in VNS (described in Section 3.5.2.3) in order to improve the routes. Next, in the 'Move or not' stage (described in Section 3.5.2.4), we save the solution as *best_solution* if *curr_cost* is less than *best_cost* and k is reset to 1. Otherwise, we continue the next iteration with *best_solution* with the increment of k by 1. Table 3.6 gives the description of the parameters and data structures used in the algorithm.

Table 3.6: Parameters and data structures used in the algorithm

Parameter/Data Structure	Description
iteration_number	number of current iteration
maximum_iteration	maximum number of iterations
curr_solution	current solution
best_solution	best solution
curr_cost	cost of current solution
best_cost	cost of best solution
best_move	best move of all feasible transfer or swap
tabu_list	stores a list of moves that are forbidden
tabu_size	number of iterations a move is held on the *tabu_list*

Step 1: Phase 1: Solve MIP model to obtain optimal w_{it}, I_{it}^{C}, I_{t}^{P} and p_t.

Step 2: Phase 2: [Initialization in VNS]
For(all days $t \in T$){
For each day t, apply Giant Tour procedure and Dijkstra's algorithm to construct the delivery routes obtained from Step 1, for all customers $i \in N$such that w_{it}>0.
}
In this step, an initial feasible solution is created and saved both as *curr_solution* as well as $best\_solution$.

Step 3: Calculate the cost of *curr_solution* and set *best_cost=curr_cost*.

Step 4: Define *maximum_iterations*, k_{max} and *tabu_size.* Initialize *tabu_list* to empty set.

Step 5: Phase 2: [Shaking Step, Local Search and Move or Not in VNS]
Do{
Set $k = 1$
Do{
Shaking Step (refer to algorithm in Figure 3.4) with feasibility checking and the move is not on *tabu list*
execute *best_move*
update *tabu_list* and adjust production levels

Local Search (refer to Section 3.3.2.3)

Move or Not
if(*curr_cost*<*best_cost*){
best_cost=curr_cost
best_solution=curr_solution
$k = 1$
}else{
$k = k + 1$
}
}While($k \leq k_{max}$)
}While(*iteration_number*<*maximum_iteration*)

Figure 3.14: MatHeuristic1 algorithm

3.9 Computational Results

All the algorithms were written in Microsoft Visual C++ 2010 and performed on 3.1 GHz processor with 8GB of RAM. The code for the allocation model was implemented as mixed integer programming using the Concert technology of Microsoft Visual Studio 2010 linked to the CPLEX 12.5 libraries. CPU times were obtained using the time function in C++.

3.9.1 Data

3.9.1.1 Data Set of Boudia *et al.* ($h_i^C = 0$)

The data sets containing 30 instances of 50, 100 and 200 customer problem, all over the planning horizon of 20 periods (Boudia *et al.*, 2005) are used in the computational testing of MatHeuristic1. Detail description of the data set was given in Section 2.4.

3.9.1.2 Randomly Generated Data Set ($h_i^C > 0$)

Since the holding cost of each customer site in Boudia's data set is set to zero, we generate another data set of 14 instances with $h_i^C > 0$ to test the performance of MatHeuristic1. The data set comprises of 12, 20, 50 and 100 customers with 5, 10, 14 and 21 periods. The locations of the customers are generated randomly on a 100×100 Euclidean grid. The locations for the 20 customers are extended from those for the 12 customers instance by adding 8 new randomly generated customers. Similarly, the locations for 50 customers are extended from those for the 20 customers by generating randomly 30 additional customers and an additional 50 customers' locations are generated randomly in the case of 100 customers instance. All of the data sets have demands in every period, except for the cases of 50 customers. The holding cost of each customer is generated randomly within the range [1, 10] while the demands are generated randomly within the range [0, 50]. The vehicle capacity is fixed at 100 and the depot is located at (0, 0) for all data sets.

3.9.2 Results and Discussions

3.9.2.1 Results of Data set of Boudia *et al.*

Tables 3.7, 3.8 and 3.9 present the total costs of our algorithm, MatHeuitic1 algorithm compared to MA|PM (Boudia & Prins, 2009), RTS (Bard & Nananukul, 2009) and SS (Moin & Yuliana, 2015) for all the instances with 50, 100 and 200 customers, respectively over a planning horizon of 20 periods. MA|PM was

implemented in the Pascal-like programming language Delphi 7 and tested on a 2.8 GHz PC under Windows XP (Boudia & Prins, 2009). RTS was implemented in Java Netban 4.1, linked to the CPLEX 8.1 libraries and run on a 2.53 GHz processor with 512 MB of RAM (Bard & Nananukul, 2009). SS was written in Matlab 7.7 and run on a 3.1GHz processor with 8GB of RAM (Moin & Yuliana, 2015).

In Table 3.7, 3.8 and 3.9, columns 2, 3 and 4 give the best solutions for MA|PM, RTS and SS, respectively. Column 5 gives the best cost of our algorithm. Columns 6, 7 and 8 report the percent deviation (see equation (16)) of our best cost from the best MA|PM, RTS and SS solutions, respectively. The total cost in bold indicates that the cost is the lowest (best) among all the compared algorithms.

$$\text{Percent deviation} = \frac{\text{Total cost of the compared algorithm} - \text{total cost of MatHeuristic1}}{\text{total cost of MatHeuristic1}} \times 100 \quad \ldots (16)$$

Table 3.7 concerns the benchmark problems with 50 customers. Our solutions are 4.70% and 3.15% better on average than the two best solutions, MA|PM and SS and 1.34% worse than the solutions from RTS. Besides, our algorithm outperforms the comparison algorithms in 8 instances. Post-hoc test (see Section 3.10.2) is conducted and the result shows that our algorithm is significantly better than MA|PM and performs equally well as RTS and SS.

Table 3.8 presents the comparisons of our algorithm with MA|PM, RTS and SS for the case of N100T20. Our algorithm outperforms all the compared algorithms in 23 instances. In addition, our solutions are 3.19%, 2.85% and 3.48% better on average than MA|PM, RTS and SS, respectively.

Table 3.9 shows the results of the benchmark problems with 200 customers with 20 periods. Our algorithm is on average 3.37% and 0.51% better than the two solutions, RTS and SS, respectively, but only 0.02% worst off when compared to MA|PM. Besides, our algorithm outperforms the comparison algorithms in 7 instances. We can conclude that our algorithm performs better than RTS and performs equally well as SS and MA|PM (see post-hoc test in Section 3.10.2).

Table 3.7: Results of N50T20

Instance	Total Cost				Percent Deviation		
	MA|PM	RTS	SS	MatH1	MA|PM	RTS	SS
1	378378	398795	**373172**	374264	-1.10	-6.55	0.29
2	403913	**373374**	380129	382679	-5.55	2.43	0.67
3	409573	**353058**	369355	374059	-9.49	5.61	1.26
4	399220	**361176**	419256	377022	-5.89	4.20	-11.20
5	422279	**364819**	410140	372027	-13.51	1.94	-10.24
6	407122	**368082**	372717	386968	-5.21	4.88	3.68
7	414977	396963	**366617**	377495	-9.93	-5.16	2.88
8	379744	**370822**	399269	383376	0.95	3.27	-4.15
9	407935	**379379**	416811	383931	-6.25	1.19	-8.56
10	396258	370655	421940	**369095**	-7.36	-0.42	-14.32
11	402475	**354025**	382085	367246	-9.59	3.60	-4.04
12	358702	**354981**	365139	370920	3.29	4.30	1.56
13	371030	**365432**	398346	372797	0.47	1.98	-6.85
14	406114	**363404**	386252	378460	-7.31	3.98	-2.06
15	373076	**367659**	406994	385600	3.25	4.65	-5.55
16	379404	360534	**360433**	367923	-3.12	2.01	2.04
17	406353	398442	400878	**389281**	-4.39	-2.35	-2.98
18	401179	368533	**365400**	381641	-5.12	3.43	4.26
19	406893	377073	392953	**373799**	-8.85	-0.88	-5.12
20	398508	372141	381826	**371942**	-7.14	-0.05	-2.66
21	397112	374743	408826	**367850**	-7.95	-1.87	-11.14
22	358749	**347329**	355774	360679	0.54	3.70	1.36
23	407369	**362619**	398696	375067	-8.61	3.32	-6.30
24	369784	375022	369717	**364765**	-1.38	-2.81	-1.36
25	411556	**374682**	414254	384851	-6.94	2.64	-7.64
26	408704	366167	**363157**	382984	-6.72	4.39	5.18
27	366197	375261	387595	**365470**	-0.20	-2.68	-6.05
28	401032	373155	416631	**365761**	-9.64	-2.02	-13.91
29	384282	379320	**371621**	389899	1.44	2.71	4.69
30	369959	369223	**364819**	371576	0.44	0.63	1.82

Table 3.8: Results of N100T20

Instance	Total Cost				Percent Deviation		
	MA|PM	RTS	SS	MatH1	MA|PM	RTS	SS
1	714401	711671	714768	**684370**	-4.39	-3.99	-4.44
2	722047	**694694**	706531	695179	-3.86	0.07	-1.63
3	**677598**	683270	702056	679257	0.24	-0.59	-3.36
4	710552	718252	715081	**659536**	-7.74	-8.90	-8.42
5	733040	731260	709207	**706692**	-3.73	-3.48	-0.36
6	696146	744927	704383	**693958**	-0.32	-7.34	-1.50
7	705322	695728	701951	**677788**	-4.06	-2.65	-3.56
8	679210	706058	709180	**670607**	-1.28	-5.29	-5.75
9	699518	705035	721338	**689304**	-1.48	-2.28	-4.65
10	705778	696521	721714	**688089**	-2.57	-1.23	-4.89
11	709122	711895	705973	**700952**	-1.17	-1.56	-0.72
12	755726	**703162**	747220	706511	-6.97	0.47	-5.76
13	695466	721066	707786	**692837**	-0.38	-4.07	-2.16
14	718260	698548	728132	**697085**	-3.04	-0.21	-4.45
15	736041	711506	729536	**680391**	-8.18	-4.57	-7.22
16	715209	714873	709663	**706658**	-1.21	-1.16	-0.43
17	737832	**702314**	702391	716302	-3.01	1.95	1.94
18	723413	720238	727677	**682024**	-6.07	-5.60	-6.69
19	720218	748734	728944	**693967**	-3.78	-7.89	-5.04
20	724727	729099	738669	**715222**	-1.33	-1.94	-3.28
21	724328	738746	712796	**700373**	-3.42	-5.48	-1.77
22	**701506**	702849	717464	706492	0.71	0.52	-1.55
23	710033	712717	723209	**680172**	-4.39	-4.78	-6.33
24	734327	727741	726290	**696555**	-5.42	-4.48	-4.27
25	725446	725869	730997	**695738**	-4.27	-4.33	-5.07
26	718939	700719	722588	**689543**	-4.26	-1.62	-4.79
27	715068	686382	719054	**684183**	-4.51	-0.32	-5.10
28	**685117**	700980	695956	685277	0.02	-2.29	-1.56
29	722571	725030	717728	**699233**	-3.34	-3.69	-2.65
30	721850	**698942**	701024	702675	-2.73	0.53	0.23

Table 3.9: Results of N200T20

Instance	Total Cost				Percent Deviation		
	MA|PM	**RTS**	**SS**	**MatH1**	**MA|PM**	**RTS**	**SS**
1	**996151**	1030684	1033864	1005064	0.89	-2.55	-2.87
2	**978373**	1010158	995600	1008993	3.03	-0.12	1.33
3	986147	1016681	**983290**	996844	1.07	-1.99	1.36
4	**962937**	1042854	1004710	994712	3.19	-4.84	-1.01
5	**970638**	1023680	1004463	999114	2.85	-2.46	-0.54
6	**965646**	1025262	967194	1006016	4.01	-1.91	3.86
7	**980562**	1038746	1011310	1008156	2.74	-3.03	-0.31
8	1014809	1066068	1001545	**992901**	-2.21	-7.37	-0.87
9	**967738**	1018420	984520	987864	2.04	-3.09	0.34
10	1093230	1035240	1024832	**999130**	-9.42	-3.61	-2.57
11	1008080	1037705	**981277**	996692	-1.14	-4.11	1.55
12	998951	1035350	**995275**	1002124	0.32	-3.32	0.68
13	**984918**	1063024	1028545	1008080	2.30	-5.45	-2.03
14	**964301**	1024491	1002466	1002613	3.82	-2.18	0.01
15	**981167**	1026787	997539	999784	1.86	-2.70	0.22
16	1017777	1033656	1028718	**998922**	-1.89	-3.48	-2.98
17	1073640	1022250	1067403	**998596**	-7.51	-2.37	-6.89
18	**1003670**	1063306	1030854	1005516	0.18	-5.75	-2.52
19	**997348**	1065705	1053589	1007677	1.03	-5.76	-4.56
20	**981788**	1027134	1001820	1003726	2.19	-2.33	0.19
21	974384	1044771	**965862**	993916	1.97	-5.12	2.82
22	1065780	1045790	1001734	**987482**	-7.93	-5.90	-1.44
23	1070520	1027042	1014335	**1003350**	-6.69	-2.36	-1.09
24	978491	1045014	**973569**	998107	1.97	-4.70	2.46
25	1029327	1024239	**983691**	996726	-3.27	-2.76	1.31
26	**961728**	1043128	988534	1017962	5.52	-2.47	2.89
27	1028006	1030753	**995478**	1014516	-1.33	-1.60	1.88
28	1011689	1032478	994108	**994009**	-1.78	-3.87	-0.01
29	1015741	1019371	1048934	**1005629**	-1.01	-1.37	-4.31
30	**985496**	1027915	1024006	1001252	1.57	-2.66	-2.27

Table 3.10: Average total costs obtained from different heuristics

	GRASP	MA|PM	RTS	SS	TSPR	ALNS	MatH1
N50T20	443264	393263	370562	387360	361704	346878	375648
N100T20	791839	714627	712294	716643	685898	636962	692566
N200T20	1070026	1001026	1034923	1006302	951638	876761	1001182

Table 3.11: Average percentage deviations of MatHeuritic1 compared with different heuristics

	GRASP	MA|PM	RTS	SS	TSPR	ALNS
N50T20	-18.00	-4.69	1.35	-3.12	3.71	7.66
N100T20	-14.33	-3.19	-2.85	-3.48	0.96	8.03
N200T20	-6.88	0.02	-3.37	-0.51	4.95	12.43

Table 3.12: Computational times of MatHeuristic1 compared with different heuristics

Inst	N50T20			N100T20			N200T20		
	MA|PM	RTS	MatH1	MA|PM	RTS	MatH1	MA|PM	RTS	MatH1
1	170.4	180.0	60.8	1147.4	1079.0	266.6	3633.8	2965.0	1792.3
2	149.0	71.0	53.7	1192.8	240.0	235.3	3755.5	1250.0	1335.0
3	135.7	234.2	49.0	925.9	1299.0	240.3	3629.6	1200.0	1179.6
4	159.7	300.0	69.9	1097.3	672.0	293.9	3851.8	2141.0	1306.3
5	193.3	290.0	55.6	1125.0	335.0	284.9	4185.5	2423.0	1475.8
6	174.7	467.1	64.3	1107.0	1143.0	229.3	3794.8	2205.0	1005.4
7	174.0	404.7	49.0	1081.8	1250.0	278.8	4732.9	2610.0	1360.5
8	170.8	329.3	29.0	1043.1	1008.0	250.4	3929.4	2978.0	1280.0
9	158.1	36.0	80.3	1136.1	1125.0	295.1	3424.8	1250.0	1401.5
10	179.4	260.0	33.7	976.9	985.0	255.1	4600.8	2625.0	1179.5
11	178.3	540.0	57.5	1154.7	835.0	277.4	5440.0	3428.0	1590.0
12	151.0	180.0	52.1	1163.3	1129.0	285.9	3933.4	2242.0	1269.6
13	193.3	208.0	30.2	1036.6	599.0	289.2	4662.0	2719.0	1545.7
14	160.8	184.0	55.9	1153.7	1065.0	288.1	3596.1	1920.0	1244.7
15	173.4	555.1	73.2	1252.7	1139.0	243.1	4023.7	2408.0	1214.4
16	186.2	493.8	55.2	1210.9	1226.0	304.0	4245.7	3200.0	1380.0
17	177.5	153.9	64.4	964.5	1218.0	379.0	4355.4	2018.0	1123.0
18	163.5	189.0	59.2	1021.0	720.0	254.7	3875.2	4740.0	1178.9
19	186.9	488.7	51.0	1164.4	1349.0	260.6	4157.7	2797.0	1655.4
20	182.7	635.9	59.6	1167.4	1131.0	296.2	4048.1	2425.0	1477.9
21	188.4	160.2	60.0	1173.5	544.0	313.8	4205.0	3900.0	1305.8
22	146.1	810.0	33.8	1179.8	998.0	290.8	4465.9	2822.0	1387.9
23	188.5	794.7	31.6	1196.9	1037.0	240.9	3147.2	1797.0	1225.4
24	180.1	232.3	34.8	1250.7	1380.0	280.7	3308.8	2908.0	1504.7
25	192.4	163.0	64.1	848.4	1240.0	270.8	4349.0	2601.0	1395.5
26	159.9	129.0	36.3	1094.4	585.0	258.2	3785.8	1673.0	1379.7
27	173.1	156.7	59.3	1033.7	454.0	261.2	4086.9	2037.0	1318.1
28	171.0	230.7	60.3	1092.3	1190.0	250.0	4456.1	2909.0	1066.5
29	198.4	543.8	56.7	1069.8	993.0	338.1	5168.0	2179.0	1221.0
30	163.5	495.7	52.6	1180.3	1300.0	302.5	4105.8	2400.0	1145.7

Table 3.10 gives the average total costs obtained from different heuristics. All the results were taken from Adulyasak *et al.* (2015b). Table 3.11 shows the average percentage deviations of MatHeuristic1 algorithm compared with different heuristics. From these two tables, we can conclude that our algorithm is superior if compared to GRASP (Boudia *et al.*, 2007), and compatible when compared to MA|PM, RTS and SS algorithms. However, the difference between the results obtained by our algorithm is between 0.96% and 4.95% worst off when compared to TSPR (Armentano *et al.*, 2011) while 7.66% and 12.43% when compared to ALNS (Adulyasak *et al.*, 2012).

Table 3.12 shows the computational times of MatHeuristic1 compared with different heuristic (MA|PM and RTS). Note that the computational time for SS is not given in Moin and Yuliana (2015). As we can see from the table, MatHeuristic1 require a shorter average computational time when compared to both MA|PM and RTS for all instances in all cases.

Table 3.13 displays the gap between MatHeuristic1 and the adjusted lower bound in all cases. Column 3, 6 and 9 give the adjusted lower bound obtained from Bard and Nananukul (2009), while column 4, 7 and 10 calculate the gaps between our best solution and the adjusted lower bound in the case of N50T20, N100T20 and N200T20, respectively. Adjusted lower bound is used instead of normal lower bound because the solution contains more feasible solution in adjusted lower bound compared to the normal lower bound. As we can see from the table, in the case of N50T20, the gap is between 10.85% and 18.16% and averaged 14.64%. While in the case of N100T20, the gap is between 14.75% and 22.37% and averaged 19.45%. At last, the gap is between 15.05% and 18.98% and averaged 16.48% in the case of N200T20, slightly better than for the 100-customers problem.

Table 3.13: Gap between MatHeuristic1 solution and the adjusted lower bound

Inst	N50T20			N100T20			N200T20		
	MatH1	**Adj LB**	**Gap**	**MatH1**	**Adj LB**	**Gap**	**MatH1**	**Adj LB**	**Gap**
1	374264	326630	14.58	684370	582090	17.57	1005064	859151	16.98
2	382679	326324	17.27	695179	584847	18.87	1008993	856489	17.81
3	374059	329607	13.49	679257	581317	16.85	996844	847677	17.60
4	377022	328467	14.78	659536	574748	14.75	994712	858620	15.85
5	372027	325044	14.45	706692	583117	21.19	999114	858986	16.31
6	386968	332449	16.40	693958	586735	18.27	1006016	863514	16.50
7	377495	328176	15.03	677788	573184	18.25	1008156	859755	17.26
8	383376	325893	17.64	670607	564981	18.70	992901	862333	15.14
9	383931	326277	17.67	689304	583295	18.17	987864	845013	16.91
10	369095	326483	13.05	688089	577928	19.06	999130	854605	16.91
11	367246	330539	11.11	700952	580831	20.68	996692	866478	15.03
12	370920	326316	13.67	706511	581643	21.47	1002124	861683	16.30
13	372797	327017	14.00	692837	572655	20.99	1008080	855760	17.80
14	378460	323498	16.99	697085	586741	18.81	1002613	860535	16.51
15	385600	326339	18.16	680391	580764	17.15	999784	866899	15.33
16	367923	328384	12.04	706658	581498	21.52	998922	859910	16.17
17	389281	329996	17.97	716302	591956	21.01	998596	859039	16.25
18	381641	324394	17.65	682024	580835	17.42	1005516	860527	16.85
19	373799	327712	14.06	693967	576318	20.41	1007677	867188	16.20
20	371942	326112	14.05	715222	593649	20.48	1003726	852918	17.68
21	367850	327367	12.37	700373	590280	18.65	993916	853126	16.50
22	360679	324015	11.32	706492	577327	22.37	987482	854260	15.60
23	375067	327824	14.41	680172	569334	19.47	1003350	862561	16.32
24	364765	321520	13.45	696555	577506	20.61	998107	858675	16.24
25	384851	331818	15.98	695738	576299	20.73	996726	864960	15.23
26	382984	334062	14.64	689543	571351	20.69	1017962	855543	18.98
27	365470	322816	13.21	684183	572684	19.47	1014516	869693	16.65
28	365761	324047	12.87	685277	569282	20.38	994009	859167	15.69
29	389899	336159	15.99	699233	588796	18.76	1005629	866925	16.00
30	371576	335212	10.85	702675	582331	20.67	1001252	865404	15.70

3.9.2.2 Result of randomly generated data set

Table 3.14 illustrates the results for randomly generated data sets (with $h_i^C > 0$). We let the CPLEX run for a maximum of 7200 seconds (3 hours). The results show that our algorithm is better except for N50T5. Note that for instances N100T10 and N100T14, CPLEX was unable to produce any significant results.

Table 3.14: Results of MatHeuristic1 on randomly generated data set

Case	CPLEX		MatH1	
	Best Integer	Time (s)	Obj	Time (s)
N12T5	4416.85	929.41	3966.91	5.554
N12T10	8674.45	1212.55	7732.44	13.608
N12T14	11814.85	3600.35	11051.5	21.006
N20T5	6882.51	1194.95	6223.26	20.14
N20T10	14820.74	1401.64	12532.4	53.163
N20T14	19448.52	897.93	17235.6	78.411
N20T21	26705.26	1989.7	25487	166.499
N50T5	12006.72	2881.62	14469.6	140.226
N50T10	31335.94	713.58	29586	548.575
N50T14	45082.01	830.42	42238.5	1044.29
N50T21	117350.35	1056.38	62914.4	1579.11
N100T5	36017.92	3352.85	27756.5	1066.84
N100T10	-	-	56601.2	3781.4
N100T14	-	-	80843.7	7763.24

3.10 Statistical Analysis

3.10.1 Friedman Test, Iman and Davenport and Friedman Aligned Rank Test

We performed non-parametric procedures, specifically the Friedman test, the Iman and Davenport and Friedman Aligned Rank tests to determine whether there is a significant difference between the algorithms. The choice of different tests is to mitigate the weakness of the Friedman test (Derrac *et al.*, 2011). Here, the null hypothesis (H_0) for the test states that there is no significant difference between the behavior of MA|PM, RTS, SS and our algorithm.

The chi-square approximation of the Friedman test statistics (χ_F^2) is calculated at the significance level of $\alpha = 0.05$ using equation (17) (Derrac *et al.*, 2011).

$$F_f = \frac{12n}{k(k+1)}\left[\sum_j R_j^2 - \frac{k(k+1)^2}{4}\right] \quad \ldots (17)$$

In view of the fact that the Friedman test is conservative, we utilize a less conservative test proposed by Iman and Davenport. By using equation (18) (Derrac *et al.,* 2011),

$$F_{ID} = \frac{(n-1)\chi_F^2}{n(k-1) - \chi_F^2} \quad \ldots (18)$$

the Friedman value (F_{ID}) in Iman and Davenport, is computed using an F distribution with $(k-1)$ and $(k-1)(n-1)$ degrees of freedom. In order to mitigate the weakness of the ranking scheme in the Friedman test, the Friedman Aligned Rank Test is adopted where the observation is aligned with respect to the problems as well as with respect to the algorithms. The Friedman Aligned Ranks test statistic is computed by using the equation (19) (Derrac *et al.*, 2011).

$$F_{AR} = \frac{(k-1)\left[\sum_{j=1}^{k} \widehat{R_j^2} - \left(\frac{kn^2}{4}\right)(kn+1)^2\right]}{\left\{\frac{[kn(kn+1)(2kn+1)]}{6}\right\} - \left(\frac{1}{k}\right)\sum_{i=1}^{n} \hat{R}_i^2} \quad \ldots (19)$$

Table 3.15 presents the results of all statistical tests. From the results, we can conclude that our algorithm is significantly different from MA|PM, RTS and SS.

Table 3.15: The results of the Friedman, Iman-Davenport and Friedman Aligned Rank tests ($\alpha = 0.05$)

Case	Friedman Value	Critical Value in χ^2	Iman-Davenport Value	Critical Value in *F* distribution	Friedman Aligned Rank Value	Critical Value in χ^2
N50T20	25.72	7.815	11.6	2.7094	34.91	7.815
N100T20	34.84		18.32		39.97	
N200T20	39.66		22.85		31.49	

3.10.2 Post-Hoc Test: Holm Procedure

However, the main drawback of the Friedman, Iman-Davenport, and Friedman Aligned tests is that they can only detect significant differences over the whole multiple comparisons between some of the algorithms considered. We carried out a post-hoc test using the Holm procedure to establish whether our algorithm (control algorithm-MatHeuristic1) is significantly better than MA|PM, RTS and SS. We chose Holm procedure because it is less conservative and more powerful than the Bonferroni method and is suitable for small sample sizes (Derrac *et al.*, 2011).

We obtained the p-value through the conversion of the rankings computed from the Friedman test by using normal approximation and compared it to those from MA|PM, RTS and SS with our algorithm as the control method. We then obtained the z-value by using equation (20) (Derrac *et al.*, 2011).

$$z = \frac{R_i - R_j}{\sqrt{\frac{k(n+1)}{6n}}} \qquad \dots (20)$$

For each z_i, the corresponding cumulative normal distribution values p_i can be obtained. Let $p_1, p_2, \dots, p_{k-1}$ be the ordered p-values (smallest to largest), so that $p_1 \leq p_2 \leq \cdots \leq p_{k-1}$, and let $H_1, H_2, \dots, H_{k-1}$ be the corresponding hypotheses. If p_i is below $\theta_i = \frac{\alpha}{k-i}$, the corresponding hypothesis H_i is rejected.

The ranking results based on the Holm procedure in the cases of N50T20, N100T20 and N200T20 are displayed in Table 3.16 Table 3.17 and Table 3.18, respectively.

Table 3.16: Ranking results based on Holm Procedure for the comparison algorithms in N50T20

i	Algorithms	z_i	p_i	θ	Hypothesis, H_i
0	MatHeuristic1 (control method)	-	-	-	-
1	MA|PM	3.1999	0.004125	0.01667	Rejected
2	RTS	-1.7	0.178262	0.025	Not rejected
3	SS	1.3	0.193601	0.05	Not rejected

Table 3.17: Ranking results based on Holm Procedure for the comparison algorithms in N100T20

i	Algorithms	z_i	p_i	θ	Hypothesis, H_i
0	MatHeuristic1 (control method)	-	-	-	-
1	SS	5.5	0	0.01667	Rejected
2	MA|PM	4.4	0.000022	0.025	Rejected
3	RTS	4.1	0.000041	0.05	Rejected

Table 3.18: Ranking results based on Holm Procedure for the compared algorithms in N200T20

i	Algorithms	z_i	p_i	θ	Hypothesis, H_i
0	MatHeuristic1 (control method)	-	-	-	-
1	RTS	4.89999	0	0.01667	Rejected
2	SS	0.39999	0.68923	0.05	Not rejected
3	MA|PM	-0.39999	1.37831	0.025	Not rejected

In N50T20, H_1 indicates that there is a significant difference between MA|PM with control algorithm, H_2 indicates that there is a significant difference between RTS with the control algorithm, while H_3 represents there is a significant difference between SS with the control algorithm. From Table 3.13, we can see that H_1 is rejected while H_2 and H_3 are not rejected. So we can conclude that our algorithm is significantly better than MA|PM but performs equally well as RTS and SS algorithms because the results

show that there are no significant differences between their algorithms with the control method, MatHeuristic1.

In the case of N100T20, from the ranking of p_i values, H_1 indicates that there is a significant difference between SS with control algorithm, H_2 respresents there is a significant difference between MA|PM with the control algorithm, while H_3 indicates that there is a significant difference between RTS with the control algorithm. As we can see in Table 3.14, we can conclude that our algorithm is significantly better than all the compared algorithms.

In N200T20, H_1 indicates that there is a significant difference between RTS with control algorithm, H_2 indicates that there is a significant difference between SS with the control algorithm, while H_3 represents there is a significant difference between MA|PM with the control algorithm. From Table 3.15, H_2 and H_3 are not rejected and H_1 is rejected, so we can conclude that our algorithm is significantly better than RTS, but performs equally well as MA|PM and SS.

3.11 Concluding Remark

In this section, we proposed a two-phase methodology to solve the production-inventory-distribution routing problem (PIDRP), called MatHeuristic1. The problem is decomposed into two parts.

Phase 1 solves the mixed integer programming allocation model. Once the amount to be delivered and the inventories were determined, the routes were constructed using a Giant Tour procedure in phase 2 to find a feasible solution. Then, the solution is improved using the well known variable neighborhood search algorithm which embeds the forward, backward, swap and transfer mechanisms to perturb the solution. In addition, local search is performed using the 1-insertion inter-route, the 2-opt inter-route, the 2-opt intra route, the swap intra route, 1-insertion intra-route, and the 2-insertion intra-route.

We compared our results on a set of benchmark instances with MA|PM, RTS and SS and we found that our results are comparable to the compared algorithms in terms of total cost, runtime as well as the gap between MatHeuristic1 and adjusted lower bound obtained by Bard and Nananukul (2009). In addition, we also tested on a randomly generated data set where holding cost of the customer sites is set more than 0 and the results showed that MatHeuristic outperformed all the best integer solutions obtained from CPLEX, except for N50T5. Statistical analysis was also conducted and the testing showed that our algorithm is significantly different from MA|PM, RTS and SS. While the post-hoc test showed that our proposed algorithm, MatHeuristic1 is significantly better when compared to MA|PM in N50T20, all the three algorithms in N100T20 and RTS in N200T20.

CHAPTER 4 INTEGRATED MATHEMATICAL PROGRAMMING AND METAHEURISTIC- MATHEURISTIC2 APPROACH

4.1 Introduction

In this chapter, we propose an alternative approach, a two-phase iterative procedure, called MatHeuristic2, to solve the PIDRP model. In MatHeuristic2 approach, we decompose the full model into two sub-problems, namely, a production and inventory control and a transportation solution (or vehicle routing) problem (Lee *et al.*, 2013). The production and inventory solution and the transportation solution are alternatively modified in an attempt to reduce the overall cost.

The major modifications are in the shaking step of variable neighborhood search and the formulation of the mixed integer linear programming problem.

4.2 Initial Solution

In MatHeuristic2, an initial feasible solution for all periods is constructed by using zero-inventory, *VNS_interchange* (see Section 4.3.1), which makes use the three neighborhood structures defined in Imran *et al.* (2009) in the shaking step, and some perturbations (forward and backward transfers).

4.3 Types of Variable Neighborhood Search (VNS)

4.3.1 VNS-Interchange

VNS-Interchange is similar to the one discussed in Section 3.5 in chapter 3. The difference is that VNS-Interchange adopts the three neighborhood structures proposed by Imran *et al.* (2009) instead of swap and transfer moves. The three neighborhoods, which are briefly described in this subsection, are used in VNS-Interchange (i.e. $k_{max} = 3$). These include the 1-1 lambda interchange, the 2-0 shift and the 2-1 interchange. The order of the neighborhoods is as follows: the 1-1 lambda interchange is used as N_1, the 2-0 shift as N_2 and lastly the 2-1 interchange as N_3.

4.3.1.1 1-1 Lambda Interchange

1-1 lambda interchange aimed at generating a feasible solution by swapping a pair of customers from two routes. This procedure starts by taking a random customer from a random route and tries to swap it with other customers by taking into consideration all other routes and the violation of the constraints. This procedure is repeated until a feasible move is found. Figure 4.1 presents an example of 1-1 lambda interchange.

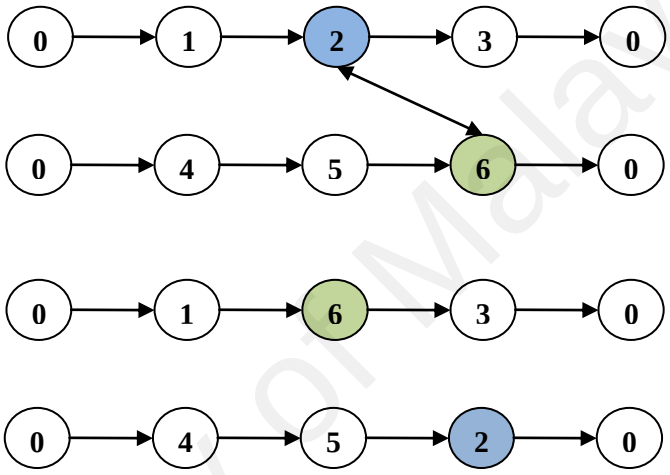

Figure 4.1: Example of a 1-1 lambda interchange

4.3.1.2 2-0 Shift

In the 2-0 shift procedure, two consecutive random customers from a randomly chosen route are selected. These two customers are considered together for possible insertion in other routes in a systematic manner, without violating any constraints. This procedure is repeated until a feasible move is found. Figure 4.2 shows an example of 2-0 shift.

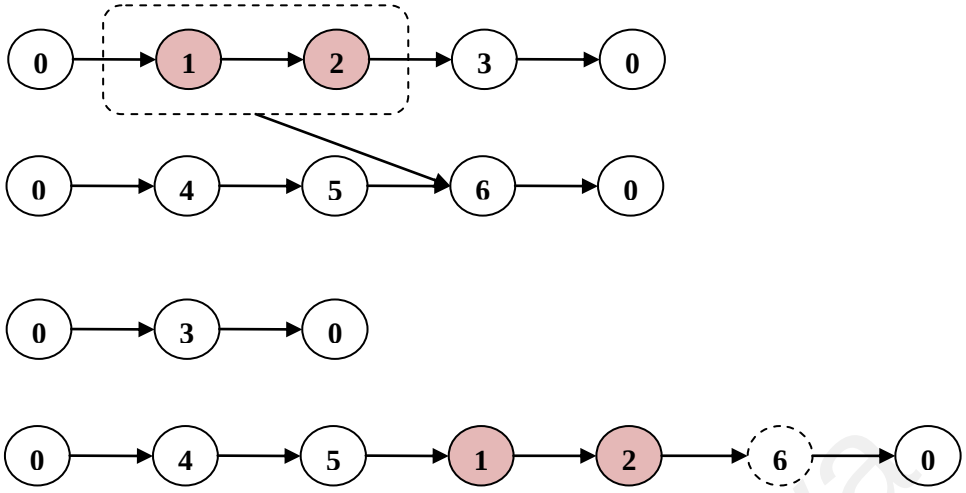

Figure 4.2: Example of a 2-0 shift

4.3.1.3 2-1 Interchange

This type of insertion attempts to shift two consecutive random customers from a randomly chosen route to another route selected systematically while getting one customer from the receiver route to be swapped until a feasible move is obtained. Figure 4.3 presents an example of 2-1 interchange.

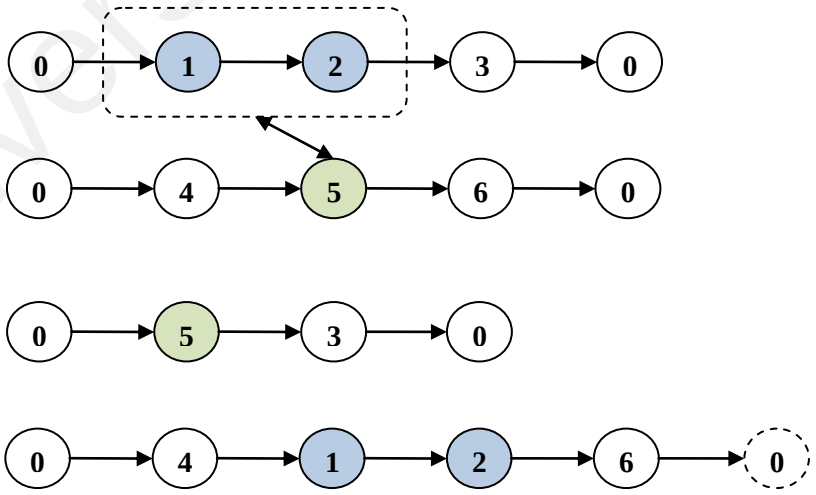

Figure 4.3: Example of a 2-1 interchange

4.3.2 VNS-Bestmove

VNS-Bestmove is also similar to the one described in Section 3.5 in Chapter 3. The neighborhood structures used in the shaking step for this type of VNS is similar to those in Section 3.6, which are forward transfer, backward transfer and swap. The only difference is that each time a transfer or a swap is performed, VNS-Bestmove will choose the best move which contributes the lowest cost (after the move) among all the feasible moves considered without violating any constraints.

4.3.3 VNS-Random

VNS-Random is exactly the same as the one discussed in Section 3.5 in Chapter 3. The moves (forward, backward transfer or swap) are chosen randomly by a defined probability (see Figure 3.4 for the algorithm of the shaking step). The reader is referred to Figure 3.1 and 3.2 for the steps of basic VNS and the flow chart of the VNS algorithm, respectively.

4.4 Enhanced Mixed Integer Program

After the routes are constructed and improved by VNS, we obtained y_{it} from the specified (improved) routes. We redefined the $y_{it} = 1$ if the customer i is visited in period t, and 0 otherwise. Note that y_{it} is a parameter which is specified by the given routes. The modified mixed integer program is presented as follows (MIP3):

$$\emptyset_{IP} = \min \sum_{t \in T} f z_t + \sum_{t \in T_0 \setminus \{\tau\}} h^P I_t^P + \sum_{t \in T \setminus \{\tau\}} \sum_{i \in N} h_i^C I_{it}^C \qquad (1b)$$

subject to: (2)-(6), (12-14) and an additional new constraint (15)

$$0 \leq w_{it} \leq D_{it}^{max} y_{it}\,, \qquad \forall i \in N, t \in T \qquad (21)$$

We introduced a new objective function that is a simplified version of objective $(1a)$ by removing the approximated cost of delivery. In addition, we newly created constraint (21) that is guided by the current routes. Solving MIP3 we obtained the optimal

production quantity, inventory at both production facility and customer sites and the delivery quantity, which serve as input into the VNS algorithm.

4.5 Algorithm Description

Figure 4.4 summarizes the important steps of the MatHeuristic2 while Figure 4.5 displays the flow chart of the MatHeuristic2 algorithm. First, an initial feasible solution is constructed by applying zero-inventory, VNS with interchange moves (described in Section 4.3.1) and some perturbations including forward and backward transfers. The algorithm iterates until a defined *maximum_iteration* is reached. In the first step of the iteration, the parameter y_{it} is assigned for each route in each period. Next, the modified mixed integer programming model is solved to obtain feasible allocations and inventories (both production and customer sites). Using the results obtained from solving the MIP1 model, the new solution is created by using Giant Tour procedure and Dijkstra's algorithm. If it is the first iteration, the solution is saved as both the *curr_solution* and *best_solution*. Whilst in the following iteration, *VNS_bestmove* (described in Section 4.3.2) is applied and determine if there is a reduction in the total cost. If there is no improvement, then *VNS_random* (described in Section 4.3.3) is applied until a defined *max_trial_count* is reached. If it reached the *max_trial_count* and there is still no improvement, then simulated annealing is used to accept a worse solution with reference to the acceptance probability. Whenever there is an improvement, the *curr_solution* and *curr_cost* is set to *best_solution* and *best_cost*.

Table 4.1 displays the additional description of the parameters and data structures used in the algorithm. The rest of the parameters and data structures are tabulated in Table 3.6.

Table 4.1: Additional parameters and data structures used in the algorithm

Parameter/Data Structure	Description
VNS_interchange	VNS with the negihborhood structures as in Section 4.4
VNS_bestmove	VNS with the best swap or/and transfer moves
VNS_random	VNS with the random swap or /and transfer moves
trial_count	Counter of the trial of *VNS_random*
max_trial_count	Maximum of *trial_count*
Improvement	1 is there is improvement of the total cost; 0 otherwise
Simulated_annealing	Simulated annealing algorithm aim at accept worse solution with the reference to the defined acceptance probability.

Step 1: Construct an initial feasible solution by zero-inventory, *VNS_iterchange* and perturbations.

Step 2: Do{
Step 2.1: Assign $y_{it} = 1$ if customer i is visited in period t.
Step 2.2: Solve MMIP model with the input of y_{it}
Step 2.3: Construct the new delivery routes by using Giant tour procedure and Dijkstra's algorithm.
Step 2.4: Evaluate the total cost:
If(*iteration_number*=1)
Set *best_cost=curr_cost* .
Else{ //The following iterations, *iteration_number*≥ 2
Apply *VNS_bestmove*.
If (*curr_cost<best_cost*){
best_cost=curr_cost
best_solution=curr_solution
}
Else{
Set *trial_count*=1
Do{
Apply *VNS_random*
If (*curr_cost<best_cost*){
best_cost=curr_cost
best_solution=curr_solution
Improvement=1
}
If (*trial_count=max_trial_count* and *Improvement*=0){
Apply *Simulated_annealing*
}While(*Improvement=0*)
}
}
iteration=iteration+1
}While(*iteration_number<maximum_iteration*)

Figure 4.4: Steps of MatHeuristic2 algorithm

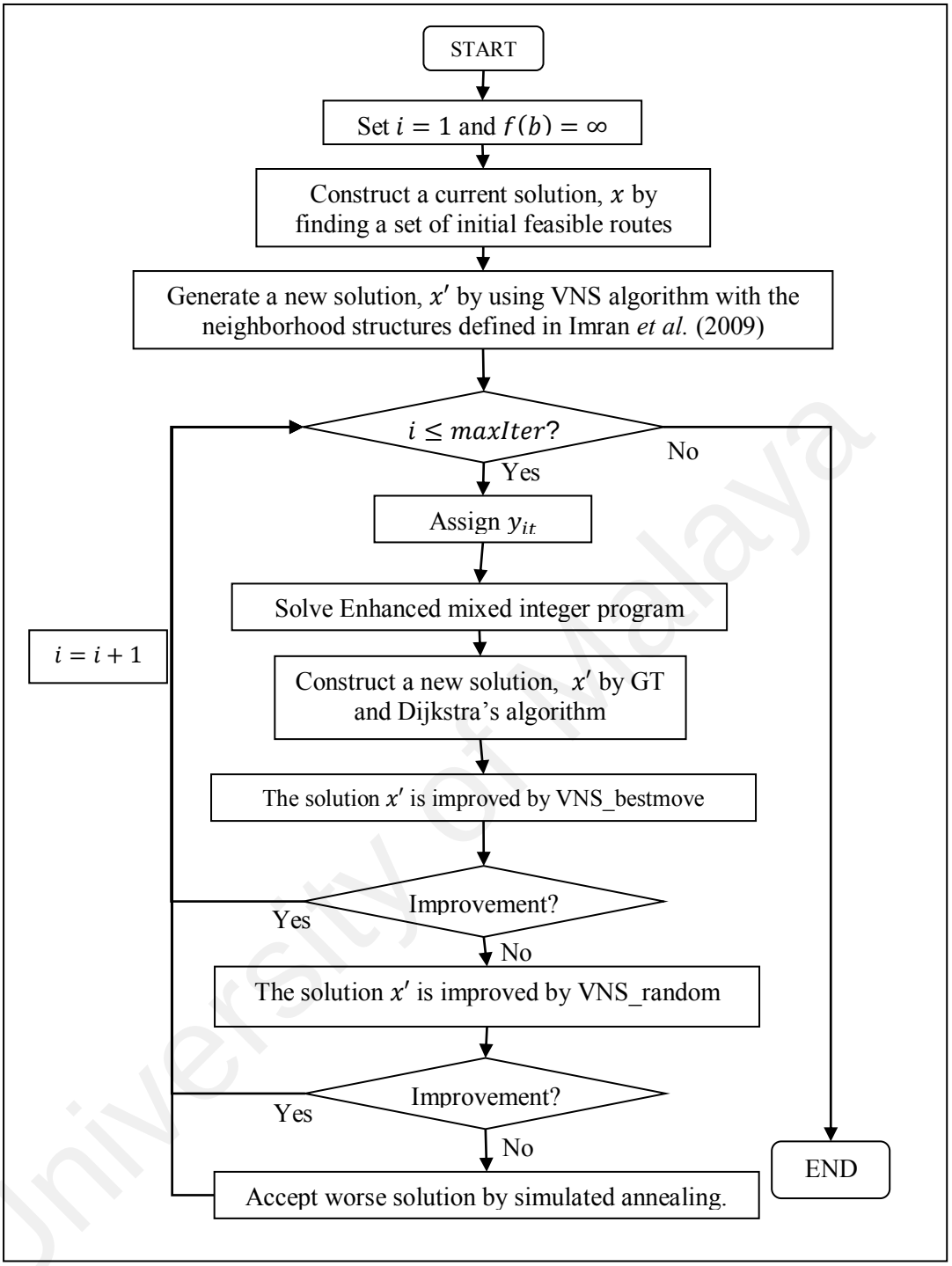

Figure 4.5: Flow chart of MatHeuristic2 algorithm

4.6 Computational Results

All the algorithms were written in Microsoft Visual C++ 2010 and performed on 3.1 GHz processor with 8GB of RAM. The code for the allocation model were implemented as mixed integer programming in Concert technology of Microsoft Visual Studio 2010 linked to the CPLEX 12.5 libraries. CPU times were obtained using the time function in C++.

4.6.1 Data

The data sets containing 30 instances of 50, 100 and 200 customer problem, all over the planning horizon of 20 periods (Boudia *et al.*, 2005) are used in the computational testing of our alternative approach, MatHeuristic2 algorithm. Detail description of the data set was given in Section 2.4.

4.6.2 Results and Discussion

Tables 4.2, 4.3 and 4.4 display the total (best) costs obtained from MatHeuristic2 compared to MA|PM (Boudia & Prins, 2009), RTS (Bard & Nananukul, 2009), SS (Moin & Yuliana, 2015), as well as MatHeuristic1 (discussed in Chapter 3) for the 30 instances with 50, 100 and 200 customers, respectively, over the 20 periods. The following tables are the extension of Tables 3.7, 3.8 and 3.9. They include the results of MatHeuristic1.

In Tables 4.2, 4.3 and 4.4, columns 2, 3, 4 and 5 present the best solutions (or costs) for MA|PM, RTS and MatHeuristic1, respectively. Column 6 gives the best cost of MatHeuristic2. Columns 7, 8, 9 and 10 report the percent deviation (see equation (16), change MatHeuristic1 to MatHeuristic2) of the best cost obtained from MatHeuristic2 from the best MA|PM, RTS, SS and MatH1 solutions, respectively. Same as in Tables 3.7, 3.8 and 3.9, the total cost in bold indicates that the cost is the lowest (or best) among all the comparison algorithms.

Table 4.2 displays the best solution of MatHeuristic2 in the benchmark problems with 50 customers and 20 periods. The solutions are 6.91%, 0.74%, 5.31% and 2.12% better on average than MA|PM, RTS, SS and MatHeuristic1. Besides, we also can see that there are 0, 12, 5, 0 and 13 instances with best cost among the five comparison algorithms. Both non-parametric procedures, the Sign test and the Wilcoxon Signed Ranks test (see Section 4.7) are conducted and the results show that MatHeuristic2 is significantly better when compared to MatHeuristic1.

Table 4.3 presents the results of the benchmark problems with 100 customers with 20 periods. MatHeuristic2 is 2.59%, 2.25% and 2.88% better on average than MA|PM, RTS and SS, respectively. However, MatHeuristic2 is only 0.58% worst off when compared to MatHeuristic1, but the difference is not significant (refer to Section 4.7).

Table 4.4 compares MatHeuristic2 with MA|PM, RTS, SS and MatHeuristic1 for the case of N200T20. MatHeuristic2 performs better than RTS by 2.3% on average, but 1.05%, 0.53% and 1.04% worst than MA|PM, SS and MatHeuristic1, respectively.

Table 4.2: Results of N50T20

Inst	Total Cost					Percent Deviation			
	MA|PM	RTS	SS	MatH1	MatH2	MA|PM	RTS	SS	MatH1
1	378378	398795	373172	374264	**370875**	-2.02	-7.53	-0.62	-0.91
2	403913	**373374**	380129	382679	375111	-7.68	0.46	-1.34	-2.02
3	409573	**353058**	369355	374059	371135	-10.36	4.87	0.48	-0.79
4	399220	**361176**	419256	377022	366328	-8.98	1.41	-14.45	-2.92
5	422279	**364819**	410140	372027	373022	-13.20	2.20	-9.95	0.27
6	407122	**368082**	372717	386968	377586	-7.82	2.52	1.29	-2.48
7	414977	396963	**366617**	377495	371137	-11.81	-6.96	1.22	-1.71
8	379744	370822	399269	383376	**370440**	-2.51	-0.10	-7.78	-3.49
9	407935	379379	416811	383931	**374140**	-9.03	-1.40	-11.41	-2.62
10	396258	370655	421940	369095	**359769**	-10.14	-3.03	-17.28	-2.59
11	402475	**354025**	382085	367246	366676	-9.76	3.45	-4.20	-0.16
12	358702	**354981**	365139	370920	356468	-0.63	0.42	-2.43	-4.05
13	371030	**365432**	398346	372797	370453	-0.16	1.36	-7.53	-0.63
14	406114	**363404**	386252	378460	364681	-11.36	0.35	-5.92	-3.78
15	373076	**367659**	406994	385600	371119	-0.53	0.93	-9.67	-3.90
16	379404	360534	**360433**	367923	362274	-4.73	0.48	0.51	-1.56
17	406353	398442	400878	389281	**382365**	-6.27	-4.20	-4.84	-1.81
18	401179	368533	**365400**	381641	368902	-8.75	0.10	0.95	-3.45
19	406893	377073	392953	373799	**368844**	-10.32	-2.23	-6.54	-1.34
20	398508	372141	381826	371942	**362015**	-10.08	-2.80	-5.47	-2.74
21	397112	374743	408826	367850	**352207**	-12.75	-6.40	-16.08	-4.44
22	358749	**347329**	355774	360679	357507	-0.35	2.85	0.48	-0.89
23	407369	**362619**	398696	375067	367723	-10.78	1.39	-8.42	-2.00
24	369784	375022	369717	364765	**360582**	-2.55	-4.00	-2.53	-1.16
25	411556	374682	414254	384851	**372288**	-10.55	-0.64	-11.27	-3.37
26	408704	366167	**363157**	382984	375902	-8.73	2.59	3.39	-1.88
27	366197	375261	387595	365470	**363711**	-0.68	-3.18	-6.57	-0.48
28	401032	373155	416631	365761	**358592**	-11.84	-4.06	-16.19	-2.00
29	384282	379320	**371621**	389899	386357	0.54	1.82	3.81	-0.92
30	369959	369223	364819	371576	**357068**	-3.61	-3.40	-2.17	-4.06

Table 4.3: Results of N100T20

Inst	Total Cost					Percent Devation			
	MA|PM	RTS	SS	MatH1	MatH2	MA|PM	RTS	SS	MatH1
1	714401	711671	714768	**684370**	694556	-2.86	-2.46	-2.91	1.47
2	722047	694694	706531	695179	**670573**	-7.68	-3.60	-5.36	-3.67
3	**677598**	683270	702056	679257	708914	4.42	3.62	0.97	4.18
4	710552	718252	715081	**659536**	680941	-4.35	-5.48	-5.01	3.14
5	733040	731260	709207	706692	**699935**	-4.73	-4.48	-1.32	-0.97
6	696146	744927	704383	693958	**682905**	-1.94	-9.08	-3.15	-1.62
7	705322	695728	701951	**677788**	680992	-3.57	-2.16	-3.08	0.47
8	679210	706058	709180	**670607**	684011	0.70	-3.22	-3.68	1.96
9	699518	705035	721338	689304	**671367**	-4.19	-5.01	-7.44	-2.67
10	705778	696521	721714	**688089**	699283	-0.93	0.39	-3.21	1.60
11	709122	711895	705973	**700952**	710272	0.16	-0.23	0.61	1.31
12	755726	703162	747220	706511	**695239**	-8.70	-1.14	-7.48	-1.62
13	695466	721066	707786	**692837**	699318	0.55	-3.11	-1.21	0.93
14	718260	698548	728132	697085	**687897**	-4.41	-1.55	-5.85	-1.34
15	736041	711506	729536	**680391**	718313	-2.47	0.95	-1.56	5.28
16	715209	714873	709663	**706658**	708177	-0.99	-0.95	-0.21	0.21
17	737832	**702314**	702391	716302	719988	-2.48	2.45	2.44	0.51
18	723413	720238	727677	**682024**	702293	-3.01	-2.56	-3.61	2.89
19	720218	748734	728944	**693967**	711100	-1.28	-5.29	-2.51	2.41
20	724727	729099	738669	**715222**	726930	0.30	-0.30	-1.61	1.61
21	724328	738746	712796	**700373**	706844	-2.47	-4.51	-0.84	0.92
22	**701506**	702849	717464	706492	716763	2.13	1.94	-0.10	1.43
23	710033	712717	723209	680172	**649341**	-9.35	-9.76	-11.38	-4.75
24	734327	727741	726290	**696555**	701365	-4.70	-3.76	-3.55	0.69
25	725446	725869	730997	**695738**	698750	-3.82	-3.88	-4.61	0.43
26	718939	700719	722588	**689543**	693440	-3.68	-1.05	-4.20	0.56
27	715068	686382	719054	**684183**	694616	-2.94	1.19	-3.52	1.50
28	685117	700980	695956	685277	**676207**	-1.32	-3.66	-2.92	-1.34
29	722571	725030	717728	**699233**	714447	-1.14	-1.48	-0.46	2.13
30	721850	698942	701024	702675	**693102**	-4.15	-0.84	-1.14	-1.38

Table 4.4: Results of N200T20

Inst	Total Cost					Percent Deviation			
	MA|PM	**RTS**	**SS**	**MatH1**	**MatH2**	**MA| PM**	**RTS**	**SS**	**Mat H1**
1	**996151**	1030684	1033864	1005064	1023870	2.71	-0.67	-0.98	1.84
2	**978373**	1010158	995600	1008993	1007123	2.85	-0.30	1.14	-0.19
3	986147	1016681	**983290**	996844	1015476	2.89	-0.12	3.17	1.83
4	**962937**	1042854	1004710	994712	1004969	4.18	-3.77	0.03	1.02
5	**970638**	1023680	1004463	999114	1014620	4.33	-0.89	1.00	1.53
6	**965646**	1025262	967194	1006016	1009793	4.37	-1.53	4.22	0.37
7	**980562**	1038746	1011310	1008156	997304	1.68	-4.16	-1.40	-1.09
8	1014809	1066068	1001545	**992901**	1013544	-0.12	-5.18	1.18	2.04
9	**967738**	1018420	984520	987864	995151	2.75	-2.34	1.07	0.73
10	1093230	1035240	1024832	**999130**	1011910	-8.04	-2.31	-1.28	1.26
11	1008080	1037705	**981277**	996692	1006529	-0.15	-3.10	2.51	0.98
12	998951	1035350	**995275**	1002124	1015183	1.60	-1.99	1.96	1.29
13	**984918**	1063024	1028545	1008080	1030030	4.38	-3.20	0.14	2.13
14	**964301**	1024491	1002466	1002613	1027311	6.13	0.27	2.42	2.40
15	**981167**	1026787	997539	999784	995542	1.44	-3.14	-0.20	-0.43
16	1017777	1033656	1028718	**998922**	1014381	-0.33	-1.90	-1.41	1.52
17	1073640	1022250	1067403	**998596**	1015209	-5.76	-0.69	-5.14	1.64
18	**1003670**	1063306	1030854	1005516	1022257	1.82	-4.02	-0.84	1.64
19	**997348**	1065705	1053589	1007677	1021451	2.36	-4.33	-3.15	1.35
20	**981788**	1027134	1001820	1003726	1009636	2.76	-1.73	0.77	0.59
21	974384	1044771	**965862**	993916	1005596	3.10	-3.90	3.95	1.16
22	1065780	1045790	1001734	**987482**	1002523	-6.31	-4.32	0.08	1.50
23	1070520	1027042	1014335	**1003350**	1014295	-5.54	-1.26	0.00	1.08
24	978491	1045014	**973569**	998107	1006545	2.79	-3.82	3.28	0.84
25	1029327	1024239	**983691**	996726	1011588	-1.75	-1.25	2.76	1.47
26	**961728**	1043128	988534	1017962	1004669	4.27	-3.83	1.61	-1.32
27	1028006	1030753	**995478**	1014516	1021082	-0.68	-0.95	2.51	0.64
28	1011689	1032478	994108	**994009**	1007917	-0.37	-2.44	1.37	1.38
29	1015741	1019371	1048934	**1005629**	1010323	-0.54	-0.90	-3.82	0.46
30	**985496**	1027915	1024006	1001252	1014027	2.81	-1.37	-0.98	1.26

Table 4.5: Average total costs obtained from different heuristics (including MatHeuristic1 algorithm)

	GRASP	MA|PM	RTS	SS	TSPR	ALNS	MatH1	MatH2
N50T20	443264	393263	370562	387360	361704	346878	375648	367843
N100T20	791839	714627	712294	716643	685898	636962	692566	696596
N200T20	1070026	1001026	1034923	1006302	951638	876761	1001182	1011662

Table 4.6: Average percentage deviations of MatHeuritic1 algorithm compared with different heuristics (including MatHeuristic1 algorithm)

	GRASP	MA|PM	RTS	SS	TSPR	ALNS	MatH1
N50T20	-20.50	-6.91	-0.74	-5.31	1.67	5.70	-2.12
N100T20	-13.67	-2.59	-2.25	-2.88	1.54	8.56	0.58
N200T20	-5.77	1.05	-2.30	0.53	5.93	13.33	1.04

Table 4.7: Computational times of MatHeuristic2 compared with different heuristics

Inst	N50T20			N100T20			N200T20		
	MA|PM	RTS	MatH2	MA|PM	RTS	MatH2	MA|PM	RTS	MatH2
1	170.4	180.0	95.4	1147.4	1079.0	626.2	3633.8	2965.0	6580.6
2	149.0	71.0	103.3	1192.8	240.0	608.8	3755.5	1250.0	4958.4
3	135.7	234.2	83.0	925.9	1299.0	720.8	3629.6	1200.0	7437.3
4	159.7	300.0	114.1	1097.3	672.0	658.7	3851.8	2141.0	6401.4
5	193.3	290.0	129.6	1125.0	335.0	845.0	4185.5	2423.0	4785.6
6	174.7	467.1	99.8	1107.0	1143.0	633.0	3794.8	2205.0	6880.5
7	174.0	404.7	90.7	1081.8	1250.0	734.0	4732.9	2610.0	6138.3
8	170.8	329.3	129.3	1043.1	1008.0	606.6	3929.4	2978.0	9012.5
9	158.1	36.0	92.4	1136.1	1125.0	818.3	3424.8	1250.0	6864.4
10	179.4	260.0	105.8	976.9	985.0	696.1	4600.8	2625.0	6811.8
11	178.3	540.0	72.6	1154.7	835.0	646.3	5440.0	3428.0	9615.1
12	151.0	180.0	94.8	1163.3	1129.0	642.1	3933.4	2242.0	7466.6
13	193.3	208.0	119.8	1036.6	599.0	883.5	4662.0	2719.0	5980.4
14	160.8	184.0	89.2	1153.7	1065.0	811.1	3596.1	1920.0	5733.3
15	173.4	555.1	131.2	1252.7	1139.0	620.4	4023.7	2408.0	6242.7
16	186.2	493.8	111.9	1210.9	1226.0	613.7	4245.7	3200.0	5340.0
17	177.5	153.9	121.2	964.5	1218.0	754.8	4355.4	2018.0	5460.6
18	163.5	189.0	127.7	1021.0	720.0	624.9	3875.2	4740.0	4983.0
19	186.9	488.7	62.1	1164.4	1349.0	677.5	4157.7	2797.0	5744.9
20	182.7	635.9	126.4	1167.4	1131.0	734.3	4048.1	2425.0	8338.5
21	188.4	160.2	113.3	1173.5	544.0	712.9	4205.0	3900.0	5411.3
22	146.1	810.0	121.6	1179.8	998.0	625.8	4465.9	2822.0	6109.3
23	188.5	794.7	137.5	1196.9	1037.0	603.2	3147.2	1797.0	5446.2
24	180.1	232.3	113.3	1250.7	1380.0	818.4	3308.8	2908.0	7903.9
25	192.4	163.0	112.7	848.4	1240.0	569.9	4349.0	2601.0	6113.7
26	159.9	129.0	141.4	1094.4	585.0	694.4	3785.8	1673.0	7584.3
27	173.1	156.7	66.3	1033.7	454.0	604.3	4086.9	2037.0	7355.7
28	171.0	230.7	117.0	1092.3	1190.0	550.8	4456.1	2909.0	5377.5
29	198.4	543.8	117.0	1069.8	993.0	595.0	5168.0	2179.0	6741.2
30	163.5	495.7	84.3	1180.3	1300.0	657.7	4105.8	2400.0	7257.3

Table 4.5 displays the average total costs obtained from different heuristics, including the MatHeuristic1 algorithm discussed in the previous chapter. Table 4.6 shows the average percentage deviations of MatHeuristic2 algorithm compared with different heuristics. Note that Table 4.5 and 4.6 are the extension of the Table 3.10 and 3.11 with the additional of MatHeuristic1 algorithm. From both tables, we can conclude that MatHeuristic2 algorithm is superior when compared to GRASP (Boudia *et al.*, 2007), and compatible if compared to MA|PM, RTS, SS and MatHeuristic1 algorithms. However, the difference between the best costs obtained by MatHeuristic2 algorithm is between 1.54% and 5.93% worst off when compared to TSPR (Armentano *et al.*, 2011) while 5.70% and 13.33% if compared to ALNS (Adulyasak *et al.*, 2012).

Table 4.7 shows the computational times of MatHeuristic2 compared with different heuristic (MA|PM and RTS). As we can see from the table, MatHeuristic1 require a shorter average computational time when compared to both MA|PM and RTS for the cases N50T20 and N100T20, but require more computational time in N200T20. We conjecture that the requirement of higher computational time is due to the large number of iterations used in MatHeuristic2 compared to MatHeuristic1.

Table 4.8 displays the gap between MatHeuristic1 and the adjusted lower bound in all cases. The description of the columns is the same of MatHeuristic1. As we can see from the table, in the case of N50T20, the gap is between 6.52% and 15.87% and averaged 12.26%. While in the case of N100T20, the gap is between 14.05% and 24.15% and averaged 20.14%. At last, the gap is between 14.84% and 20.36% and averaged 17.70% in the case of N200T20, slightly better than for the 100-customers problem.

Table 4.8: Gap between MatHeuristic2 solution and the adjusted lower bound

Inst	N50T20			N100T20			N200T20		
	MatH2	Adj LB	Gap	MatH2	Adj LB	Gap	MatH2	Adj LB	Gap
1	370875	326630	13.55	694556	582090	19.32	1023870	859151	19.17
2	375111	326324	14.95	670573	584847	14.66	1007123	856489	17.59
3	371135	329607	12.60	708914	581317	21.95	1015476	847677	19.80
4	366328	328467	11.53	680941	574748	18.48	1004969	858620	17.04
5	373022	325044	14.76	699935	583117	20.03	1014620	858986	18.12
6	377586	332449	13.58	682905	586735	16.39	1009793	863514	16.94
7	371137	328176	13.09	680992	573184	18.81	997304	859755	16.00
8	370440	325893	13.67	684011	564981	21.07	1013544	862333	17.54
9	374140	326277	14.67	671367	583295	15.10	995151	845013	17.77
10	359769	326483	10.20	699283	577928	21.00	1011910	854605	18.41
11	366676	330539	10.93	710272	580831	22.29	1006529	866478	16.16
12	356468	326316	9.24	695239	581643	19.53	1015183	861683	17.81
13	370453	327017	13.28	699318	572655	22.12	1030030	855760	20.36
14	364681	323498	12.73	687897	586741	17.24	1027311	860535	19.38
15	371119	326339	13.72	718313	580764	23.68	995542	866899	14.84
16	362274	328384	10.32	708177	581498	21.78	1014381	859910	17.96
17	382365	329996	15.87	719988	591956	21.63	1015209	859039	18.18
18	368902	324394	13.72	702293	580835	20.91	1022257	860527	18.79
19	368844	327712	12.55	711100	576318	23.39	1021451	867188	17.79
20	362015	326112	11.01	726930	593649	22.45	1009636	852918	18.37
21	352207	327367	7.59	706844	590280	19.75	1005596	853126	17.87
22	357507	324015	10.34	716763	577327	24.15	1002523	854260	17.36
23	367723	327824	12.17	649341	569334	14.05	1014295	862561	17.59
24	360582	321520	12.15	701365	577506	21.45	1006545	858675	17.22
25	372288	331818	12.20	698750	576299	21.25	1011588	864960	16.95
26	375902	334062	12.52	693440	571351	21.37	1004669	855543	17.43
27	363711	322816	12.67	694616	572684	21.29	1021082	869693	17.41
28	358592	324047	10.66	676207	569282	18.78	1007917	859167	17.31
29	386357	336159	14.93	714447	588796	21.34	1010323	866925	16.54
30	357068	335212	6.52	693102	582331	19.02	1014027	865404	17.17

4.7 Statistical Analysis

We performed two non-parametric procedures, namely the Sign test and the Wilcoxon signed ranks test for pairwise statistical comparisons between MatHeuristic1 and MatHeuristic2 algorithms. These two procedures are used to detect significant differences (or the behavior) of the two algorithms, especially when the normality assumption of the data is violated. Here, in both procedures, the null hypothesis (H_0) for the test states that there is no significant difference between the behavior of the two algorithms, MatHeuristic1 and MatHeuristic2.

The Sign test is a popular way to compare the overall performance of algorithms by counting the number of cases on which an algorithm is the overall winner. The number

of wins is distributed based on a binomial distribution; for a greater number of cases, as in our study, the number of wins is under the null hypothesis distributed according to $\left(\frac{n}{2}, \frac{\sqrt{n}}{2}\right)$, where n represents the number of cases or instances, which allows for the use of the z-test. If the number of wins is at least $\frac{n}{2} + 1.96\frac{\sqrt{n}}{2}$, then the algorithm is significantly better with $p < 0.05$. Note that we set $\propto = 0.05$ in our study.

Table 4.9 displays the results of the Sign test in all cases. For N50T20, the sign value is more than 15 $\left(= \frac{30}{2}\right)$ and the p-value is less than 0.05, so we can conclude that the hypothesis is rejected. In other words, MatHeuristic2 algorithm is significantly better than MatHeuristic1 algorithm. However, in the cases of N100T20 and N200T20, the sign value is less than 15 and p-values are less than 0.05 in both cases, the hypotheses are rejected, meaning that MatHeuristic1 is significantly better than MatHeuristic2 algorithm.

Table 4.9: The results of the Sign test

Cases	p-value	$\propto$	Sign	Hypothesis, H_i
N50T20	5.77×10^{-8}	0.05	29	Rejected
N100T20	0.0428		9	Rejected
N200T20	5.95×10^{-5}		4	Rejected

Wilcoxon signed rank test is a nonparametric pairwise test that aims to detect significant differences between two sample means, that is, the behavior of the two algorithms. Wilcoxon's test is used because it is more sensitive than the Sign test. In Wilcoxon's test, we first compute the R^+ and R^-, where $R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i)$ and $R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i)$ (Derrac et al. (2011), related to the comparisons between the two algorithms. Let $M = \min(R^+, R^-)$, if M is less than or equal to the value of the distribution of Wilcoxon for n degrees of freedom

(refer to Table B.12 in Zar (2009)), then the null hypothesis is rejected. Computation of p-value is also included; if the corresponding p-value is less than $\propto$, then the null hypothesis is rejected as well. Table 4.10 shows the results of the Wilcoxon Signed Ranks test in all the cases.

Table 4.10: The results of the Wilcoxon Signed Ranks test

Cases	p-value	$\propto$	z-value	Signed rank	Hypothesis, H_i
N50T20	0.0000021	0.05	4.741	463	Rejected
N100T20	0.1306		-1.5118	159	Not rejected
N200T20	0.000037		-4.124	32	Rejected

From the table, we can see that the hypothesis is rejected in the case of N50T20, as the p-value is less than 0.05. Also, it has a positive z-value; meaning that MatHeuristic2 is significantly better than MatHeuristic1 algorithm. Whilst in the case of N100T20, the hypothesis is not rejected, so we can conclude that MatHeuristic2 approach performs equally well as MatHeuristic1. Lastly, the hypothesis is rejected in the case of N200T20, since it has a negative z-value, we can conclude that MatHeuristic1 is significantly better than MatHeuristic2 algorithm.

4.8 Concluding Remarks

In this chapter, we proposed an alternative approach, called the MatHeuristic2 approach to solve the PIDRP. The problem is subdivided into two parts: the production and inventory control and the vehicle routing problem. First, an initial feasible solution is constructed by using zero-inventory, VNS with interchange moves as the neighborhood structures in shaking step and some perturbations. Then, a modified mixed integer problem is solved with the input of a routing parameter to obtain the optimal amount to be delivered, inventories at both the production facility and customer sites and the production quantity over the planning horizon.

The routes are formed again using the optimal allocations obtained by using Giant Tour procedure and Dijkstra's algorithm. The solution is then improved using VNS with best move property, which always find the move (forward, backward transfer or swap) that produces the most savings. If there is no improvement, then VNS with random moves is applied. If there is still no improvement, simulated annealing algorithm is used to accept the worse solution with the reference of acceptance probability. Next, the routing parameters are updated and the modified mixed integer programming model is solved again with the input of the parameter. The process is iterated until a defined maximum number of iterations is reached.

In this chapter, we also compare our results of MatHeuristic2 on a set of benchmark instances with MA|PM, RTS, SS and MatHeuristic1 in terms of total cost and runtime. By comparing the total cost of the best solutions, MatHeuristic2 outperforms all the comparison algorithms in smaller cases. However, MatHeuristic2 performs quite poorly when compared to MA|PM, SS as well as MatHeuristic1 for large instances (N200T20). Besides, MatHeuristic2 require shorter average computational time as compared to MA|PM and RTS in all cases, except for the case of N200T20, which require much

higher computational time due to large number of iterations used in the algorithm. In addition, the gaps between MatHeuristic2 and the adjusted lower bound are also computed and the results showed that the average gap is 12.26%, 20.14% and 17.70% in the cases of N50T20, N100T20 and N200T20, respectively. In terms of total cost, MatHeuristic2 outperforms all the comparison algorithms in smaller cases. However, MatHeuristic2 performs quite poorly when compared to MA|PM, SS as well as MatHeuristic1 for large instances (N200T20).

Statistical analysis was also conducted to test the significant difference between MatHeuristic1 and MatHeuristic2. The results show that MatHeuristic2 is significantly better than MatHeuristic1 approach for the smaller case, but performs poorly for the bigger case. We conjecture that the poor performance is due to the limited number of iterations.

CHAPTER 5 CONCLUSION

This research considered two different MatHeuristics, namely MatHeuristic1 and MatHeuristic2, for solving the production, inventory and distribution routing problem (PIDRP). The results of both algorithms were compared with the MA|PM (Boudia & Prins, 2009), RTS (Bard & Nananukul, 2009) and SS (Moin & Yuliana, 2015). The summary and conclusion of each chapter and future research suggestions are presented in the next sections.

5.1 Summary of Research and Conclusions

In Chapter 2, a brief introduction and definition of PIDRP were given. The solution methodologies for PIDRP were explained and classified into exact algorithm, heuristic algorithm and metaheuristic algorithm. Some examples of several well-known algorithms were presented based on the respective classifications, such as branch-and-cut with LSP reformulation (Ruokokoski *et al.*, 2010), MIP-based hybrid heuristic (Archetti *et al.*, 2011) and Op-ALNS (Adulyasak *et al.*, 2012).

In Chapter 3, we proposed a two-phase methodology, known as MatHeuristic1 algorithm, to solve PIDRP. The PIDRP comprises of a single production facility that produces a single product and multiple customer sites over a finite planning horizon. The problem was decomposed into two phases: Phase 1 solved the mixed integer programming (MIP) allocation model to obtain the optimal delivery quantities and inventories. Then, the routes were constructed using a Giant Tour procedure in phase 2 to find an initial feasible solution. Then, the solution was improved using the variable neighborhood search (VNS) algorithm, including the three type of neighborhood structures: forward, backward and swap; and the local searches: 1-insertion inter-route, the 2-opt inter-route, the 2-opt intra route, the swap intra route, 1-insertion intra-route,

and the 2-insertion intra-route. Besides, we also embed the concept of tabu search into the VNS algorithm to escape from local optima.

We tested the MatHeuristic1 algorithm on a set of benchmark instances (Boudia *et al.*, 2005) and compared the results with MA|PM, RTS and SS in terms of both total operating cost and runtimes. From all the results obtained, we can conclude that our results are comparable to the comparison algorithms. The gap between our solution and the adjusted lower bound is also computed and the results showed that the gaps are averaged at 18.16%, 19.45% and 16.48% for the cases of N50T20, N100T20 and N200T20, respectively. In addition, we also tested our algorithm on a randomly generated data set for which the holding cost is between 1 and 10 instead of 0, and the results showed that our algorithm outperformed all the best integer solution obtained from CPLEX, except for N50T5. Three non-parametric tests, such as Friedman test, were also performed and the testing showed that MatHeuristic1 is significantly different from MA|PM, RTS and SS. While the post-hoc test, the Holm procedure showed that MatHeuristic1 is significantly better when compared to MA|PM in N50T20, all the three algorithms in N100T20 and RTS in N200T20.

In Chapter 4, we proposed an alternative approach, called the MatHeuristic2 approach to solve the PIDRP. The problem was subdivided into two parts: the production and inventory control and the vehicle routing problem. The major modification (compared to MatHeuristic1) was the iterative procedure between the MIP and VNS. In MatHeuristic1, the mixed linear programming problem was first solved and the output was used as input in VNS. However, the algorithm did not iterate. In addition, in MatHeuristic2 the formulation of the MIP problem was modified to include the routing component instead of using the approximated routing as in MatHeuristic1, and the shaking step of VNS that considered additional three neighborhood structures

was also embedded. To obtain initial routing, an initial feasible solution over a finite planning horizon was formed by using zero-inventory and VNS with interchange moves which adopted the three neighbourhood structures (Imran *et al.*, 2009): 1-1 lambda interchange, 2-0 shift and 2-1 interchange in shaking step and some perturbations was applied. A modified mixed integer problem (MIP) was solved with the input of a routing parameter to obtain the optimal delivered quantities, inventories at both the production facility and customer sites and the production quantity for all periods.

The routes were constructed again using the optimal allocations that obtained from the modified MIP by using Giant Tour procedure and Dijkstra's algorithm. The solution was then improved using VNS with best move property, which always finds the move (forward, backward transfer or swap) that gives the most savings. VNS with random moves was applied if there is no improvement. If there is still no improvement, simulated annealing algorithm was performed to accept the worse solution with the reference of acceptance probability. Next, the routing parameters were updated and the modified MIP model was solved again using the input of the parameter. The process was iterated until a maximum number of iterations is reached.

In this chapter, we also tested the MatHeuritic2 algorithm on a set of benchmark instances (as in Chapter 3) and compared the results with MA|PM, RTS, SS and MatHeuristic1 algorithms in terms of total cost and runtime. For total cost, MatHeuristic2 outperformed all the comparison algorithms in the case of N50T20. However, MatHeuristic2 performed quite poorly when compared to MA|PM, SS as well as MatHeuristic1 for large instances (N200T20). Besides, MatHeuristic2 require shorter average computational time when compared to MA|PM and RTS for the cases N50T20 and N100T20, but require more runtime in N200T20, this is due to the large number of iterations used in MatHeuristic2 as compared to MatHeuristic1. Two non-parametric

tests, namely the Sign test and the Wilcoxon signed ranks test, were also performed to test the significant difference between MatHeuristic1 and MatHeuristic2 algorithms. The results showed that MaHeuristic2 is significantly better than MatHeuristic1 approach for the smaller case (N50T20), but performed poorly for the bigger case (N200T20).

5.2 Future Research Directions

There are several potential extensions of this research.

For VNS, different local search algorithms such as relax-and-fix (Pochet & Wolsey, 2006) and adaptive large neighborhood search (ALNS) (Adulyasak *et al.*, 2012) can be applied to improve the solution. Alternatively, a different neighborhood structure can be explored to improve the solution.

The algorithm is suggested to be extended to PIDRP with stochastic demand where the future demand is uncertain, in a MatHeuristic framework. Capturing the stochastic nature of demand in the PIDRP model would be advantageous, especially in those industries where markets shift regularly. In addition, PIDRP with stochastic demand is more related to the real-life inventory system in which demands for many products are uncertain and the model can be easily run out of stock. PIDRP with stochastic demand is also known as Stochastic Production Routing Problem (SPRP) (Adulyasak *et al.*, 2015a). We also refer the reader to Adulyasak *et al.* (2015a) for the detailed explanation of the SPRP.

Other variations of the PIDRP problem can be applied, for example PIDRP with backordering, also in a MatHeuristic framework. The PIDRP with backordering is also known as the production routing problem with backordering (PRP-B) (Brahimi & Aouam, 2016). The PRP-B integrates lot-sizing and vehicle routing decisions, where

backordering of end customer demands is allowed at a penalty. However, some modifications to the algorithm are needed by applying these problems.

Other metaheuristic algorithms can be used instead of VNS in MatHeursitic2 such as memetic algorithm with population management (MA|PM) (Boudia *et al.*, 2007), tabu search (Bard & Nananukul, 2009 and Armentano *et al.*, 2011), optimization based adaptive large neighborhood search (Op-ALNS) (Adulyasak *et al.*, 2012) and Scatter Search (SS) (Moin & Yuliana, 2015).

REFERENCES

Abdelmaguid, T. F., & Dessouky, M. M. (2006). A genetic algorithm approach to the integrated inventory-distribution problem. *International Journal of Production Research*, *44*(21), 4445-4464.

Absi, N., Archetti, C., Dauzère-Pérès, S., & Feillet, D. (2014). A two-phase iterative heuristic approach for the production routing problem. *Transportation Science*, *49*(4), 784-795.

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2012). Optimization-based adaptive large neighborhood search for the production routing problem. *Transportation Science*, *48*(1), 20-45.

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2013). Formulations and branch-and-cut algorithms for multivehicle production and inventory routing problems. *INFORMS Journal on Computing*, *26*(1), 103-120.

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2015a). Benders decomposition for production routing under demand uncertainty. *Operations Research*, *63*(4), 851-867.

Adulyasak, Y., Cordeau, J. F., & Jans, R. (2015b). The production routing problem: A review of formulations and solution algorithms. *Computers & Operations Research*, *55*, 141-152.

Aghezzaf, E. H., Zhong, Y., Raa, B., & Mateo, M. (2012). Analysis of the single-vehicle cyclic inventory routing problem. *International Journal of Systems Science*, *43*(11), 2040-2049.

Andersson, H., Hoff, A., Christiansen, M., Hasle, G., & Lokketangen, A. (2010). Industrial aspects and literature survey: Combined inventory management and routing. *Computers & Operations Research, 37*(9), 1515-1536.

Archetti, C., Bertazzi, L., Paletta, G., & Speranza, M. G. (2011). Analysis of the maximum level policy in a production-distribution system. *Computers & Operations Research*, *38*(12), 1731-1746.

Armentano, V. A., Shiguemoto, A. L., & Løkketangen, A. (2011). Tabu search with path relinking for an integrated production–distribution problem. *Computers & Operations Research*, *38*(8), 1199-1209.

Bard, J. F., & Nananukul, N. (2009). The integrated production–inventory–distribution–routing problem. *Journal of Scheduling*, *12*(3), 257-280

Bard, J. F., & Nananukul, N. (2010). A branch-and-price algorithm for an integrated production and inventory routing problem. *Computers & Operations Research*, *37*(12), 2202-2217.

Boudia, M., Louly, M. A. O., & Prins, C. (2005). Combined optimization of production and distribution. In *CD-ROM Proceedings of the International Conference on Industrial Engineering and Systems Management, IESM* (Vol. 5).

Boudia, M., Louly, M. A. O., & Prins, C. (2007). A reactive GRASP and path relinking for a combined production–distribution problem. *Computers & Operations Research*, *34*(11), 3402-3419.

Boudia, M., Louly, M. A. O., & Prins, C. (2008). Fast heuristics for a combined production planning and vehicle routing problem. *Production Planning and Control*, *19*(2), 85-96.

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.

Brahimi, N., & Aouam, T. (2016). Multi-item production routing problem with backordering: a MILP approach. *International Journal of Production Research*, *54*(4), 1076-1093.

Brahimi, N., Dauzere-Peres, S., Najid, N. M., & Nordli, A. (2006). Single item lot sizing problems. *European Journal of Operational Research*, *168*(1), 1-16.

Carlton, B. (1995). *A tabu search approach to the general vehicle routing problem.* (Doctoral Dissertation, The University of Texas at Austin). Retrieved from https://dl.acm.org/citation.cfm?id=2739491.

Chandra, P. (1991). *On coordination of production and distribution decisions*. (Doctoral dissertation, University of Pennsylvania). Retrieved from https://pdfs.semanticscholar.org/c5ab/0c98353df6e40266e07be71f7e0e1dc74946.pdf.

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

Chandra, P., & Fisher, M. L. (1994). Coordination of production and distribution planning. *European Journal of Operational Research*, *72*(3), 503-517.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*(4), 568-581.

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

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3-18.

Evans, J. R. (1985). An efficient implementation of the Wagner-Whitin algorithm for dynamic lot-sizing. *Journal of Operations Management*, *5*(2), 229-235.

Fischetti, M., & Lodi, A. (2003). Local branching. *Mathematical Programming*, *98*(1-3), 23-47.

Fischetti, M., Salazar González, J. J., & Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, *45*(3), 378-394.

Gaudioso, M., & Paletta, G. (1992). A heuristic for the periodic vehicle routing problem. *Transportation Science*, *26*(2), 86-92.

Gillett, B. E., & Miller, L. R. (1974). A heuristic algorithm for the vehicle-dispatch problem. *Operations Research*, *22*(2), 340-349.

Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, *130*(3), 449-467.

Herer, Y. T., Tzur, M., & Yücesan, E. (2006). The multilocation transshipment problem. *IIE Transactions*, *38*(3), 185-200.

Imran, A., Salhi, S., & Wassan, N. A. (2009). A variable neighborhood-based heuristic for the heterogeneous fleet vehicle routing problem. *European Journal of Operational Research*, *197*(2), 509-518.

Karimi, B., Ghomi, S. F., & Wilson, J. M. (2003). The capacitated lot sizing problem: a review of models and algorithms. *Omega*, *31*(5), 365-378.

Lee, C. G., Bozer, Y. A., & White III, C. C. (2003). *A heuristic approach and properties of optimal solutions to the dynamic inventory routing problem*. Working Paper.

Lei, L., Liu, S., Ruszczynski, A., & Park, S. (2006). On the integrated production, inventory, and distribution routing problem. *IIE Transactions*, *38*(11), 955-970.

Levi, D. S., Kaminsky, P., & Levi, E. S. (2003). *Designing and managing the supply chain: Concepts, strategies, and case studies*. McGraw-Hill.

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, *21*(2), 498-516.

Lysgaard, J., Letchford, A. N., & Eglese, R. W. (2004). A new branch-and-cut algorithm for the capacitated vehicle routing problem. *Mathematical Programming*, *100*(2), 423-445.

Magnanti, T. L., & Wong, R. T. (1981). Accelerating Benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research*, *29*(3), 464-484

Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research*, *24*(11), 1097-1100.

Moin, N. H., & Salhi, S. (2007). Inventory routing problems: a logistical overview. *Journal of the Operational Research Society*, *58*(9), 1185-1194.

Moin, N. H., & Yuliana, T. (2015). Three-phase methodology incorporating scatter search for integrated production, inventory, and distribution routing problem. *Mathematical Problems in Engineering*, *2015*, 1-11.

Mourgaya, M., & Vanderbeck, F. (2007). Column generation based heuristic for tactical planning in multi-period vehicle routing. *European Journal of Operational Research*, *183*(3), 1028-1041.

Nananukul, N. (2008). *Lot-sizing and inventory routing for a production, inventory supply chain*. (Doctoral dissertation, The University of Texas at Austin). Retrieved from http://hdl.handle.net/2152/3960.

Parthanadee, P., & Logendran, R. (2006). Periodic product distribution from multi-depots under limited supplies. *IIE Transactions*, *38*(11), 1009-1026.

Pochet, Y., & Wolsey, L. A. (2006). *Production planning by mixed integer programming*. Springer Science & Business Media.

Prais, M., & Ribeiro, C. C. (2000). Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *INFORMS Journal on Computing*, *12*(3), 164-176.

Resendel, M. G., & Ribeiro, C. C. (2005). GRASP with path-relinking: Recent advances and applications. In *Metaheuristics: progress as real problem solvers* (pp. 29-63). Springer, Boston, MA.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science*, *40*(4), 455-472.

Rosenkrantz, D. J., Stearns, R. E., & Lewis, P. M. (1974, October). Approximate algorithms for the traveling salesperson problem. In *Switching and Automata Theory, 1974., IEEE Conference Record of 15th Annual Symposium on* (pp. 33-42). IEEE.

Ruokokoski, M., Solyali, O. G. U. Z., Cordeau, J. F., Jans, R., & Süral, H. (2010). *Efficient formulations and a branch-and-cut algorithm for a production-routing problem*. GERAD Technical Report G-2010-66. Retrieved from https://pdfs.semanticscholar.org/4232/e0596984f1b71af7fc31f6beda6b92a44438.pdf.

Savelsbergh, M., & Song, J. H. (2007). Inventory routing with continuous moves. *Computers & Operations Research*, *34*(6), 1744-1763.

Simchi-Levi, D., Kaminsky, P., & Simchi-Levi, E. (2003). *Designing and managing the supply chain: Concepts, strategies, and case studies*. McGraw-Hill.

Solyalı, O., & Süral, H. (2008). 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.

Sörensen, K., & Sevaux, M. (2006). MA| PM: memetic algorithms with population management. *Computers & Operations Research*, *33*(5), 1214-1225.

Torabi, S. A., Pourghaderi, A. R., & Sekhavat, S. (2009, December). A two step approach including scatter search algorithm for the integrated procurement, production and distribution planning. In *Industrial Engineering and Engineering Management, 2009. IEEM 2009. IEEE International Conference on* (pp. 354-359). IEEE.

Toth, P., & Vigo, D. (2014). *Vehicle routing: problems, methods and applications*. In 2nd edn, MOS-Siam series on optimization. Siam, Philadelphia.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, *48*(1), 111-128.

Wagner, H. M., & Whitin, T. M. (1958). Dynamic version of the economic lot size model. *Management Science*, *5*(1), 89-96.

Watanabe, H., Cirino, R., Soler, W., & Santos, M. (2017). Solution Methods for the Integrated Production Routing Problem. *Salesian Journal on Information Systems*, *1*(19), 53-61.

Wolsey, L. A. (1998). *Integer Programming. Series in Discrete Mathematics and Optimization*. Wiley Interscience New Jersey.

Zar, J. H. (2009). *Biostatistical Analysis*. Prentice Hall.

LIST OF PUBLICATIONS AND PAPERS PRESENTED

Dicky Lim, T.K. & Moin, N.H. (2018). MatHeuristic approach for production-inventory-distribution routing problem. *Chiang Mai Journal of Science, 45*(2), 1145-1160.

Dicky Lim, T.K. & Moin, N. H. (2016). A two-phase iterative procedure for the production, inventory and distribution routing problem. In *AIP Conference Proceedings* (Vol. 1750, p. 030032). AIP Publishing.

Dicky Lim, T.K. & Moin, N.H. (2015). *An iterative procedure for production-inventory-distribution routing problem*. Paper presented at the 10th Triennial Conference of the Association of Asia-Pacific Operational Research Societies 2015, Kuching, Sarawak.

Dicky Lim, T.K. & Moin, N.H. (2014). *The development of MatHeuristics for production-inventory-distribution routing problem*. Paper presented at the Simposium Kebangsaan Sains Matematik ke-22, Shah Alam, Selangor.

APPENDIX

APPENDIX A: DETAILED RESULTS OF THE ALGORITHMS

Results of MatHeuristic1 Algorithm

TC: Total cost, PS: Production setup cost, PI: Inventory holding cost at the production facility, CI: Holding cost at the customer sites, TC: Transportation cost TV: Total vehicles used

Case N50T20

Instance 1

Run	TC	PS	PI	CI	TC	TV	Time
1	**374264**	**200000**	**95925**	**0**	**78339**	**29**	**60.814**
2	375328	200000	96600	0	78728	30	53.714
3	374983	200000	95490	0	79493	29	64.991
4	374970	200000	97275	0	77695	29	55.129
5	375238	200000	96960	0	78278	29	58.388
6	375670	200000	97860	0	77810	30	53.133
7	379039	200000	99600	0	79439	30	52.983
8	378760	200000	99195	0	79565	30	57.508
9	375062	200000	97500	0	77562	31	57.69
10	374448	200000	95325	0	79123	29	62.413

Instance 2

Run	TC	PS	PI	CI	TC	TV	Time
1	385039	200000	103965	0	81074	30	54.656
2	382683	200000	101520	0	81163	29	51.637
3	383418	200000	102915	0	80503	28	51.422
4	**382679**	**200000**	**100635**	**0**	**82044**	**27**	**53.696**
5	382756	200000	99330	0	83426	27	50.173
6	385409	200000	104580	0	80829	29	60.134
7	385085	200000	104475	0	80610	29	52.135
8	383308	200000	101130	0	82178	28	52.486
9	385221	200000	102480	0	82741	29	53.165
10	385737	200000	104595	0	81142	30	54.999

Instance 3

Run	TC	PS	PI	CI	TC	TV	Time
1	375378	200000	97185	0	78193	30	51.296
2	374452	200000	98685	0	75767	30	56.876
3	375201	200000	99300	0	75901	30	55.068
4	376642	200000	99000	0	77642	31	53.441
5	375187	200000	96795	0	78392	29	56.48
6	**374059**	**200000**	**98715**	**0**	**75344**	**30**	**49.049**
7	375628	200000	97260	0	78368	29	52.306
8	374861	200000	98025	0	76836	30	54.731
9	374326	200000	96375	0	77951	30	52.161
10	375056	200000	99165	0	75891	30	52.765

Instance 4

Run	TC	PS	PI	CI	TC	TV	Time
1	379026	200000	96975	0	82051	33	65.264
2	**377022**	**200000**	**96795**	**0**	**80227**	**33**	**69.877**
3	379275	200000	94695	0	84580	32	59.625
4	381641	200000	99090	0	82551	35	64.669
5	379633	200000	95430	0	84203	32	61.868
6	378183	200000	96810	0	81373	33	63.451
7	377550	200000	94710	0	82840	31	62.186
8	377647	200000	96720	0	80927	32	67.106
9	377752	200000	96540	0	81212	34	65.804
10	378115	200000	97245	0	80870	32	68.365

Instance 5

Run	TC	PS	PI	CI	TC	TV	Time
1	374053	200000	94695	0	79358	28	60.137
2	**372027**	**200000**	**93735**	**0**	**78292**	**28**	**55.638**
3	374082	200000	94305	0	79777	27	60.034
4	374686	200000	94620	0	80066	28	54.847
5	373528	200000	93540	0	79988	28	58.504
6	374290	200000	97170	0	77120	29	63.761
7	376446	200000	97620	0	78826	29	52.018
8	374142	200000	94800	0	79342	28	55.079
9	373709	200000	96180	0	77529	29	55.279
10	375059	200000	97185	0	77874	29	55.527

Instance 6

Run	TC	PS	PI	CI	TC	TV	Time
1	387821	200000	103080	0	84741	31	70.393
2	387310	200000	100005	0	87305	30	60.678
3	388634	200000	105660	0	82974	32	64.063
4	**386968**	**200000**	**102045**	**0**	**84923**	**31**	**64.324**
5	390889	200000	102840	0	88049	31	58.737
6	388925	200000	103395	0	85530	31	60.019
7	387619	200000	102930	0	84689	31	61.131
8	388432	200000	106650	0	81782	31	63.128
9	387329	200000	101745	0	85584	30	59.486
10	389456	200000	105585	0	83871	32	62.978

Instance 7

Run	TC	PS	PI	CI	TC	TV	Time
1	379448	200000	96465	0	82983	27	54.253
2	379916	200000	96705	0	83211	27	48.509
3	382036	200000	102720	0	79316	29	50.522
4	381234	200000	101955	0	79279	29	57.125
5	382112	200000	98610	0	83502	28	46.763
6	381824	200000	101100	0	80724	28	47.53
7	382280	200000	105840	0	76440	30	63.195
8	**377495**	**200000**	**96930**	**0**	**80565**	**28**	**49.008**
9	380121	200000	98625	0	81496	28	47.262
10	380106	200000	101655	0	78451	28	46.416

Instance 8

Run	TC	PS	PI	CI	TC	TV	Time
1	385232	200000	111210	0	74022	30	40.182
2	**383376**	**200000**	**107685**	**0**	**75691**	**28**	**29.003**
3	386581	200000	108225	0	78356	28	30.051
4	387862	200000	111660	0	76202	30	27.839
5	390132	200000	113760	0	76372	30	34.325
6	383503	200000	108375	0	75128	29	29.168
7	384516	200000	106680	0	77836	28	30.45
8	384895	200000	107835	0	77060	28	34.558
9	384359	200000	106155	0	78204	28	30.782
10	384929	200000	110475	0	74454	29	32.961

Instance 9

Run	TC	PS	PI	CI	TC	TV	Time
1	385141	200000	103485	0	81656	31	62.115
2	385878	200000	102105	0	83773	31	60.585
3	387705	200000	103545	0	84160	31	62.861
4	385839	200000	105405	0	80434	32	66.918
5	**383931**	**200000**	**101160**	**0**	**82771**	**31**	**80.277**
6	386961	200000	102150	0	84811	31	60.946
7	388096	200000	106305	0	81791	32	58.419
8	385203	200000	105255	0	79948	33	50.623
9	388643	200000	107220	0	81423	32	61.179
10	385060	200000	102105	0	82955	31	57.434

Instance 10

Run	TC	PS	PI	CI	TC	TV	Time
1	370942	200000	95040	0	75902	28	40.966
2	369977	200000	95235	0	74742	29	39.48
3	**369095**	**200000**	**94260**	**0**	**74835**	**28**	**33.691**
4	371191	200000	97125	0	74066	30	39.996
5	373803	200000	95910	0	77893	28	39.398
6	370419	200000	94500	0	75919	28	33.425
7	370880	200000	94590	0	76290	29	34.69
8	371113	200000	95850	0	75263	30	37.518
9	372347	200000	94755	0	77592	30	34.608
10	371924	200000	97545	0	74379	30	37.684

Instance 11

Run	TC	PS	PI	CI	TC	TV	Time
1	368581	200000	95355	0	73226	28	62.418
2	369744	200000	94560	0	75184	28	53.867
3	367812	200000	93615	0	74197	28	57.738
4	369286	200000	94080	0	75206	28	57.529
5	**367246**	**200000**	**93825**	**0**	**73421**	**28**	**57.523**
6	369011	200000	94260	0	74751	28	58.119
7	368723	200000	95235	0	73488	29	60.682
8	371347	200000	96420	0	74927	30	65.223
9	369570	200000	95505	0	74065	29	56.774
10	369164	200000	93450	0	75714	28	60.701

Instance 12

Run	TC	PS	PI	CI	TC	TV	Time
1	372917	200000	95520	0	77397	28	57.163
2	375809	200000	99210	0	76599	29	66.588
3	372278	200000	95910	0	76368	28	50.821
4	372132	200000	95880	0	76252	29	54.481
5	373973	200000	98820	0	75153	30	54.912
6	371362	200000	94665	0	76697	29	55.162
7	372590	200000	96405	0	76185	29	55.862
8	375634	200000	99420	0	76214	29	58.886
9	**370920**	**200000**	**95970**	**0**	**74950**	**29**	**52.134**
10	373297	200000	92940	0	80357	27	63.894

Instance 13

Run	TC	PS	PI	CI	TC	TV	Time
1	373890	200000	101730	0	72160	30	41.81
2	375152	200000	101820	0	73332	30	35.41
3	375027	200000	101370	0	73657	30	39.094
4	**372797**	**200000**	**99630**	**0**	**73167**	**30**	**30.168**
5	373596	200000	100725	0	72871	29	36.271
6	373753	200000	100635	0	73118	30	43.304
7	373901	200000	99690	0	74211	31	36.103
8	374677	200000	99285	0	75392	28	33.892
9	374700	200000	100530	0	74170	29	34.141
10	374626	200000	101430	0	73196	28	35.439

Instance 14

Run	TC	PS	PI	CI	TC	TV	Time
1	379389	200000	102000	0	77389	31	61.564
2	**378460**	**200000**	**101040**	**0**	**77420**	**31**	**55.876**
3	382204	200000	103875	0	78329	33	60.251
4	380688	200000	102105	0	78583	32	56.426
5	381246	200000	104880	0	76366	33	61.448
6	379336	200000	103680	0	75656	32	56.028
7	380033	200000	102630	0	77403	31	54.413
8	380003	200000	102585	0	77418	32	55.562
9	380882	200000	103530	0	77352	32	70.66
10	379721	200000	103170	0	76551	32	68.616

Instance 15

Run	TC	PS	PI	CI	TC	TV	Time
1	386343	200000	98775	0	87568	31	68.649
2	387415	200000	100260	0	87155	31	63.673
3	**385600**	**200000**	**96495**	**0**	**89105**	**30**	**73.224**
4	388873	200000	102060	0	86813	32	65.588
5	388716	200000	100305	0	88411	31	68.287
6	389258	200000	98730	0	90528	30	64.027
7	385729	200000	96060	0	89669	30	60.682
8	386866	200000	98220	0	88646	30	66.296
9	387740	200000	103140	0	84600	33	72.311
10	386329	200000	96600	0	89729	30	61.733

Instance 16

Run	TC	PS	PI	CI	TC	TV	Time
1	371522	200000	97515	0	74007	27	55.047
2	370407	200000	97740	0	72667	28	51.985
3	**367923**	**200000**	**97080**	**0**	**70843**	**28**	**55.196**
4	369252	200000	97830	0	71422	28	57.855
5	369634	200000	98070	0	71564	28	55.428
6	371096	200000	99150	0	71946	29	48.993
7	369594	200000	94905	0	74689	27	52.252
8	369376	200000	98445	0	70931	30	53.98
9	373041	200000	100905	0	72136	30	55.461
10	371114	200000	98055	0	73059	28	41.726

Instance 17

Run	TC	PS	PI	CI	TC	TV	Time
1	389852	200000	104340	0	85512	30	66.616
2	390136	200000	105210	0	84926	32	64.409
3	390266	200000	106110	0	84156	31	64.009
4	**389281**	**200000**	**104130**	**0**	**85151**	**32**	**64.391**
5	390190	200000	105105	0	85085	31	62.797
6	390236	200000	104835	0	85401	31	61.131
7	393984	200000	106440	0	87544	33	59.503
8	390869	200000	104445	0	86424	31	62.849
9	391889	200000	106125	0	85764	31	63.705
10	392653	200000	105405	0	87248	30	63.263

Instance 18

Run	TC	PS	PI	CI	TC	TV	Time
1	382445	200000	103200	0	79245	31	65.674
2	383285	200000	103410	0	79875	31	58.139
3	382545	200000	101355	0	81190	30	57.922
4	**381641**	**200000**	**99690**	**0**	**81951**	**29**	**59.171**
5	383041	200000	102555	0	80486	31	58.987
6	385876	200000	104115	0	81761	30	60.468
7	385374	200000	102300	0	83074	31	62.214
8	384559	200000	101985	0	82574	29	57.899
9	384337	200000	101775	0	82562	31	57.67
10	385619	200000	103845	0	81774	31	52.924

Instance 19

Run	TC	PS	PI	CI	TC	TV	Time
1	374399	200000	97950	0	76449	29	56.115
2	374836	200000	100395	0	74441	29	55.355
3	374512	200000	100110	0	74402	30	49.941
4	379449	200000	104715	0	74734	32	56.493
5	**373799**	**200000**	**99120**	**0**	**74679**	**30**	**50.988**
6	373982	200000	101595	0	72387	30	52.185
7	374781	200000	100995	0	73786	30	51.554
8	375192	200000	100650	0	74542	30	48.909
9	379101	200000	104040	0	75061	31	53.866
10	375863	200000	100380	0	75483	31	51.169

Instance 20

Run	TC	PS	PI	CI	TC	TV	Time
1	375113	200000	95580	0	79533	29	51.626
2	376352	200000	100350	0	76002	32	50.002
3	374298	200000	99930	0	74368	32	57.729
4	375291	200000	97080	0	78211	31	49.834
5	374012	200000	95295	0	78717	30	46.102
6	377387	200000	98220	0	79167	31	57.142
7	374344	200000	98010	0	76334	30	47.977
8	374260	200000	99060	0	75200	31	48.279
9	372586	200000	93945	0	78641	30	57.208
10	**371942**	**200000**	**95595**	**0**	**76347**	**31**	**59.563**

Instance 21

Run	TC	PS	PI	CI	TC	TV	Time
1	370000	200000	96060	0	73940	31	73.562
2	369745	200000	93765	0	75980	31	59.417
3	368498	200000	92820	0	75678	31	59.67
4	368219	200000	95520	0	72699	31	65.573
5	369644	200000	93690	0	75954	30	62.228
6	371947	200000	95415	0	76532	31	63.814
7	**367850**	**200000**	**92970**	**0**	**74880**	**31**	**60.014**
8	368846	200000	91695	0	77151	29	54.581
9	366690	200000	93885	0	72805	32	61.897
10	367853	200000	95655	0	72198	30	61.897

Instance 22

Run	TC	PS	PI	CI	TC	TV	Time
1	363777	200000	95100	0	68677	30	38.956
2	**360679**	**200000**	**91050**	**0**	**69629**	**28**	**33.793**
3	363581	200000	94860	0	68721	31	38.648
4	362422	200000	94410	0	68012	30	33.81
5	362538	200000	93330	0	69208	30	36.885
6	365314	200000	97605	0	67709	31	33.659
7	359117	200000	88695	0	70422	28	28.769
8	363949	200000	92100	0	71849	29	34.109
9	366223	200000	94995	0	71228	30	38.982
10	364070	200000	95460	0	68610	31	29.235

Instance 23

Run	TC	PS	PI	CI	TC	TV	Time
1	375820	200000	99750	0	76070	29	38.231
2	375072	200000	96495	0	78577	29	35.974
3	**375067**	**200000**	**99780**	**0**	**75287**	**31**	**31.559**
4	376885	200000	100440	0	76445	30	31.912
5	376435	200000	100485	0	75950	30	32.811
6	375667	200000	98655	0	77012	31	33.077
7	376855	200000	98130	0	78725	30	35.173
8	377396	200000	100230	0	77166	30	38.915
9	375917	200000	100530	0	75387	29	33.126
10	375397	200000	99105	0	76292	30	33.344

Instance 24

Run	TC	PS	PI	CI	TC	TV	Time
1	368415	200000	95820	0	72595	29	41.967
2	367100	200000	92985	0	74115	27	31.809
3	368149	200000	95460	0	72689	30	43.653
4	367334	200000	96135	0	71199	29	36.903
5	369327	200000	98385	0	70942	30	36.251
6	370397	200000	96390	0	74007	30	36.637
7	365231	200000	92610	0	72621	28	32.945
8	371114	200000	100185	0	70929	31	40.578
9	**364765**	**200000**	**92415**	**0**	**72350**	**29**	**34.79**
10	371495	200000	96600	0	74895	31	35.505

Instance 25

Run	TC	PS	PI	CI	TC	TV	Time
1	**384851**	**200000**	**104370**	**0**	**80481**	**29**	**64.056**
2	386917	200000	107085	0	79832	29	61.798
3	389059	200000	109275	0	79784	31	59.719
4	389073	200000	108000	0	81073	32	60.667
5	387282	200000	108465	0	78817	31	62.346
6	387759	200000	108630	0	79129	31	60.067
7	385094	200000	107925	0	77169	32	61.265
8	384965	200000	106695	0	78270	31	60.983
9	387579	200000	107340	0	80239	31	59.601
10	387681	200000	108030	0	79651	32	60.583

Instance 26

Run	TC	PS	PI	CI	TC	TV	Time
1	385286	200000	100830	0	84456	30	39.783
2	384907	200000	103290	0	81617	30	35.729
3	**382984**	**200000**	**98700**	**0**	**84284**	**29**	**36.3**
4	385284	200000	99990	0	85294	30	35.354
5	385911	200000	102180	0	83731	30	37.083
6	386309	200000	101940	0	84369	30	37.039
7	385296	200000	101790	0	83506	30	37.862
8	385151	200000	97980	0	87171	29	33.94
9	386132	200000	103815	0	82317	30	36.774
10	388023	200000	101715	0	86308	30	37.025

Instance 27

Run	TC	PS	PI	CI	TC	TV	Time
1	**365470**	**200000**	**94155**	**0**	**71315**	**29**	**59.261**
2	370000	200000	96930	0	73070	31	61.215
3	369761	200000	99540	0	70221	31	59.686
4	368416	200000	94290	0	74126	30	61.464
5	367271	200000	95415	0	71856	30	53.149
6	371959	200000	99960	0	71999	30	60.418
7	369817	200000	98520	0	71297	29	53.317
8	369093	200000	96705	0	72388	31	57.989
9	368619	200000	95640	0	72979	30	55.127
10	369243	200000	96915	0	72328	29	55.545

Instance 28

Run	TC	PS	PI	CI	TC	TV	Time
1	**365761**	**200000**	**93420**	**0**	**72341**	**28**	**60.314**
2	369628	200000	94395	0	75233	28	51.519
3	372688	200000	96630	0	76058	29	55.927
4	366309	200000	94350	0	71959	29	51.27
5	368185	200000	94110	0	74075	27	50.673
6	366134	200000	92235	0	73899	28	51.236
7	368653	200000	96210	0	72443	29	57.907
8	368219	200000	92250	0	75969	27	49.608
9	371337	200000	97845	0	73492	29	65.422
10	366886	200000	92535	0	74351	27	50.473

Instance 29

Run	TC	PS	PI	CI	TC	TV	Time
1	390375	200000	102555	0	87820	30	61.483
2	391800	200000	103635	0	88165	30	58.635
3	392755	200000	104625	0	88130	31	63.033
4	390949	200000	104655	0	86294	30	67.351
5	393242	200000	105660	0	87582	30	64.827
6	393117	200000	105135	0	87982	30	58.929
7	391895	200000	103305	0	88590	31	57.279
8	392704	200000	104385	0	88319	31	54.881
9	**389899**	**200000**	**102735**	**0**	**87164**	**30**	**56.675**
10	392834	200000	104220	0	88614	29	61.967

Instance 30

Run	TC	PS	PI	CI	TC	TV	Time
1	373262	200000	96900	0	76362	30	54.672
2	373729	200000	95940	0	77789	30	52.08
3	**371576**	**200000**	**96570**	**0**	**75006**	**29**	**52.623**
4	371795	200000	93390	0	78405	30	51.163
5	373822	200000	96045	0	77777	30	54.152
6	373772	200000	94155	0	79617	28	59.356
7	372338	200000	96450	0	75888	30	64.654
8	375158	200000	97245	0	77913	31	59.54
9	372806	200000	95115	0	77691	28	54.252
10	375023	200000	95925	0	79098	29	56.073

Case N100T20

Instance 1

Run	TC	PS	PI	CI	TC	TV	Time
1	**684370**	**350000**	**203940**	**0**	**130430**	**61**	**266.581**
2	687994	350000	206445	0	131549	61	237.615
3	691687	350000	211845	0	129842	63	277.621
4	685951	350000	207345	0	128606	62	218.217
5	688163	350000	204240	0	133923	61	264.458
6	689777	350000	205890	0	133887	61	237.386
7	689737	350000	208080	0	131657	61	246.549
8	696319	350000	210975	0	135344	63	287.083
9	688564	350000	206820	0	131744	61	237.863
10	691606	350000	207390	0	134216	61	278.352

Instance 2

Run	TC	PS	PI	CI	TC	TV	Time
1	696072	350000	204795	0	141277	62	299.001
2	701951	350000	206370	0	145581	62	304.517
3	697527	350000	207390	0	140137	64	278.271
4	697921	350000	207525	0	140396	61	257.995
5	**695179**	**350000**	**205215**	**0**	**139964**	**62**	**235.325**
6	697545	350000	203220	0	144325	61	237.27
7	697875	350000	209820	0	138055	61	266.648
8	700846	350000	206025	0	144821	60	260.762
9	703445	350000	209085	0	144360	62	276.514
10	702147	350000	207675	0	144472	60	252.715

Instance 3

Run	TC	PS	PI	CI	TC	TV	Time
1	683461	350000	200700	0	132761	59	274.163
2	684139	350000	202140	0	131999	61	260.086
3	682879	350000	201105	0	131774	61	257.407
4	**679257**	**350000**	**196425**	**0**	**132832**	**60**	**240.289**
5	683681	350000	198225	0	135456	59	229.371
6	681918	350000	196530	0	135388	58	263.957
7	686497	350000	201480	0	135017	61	234.36
8	690960	350000	202695	0	138265	61	280.796
9	687606	350000	197985	0	139621	61	238.197
10	689697	350000	201795	0	137902	61	225.58

Instance 4

Run	TC	PS	PI	CI	TC	TV	Time
1	**659536**	**350000**	**190200**	**0**	**119336**	**61**	**293.935**
2	664774	350000	196920	0	117854	62	225.912
3	660365	350000	188265	0	122100	60	236.836
4	662596	350000	192855	0	119741	60	267.614
5	660890	350000	190020	0	120870	60	241.217
6	667782	350000	193515	0	124267	61	276.476
7	661860	350000	187920	0	123940	61	231.512
8	667558	350000	193020	0	124538	62	279.532
9	669234	350000	191505	0	127729	62	300.876
10	669891	350000	197625	0	122266	61	277.375

Instance 5

Run	TC	PS	PI	CI	TC	TV	Time
1	709620	350000	212910	0	146710	64	315.226
2	711457	350000	215310	0	146147	64	295.104
3	710596	350000	211920	0	148676	65	286.836
4	709793	350000	214215	0	145578	63	280.751
5	711012	350000	211665	0	149347	64	263.959
6	716943	350000	215430	0	151513	64	297.792
7	**706692**	**350000**	**209280**	**0**	**147412**	**62**	**284.901**
8	714342	350000	212910	0	151432	64	300.491
9	714548	350000	215505	0	149043	63	283.008
10	713302	350000	216300	0	147002	64	262.999

Instance 6

Run	TC	PS	PI	CI	TC	TV	Time
1	694747	350000	194880	0	149867	62	332.02
2	694793	350000	199320	0	145473	63	256.314
3	704814	350000	201030	0	153784	65	255.208
4	**693958**	**350000**	**196080**	**0**	**147878**	**64**	**229.259**
5	694041	350000	197160	0	146881	62	256.47
6	700769	350000	197640	0	153129	64	246.313
7	698588	350000	198000	0	150588	63	287.486
8	694429	350000	197685	0	146744	62	253.075
9	695585	350000	196095	0	149490	64	261.242
10	695300	350000	192300	0	153000	62	258.346

Instance 7

Run	TC	PS	PI	CI	TC	TV	Time
1	680951	350000	190515	0	140436	60	326.401
2	679027	350000	191940	0	137087	62	300.58
3	678482	350000	189060	0	139422	61	252.382
4	680215	350000	195180	0	135035	62	249.921
5	684748	350000	199290	0	135458	62	241.353
6	684006	350000	194865	0	139141	63	234.248
7	679774	350000	190395	0	139379	61	261.525
8	680381	350000	192150	0	138231	63	234.251
9	684228	350000	192765	0	141463	61	280.346
10	**677788**	**350000**	**188730**	**0**	**139058**	**61**	**278.828**

Instance 8

Run	TC	PS	PI	CI	TC	TV	Time
1	675129	350000	195600	0	129529	59	282.138
2	673411	350000	195135	0	128276	60	249.378
3	676090	350000	195165	0	130925	60	250.045
4	671787	350000	193635	0	128152	59	248.114
5	675626	350000	193485	0	132141	59	259.963
6	673072	350000	190410	0	132662	59	255.182
7	676635	350000	197160	0	129475	60	243.82
8	679658	350000	193320	0	136338	59	297.633
9	**670607**	**350000**	**191430**	**0**	**129177**	**59**	**250.366**
10	676138	350000	195855	0	130283	60	239.484

Instance 9

Run	TC	PS	PI	CI	TC	TV	Time
1	694296	350000	205740	0	138556	63	326.237
2	691847	350000	201105	0	140742	63	267.155
3	697849	350000	206820	0	141029	64	265.12
4	699588	350000	208065	0	141523	63	272.851
5	693055	350000	202560	0	140495	63	254.892
6	698894	350000	205560	0	143334	63	267.684
7	701173	350000	205605	0	145568	61	280.032
8	**689304**	**350000**	**205020**	**0**	**134284**	**62**	**295.138**
9	691648	350000	201765	0	139883	63	311.038
10	697132	350000	203850	0	143282	62	281.309

Instance 10

Run	TC	PS	PI	CI	TC	TV	Time
1	696625	350000	209325	0	137300	62	310.128
2	697996	350000	202155	0	145841	62	245.527
3	691731	350000	203610	0	138121	61	260.736
4	701801	350000	204540	0	147261	63	287.757
5	696332	350000	208020	0	138312	61	264.602
6	697827	350000	207030	0	140797	63	255.079
7	698225	350000	210195	0	138030	62	254.043
8	698574	350000	208425	0	140149	65	296.155
9	**688089**	**350000**	**200355**	**0**	**137734**	**61**	**255.144**
10	700391	350000	210015	0	140376	63	250.829

Instance 11

Run	TC	PS	PI	CI	TC	TV	Time
1	709381	350000	215445	0	143936	63	322.511
2	707935	350000	209355	0	148580	63	343.494
3	703286	350000	207615	0	145671	62	278.629
4	704981	350000	210330	0	144651	63	274.43
5	**700952**	**350000**	**210120**	**0**	**140832**	**62**	**277.371**
6	705221	350000	210255	0	144966	62	288.923
7	712069	350000	214275	0	147794	63	337.805
8	712724	350000	217695	0	145029	66	303.924
9	702342	350000	209445	0	142897	63	248.932
10	706178	350000	211350	0	144828	63	278.113

Instance 12

Run	TC	PS	PI	CI	TC	TV	Time
1	710337	350000	214605	0	145732	63	342.646
2	709170	350000	211575	0	147595	63	281.283
3	709807	350000	213435	0	146372	63	315.91
4	709095	350000	212610	0	146485	62	279.351
5	712935	350000	217440	0	145495	64	257.323
6	715862	350000	209175	0	156687	63	309.441
7	**706511**	**350000**	**213570**	**0**	**142941**	**65**	**285.921**
8	714381	350000	214485	0	149896	63	315.714
9	711390	350000	213375	0	148015	64	281.896
10	713680	350000	214290	0	149390	63	310.565

Instance 13

Run	TC	PS	PI	CI	TC	TV	Time
1	696876	350000	206985	0	139891	64	334.465
2	694454	350000	196920	0	147534	63	276.625
3	694093	350000	203850	0	140243	64	247.688
4	**692837**	**350000**	**204345**	**0**	**138492**	**63**	**289.216**
5	697204	350000	203685	0	143519	65	303.114
6	700797	350000	205095	0	145702	65	294.5
7	699200	350000	208065	0	141135	64	283.201
8	696002	350000	204375	0	141627	63	288.838
9	697073	350000	209625	0	137448	64	268.11
10	699139	350000	208935	0	140204	64	329.783

Instance 14

Run	TC	PS	PI	CI	TC	TV	Time
1	697892	350000	205290	0	142602	61	324.604
2	**697085**	**350000**	**206535**	**0**	**140550**	**62**	**288.119**
3	699844	350000	207090	0	142754	62	253.807
4	699549	350000	204750	0	144799	62	279.073
5	702444	350000	208005	0	144439	63	291.265
6	701748	350000	208005	0	143743	63	252.306
7	700838	350000	206010	0	144828	63	263.204
8	697564	350000	205920	0	141644	62	246.312
9	697431	350000	204570	0	142861	62	241.986
10	699613	350000	203400	0	146213	62	269.037

Instance 15

Run	TC	PS	PI	CI	TC	TV	Time
1	684741	350000	204765	0	129976	61	317.268
2	687744	350000	205785	0	131959	62	246.197
3	680940	350000	197220	0	133720	61	255.366
4	684703	350000	205710	0	128993	64	250.481
5	682688	350000	198405	0	134283	61	272.92
6	684975	350000	203580	0	131395	60	232.784
7	**680391**	**350000**	**200160**	**0**	**130231**	**60**	**243.058**
8	685506	350000	200355	0	135151	59	264.989
9	688307	350000	203220	0	135087	62	242.04
10	682228	350000	200415	0	131813	61	244.131

Instance 16

Run	TC	PS	PI	CI	TC	TV	Time
1	710088	350000	211650	0	148438	62	337.817
2	709805	350000	211275	0	148530	62	289.908
3	716393	350000	212325	0	154068	61	305.974
4	708461	350000	207660	0	150801	62	354.896
5	718607	350000	211050	0	157557	62	376.238
6	712786	350000	212265	0	150521	63	357.654
7	711399	350000	209160	0	152239	63	346.844
8	719017	350000	213015	0	156002	62	347.407
9	716942	350000	213330	0	153612	64	300.251
10	**706658**	**350000**	**206385**	**0**	**150273**	**61**	**303.964**

Instance 17

Run	TC	PS	PI	CI	TC	TV	Time
1	**716302**	**420000**	**213450**	**0**	**82852**	**64**	**378.958**
2	720890	420000	219450	0	81440	64	444.136
3	719254	420000	217020	0	82234	63	298.147
4	717102	420000	216330	0	80772	65	321.611
5	717592	420000	215220	0	82372	62	328.536
6	726043	420000	220245	0	85798	64	391.434
7	728369	420000	219660	0	88709	63	334.914
8	721097	420000	216210	0	84887	64	306.611
9	728553	420000	221085	0	87468	64	275.754
10	722632	420000	218280	0	84352	63	338.24

Instance 18

Run	TC	PS	PI	CI	TC	TV	Time
1	687034	350000	197520	0	139514	59	326.215
2	687240	350000	198945	0	138295	62	277.122
3	687673	350000	198285	0	139388	61	289.611
4	**682024**	**350000**	**192900**	**0**	**139124**	**59**	**254.66**
5	689225	350000	196485	0	142740	63	245.779
6	688363	350000	196620	0	141743	61	281.906
7	687579	350000	198360	0	139219	60	287.42
8	692303	350000	200160	0	142143	62	267.712
9	693517	350000	202695	0	140822	62	317.093
10	687546	350000	200670	0	136876	60	285.288

Instance 19

Run	TC	PS	PI	CI	TC	TV	Time
1	696774	350000	214185	0	132589	62	335.815
2	698012	350000	215100	0	132912	62	261.524
3	696437	350000	211740	0	134697	63	277.22
4	699498	350000	218835	0	130663	63	251.793
5	694791	350000	214500	0	130291	63	273.05
6	702539	350000	215790	0	136749	63	289.456
7	694113	350000	211200	0	132913	61	256.318
8	694298	350000	210120	0	134178	62	251.26
9	696516	350000	212655	0	133861	63	290.926
10	**693967**	**350000**	**216210**	**0**	**127757**	**62**	**260.592**

Instance 20

Run	TC	PS	PI	CI	TC	TV	Time
1	720552	350000	216240	0	154312	65	385.446
2	**715222**	**350000**	**212955**	**0**	**152267**	**63**	**296.216**
3	722343	350000	218715	0	153628	66	293.322
4	719155	350000	215925	0	153230	65	279.07
5	727870	350000	220320	0	157550	66	269.093
6	720567	350000	216465	0	154102	64	272.048
7	719835	350000	216750	0	153085	65	276.175
8	721915	350000	216510	0	155405	64	277.235
9	719010	350000	215610	0	153400	67	272.811
10	720580	350000	213330	0	157250	63	283.594

Instance 21

Run	TC	PS	PI	CI	TC	TV	Time
1	707889	350000	210300	0	147589	64	338.775
2	709167	350000	206835	0	152332	64	275.686
3	712844	350000	214980	0	147864	65	259.039
4	713838	350000	214050	0	149788	64	237.859
5	706229	350000	210180	0	146049	63	277.458
6	**700373**	**350000**	**198060**	**0**	**152313**	**63**	**313.813**
7	704666	350000	206895	0	147771	63	239.007
8	711073	350000	207840	0	153233	65	269.483
9	708837	350000	211665	0	147172	65	257.975
10	706958	350000	208875	0	148083	64	280.617

Instance 22

Run	TC	PS	PI	CI	TC	TV	Time
1	**706492**	**350000**	**213645**	**0**	**142847**	**61**	**290.829**
2	710966	350000	217035	0	143931	62	235.565
3	706624	350000	214380	0	142244	62	251.321
4	714564	350000	216480	0	148084	62	269.235
5	707589	350000	213915	0	143674	62	229.19
6	712897	350000	214740	0	148157	62	258.48
7	707567	350000	211440	0	146127	62	245.837
8	709057	350000	213870	0	145187	62	257.906
9	710792	350000	213570	0	147222	61	276.403
10	715210	350000	219315	0	145895	63	298.986

Instance 23

Run	TC	PS	PI	CI	TC	TV	Time
1	686350	350000	197415	0	138935	63	317.442
2	685798	350000	195390	0	140408	60	254.689
3	683841	350000	193650	0	140191	60	241.065
4	**680172**	**350000**	**192060**	**0**	**138112**	**60**	**240.876**
5	694223	350000	201210	0	143013	62	288.857
6	685057	350000	199185	0	135872	61	271.935
7	684989	350000	192225	0	142764	61	251.1
8	692349	350000	205935	0	136414	63	241.236
9	684390	350000	195810	0	138580	60	284.615
10	685065	350000	195405	0	139660	61	278.178

Instance 24

Run	TC	PS	PI	CI	TC	TV	Time
1	703435	350000	208455	0	144980	62	366.191
2	696786	350000	205665	0	141121	62	283.661
3	704261	350000	210615	0	143646	64	283.343
4	702887	350000	206985	0	145902	62	302.65
5	702687	350000	212925	0	139762	63	286.555
6	**696555**	**350000**	**208335**	**0**	**138220**	**61**	**280.682**
7	705860	350000	207780	0	148080	64	277.989
8	703918	350000	210315	0	143603	63	347.568
9	699696	350000	205065	0	144631	63	294.471
10	704220	350000	209115	0	145105	62	329.659

Instance 25

Run	TC	PS	PI	CI	TC	TV	Time
1	701142	350000	207330	0	143812	63	337.419
2	703783	350000	206640	0	147143	65	246.23
3	704290	350000	207330	0	146960	63	269.259
4	699771	350000	209040	0	140731	63	245.36
5	703763	350000	208170	0	145593	63	237.86
6	**695738**	**350000**	**205920**	**0**	**139818**	**62**	**270.837**
7	704771	350000	210015	0	144756	65	289.098
8	706415	350000	208275	0	148140	63	297.545
9	709918	350000	215460	0	144458	65	280.171
10	706296	350000	203925	0	152371	62	298.959

Instance 26

Run	TC	PS	PI	CI	TC	TV	Time
1	693247	350000	205575	0	137672	61	297.893
2	689677	350000	202575	0	137102	61	237.289
3	**689543**	**350000**	**207780**	**0**	**131763**	**62**	**258.169**
4	693857	350000	211890	0	131967	62	250.266
5	692017	350000	205815	0	136202	61	236.637
6	689747	350000	207210	0	132537	62	249.224
7	690098	350000	208395	0	131703	61	243.667
8	695061	350000	209730	0	135331	62	271.386
9	692058	350000	209070	0	132988	62	227.692
10	692363	350000	208350	0	134013	61	251.48

Instance 27

Run	TC	PS	PI	CI	TC	TV	Time
1	688356	350000	199620	0	138736	62	322.53
2	688581	350000	200175	0	138406	63	286.096
3	691877	350000	204945	0	136932	63	305.925
4	**684183**	**350000**	**199290**	**0**	**134893**	**62**	**261.191**
5	694135	350000	207000	0	137135	63	291.281
6	687652	350000	197175	0	140477	61	264.804
7	699118	350000	205590	0	143528	63	296.209
8	688539	350000	202410	0	136129	63	297.3
9	694065	350000	203280	0	140785	63	267.392
10	689468	350000	201660	0	137808	62	264.006

Instance 28

Run	TC	PS	PI	CI	TC	TV	Time
1	688962	350000	200835	0	138127	61	293.222
2	687753	350000	201375	0	136378	60	262.784
3	686870	350000	200520	0	136350	62	253.693
4	**685277**	**350000**	**195780**	**0**	**139497**	**60**	**250**
5	686811	350000	197115	0	139696	60	266.572
6	689426	350000	199635	0	139791	62	253.359
7	686814	350000	197655	0	139159	60	246.116
8	693730	350000	200655	0	143075	63	262.378
9	690757	350000	201270	0	139487	61	300.86
10	685830	350000	198600	0	137230	60	234.837

Instance 29

Run	TC	PS	PI	CI	TC	TV	Time
1	**699233**	**350000**	**205065**	**0**	**144168**	**62**	**338.053**
2	705641	350000	208125	0	147516	61	275.617
3	705588	350000	203820	0	151768	63	275.971
4	706280	350000	207390	0	148890	62	263.415
5	709665	350000	214890	0	144775	63	272.678
6	703704	350000	209415	0	144289	62	272.555
7	706953	350000	207000	0	149953	63	264.674
8	708311	350000	207315	0	150996	63	260.465
9	704915	350000	208140	0	146775	61	270.401
10	716923	350000	216540	0	150383	65	311.69

Instance 30

Run	TC	PS	PI	CI	TC	TV	Time
1	**702675**	**420000**	**199875**	**0**	**82800**	**62**	**302.507**
2	705644	420000	198600	0	87044	62	288.135
3	707544	420000	201075	0	86469	62	275.999
4	708381	420000	202365	0	86016	63	331.224
5	708818	420000	199665	0	89153	64	310.597
6	714943	420000	202695	0	92248	62	306.348
7	708585	420000	197820	0	90765	60	346.27
8	716231	420000	204930	0	91301	63	380.761
9	709974	420000	198855	0	91119	60	290.586
10	712004	420000	202170	0	89834	63	325.011

Case N200T20

Instance 1

Run	TC	PS	PI	CI	TC	TV	Time
1	1007834	720000	253500	0	34334	74	1525.95
2	1007008	720000	252300	0	34708	75	2147.77
3	1013442	720000	255225	0	38217	76	1891.41
4	1010310	720000	252090	0	38220	75	1443.21
5	1017517	720000	258120	0	39397	75	1652.34
6	1005463	720000	251670	0	33793	75	2366.62
7	1016934	720000	254295	0	42639	75	2613.59
8	1011030	720000	251460	0	39570	74	1514.03
9	**1005064**	**720000**	**250950**	**0**	**34114**	**74**	**1792.29**
10	1009006	720000	251295	0	37711	74	1810.87

Instance 2

Run	TC	PS	PI	CI	TC	TV	Time
1	1012820	720000	228585	0	64235	73	1557.47
2	1021148	720000	234525	0	66623	72	1167.56
3	1013211	720000	232590	0	60621	72	1239.43
4	1013339	720000	229635	0	63704	71	1437.3
5	1017514	720000	233730	0	63784	70	1400.94
6	1013476	720000	229290	0	64186	73	1225.46
7	1021890	720000	234255	0	67635	72	1274.6
8	1016955	720000	229005	0	67950	72	1380.17
9	**1008993**	**720000**	**225825**	**0**	**63168**	**70**	**1334.95**
10	1015539	720000	232635	0	62904	73	1360.83

Instance 3

Run	TC	PS	PI	CI	TC	TV	Time
1	1001139	720000	247860	0	33279	73	1312.81
2	1001579	720000	245775	0	35804	72	1217.85
3	999467	720000	246045	0	33422	73	1252.81
4	1008206	720000	248355	0	39851	73	1325.74
5	1005360	720000	249705	0	35655	72	1182.67
6	1010011	720000	249315	0	40696	75	1225.07
7	**996844**	**720000**	**241950**	**0**	**34894**	**72**	**1179.64**
8	1007950	720000	251805	0	36145	76	1422.63
9	1009794	720000	253755	0	36039	75	1398.26
10	1012217	720000	252735	0	39482	74	1497.58

Instance 4

Run	TC	PS	PI	CI	TC	TV	Time
1	997482	720000	244680	0	32802	73	1241.72
2	995561	720000	239055	0	36506	74	1517.8
3	**994712**	**720000**	**241575**	**0**	**33137**	**73**	**1306.3**
4	1007470	720000	249870	0	37600	75	1378
5	1000463	720000	242730	0	37733	73	1356.69
6	998105	720000	240495	0	37610	73	1506.15
7	1006725	720000	245505	0	41220	74	1525.74
8	999455	720000	244065	0	35390	73	1570.82
9	1003380	720000	248220	0	35160	75	1834.11
10	1002382	720000	244200	0	38182	73	1461.38

Instance 5

Run	TC	PS	PI	CI	TC	TV	Time
1	1000694	720000	246240	0	34454	74	1379.56
2	1006978	720000	242880	0	44098	73	1199.14
3	1001309	720000	239505	0	41804	74	1316.33
4	1005351	720000	246420	0	38931	74	1314.96
5	1004341	720000	244785	0	39556	74	1334.29
6	1007857	720000	246120	0	41737	76	2246.09
7	1005398	720000	245460	0	39938	75	1470.76
8	**999114**	**720000**	**242115**	**0**	**36999**	**74**	**1475.75**
9	999838	720000	242970	0	36868	74	1457.9
10	1002562	720000	246045	0	36517	74	1539.34

Instance 6

Run	TC	PS	PI	CI	TC	TV	Time
1	1011856	720000	245460	0	46396	75	1424.99
2	1007608	720000	242070	0	45538	76	1119.83
3	1006899	720000	241860	0	45039	75	1157.36
4	**1006016**	**720000**	**241785**	**0**	**44231**	**74**	**1005.4**
5	1009466	720000	243090	0	46376	75	1507.21
6	1014429	720000	248145	0	46284	74	1586.84
7	1008016	720000	246600	0	41416	76	1199.42
8	1010500	720000	244545	0	45955	74	1278.06
9	1014825	720000	247110	0	47715	76	1082.78
10	1008109	720000	242940	0	45169	75	1115.6

Instance 7

Run	TC	PS	PI	CI	TC	TV	Time
1	1018953	720000	254790	0	44163	75	1360.28
2	1011535	720000	248625	0	42910	74	1362.32
3	1014454	720000	250380	0	44074	74	1481.16
4	1017591	720000	251520	0	46071	73	1106.73
5	1015045	720000	248055	0	46990	74	1344.81
6	1013499	720000	249975	0	43524	73	1443.48
7	1011706	720000	248370	0	43336	74	1114.08
8	1019218	720000	252900	0	46318	75	1346.32
9	1018844	720000	251310	0	47534	74	1303.63
10	**1008156**	**720000**	**246840**	**0**	**41316**	**73**	**1360.5**

Instance 8

Run	TC	PS	PI	CI	TC	TV	Time
1	999169	720000	242520	0	36649	75	1548.57
2	994683	720000	237810	0	36873	76	1413.75
3	995939	720000	237825	0	38114	75	1392.02
4	997390	720000	240555	0	36835	77	1414.93
5	999089	720000	240480	0	38609	76	1318.01
6	995083	720000	238230	0	36853	76	1494.92
7	**992901**	**720000**	**234795**	**0**	**38106**	**75**	**1280.04**
8	999231	720000	240600	0	38631	76	1386.08
9	997478	720000	240075	0	37403	77	1490.08
10	998830	720000	241200	0	37630	76	1480.72

Instance 9

Run	TC	PS	PI	CI	TC	TV	Time
1	989082	720000	233580	0	35502	72	1487.93
2	990925	720000	234720	0	36205	72	1360.51
3	991376	720000	238080	0	33296	73	1337.41
4	991225	720000	233850	0	37375	72	1441.11
5	992325	720000	235185	0	37140	72	1410.35
6	992329	720000	236190	0	36139	73	1485.35
7	992196	720000	236010	0	36186	72	1659.98
8	**987864**	**720000**	**232185**	**0**	**35679**	**71**	**1401.45**
9	995794	720000	237960	0	37834	74	1380.53
10	993857	720000	237855	0	36002	72	1335.93

Instance 10

Run	TC	PS	PI	CI	TC	TV	Time
1	**999130**	**720000**	**240045**	**0**	**39085**	**74**	**1179.54**
2	1005171	720000	240315	0	44856	74	1244.71
3	1003761	720000	242610	0	41151	74	1400.35
4	1003397	720000	242175	0	41222	74	1237.77
5	1007422	720000	245970	0	41452	76	1351.31
6	1005666	720000	244005	0	41661	75	1477.99
7	1012172	720000	246060	0	46112	76	1930.97
8	1007018	720000	244155	0	42863	74	1753.49
9	1008815	720000	250230	0	38585	76	1549.1
10	1004896	720000	243030	0	41866	74	1360.19

Instance 11

Run	TC	PS	PI	CI	TC	TV	Time
1	1001741	720000	243105	0	38636	75	1615
2	1008685	720000	247395	0	41290	76	1496.43
3	1001318	720000	241710	0	39608	74	1337.22
4	1006906	720000	245910	0	40996	74	1458.88
5	999274	720000	245415	0	33859	75	1251.56
6	997974	720000	236835	0	41139	73	1257.86
7	**996692**	**720000**	**240390**	**0**	**36302**	**73**	**1589.98**
8	1004745	720000	246495	0	38250	75	1340.52
9	1014388	720000	243300	0	51088	74	1562.19
10	1001456	720000	245730	0	35726	75	1365.86

Instance 12

Run	TC	PS	PI	CI	TC	TV	Time
1	1003824	720000	246825	0	36999	73	1198.94
2	1010311	720000	253800	0	36511	74	1289.55
3	1004720	720000	243225	0	41495	73	1036.28
4	1002998	720000	245190	0	37808	73	1300.36
5	1006970	720000	244215	0	42755	73	1221.1
6	1006055	720000	244785	0	41270	74	1291.25
7	**1002124**	**720000**	**244695**	**0**	**37429**	**73**	**1269.62**
8	1012685	720000	247995	0	44690	76	1335.78
9	1009045	720000	249495	0	39550	75	1462.69
10	1004209	720000	247395	0	36814	74	1374.54

Instance 13

Run	TC	PS	PI	CI	TC	TV	Time
1	**1008080**	**720000**	**251595**	**0**	**36485**	**74**	**1545.69**
2	1008875	720000	252900	0	35975	74	1315.83
3	1015667	720000	259290	0	36377	75	1435.54
4	1009870	720000	251010	0	38860	74	1351.73
5	1011809	720000	251790	0	40019	74	1606.71
6	1013491	720000	255645	0	37846	74	1321.36
7	1010629	720000	253785	0	36844	74	1222.25
8	1013135	720000	255795	0	37340	76	1252.69
9	1013944	720000	258030	0	35914	75	1336.86
10	1014672	720000	255795	0	38877	74	1285.57

Instance 14

Run	TC	PS	PI	CI	TC	TV	Time
1	**1002613**	**720000**	**248280**	**0**	**34333**	**75**	**1244.73**
2	1012684	720000	249270	0	43414	75	1699.35
3	1011565	720000	248745	0	42820	75	1328.13
4	1012277	720000	255150	0	37127	75	1612.33
5	1010269	720000	252735	0	37534	75	1440.55
6	1005228	720000	246915	0	38313	74	1404.01
7	1010136	720000	251280	0	38856	75	1293.72
8	1015568	720000	252195	0	43373	74	1565.79
9	1007461	720000	249420	0	38041	75	1355.75
10	1011718	720000	252150	0	39568	75	1613.33

Instance 15

Run	TC	PS	PI	CI	TC	TV	Time
1	1001901	720000	243540	0	38361	72	1341.55
2	**999784**	**720000**	**242235**	**0**	**37549**	**74**	**1214.4**
3	1005298	720000	243555	0	41743	74	1128.54
4	1006007	720000	241590	0	44417	74	1114.14
5	999958	720000	242310	0	37648	74	1266.52
6	1005960	720000	244515	0	41445	74	1247.01
7	1006748	720000	245445	0	41303	74	1452.89
8	1008573	720000	247695	0	40878	75	1699.93
9	1006307	720000	241785	0	44522	75	1227.77
10	1001256	720000	240945	0	40311	74	1164.97

Instance 16

Run	TC	PS	PI	CI	TC	TV	Time
1	1001591	720000	243450	0	38141	74	1535.44
2	1004779	720000	245100	0	39679	74	1277.76
3	1005005	720000	245535	0	39470	74	1896.67
4	1000898	720000	244065	0	36833	74	1275.51
5	1003763	720000	245430	0	38333	74	1366.56
6	1003648	720000	244965	0	38683	74	1608.1
7	**998922**	**720000**	**242085**	**0**	**36837**	**74**	**1379.95**
8	1002676	720000	244590	0	38086	74	1308.17
9	1004028	720000	244785	0	39243	74	1541.61
10	1002729	720000	244410	0	38319	74	1324.75

Instance 17

Run	TC	PS	PI	CI	TC	TV	Time
1	998977	720000	246960	0	32017	74	1184.84
2	999769	720000	248910	0	30859	75	1142.08
3	999836	720000	248730	0	31106	75	1256.41
4	**998596**	**720000**	**245085**	**0**	**33511**	**74**	**1123**
5	1002097	720000	246690	0	35407	74	1373.21
6	1000873	720000	246855	0	34018	76	1292.35
7	1002063	720000	248985	0	33078	75	1241.86
8	1002814	720000	248295	0	34519	76	1211.45
9	1000966	720000	248160	0	32806	74	1209.74
10	999362	720000	245385	0	33977	74	1158.32

Instance 18

Run	TC	PS	PI	CI	TC	TV	Time
1	1008279	720000	247095	0	41184	73	1358.64
2	1012376	720000	247350	0	45026	73	1330.99
3	1010276	720000	248055	0	42221	74	1329.26
4	1010957	720000	247905	0	43052	74	1174.99
5	1011197	720000	246735	0	44462	74	1144.1
6	1009427	720000	246180	0	43247	73	1167.35
7	1017983	720000	249225	0	48758	74	1167.74
8	**1005516**	**720000**	**241215**	**0**	**44301**	**73**	**1178.85**
9	1012146	720000	248205	0	43941	74	1162.82
10	1009984	720000	245700	0	44284	74	1335.09

Instance 19

Run	TC	PS	PI	CI	TC	TV	Time
1	1020592	720000	251100	0	49492	76	1625.65
2	1015310	720000	254370	0	40940	76	1253.3
3	1017342	720000	252390	0	44952	76	1379.61
4	1016069	720000	251385	0	44684	76	1320.96
5	1018139	720000	253890	0	44249	77	1379.32
6	1010477	720000	245730	0	44747	75	1247.86
7	**1007677**	**720000**	**247035**	**0**	**40642**	**75**	**1655.44**
8	1020225	720000	252195	0	48030	77	1356.95
9	1020584	720000	253350	0	47234	76	2200.49
10	1011221	720000	250710	0	40511	76	1250.93

Instance 20

Run	TC	PS	PI	CI	TC	TV	Time
1	1005714	720000	242760	0	42954	73	1433.02
2	1006333	720000	245055	0	41278	72	1466.47
3	1009971	720000	247245	0	42726	73	1463.76
4	1007590	720000	246930	0	40660	73	1334.69
5	1007899	720000	246795	0	41104	73	1280.8
6	1005438	720000	247185	0	38253	72	1299.69
7	1013821	720000	252300	0	41521	74	1360.54
8	1009748	720000	246270	0	43478	73	1350.12
9	1006749	720000	247455	0	39294	73	1316.46
10	**1003726**	**720000**	**246165**	**0**	**37561**	**74**	**1477.91**

Instance 21

Run	TC	PS	PI	CI	TC	TV	Time
1	995350	720000	240480	0	34870	74	1491.87
2	1002568	720000	243990	0	38578	74	1455.49
3	**993916**	**720000**	**237975**	**0**	**35941**	**73**	**1305.76**
4	996659	720000	240975	0	35684	73	1457.74
5	999629	720000	242775	0	36854	74	1324.3
6	1003820	720000	246300	0	37520	75	2214.94
7	1007360	720000	247545	0	39815	76	1419.95
8	998376	720000	241035	0	37341	75	1559.59
9	997068	720000	243345	0	33723	74	1423.79
10	1009962	720000	249285	0	40677	74	1622.34

Instance 22

Run	TC	PS	PI	CI	TC	TV	Time
1	992408	720000	238380	0	34028	73	1608.66
2	995139	720000	243090	0	32049	74	1538.86
3	993523	720000	239145	0	34378	74	1567.27
4	992132	720000	239595	0	32537	73	1611.63
5	993978	720000	239790	0	34188	74	1488.5
6	989916	720000	238350	0	31566	73	1770.97
7	995133	720000	243180	0	31953	74	1672.13
8	1000345	720000	242850	0	37495	75	1624.93
9	**987482**	**720000**	**235500**	**0**	**31982**	**73**	**1387.92**
10	992225	720000	238800	0	33425	73	1394.08

Instance 23

Run	TC	PS	PI	CI	TC	TV	Time
1	1003619	720000	242295	0	41324	73	1461.13
2	1009161	720000	248400	0	40761	75	1453.95
3	1008741	720000	249510	0	39231	75	1398.6
4	1007136	720000	248085	0	39051	74	1330.14
5	1009629	720000	248670	0	40959	74	1283.55
6	1004627	720000	244725	0	39902	75	1421.24
7	1007210	720000	249690	0	37520	74	1467.56
8	1005769	720000	247725	0	38044	75	1303.24
9	1004518	720000	246210	0	38308	74	1483.87
10	**1003350**	**720000**	**246285**	**0**	**37065**	**75**	**1225.39**

Instance 24

Run	TC	PS	PI	CI	TC	TV	Time
1	**998107**	**720000**	**241620**	**0**	**36487**	**74**	**1504.69**
2	1000176	720000	242160	0	38016	75	1332.49
3	998557	720000	239665	0	38892	74	1191.15
4	999649	720000	238800	0	40849	75	1266.01
5	1005872	720000	244965	0	40907	76	1514.2
6	1000639	720000	244815	0	35824	75	1589.1
7	1000774	720000	242145	0	38629	75	1401.02
8	1000378	720000	241230	0	39148	75	1478.13
9	1006553	720000	242325	0	44228	75	1268.2
10	1004558	720000	243900	0	40658	76	1408.2

Instance 25

Run	TC	PS	PI	CI	TC	TV	Time
1	**996726**	**720000**	**242310**	**0**	**34416**	**74**	**1395.48**
2	1004767	720000	244575	0	40192	75	1631.54
3	1005485	720000	246750	0	38735	75	1394.38
4	999026	720000	245745	0	33281	76	1703.89
5	997180	720000	244065	0	33115	74	1225.43
6	999306	720000	242700	0	36606	74	1324.63
7	1002091	720000	247110	0	34981	76	1776.37
8	999382	720000	244245	0	35137	74	1093.36
9	997044	720000	241920	0	35124	75	1463.18
10	998563	720000	246165	0	32398	75	1475.53

Instance 26

Run	TC	PS	PI	CI	TC	TV	Time
1	1021908	720000	245820	0	56088	74	1655.26
2	1019890	720000	244740	0	55150	74	1647.63
3	1022053	720000	245730	0	56323	75	1468.96
4	1024612	720000	249345	0	55267	75	1715.87
5	1020710	720000	243765	0	56945	75	1524.36
6	1019110	720000	244950	0	54160	74	1416.3
7	1027804	720000	246975	0	60829	74	1879.49
8	**1017962**	**720000**	**245400**	**0**	**52562**	**75**	**1379.68**
9	1022264	720000	247185	0	55079	74	1476.74
10	1026346	720000	251280	0	55066	75	1574.76

Instance 27

Run	TC	PS	PI	CI	TC	TV	Time
1	1014517	720000	255675	0	38842	75	1439.51
2	**1014516**	**720000**	**255225**	**0**	**39291**	**75**	**1318.05**
3	1014558	720000	256320	0	38238	75	1541.13
4	1015040	720000	255000	0	40040	75	1029.5
5	1020401	720000	258165	0	42236	76	1592.6
6	1017887	720000	257415	0	40472	75	1296.46
7	1016594	720000	256320	0	40274	74	1299.93
8	1017443	720000	257235	0	40208	75	1299.14
9	1019329	720000	257535	0	41794	75	1416.82
10	1018600	720000	258825	0	39775	76	1500.81

Instance 28

Run	TC	PS	PI	CI	TC	TV	Time
1	999135	720000	244380	0	34755	75	1310.74
2	996006	720000	239550	0	36456	74	1285.91
3	999761	720000	241350	0	38411	75	1141.18
4	1004091	720000	244725	0	39366	75	1508.53
5	**994009**	**720000**	**240645**	**0**	**33364**	**74**	**1066.5**
6	998043	720000	242775	0	35268	75	1202.89
7	999551	720000	241845	0	37706	75	1144.48
8	1007776	720000	242055	0	45721	75	1510.01
9	996864	720000	240840	0	36024	76	1299.53
10	1004167	720000	240750	0	43417	76	1300.64

Instance 29

Run	TC	PS	PI	CI	TC	TV	Time
1	1008410	720000	245400	0	43010	77	1456.73
2	1012561	720000	245505	0	47056	76	1182.15
3	1009996	720000	246765	0	43231	76	1480.18
4	**1005629**	**720000**	**243585**	**0**	**42044**	**75**	**1221.03**
5	1010254	720000	243405	0	46849	76	1196.22
6	1007099	720000	245850	0	41249	77	1295.74
7	1008088	720000	246240	0	41848	77	1346.72
8	1006350	720000	241845	0	44505	77	1499.08
9	1011540	720000	243810	0	47730	77	1525.4
10	1006205	720000	240555	0	45650	74	1337.67

Instance 30

Run	TC	PS	PI	CI	TC	TV	Time
1	1003744	720000	244425	0	39319	76	1100.47
2	1003033	720000	244200	0	38833	76	1198.41
3	**1001252**	**720000**	**244710**	**0**	**36542**	**75**	**1145.68**
4	1004090	720000	243225	0	40865	75	1398.48
5	1003364	720000	245445	0	37919	76	1231.43
6	1006462	720000	245835	0	40627	75	1451.26
7	1004200	720000	245820	0	38380	76	1665.35
8	1010997	720000	249150	0	41847	77	1208.57
9	1014228	720000	251895	0	42333	77	1307.75
10	1005951	720000	245985	0	39966	75	1202.39

Results of MatHeuristic2 Algorithm

Case N50T20

Instance 1

Run	TC	PS	PI	CI	TC	TV	Time
1	383348	200000	99660	0	83688	32	128.011
2	372245	200000	94140	0	78105	32	87.46
3	373567	200000	99015	0	74552	31	76.365
4	386979	200000	103140	0	83839	33	100.12
5	381011	200000	99450	0	81561	33	117.896
6	371126	200000	98985	0	72141	31	135.265
7	374857	200000	97155	0	77702	32	81.653
8	375158	200000	97845	0	77313	31	80.677
9	371146	200000	98910	0	72236	32	83.992
10	**370875**	**200000**	**96000**	**0**	**74875**	**31**	**95.414**

Instance 2

Run	TC	PS	PI	CI	TC	TV	Time
1	389580	200000	111195	0	78385	31	110.306
2	390399	200000	108975	0	81424	31	75.316
3	381423	200000	109890	0	71533	30	93.236
4	382677	200000	108135	0	74542	30	93.94
5	389093	200000	109965	0	79128	31	92.171
6	382426	200000	110085	0	72341	31	88.855
7	379312	200000	107370	0	71942	31	91.499
8	386027	200000	109395	0	76632	30	125.292
9	384684	200000	105885	0	78799	30	96.986
10	**375111**	**200000**	**106710**	**0**	**68401**	**30**	**103.324**

Instance 3

Run	TC	PS	PI	CI	TC	TV	Time
1	374485	200000	102030	0	72455	31	92.74
2	385909	200000	102180	0	83729	32	83.467
3	379519	200000	103230	0	76289	31	75.161
4	379809	200000	101595	0	78214	31	80.695
5	387232	200000	105675	0	81557	32	106.637
6	383716	200000	108000	0	75716	32	121.49
7	**371135**	**200000**	**103860**	**0**	**67275**	**30**	**83.046**
8	374903	200000	103650	0	71253	32	76.003
9	379730	200000	104250	0	75480	32	91.708
10	379396	200000	103425	0	75971	32	116.854

Instance 4

Run	TC	PS	PI	CI	TC	TV	Time
1	373892	200000	97275	0	76617	31	116.466
2	373016	200000	94650	0	78366	32	139.946
3	377044	200000	97410	0	79634	32	124.662
4	380329	200000	99795	0	80534	32	128.935
5	369382	200000	97005	0	72377	31	128.332
6	367545	200000	96885	0	70660	30	129.602
7	377082	200000	97740	0	79342	32	119.348
8	**366328**	**200000**	**97185**	**0**	**69143**	**30**	**114.129**
9	380410	200000	102240	0	78170	34	73.663
10	381314	200000	96615	0	84699	31	103.347

Instance 5

Run	TC	PS	PI	CI	TC	TV	Time
1	**373022**	**200000**	**96300**	**0**	**76722**	**30**	**129.567**
2	378070	200000	99195	0	78875	31	107.589
3	380452	200000	99690	0	80762	30	128.378
4	375275	200000	99075	0	76200	32	65.039
5	375206	200000	100350	0	74856	31	83.764
6	378250	200000	97020	0	81230	30	105.97
7	384975	200000	97380	0	87595	32	146.277
8	375793	200000	98325	0	77468	32	139.027
9	382473	200000	99225	0	83248	33	96.57
10	379901	200000	101025	0	78876	32	121.731

Instance 6

Run	TC	PS	PI	CI	TC	TV	Time
1	381712	200000	103845	0	77867	32	119.651
2	378640	200000	101445	0	77195	32	129.944
3	386029	200000	104835	0	81194	32	82.077
4	382473	200000	101670	0	80803	32	162.499
5	378219	200000	105015	0	73204	32	87.308
6	381199	200000	103800	0	77399	32	132.808
7	382383	200000	103590	0	78793	33	109.503
8	382504	200000	103005	0	79499	31	135.246
9	**377586**	**200000**	**101370**	**0**	**76216**	**31**	**99.771**
10	385564	200000	105135	0	80429	33	103.202

Instance 7

Run	TC	PS	PI	CI	TC	TV	Time
1	382175	200000	105060	0	77115	32	89.598
2	384506	200000	107565	0	76941	32	97.503
3	382786	200000	104310	0	78476	32	96.697
4	383230	200000	104670	0	78560	32	147.94
5	379904	200000	105405	0	74499	32	114.369
6	375184	200000	103365	0	71819	31	86.727
7	381343	200000	105990	0	75353	32	80.121
8	**371137**	**200000**	**104700**	**0**	**66437**	**30**	**90.717**
9	384906	200000	106740	0	78166	32	110.371
10	386876	200000	104565	0	82311	33	111.208

Instance 8

Run	TC	PS	PI	CI	TC	TV	Time
1	384934	200000	107820	0	77114	31	95.851
2	385712	200000	113745	0	71967	32	99.373
3	375183	200000	107310	0	67873	29	92.857
4	386277	200000	110460	0	75817	32	93.308
5	382914	200000	114780	0	68134	33	88.568
6	374943	200000	110595	0	64348	31	136.643
7	**370440**	**200000**	**106860**	**0**	**63580**	**29**	**129.333**
8	386245	200000	109215	0	77030	30	134.903
9	373254	200000	109605	0	63649	31	117.432
10	386827	200000	112635	0	74192	32	112.84

Instance 9

Run	TC	PS	PI	CI	TC	TV	Time
1	**374140**	**200000**	**105435**	**0**	**68705**	**30**	**92.429**
2	379040	200000	101595	0	77445	31	105.108
3	377835	200000	105150	0	72685	32	105.859
4	381780	200000	107895	0	73885	33	96.064
5	378995	200000	103365	0	75630	31	107.195
6	385737	200000	106770	0	78967	33	96.58
7	388959	200000	109470	0	79489	33	100.968
8	374293	200000	101355	0	72938	30	109.445
9	375618	200000	100890	0	74728	28	96.93
10	381062	200000	105375	0	75687	31	88.209

Instance 10

Run	TC	PS	PI	CI	TC	TV	Time
1	373028	200000	100575	0	72453	31	117.016
2	368335	200000	96345	0	71990	30	110.204
3	369310	200000	96585	0	72725	31	126.739
4	**359769**	**200000**	**94380**	**0**	**65389**	**30**	**105.815**
5	366877	200000	99255	0	67622	30	70.089
6	369703	200000	99750	0	69953	31	110.531
7	370983	200000	100950	0	70033	31	133.434
8	374013	200000	101280	0	72733	32	122.175
9	366036	200000	98790	0	67246	32	98.846
10	367403	200000	94275	0	73128	30	122.064

Instance 11

Run	TC	PS	PI	CI	TC	TV	Time
1	368952	200000	95985	0	72967	32	110.661
2	377347	200000	100545	0	76802	32	109.16
3	368548	200000	99975	0	68573	30	65.989
4	371002	200000	94230	0	76772	31	92.451
5	372900	200000	98310	0	74590	32	132.419
6	372813	200000	95925	0	76888	32	103.247
7	373805	200000	97455	0	76350	32	69.152
8	371203	200000	96840	0	74363	32	102.531
9	367923	200000	96075	0	71848	30	75.689
10	**366676**	**200000**	**97740**	**0**	**68936**	**29**	**72.641**

Instance 12

Run	TC	PS	PI	CI	TC	TV	Time
1	369884	200000	96900	0	72984	31	92.576
2	377940	200000	99585	0	78355	32	93.241
3	376576	200000	101745	0	74831	32	97.959
4	373379	200000	95535	0	77844	31	104.414
5	360006	200000	97890	0	62116	29	68.844
6	**356468**	**200000**	**94680**	**0**	**61788**	**28**	**94.789**
7	370845	200000	100695	0	70150	31	134.082
8	366891	200000	99990	0	66901	31	125.724
9	375383	200000	101145	0	74238	31	108.23
10	368849	200000	100830	0	68019	31	105.267

Instance 13

Run	TC	PS	PI	CI	TC	TV	Time
1	380755	200000	106095	0	74660	32	113.601
2	376458	200000	101610	0	74848	30	107.991
3	371030	200000	100875	0	70155	30	153.21
4	374769	200000	101475	0	73294	30	128.101
5	372500	200000	102585	0	69915	30	146.703
6	**370453**	**200000**	**102930**	**0**	**67523**	**30**	**119.772**
7	379582	200000	102000	0	77582	31	96.423
8	378937	200000	103890	0	75047	32	105.492
9	378874	200000	103245	0	75629	31	105.428
10	378372	200000	103260	0	75112	31	95.052

Instance 14

Run	TC	PS	PI	CI	TC	TV	Time
1	380057	200000	101385	0	78672	31	137.789
2	375041	200000	100095	0	74946	32	107.479
3	375533	200000	102900	0	72633	31	97.125
4	370563	200000	102285	0	68278	30	110.1
5	**364681**	**200000**	**100170**	**0**	**64511**	**29**	**89.164**
6	380571	200000	101805	0	78766	30	130.571
7	375466	200000	100860	0	74606	30	84.738
8	375302	200000	100155	0	75147	31	109.387
9	373514	200000	98790	0	74724	30	156.489
10	378867	200000	106290	0	72577	32	110.128

Instance 15

Run	TC	PS	PI	CI	TC	TV	Time
1	380461	200000	104040	0	76421	33	102.708
2	378693	200000	100950	0	77743	30	121.942
3	383367	200000	102135	0	81232	31	93.571
4	375946	200000	100635	0	75311	31	124.602
5	377331	200000	100725	0	76606	32	128.545
6	378082	200000	101400	0	76682	31	119.335
7	**371119**	**200000**	**98100**	**0**	**73019**	**30**	**131.232**
8	377555	200000	103560	0	73995	32	118.658
9	382724	200000	104055	0	78669	32	109.604
10	377090	200000	101355	0	75735	33	110.888

Instance 16

Run	TC	PS	PI	CI	TC	TV	Time
1	372393	200000	100935	0	71458	30	120.468
2	378385	200000	103440	0	74945	31	119.618
3	368540	200000	102765	0	65775	31	108.1
4	376634	200000	106515	0	70119	33	109.903
5	373501	200000	103245	0	70256	31	112.784
6	368874	200000	106170	0	62704	32	59.598
7	**362274**	**200000**	**101910**	**0**	**60364**	**30**	**111.893**
8	375575	200000	103740	0	71835	31	106.274
9	372090	200000	105570	0	66520	31	119.969
10	367912	200000	104205	0	63707	31	136.219

Instance 17

Run	TC	PS	PI	CI	TC	TV	Time
1	385257	200000	109200	0	76057	32	110.771
2	385735	200000	106485	0	79250	32	108.575
3	384785	200000	110745	0	74040	32	131.417
4	**382365**	**200000**	**106710**	**0**	**75655**	**33**	**121.21**
5	385690	200000	105555	0	80135	32	81.496
6	382629	200000	105135	0	77494	31	115.687
7	389489	200000	106230	0	83259	31	61.326
8	385662	200000	106755	0	78907	32	98.845
9	389962	200000	109185	0	80777	33	66.148
10	394368	200000	109980	0	84388	33	62.724

Instance 18

Run	TC	PS	PI	CI	TC	TV	Time
1	386702	200000	105930	0	80772	31	113.599
2	**368902**	**200000**	**101925**	**0**	**66977**	**30**	**127.744**
3	385611	200000	104505	0	81106	32	121.155
4	378242	200000	105345	0	72897	31	116.428
5	382649	200000	102270	0	80379	33	139.035
6	390266	200000	108735	0	81531	33	61.788
7	384654	200000	105465	0	79189	34	130.203
8	385877	200000	103860	0	82017	32	90.983
9	379805	200000	103305	0	76500	32	59.956
10	385770	200000	107340	0	78430	32	114.483

Instance 19

Run	TC	PS	PI	CI	TC	TV	Time
1	374391	200000	106755	0	67636	32	131.022
2	380116	200000	108810	0	71306	31	108.698
3	379729	200000	104445	0	75284	30	87.08
4	374613	200000	108540	0	66073	30	125.076
5	382406	200000	109275	0	73131	31	138.728
6	383320	200000	106290	0	77030	31	69.553
7	**368844**	**200000**	**104745**	**0**	**64099**	**30**	**62.095**
8	378188	200000	105990	0	72198	32	134.587
9	379912	200000	104985	0	74927	32	78.4
10	379988	200000	105630	0	74358	30	118.204

Instance 20

Run	TC	PS	PI	CI	TC	TV	Time
1	371661	200000	101430	0	70231	31	112.859
2	371785	200000	98685	0	73100	31	124.648
3	376269	200000	99345	0	76924	32	84.313
4	373859	200000	99675	0	74184	31	53.801
5	**362015**	**200000**	**94590**	**0**	**67425**	**29**	**126.373**
6	369309	200000	97545	0	71764	31	76.661
7	375369	200000	98745	0	76624	31	80.086
8	371341	200000	97845	0	73496	31	85.608
9	370549	200000	96975	0	73574	31	76.726
10	366303	200000	97965	0	68338	31	110.104

Instance 21

Run	TC	PS	PI	CI	TC	TV	Time
1	371324	200000	96375	0	74949	30	101.499
2	370657	200000	98595	0	72062	30	152.484
3	366049	200000	94275	0	71774	32	80.843
4	372329	200000	93585	0	78744	31	79.515
5	369370	200000	98805	0	70565	30	81.126
6	357727	200000	93990	0	63737	31	103.537
7	363310	200000	93345	0	69965	31	98.37
8	359570	200000	94755	0	64815	32	108.726
9	369347	200000	96615	0	72732	32	94.32
10	**352207**	**200000**	**95145**	**0**	**57062**	**30**	**113.322**

Instance 22

Run	TC	PS	PI	CI	TC	TV	Time
1	364214	200000	92925	0	71289	31	109.481
2	363074	200000	93405	0	69669	31	94.219
3	359277	200000	92490	0	66787	29	97.906
4	364225	200000	91890	0	72335	30	135.457
5	361575	200000	95745	0	65830	30	135.332
6	367438	200000	95010	0	72428	32	111.138
7	358701	200000	94380	0	64321	30	95.015
8	363925	200000	94530	0	69395	30	80.004
9	361760	200000	94995	0	66765	31	114.803
10	**357507**	**200000**	**93945**	**0**	**63562**	**30**	**112.907**

Instance 23

Run	TC	PS	PI	CI	TC	TV	Time
1	382364	200000	102360	0	80004	33	97.709
2	377457	200000	100185	0	77272	30	84.591
3	**367723**	**200000**	**103560**	**0**	**64163**	**31**	**121.629**
4	373872	200000	99375	0	74497	31	91.616
5	382734	200000	102495	0	80239	33	100.181
6	383072	200000	100920	0	82152	32	134.738
7	373951	200000	98190	0	75761	31	146.616
8	380988	200000	102900	0	78088	32	97.4
9	379705	200000	102615	0	77090	32	107.827
10	377449	200000	99465	0	77984	32	107.949

Instance 24

Run	TC	PS	PI	CI	TC	TV	Time
1	**360582**	**200000**	**97005**	**0**	**63577**	**30**	**137.524**
2	371890	200000	97785	0	74105	31	94.998
3	368716	200000	97170	0	71546	30	99.148
4	367028	200000	96195	0	70833	29	122.725
5	373467	200000	101130	0	72337	32	93.653
6	373078	200000	101625	0	71453	32	103.141
7	369602	200000	96765	0	72837	29	92.768
8	368191	200000	99360	0	68831	32	74.955
9	369710	200000	100095	0	69615	30	86.347
10	367474	200000	99150	0	68324	30	183.531

Instance 25

Run	TC	PS	PI	CI	TC	TV	Time
1	381741	200000	107145	0	74596	30	84.758
2	380360	200000	103725	0	76635	30	134.667
3	**372288**	**200000**	**105330**	**0**	**66958**	**31**	**112.703**
4	389373	200000	112455	0	76918	34	123.295
5	390700	200000	109710	0	80990	32	131.628
6	384870	200000	106755	0	78115	32	121.571
7	389518	200000	107775	0	81743	32	70.137
8	378266	200000	105735	0	72531	30	128.071
9	386718	200000	108300	0	78418	32	123.817
10	380308	200000	106770	0	73538	31	173.827

Instance 26

Run	TC	PS	PI	CI	TC	TV	Time
1	380476	200000	106440	0	74036	30	129.344
2	379261	200000	101565	0	77696	31	115.617
3	386115	200000	100980	0	85135	30	123.165
4	380156	200000	103260	0	76896	31	92.954
5	**375902**	**200000**	**106020**	**0**	**69882**	**30**	**141.445**
6	379238	200000	103290	0	75948	32	116.972
7	379638	200000	105180	0	74458	31	109.528
8	383403	200000	105480	0	77923	32	118.768
9	378890	200000	106665	0	72225	31	91.119
10	383508	200000	106575	0	76933	32	139.578

Instance 27

Run	TC	PS	PI	CI	TC	TV	Time
1	375365	200000	99840	0	75525	32	68.626
2	374623	200000	97560	0	77063	32	62.777
3	372591	200000	97710	0	74881	31	115.477
4	369883	200000	96135	0	73748	30	123.758
5	**363711**	**200000**	**96795**	**0**	**66916**	**31**	**66.335**
6	372416	200000	98400	0	74016	33	116.74
7	366306	200000	98295	0	68011	31	125.052
8	366757	200000	99615	0	67142	30	70.798
9	373390	200000	99255	0	74135	32	115.42
10	367686	200000	95535	0	72151	30	91.5

Instance 28

Run	TC	PS	PI	CI	TC	TV	Time
1	367733	200000	98355	0	69378	31	124.71
2	364658	200000	97080	0	67578	30	118.206
3	377820	200000	104025	0	73795	32	125.427
4	370375	200000	98205	0	72170	31	128.573
5	**358592**	**200000**	**98820**	**0**	**59772**	**31**	**116.985**
6	376829	200000	98415	0	78414	32	122.912
7	366205	200000	97890	0	68315	31	117.245
8	372729	200000	100290	0	72439	31	94.702
9	373814	200000	94845	0	78969	30	106.446
10	365430	200000	99090	0	66340	31	130.096

Instance 29

Run	TC	PS	PI	CI	TC	TV	Time
1	387851	200000	105315	0	82536	31	93.999
2	389241	200000	106320	0	82921	33	117.415
3	389901	200000	109035	0	80866	32	113.854
4	392533	200000	106800	0	85733	32	73.526
5	390912	200000	106710	0	84202	33	125.804
6	392734	200000	108375	0	84359	33	144.196
7	391027	200000	109950	0	81077	33	95.92
8	**386357**	**200000**	**107400**	**0**	**78957**	**32**	**116.962**
9	390501	200000	107850	0	82651	33	104.046
10	387228	200000	107220	0	80008	32	104.836

Instance 30

Run	TC	PS	PI	CI	TC	TV	Time
1	**357068**	**200000**	**95955**	**0**	**61113**	**30**	**84.297**
2	368933	200000	97875	0	71058	29	85.834
3	373950	200000	97890	0	76060	32	144.162
4	374602	200000	100995	0	73607	32	116.695
5	375736	200000	99090	0	76646	32	80.213
6	376269	200000	98130	0	78139	30	108.307
7	373035	200000	96465	0	76570	30	84.09
8	375039	200000	97590	0	77449	33	87.334
9	370529	200000	98610	0	71919	31	106.958
10	368491	200000	97650	0	70841	29	111.552

Case N100T20

Instance 1

Run	TC	PS	PI	CI	TC	TV	Time
1	711661	350000	202200	0	159461	63	599.496
2	713637	350000	207780	0	155857	67	686.2
3	717025	350000	209850	0	157175	64	713.201
4	704979	350000	201930	0	153049	62	606.847
5	705426	350000	202245	0	153181	62	693.084
6	695251	350000	203925	0	141326	63	710.221
7	715599	350000	207990	0	157609	65	539.996
8	715768	350000	203880	0	161888	64	591.337
9	**694556**	**350000**	**207960**	**0**	**136596**	**62**	**626.238**
10	704887	350000	204885	0	150002	63	603.608

Instance 2

Run	TC	PS	PI	CI	TC	TV	Time
1	682037	350000	192645	0	139392	60	459.393
2	709706	350000	202950	0	156756	63	799.592
3	**670573**	**350000**	**191190**	**0**	**129383**	**60**	**608.846**
4	692205	350000	195195	0	147010	62	651.838
5	697860	350000	197535	0	150325	60	500.317
6	704486	350000	196305	0	158181	62	536.23
7	697376	350000	194280	0	153096	59	735.86
8	691831	350000	199545	0	142286	61	587.535
9	701236	350000	195870	0	155366	61	768.124
10	703713	350000	196950	0	156763	62	696.679

Instance 3

Run	TC	PS	PI	CI	TC	TV	Time
1	720600	350000	208515	0	162085	64	854.848
2	724211	350000	212700	0	161511	64	672.491
3	709775	350000	204300	0	155475	66	673.173
4	**708914**	**350000**	**207510**	**0**	**151404**	**63**	**720.849**
5	708924	350000	207210	0	151714	62	657.986
6	716716	350000	214050	0	152666	64	763.995
7	714711	350000	201210	0	163501	62	762.933
8	710867	350000	206460	0	154407	63	655.176
9	720150	350000	209160	0	160990	64	708.51
10	717178	350000	211275	0	155903	65	602.101

Instance 4

Run	TC	PS	PI	CI	TC	TV	Time
1	700883	350000	195405	0	155478	65	724.301
2	699613	350000	192495	0	157118	65	650.265
3	702906	350000	195855	0	157051	64	711.437
4	**680941**	**350000**	**188130**	**0**	**142811**	**64**	**658.744**
5	702332	350000	190245	0	162087	62	607.335
6	687127	350000	192135	0	144992	60	561.896
7	688047	350000	196440	0	141607	63	812.28
8	697926	350000	194190	0	153736	65	738.681
9	699942	350000	192705	0	157237	64	628.579
10	684998	350000	190200	0	144798	62	560.725

Instance 5

Run	TC	PS	PI	CI	TC	TV	Time
1	732815	350000	213705	0	169110	64	694.404
2	734070	350000	218655	0	165415	64	716.81
3	728348	350000	213345	0	165003	63	826.604
4	724643	350000	215190	0	159453	66	681.576
5	729641	350000	213390	0	166251	66	603.489
6	723159	350000	213420	0	159739	64	766.404
7	730834	350000	215130	0	165704	66	732.891
8	**699935**	**350000**	**210555**	**0**	**139380**	**63**	**844.958**
9	723310	350000	213855	0	159455	64	683.154
10	735822	350000	219165	0	166657	67	791.563

Instance 6

Run	TC	PS	PI	CI	TC	TV	Time
1	713734	350000	202950	0	160784	65	770.362
2	713421	350000	197760	0	165661	66	757.848
3	690894	350000	195375	0	145519	61	732.801
4	705862	350000	198000	0	157862	62	684.406
5	**682905**	**350000**	**190725**	**0**	**142180**	**62**	**632.957**
6	716210	350000	202245	0	163965	68	810.745
7	702469	350000	198360	0	154109	65	675.541
8	704278	350000	197115	0	157163	65	628.005
9	695923	350000	198390	0	147533	62	675.5
10	716312	350000	202470	0	163842	65	683.787

Instance 7

Run	TC	PS	PI	CI	TC	TV	Time
1	699799	350000	194625	0	155174	64	586.483
2	705869	350000	195435	0	160434	63	831.617
3	703467	350000	194310	0	159157	64	599.594
4	706524	350000	191340	0	165184	62	640.162
5	**680992**	**350000**	**190140**	**0**	**140852**	**62**	**734.041**
6	707820	350000	196065	0	161755	64	698.629
7	701240	350000	196140	0	155100	64	656.948
8	690418	350000	191895	0	148523	62	562.969
9	705814	350000	192420	0	163394	63	611.289
10	711310	350000	197250	0	164060	64	657.733

Instance 8

Run	TC	PS	PI	CI	TC	TV	Time
1	702463	350000	201750	0	150713	61	582.026
2	711106	350000	199725	0	161381	62	725.391
3	700648	350000	196560	0	154088	60	567.913
4	697465	350000	198465	0	149000	62	601.382
5	687539	350000	192345	0	145194	61	574.399
6	694779	350000	195345	0	149434	61	734.689
7	695759	350000	192210	0	153549	60	660.277
8	711308	350000	202335	0	158973	62	658.989
9	**684011**	**350000**	**194280**	**0**	**139731**	**61**	**606.626**
10	697611	350000	195855	0	151756	61	656.018

Instance 9

Run	TC	PS	PI	CI	TC	TV	Time
1	725428	350000	208995	0	166433	63	794.818
2	724674	350000	205050	0	169624	63	770.013
3	719914	350000	205770	0	164144	63	652.677
4	715618	350000	203835	0	161783	63	645.004
5	722429	350000	203985	0	168444	62	660.674
6	737741	350000	217905	0	169836	65	832.008
7	**671367**	**350000**	**193245**	**0**	**128122**	**58**	**818.265**
8	680433	350000	193830	0	136603	62	687.474
9	710591	350000	198960	0	161631	63	663.772
10	722855	350000	210345	0	162510	62	676.813

Instance 10

Run	TC	PS	PI	CI	TC	TV	Time
1	718932	350000	212190	0	156742	65	775.221
2	**699283**	**350000**	**209565**	**0**	**139718**	**63**	**696.077**
3	719228	350000	205560	0	163668	64	897.468
4	706899	350000	206640	0	150259	65	633.94
5	712263	350000	207930	0	154333	64	697.213
6	714958	350000	209220	0	155738	67	662.421
7	705293	350000	207480	0	147813	63	636.459
8	715017	350000	211050	0	153967	64	666.714
9	723031	350000	208920	0	164111	64	636.896
10	706690	350000	205845	0	150845	63	670.041

Instance 11

Run	TC	PS	PI	CI	TC	TV	Time
1	720158	350000	208080	0	162078	63	618.391
2	717207	350000	206175	0	161032	65	636.253
3	**710272**	**350000**	**205575**	**0**	**154697**	**63**	**646.294**
4	716223	350000	209505	0	156718	63	743.99
5	714934	350000	204795	0	160139	63	764.755
6	713405	350000	207015	0	156390	63	597.813
7	728631	350000	213285	0	165346	63	776.469
8	726252	350000	208785	0	167467	66	625.239
9	723069	350000	210255	0	162814	65	581.309
10	724942	350000	212025	0	162917	63	621.432

Instance 12

Run	TC	PS	PI	CI	TC	TV	Time
1	721323	350000	215550	0	155773	65	609.236
2	728887	350000	212325	0	166562	64	556.326
3	733597	350000	216825	0	166772	63	622.641
4	733616	350000	216285	0	167331	65	528.726
5	723079	350000	213240	0	159839	65	672.028
6	729354	350000	215190	0	164164	63	721.888
7	**695239**	**350000**	**208965**	**0**	**136274**	**63**	**642.146**
8	730672	350000	214290	0	166382	66	623.941
9	724502	350000	208485	0	166017	64	611.483
10	700054	350000	209010	0	141044	63	606.933

Instance 13

Run	TC	PS	PI	CI	TC	TV	Time
1	721174	350000	208545	0	162629	66	597.505
2	723824	350000	209595	0	164229	66	671.059
3	716402	350000	201585	0	164817	65	561.895
4	701059	350000	202065	0	148994	63	645.129
5	707234	350000	200520	0	156714	65	652.809
6	711904	350000	206475	0	155429	65	663.58
7	713055	350000	204195	0	158860	62	723.581
8	707721	350000	204975	0	152746	63	617.359
9	**699318**	**350000**	**203760**	**0**	**145558**	**65**	**883.529**
10	700351	350000	200820	0	149531	65	597.372

Instance 14

Run	TC	PS	PI	CI	TC	TV	Time
1	710768	350000	206235	0	154533	61	802.951
2	704065	350000	205665	0	148400	61	594.511
3	716658	350000	211740	0	154918	65	767.828
4	**687897**	**350000**	**199875**	**0**	**138022**	**61**	**811.129**
5	720554	350000	202560	0	167994	62	643.156
6	703630	350000	206040	0	147590	63	552.044
7	713693	350000	207000	0	156693	63	653.426
8	718654	350000	209160	0	159494	63	717.353
9	720909	350000	210375	0	160534	64	691.725
10	705396	350000	205575	0	149821	64	667.906

Instance 15

Run	TC	PS	PI	CI	TC	TV	Time
1	**718313**	**350000**	**207930**	**0**	**160383**	**63**	**620.363**
2	725716	350000	213030	0	162686	65	678.841
3	731235	350000	212865	0	168370	66	588.213
4	721769	350000	205650	0	166119	64	581.618
5	718521	350000	207720	0	160801	65	563.722
6	730122	350000	206310	0	173812	64	714.719
7	725993	350000	209655	0	166338	65	707.322
8	720007	350000	206910	0	163097	64	537.049
9	727082	350000	204060	0	173022	64	511.728
10	719439	350000	206685	0	162754	65	584.125

Instance 16

Run	TC	PS	PI	CI	TC	TV	Time
1	724517	350000	210300	0	164217	64	629.224
2	**708177**	**350000**	**207975**	**0**	**150202**	**62**	**613.669**
3	726681	350000	214425	0	162256	64	668.743
4	709748	350000	211395	0	148353	62	731.933
5	728465	350000	212430	0	166035	64	703.127
6	717323	350000	210105	0	157218	62	670.524
7	729236	350000	211275	0	167961	63	744.668
8	725962	350000	214515	0	161447	63	750.692
9	723393	350000	211755	0	161638	62	898.848
10	722065	350000	210240	0	161825	64	618.135

Instance 17

Run	TC	PS	PI	CI	TC	TV	Time
1	728504	350000	214350	0	164154	64	603.917
2	728622	350000	215610	0	163012	64	772.715
3	735776	350000	215370	0	170406	64	725.188
4	724752	350000	215145	0	159607	66	845.189
5	721392	350000	212685	0	158707	63	872.19
6	727788	350000	214065	0	163723	64	629.27
7	**719988**	**350000**	**216720**	**0**	**153268**	**65**	**754.819**
8	731003	350000	213795	0	167208	64	581.945
9	730659	350000	212730	0	167929	65	640.912
10	737086	350000	219420	0	167666	66	501.693

Instance 18

Run	TC	PS	PI	CI	TC	TV	Time
1	705406	350000	201735	0	153671	62	787.73
2	712680	350000	205725	0	156955	62	628.862
3	720365	350000	201360	0	169005	63	696.045
4	712572	350000	198330	0	164242	66	743.768
5	716945	350000	203025	0	163920	64	563.53
6	713700	350000	206880	0	156820	63	691.119
7	715299	350000	203985	0	161314	63	586.846
8	704887	350000	201555	0	153332	62	731.08
9	721178	350000	205395	0	165783	64	654.15
10	**702293**	**350000**	**201180**	**0**	**151113**	**62**	**624.862**

Instance 19

Run	TC	PS	PI	CI	TC	TV	Time
1	725207	350000	216345	0	158862	64	677.609
2	725163	350000	213360	0	161803	63	634.868
3	720212	350000	211350	0	158862	62	603.773
4	**711100**	**350000**	**213330**	**0**	**147770**	**64**	**677.513**
5	717386	350000	211965	0	155421	65	619.528
6	715658	350000	210915	0	154743	63	665.824
7	728482	350000	210540	0	167942	65	686.459
8	724913	350000	217605	0	157308	67	675.341
9	715211	350000	216540	0	148671	64	681.492
10	731193	350000	213195	0	167998	64	625.922

Instance 20

Run	TC	PS	PI	CI	TC	TV	Time
1	736703	350000	217020	0	169683	65	573.64
2	741465	350000	220680	0	170785	67	596.61
3	736882	350000	217035	0	169847	67	737.748
4	739199	350000	219525	0	169674	67	670.602
5	733528	350000	214035	0	169493	65	588.947
6	738471	350000	217530	0	170941	65	729.017
7	741073	350000	214545	0	176528	65	591.683
8	**726930**	**350000**	**212910**	**0**	**164020**	**64**	**734.321**
9	741456	350000	216885	0	174571	64	614.32
10	735458	350000	217170	0	168288	65	604.202

Instance 21

Run	TC	PS	PI	CI	TC	TV	Time
1	710610	350000	204360	0	156250	63	674.394
2	714008	350000	204870	0	159138	67	622.336
3	729849	350000	214530	0	165319	67	735.641
4	717593	350000	205200	0	162393	66	714.749
5	720391	350000	202245	0	168146	67	685.031
6	713296	350000	201960	0	161336	64	627.599
7	726413	350000	208785	0	167628	65	697.699
8	**706844**	**350000**	**200520**	**0**	**156324**	**66**	**712.852**
9	708744	350000	201645	0	157099	64	625.019
10	729556	350000	213015	0	166541	65	635.595

Instance 22

Run	TC	PS	PI	CI	TC	TV	Time
1	720239	350000	211620	0	158619	63	603.966
2	727186	350000	215925	0	161261	63	667.393
3	722855	350000	210645	0	162210	64	625.867
4	717456	350000	206355	0	161101	62	706.8
5	720620	350000	213375	0	157245	64	572.855
6	727936	350000	212685	0	165251	63	619.647
7	737085	350000	212040	0	175045	63	668.368
8	**716763**	**350000**	**213705**	**0**	**153058**	**63**	**625.767**
9	728547	350000	214410	0	164137	64	624.346
10	719654	350000	212805	0	156849	62	683.835

Instance 23

Run	TC	PS	PI	CI	TC	TV	Time
1	705564	350000	192900	0	162664	65	677.305
2	699468	350000	194145	0	155323	65	557.696
3	701754	350000	195285	0	156469	66	574.946
4	685605	350000	190500	0	145105	62	745.036
5	710050	350000	195780	0	164270	62	711.081
6	**649341**	**350000**	**183015**	**0**	**116326**	**59**	**603.156**
7	690289	350000	198915	0	141374	66	611.654
8	677800	350000	190755	0	137045	62	634.127
9	697465	350000	195690	0	151775	62	579.952
10	693178	350000	196350	0	146828	64	679.334

Instance 24

Run	TC	PS	PI	CI	TC	TV	Time
1	736381	350000	215730	0	170651	67	665.315
2	730854	350000	216165	0	164689	66	738.331
3	**701365**	**350000**	**204315**	**0**	**147050**	**63**	**818.377**
4	723505	350000	215250	0	158255	63	723.138
5	721545	350000	208155	0	163390	64	816.329
6	717075	350000	207240	0	159835	62	710.532
7	709853	350000	201915	0	157938	63	706.271
8	724727	350000	211740	0	162987	65	824.422
9	716844	350000	211935	0	154909	69	666.414
10	719991	350000	209310	0	160681	63	786.087

Instance 25

Run	TC	PS	PI	CI	TC	TV	Time
1	717395	350000	206520	0	160875	64	624.181
2	708174	350000	207420	0	150754	62	623.824
3	708182	350000	206970	0	151212	65	670.313
4	718590	350000	205395	0	163195	63	691.939
5	727011	350000	211320	0	165691	66	677.319
6	713970	350000	205635	0	158335	62	547.741
7	716223	350000	205380	0	160843	63	660.483
8	**698750**	**350000**	**198825**	**0**	**149925**	**63**	**569.909**
9	730310	350000	210510	0	169800	64	806.323
10	720803	350000	210225	0	160578	62	619.53

Instance 26

Run	TC	PS	PI	CI	TC	TV	Time
1	714834	350000	205170	0	159664	63	556.496
2	705460	350000	205665	0	149795	62	670.405
3	718499	350000	212040	0	156459	63	740.889
4	716878	350000	208185	0	158693	64	695.794
5	**693440**	**350000**	**199050**	**0**	**144390**	**61**	**694.392**
6	712362	350000	210750	0	151612	63	616.365
7	709930	350000	207885	0	152045	61	622.819
8	716286	350000	205830	0	160456	64	587.566
9	713301	350000	206715	0	156586	64	695.16
10	711242	350000	204330	0	156912	63	662.213

Instance 27

Run	TC	PS	PI	CI	TC	TV	Time
1	716708	350000	202245	0	164463	64	616.408
2	719101	350000	204240	0	164861	63	676.24
3	712443	350000	201315	0	161128	63	607.651
4	714788	350000	202680	0	162108	64	621.888
5	708379	350000	200610	0	157769	65	564.036
6	714803	350000	200715	0	164088	63	598.942
7	706646	350000	198165	0	158481	64	585.87
8	710786	350000	197910	0	162876	62	710.476
9	704533	350000	196590	0	157943	64	573.975
10	**694616**	**350000**	**194940**	**0**	**149676**	**62**	**604.318**

Instance 28

Run	TC	PS	PI	CI	TC	TV	Time
1	711451	350000	203520	0	157931	64	600.721
2	693836	350000	197085	0	146751	64	563.892
3	699880	350000	196245	0	153635	61	725.335
4	683824	350000	197910	0	135914	62	674.019
5	679991	350000	195780	0	134211	62	576.063
6	699875	350000	199575	0	150300	62	607.927
7	**676207**	**350000**	**194625**	**0**	**131582**	**63**	**550.833**
8	682305	350000	193350	0	138955	60	605.811
9	697433	350000	197550	0	149883	62	540.75
10	706356	350000	201405	0	154951	64	526.75

Instance 29

Run	TC	PS	PI	CI	TC	TV	Time
1	718388	350000	208260	0	160128	64	613.018
2	**714447**	**350000**	**208875**	**0**	**155572**	**65**	**595.046**
3	727334	350000	210825	0	166509	64	560.791
4	721825	350000	207930	0	163895	62	658.919
5	719528	350000	210915	0	158613	67	660.833
6	726374	350000	208695	0	167679	65	532.558
7	719850	350000	203790	0	166060	65	559.043
8	725138	350000	209100	0	166038	65	592.903
9	727780	350000	211305	0	166475	64	622.108
10	725929	350000	213150	0	162779	63	621.488

Instance 30

Run	TC	PS	PI	CI	TC	TV	Time
1	702779	350000	196290	0	156489	62	750.333
2	701486	350000	197520	0	153966	64	712.343
3	**693102**	**350000**	**195885**	**0**	**147217**	**62**	**657.662**
4	714118	350000	202620	0	161498	64	757.314
5	714451	350000	202800	0	161651	62	631.349
6	707542	350000	201045	0	156497	65	817.593
7	715418	350000	203775	0	161643	64	606.043
8	716072	350000	198495	0	167577	62	895.81
9	719630	350000	195705	0	173925	64	758.553
10	714218	350000	202290	0	161928	63	726.299

Case N200T20

Instance 1

Run	TC	PS	PI	CI	TC	TV	Time
1	1029109	720000	249270	0	59839	75	5437.92
2	1048302	720000	254160	0	74142	76	5050.19
3	1039859	720000	258690	0	61169	76	7070.77
4	1052846	720000	259140	0	73706	79	4723.1
5	**1023870**	**720000**	**259440**	**0**	**44430**	**77**	**6580.62**
6	1043315	720000	259740	0	63575	77	6119.3
7	1049350	720000	257595	0	71755	77	4867.13
8	1044359	720000	252240	0	72119	73	6174.87
9	1037107	720000	255870	0	61237	76	5496.6
10	1048394	720000	251205	0	77189	73	7461.9

Instance 2

Run	TC	PS	PI	CI	TC	TV	Time
1	1029597	720000	244830	0	64767	74	6359.91
2	1034197	720000	246795	0	67402	73	4982.18
3	1020255	720000	244440	0	55815	76	7041.43
4	**1007123**	**720000**	**246060**	**0**	**41063**	**74**	**4958.36**
5	1035808	720000	242370	0	73438	74	7597.79
6	1034454	720000	238215	0	76239	73	7224.1
7	1025752	720000	248805	0	56947	74	7063.87
8	1032678	720000	251895	0	60783	77	6025.17
9	1011851	720000	256305	0	35546	74	7732.45
10	1026925	720000	243735	0	63190	75	6436.31

Instance 3

Run	TC	PS	PI	CI	TC	TV	Time
1	1026367	720000	254340	0	52027	76	15950.5
2	1045402	720000	252600	0	72802	76	5934.22
3	1035861	720000	256590	0	59271	76	6313.35
4	1039611	720000	248550	0	71061	74	4682.56
5	1021210	720000	243675	0	57535	76	5897.35
6	1023663	720000	245520	0	58143	73	6472.91
7	1017729	720000	253125	0	44604	75	4672.4
8	1030406	720000	247335	0	63071	74	5047.43
9	1031971	720000	255045	0	56926	75	5277.57
10	**1015476**	**720000**	**248565**	**0**	**46911**	**72**	**7437.31**

Instance 4

Run	TC	PS	PI	CI	TC	TV	Time
1	**1004969**	**720000**	**248700**	**0**	**36269**	**75**	**6401.47**
2	1017982	720000	242400	0	55582	73	8298.86
3	1039780	720000	246030	0	73750	76	5572.46
4	1019216	720000	248700	0	50516	75	6014.07
5	1017066	720000	249780	0	47286	76	6214.96
6	1014069	720000	246750	0	47319	75	7916.74
7	1026274	720000	234180	0	72094	73	5560.3
8	1022286	720000	243825	0	58461	74	5386.39
9	1039396	720000	240960	0	78436	74	5801.16
10	1030901	720000	248580	0	62321	75	5944.88

Instance 5

Run	TC	PS	PI	CI	TC	TV	Time
1	1024915	720000	245655	0	59260	74	5968.93
2	1025084	720000	245745	0	59339	74	6641.68
3	1030754	720000	243000	0	67754	76	8024.36
4	1034884	720000	242205	0	72679	74	5666.27
5	**1014620**	**720000**	**244320**	**0**	**50300**	**75**	**4785.6**
6	1028860	720000	245625	0	63235	77	6729.97
7	1040633	720000	239430	0	81203	75	5741.9
8	1018055	720000	244425	0	53630	76	6040.72
9	1018604	720000	241110	0	57494	75	5858.92
10	1023530	720000	246090	0	57440	75	5639.03

Instance 6

Run	TC	PS	PI	CI	TC	TV	Time
1	1042426	720000	246285	0	76141	77	5526.68
2	**1009793**	**720000**	**247590**	**0**	**42203**	**77**	**6880.47**
3	1029029	720000	248880	0	60149	76	7352.54
4	1037524	720000	237915	0	79609	78	6108.23
5	1032471	720000	246765	0	65706	75	6635.35
6	1039725	720000	247155	0	72570	78	5236.49
7	1039083	720000	240930	0	78153	76	5612.6
8	1044383	720000	245355	0	79028	76	5346.25
9	1031749	720000	243435	0	68314	75	5657.81
10	1038774	720000	243870	0	74904	74	5258.21

Instance 7

Run	TC	PS	PI	CI	TC	TV	Time
1	1014518	720000	246990	0	47528	74	7486.61
2	1021336	720000	258900	0	42436	76	6141.58
3	1032632	720000	253815	0	58817	76	6280.3
4	1041307	720000	254685	0	66622	76	5554.22
5	1027353	720000	253350	0	54003	74	5430.26
6	**997304**	**720000**	**239415**	**0**	**37889**	**71**	**6138.27**
7	1032450	720000	245970	0	66480	72	6405.88
8	1022569	720000	250470	0	52099	76	5389.9
9	1047665	720000	256605	0	71060	76	5622.63
10	1015786	720000	256035	0	39751	75	5810.23

Instance 8

Run	TC	PS	PI	CI	TC	TV	Time
1	1029560	720000	232050	0	77510	77	5667.29
2	1023880	720000	234735	0	69145	74	5489.7
3	1028753	720000	245415	0	63338	78	5520.16
4	**1013544**	**720000**	**247560**	**0**	**45984**	**74**	**9012.51**
5	1015294	720000	236295	0	58999	76	6698.99
6	1021962	720000	234600	0	67362	75	8276.22
7	1014524	720000	231945	0	62579	77	6114.75
8	1017305	720000	229245	0	68060	75	6276.76
9	1019263	720000	242340	0	56923	73	6347.31
10	1025544	720000	236790	0	68754	76	6193.15

Instance 9

Run	TC	PS	PI	CI	TC	TV	Time
1	1002120	720000	237570	0	44550	75	5192.87
2	1002529	720000	229965	0	52564	75	6847.74
3	1012661	720000	234000	0	58661	73	5064.24
4	1009489	720000	240345	0	49144	72	5839.47
5	**995151**	**720000**	**235665**	**0**	**39486**	**76**	**6364.36**
6	1006515	720000	234210	0	52305	75	6194.31
7	1011231	720000	230865	0	60366	74	4988.05
8	1006750	720000	228405	0	58345	72	5456.34
9	1011883	720000	235410	0	56473	73	6299.84
10	1007037	720000	235200	0	51837	74	4886.18

Instance 10

Run	TC	PS	PI	CI	TC	TV	Time
1	1022175	720000	242235	0	59940	78	4854.32
2	1032225	720000	244065	0	68160	76	5689.9
3	1013218	720000	243420	0	49798	76	5808.68
4	1019659	720000	242940	0	56719	74	7165.2
5	**1011910**	**720000**	**247110**	**0**	**44800**	**77**	**6811.81**
6	1033844	720000	250440	0	63404	75	7115.34
7	1012222	720000	240840	0	51382	77	5768.6
8	1037350	720000	233040	0	84310	75	6930.3
9	1034545	720000	239295	0	75250	77	6337.85
10	1023297	720000	242220	0	61077	76	5024.13

Instance 11

Run	TC	PS	PI	CI	TC	TV	Time
1	1017487	720000	250830	0	46657	75	7354.48
2	1028598	720000	252450	0	56148	77	7606.08
3	1041715	720000	250005	0	71710	75	6576.61
4	1014756	720000	246465	0	48291	74	4752.69
5	1037972	720000	243690	0	74282	75	6136.3
6	1020240	720000	250365	0	49875	77	6733.55
7	1034061	720000	248430	0	65631	75	6616.99
8	**1006529**	**720000**	**247950**	**0**	**38579**	**74**	**9615.09**
9	1023020	720000	247680	0	55340	74	5452.64
10	1047395	720000	246000	0	81395	76	8961.78

Instance 12

Run	TC	PS	PI	CI	TC	TV	Time
1	1034747	720000	249975	0	64772	77	5469.3
2	1042786	720000	245025	0	77761	75	5810.64
3	1045756	720000	257160	0	68596	75	5225.76
4	1046827	720000	251775	0	75052	77	6280.98
5	1028788	720000	251895	0	56893	73	7634.15
6	1040905	720000	254250	0	66655	76	5647.21
7	1019159	720000	234600	0	64559	75	5250.88
8	1045009	720000	251370	0	73639	76	5162.21
9	**1015183**	**720000**	**251280**	**0**	**43903**	**74**	**7466.57**
10	1024584	720000	246390	0	58194	74	7734.89

Instance 13

Run	TC	PS	PI	CI	TC	TV	Time
1	1031057	720000	251955	0	59102	74	5129.46
2	1047888	720000	248610	0	79278	78	5237.44
3	1030213	720000	258285	0	51928	75	5261.24
4	1046587	720000	257145	0	69442	76	4989.76
5	1039534	720000	256635	0	62899	76	5780.17
6	1035824	720000	248415	0	67409	76	5781.41
7	**1030030**	**720000**	**258120**	**0**	**51910**	**76**	**5980.42**
8	1044068	720000	256065	0	68003	77	4772.69
9	1033595	720000	261000	0	52595	76	5531.81
10	1040249	720000	261675	0	58574	75	6011.01

Instance 14

Run	TC	PS	PI	CI	TC	TV	Time
1	1043017	720000	244500	0	78517	76	5321.06
2	**1027311**	**720000**	**251055**	**0**	**56256**	**78**	**5733.34**
3	1028693	720000	245115	0	63578	76	6054.25
4	1031216	720000	249270	0	61946	74	7120.56
5	1042892	720000	241710	0	81182	76	5163.54
6	1047468	720000	248040	0	79428	75	5619.17
7	1030311	720000	244590	0	65721	75	7282.31
8	1030809	720000	252765	0	58044	79	5332
9	1041515	720000	259485	0	62030	76	5272.51
10	1037398	720000	246675	0	70723	75	5010.72

Instance 15

Run	TC	PS	PI	CI	TC	TV	Time
1	1019853	720000	245610	0	54243	76	6907.69
2	**995542**	**720000**	**236685**	**0**	**38857**	**76**	**6242.69**
3	1011636	720000	240330	0	51306	73	5766.99
4	1029533	720000	244185	0	65348	77	8156.91
5	1017965	720000	243255	0	54710	73	6013.88
6	1020923	720000	241500	0	59423	78	6314.42
7	1016277	720000	252570	0	43707	76	6804.95
8	1033372	720000	240945	0	72427	75	4735.28
9	1030767	720000	241335	0	69432	75	5512.13
10	996793	720000	251280	0	25513	72	9294.8

Instance 16

Run	TC	PS	PI	CI	TC	TV	Time
1	1027523	720000	240330	0	67193	75	5447.25
2	1028048	720000	240240	0	67808	76	5135.14
3	1023508	720000	240555	0	62953	74	6753.95
4	1020935	720000	246525	0	54410	78	6372.65
5	**1014381**	**720000**	**247545**	**0**	**46836**	**75**	**5340**
6	1020501	720000	240930	0	59571	76	6036.67
7	1037430	720000	246930	0	70500	76	6377.13
8	1034778	720000	242100	0	72678	74	5578.36
9	1024127	720000	244980	0	59147	74	6275.07
10	1015352	720000	251370	0	43982	75	6554.13

Instance 17

Run	TC	PS	PI	CI	TC	TV	Time
1	1040528	720000	247440	0	73088	76	5002.09
2	1041056	720000	255030	0	66026	75	5152
3	1033528	720000	253365	0	60163	76	6384.21
4	1031066	720000	250200	0	60866	76	4793.07
5	1030726	720000	250860	0	59866	73	5780.62
6	1025723	720000	254610	0	51113	78	5661.67
7	1015564	720000	237075	0	58489	73	4820.18
8	**1015209**	**720000**	**246015**	**0**	**49194**	**74**	**5460.62**
9	1024220	720000	248865	0	55355	72	5662.55
10	1036575	720000	253125	0	63450	75	6941.73

Instance 18

Run	TC	PS	PI	CI	TC	TV	Time
1	1034434	720000	249645	0	64789	74	5472.65
2	1036484	720000	248910	0	67574	76	8183.29
3	1037715	720000	250980	0	66735	73	4737.39
4	1028471	720000	249480	0	58991	77	4837.52
5	1028105	720000	251880	0	56225	74	5236.31
6	1023898	720000	243345	0	60553	76	3660.99
7	**1022257**	**720000**	**245715**	**0**	**56542**	**75**	**4982.98**
8	1029370	720000	250365	0	59005	74	5774.95
9	1025432	720000	252765	0	52667	75	4889.17
10	1044347	720000	251415	0	72932	75	9386.31

Instance 19

Run	TC	PS	PI	CI	TC	TV	Time
1	1027504	720000	253395	0	54109	74	6614.56
2	1030672	720000	262515	0	48157	76	6545.23
3	1036324	720000	244440	0	71884	75	6464.88
4	1036920	720000	242490	0	74430	76	7722.1
5	1029312	720000	251715	0	57597	77	5826.19
6	1040890	720000	252465	0	68425	75	5622.73
7	1039521	720000	251865	0	67656	79	5122.87
8	1040804	720000	255195	0	65609	76	6127.1
9	1021623	720000	247110	0	54513	76	5569.69
10	**1021451**	**720000**	**252360**	**0**	**49091**	**76**	**5744.86**

Instance 20

Run	TC	PS	PI	CI	TC	TV	Time
1	1024580	720000	243450	0	61130	74	4811.36
2	1017155	720000	241425	0	55730	75	6555.25
3	1026231	720000	251595	0	54636	73	7373.82
4	**1009636**	**720000**	**240675**	**0**	**48961**	**74**	**8338.49**
5	1019300	720000	236505	0	62795	73	5552.34
6	1033669	720000	252900	0	60769	74	5427.04
7	1022242	720000	248895	0	53347	76	6242.07
8	1015826	720000	244470	0	51356	76	6200.54
9	1028592	720000	246570	0	62022	75	6224.86
10	1020048	720000	248955	0	51093	74	5755.15

Instance 21

Run	TC	PS	PI	CI	TC	TV	Time
1	1023818	720000	235305	0	68513	74	5345.17
2	1019978	720000	244680	0	55298	74	5615.62
3	1035927	720000	249600	0	66327	77	5825.49
4	**1005596**	**720000**	**241650**	**0**	**43946**	**77**	**5411.34**
5	1022371	720000	237300	0	65071	75	6433.24
6	1028844	720000	246060	0	62784	75	5196.02
7	1022911	720000	252090	0	50821	77	7616.4
8	1025501	720000	247725	0	57776	74	6780.54
9	1034848	720000	242865	0	71983	75	4710
10	1028901	720000	237765	0	71136	74	4777.95

Instance 22

Run	TC	PS	PI	CI	TC	TV	Time
1	1033859	720000	241695	0	72164	75	5712.13
2	1027007	720000	245625	0	61382	74	6723.52
3	1024197	720000	234270	0	69927	74	5522.34
4	1006098	720000	243030	0	43068	74	5224.12
5	1013436	720000	239265	0	54171	76	4924.41
6	1024110	720000	247395	0	56715	77	4757.96
7	1004740	720000	246540	0	38200	75	5973.81
8	1024644	720000	240705	0	63939	74	5755.26
9	1011207	720000	242220	0	48987	74	6792.15
10	**1002523**	**720000**	**240495**	**0**	**42028**	**73**	**6109.26**

Instance 23

Run	TC	PS	PI	CI	TC	TV	Time
1	1033702	720000	251670	0	62032	75	7221.93
2	1038326	720000	245730	0	72596	75	4794.48
3	1027168	720000	244395	0	62773	74	7078.02
4	**1014295**	**720000**	**246000**	**0**	**48295**	**78**	**5446.19**
5	1038699	720000	253245	0	65454	75	6252.77
6	1023459	720000	248400	0	55059	74	8191.54
7	1038141	720000	251940	0	66201	75	5065.91
8	1020725	720000	246540	0	54185	74	5765.97
9	1035354	720000	249810	0	65544	75	6945.45
10	1027857	720000	258405	0	49452	76	7159.71

Instance 24

Run	TC	PS	PI	CI	TC	TV	Time
1	1024186	720000	231015	0	73171	75	5803.52
2	1026952	720000	237420	0	69532	74	6347.88
3	**1006545**	**720000**	**242820**	**0**	**43725**	**75**	**7903.89**
4	1022663	720000	240300	0	62363	74	5865.87
5	1024083	720000	240285	0	63798	75	10424.2
6	1032775	720000	239490	0	73285	75	6023.52
7	1024134	720000	239625	0	64509	75	6584.4
8	1017591	720000	246165	0	51426	74	6518.24
9	1014534	720000	235965	0	58569	75	6785.11
10	1030695	720000	231960	0	78735	72	5748.82

Instance 25

Run	TC	PS	PI	CI	TC	TV	Time
1	1032138	720000	253965	0	58173	76	6440.99
2	1025002	720000	247890	0	57112	74	5306.21
3	1036862	720000	246960	0	69902	75	6058.77
4	1017401	720000	249345	0	48056	76	6835.65
5	**1011588**	**720000**	**246105**	**0**	**45483**	**74**	**6613.72**
6	1021879	720000	251670	0	50209	76	5890.42
7	1037024	720000	249015	0	68009	74	7588.99
8	1025991	720000	248535	0	57456	75	5587.48
9	1019137	720000	251610	0	47527	75	6655.94
10	1020509	720000	245220	0	55289	74	5905.6

Instance 26

Run	TC	PS	PI	CI	TC	TV	Time
1	1027064	720000	242325	0	64739	73	5358.15
2	1012027	720000	228990	0	63037	75	5422.48
3	1009316	720000	241560	0	47756	74	6853.21
4	1026642	720000	244650	0	61992	74	6360.22
5	**1004669**	**720000**	**245955**	**0**	**38714**	**74**	**7584.26**
6	1019352	720000	246135	0	53217	76	5519.1
7	1026293	720000	249945	0	56348	76	5317.06
8	1005519	720000	241785	0	43734	74	6643.25
9	1006418	720000	237615	0	48803	75	5177.17
10	1012520	720000	236340	0	56180	74	5869.16

Instance 27

Run	TC	PS	PI	CI	TC	TV	Time
1	1034329	720000	253995	0	60334	75	6561.95
2	1033901	720000	258870	0	55031	76	6376.05
3	1066184	720000	256785	0	89399	77	6013.97
4	1032263	720000	253035	0	59228	76	7132.17
5	1024367	720000	259635	0	44732	78	7368.95
6	1038952	720000	260445	0	58507	76	5852.95
7	1042845	720000	244170	0	78675	74	5373.9
8	1053910	720000	255300	0	78610	75	6376.83
9	**1021082**	**720000**	**261015**	**0**	**40067**	**75**	**7355.74**
10	1036672	720000	255570	0	61102	75	6512.93

Instance 28

Run	TC	PS	PI	CI	TC	TV	Time
1	1036481	720000	240960	0	75521	75	7700.85
2	1039477	720000	237960	0	81517	73	10532.5
3	1015542	720000	244905	0	50637	75	8297.24
4	1014253	720000	238425	0	55828	75	5902.53
5	1020996	720000	236115	0	64881	76	5325.89
6	1031340	720000	239805	0	71535	78	6474.75
7	1021856	720000	234285	0	67571	75	4743.67
8	1017333	720000	238800	0	58533	76	8066.18
9	1027138	720000	247410	0	59728	76	6226.78
10	**1007917**	**720000**	**231180**	**0**	**56737**	**74**	**5377.46**

Instance 29

Run	TC	PS	PI	CI	TC	TV	Time
1	1025655	720000	233310	0	72345	77	4833.06
2	1017582	720000	241110	0	56472	77	4747.16
3	1033277	720000	231510	0	81767	75	5503.74
4	**1010323**	**720000**	**246990**	**0**	**43333**	**76**	**6741.24**
5	1017606	720000	245940	0	51666	77	6102.39
6	1011984	720000	244950	0	47034	75	7742.9
7	1028507	720000	240195	0	68312	75	6779.11
8	1014359	720000	242310	0	52049	76	6636.89
9	1018296	720000	235875	0	62421	74	6310.12
10	1030027	720000	246255	0	63772	77	6886.29

Instance 30

Run	TC	PS	PI	CI	TC	TV	Time
1	1034026	720000	253095	0	60931	75	6118.63
2	1032354	720000	244215	0	68139	76	5765.44
3	1018793	720000	247395	0	51398	74	6740.38
4	1025769	720000	249195	0	56574	76	7166.67
5	1023372	720000	255255	0	48117	76	6714.44
6	1026632	720000	243660	0	62972	74	5919.04
7	1039137	720000	246720	0	72417	77	5437.75
8	**1014027**	**720000**	**250455**	**0**	**43572**	**76**	**7257.26**
9	1034456	720000	251700	0	62756	75	5157.47
10	1014252	720000	243045	0	51207	73	6648.79