THE UNIVERSITY OF QUEENSLAND

AUSTRALIA

Rich Variants of the Vehicle Routing Problem

Ali Mehsin Alyasiry

B.Sc., Statistics

M.Sc., Operations Research

A thesis submitted for the degree of Doctor of Philosophy at

The University of Queensland in 2020

School of Mathematics and Physics

Abstract

The *Vehicle Routing Problem* (VRP) is one of the most important optimization problems in the field of *Operations Research* (OR) and it has been an interesting and challenging subject for OR researchers for more than fifty years. Moreover, routing problems are crucial in real life when people or goods are carried from one place to another. The solution of the VRP requires designing optimal routes from one or more depots to a number of customers. This will usually involve the minimization of some combination of the number of vehicles used and the total distance traveled. The term *Rich Vehicle Routing Problems* (RVRPs) arises when applying real-life extensions and when adding realistic constraints to the problem.

The *Pickup and Delivery Problem with Time Windows* (PDPTW) is a generalization of the VRP. In the PDPTW vehicles with limited capacity must be routed to serve given requests each of which consists of a pickup and a corresponding delivery. For each request, the pickup must precede the delivery (precedence) and both must be performed by the same vehicle (pairing). Routes must respect precedence, pairing, vehicle-capacity, and time-window constraints, as well as constraints which apply to specific problem variants. Incorporating loading constraints to the routing will add more complexity to the problem and make it very challenging.

Solution techniques for the PDPTW have focused on either heuristic approaches or increasingly complex exact algorithms based on branch-and-cut-and-price schemes. Very little work has been done on other possible exact solution techniques for variants of the PDPTW.

This thesis proposes a novel exact method by introducing a new methodology and formulation for exactly solving the PDPTW and its variants. This method is based on *fragments* - a series of pickup and delivery requests starting and ending with an empty vehicle. Using fragments, we formulate a relaxed network flow model with side constraint and use lazy constraints to cut off any illegal solutions generated while solving the resultant integer program. This method is easy to implement and can be extended in a straightforward way to solve most variants of the PDPTW for problems where it is possible to generate all fragments. Computational results confirm that this method outperforms the current state-of-the-art algorithms for solving the *Pickup and Delivery Problem with Time Windows and Last-in-First-Out Loading* (PDPTWL) and the *Pickup and Delivery Problem with Time Windows and Multiple Stacks* (PDPTWMS). This thesis also introduces for the first time an algorithm for solving a real-life extension of the PDPTWMS. Moreover, new valid inequalities for solving the PDPTW and its variants are introduced.

Declaration by author

Publications included in this thesis

peer-reviewed papers

Alyasiry, Forbes, and Bulmer (2019) **Alyasiry A**, Forbes M, and Bulmer M (2019) An exact algorithm for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science* 53(6):1695–1705. – Incorporated as Chapter 3.

Submitted manuscripts included in this thesis

Alyasiry A, and Forbes M (2019) An exact algorithm for the pickup and delivery problem with time windows and multiple stacks, submitted to *Computers and Operations Research* on September 26, 2019. – Incorporated as Chapter 4.

Other publications during candidature

No other publications

Contributions by others to the thesis

My supervisors Dr. Michael Forbes and Dr. Michael Bulmer supervised the whole research project and assisted in proof reading the thesis. Two anonymous reviewers contributed to chapter 3.

Statement of parts of the thesis submitted to qualify for the award of another degree

No works submitted towards another degree have been included in this thesis.

Research involving human or animal subjects

No animal or human subjects were involved in this research.

Acknowledgments

First and foremost, I would like to thank my principal supervisor Dr. Michael Forbes for his guidance and support, due to his wide knowledge and practical experience he has proven his smart leadership along the way during my PhD study. Also, I would like to extend my thanks to Dr. Michael Bulmer the associate advisor for his support and encouragement. Finally, thanks to my family for their patience for years during my study.

Financial support

This research was supported by the Iraqi Ministry of Higher Education and Scientific Research.

Keywords

vehicle routing, pickup and delivery, LIFO loading, lazy constraints.

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 010206, Operations Research, 100%

Fields of Research (FoR) Classification

FoR code: 0102, Applied Mathematics, 100%

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations	
AGV	Automated Guided Vehicles
CVRP	Capacitated Vehicle Routing Problem
DARP	Dial-a-Ride Problem
DTSPMS	Double Traveling Salesman Problem with Multiple Stacks
DVRPMS	Double Vehicle Routing Problem with Multiple Stacks
FIFO	First-in-First-Out
LIFO	Last-in-First-Out
LNS	Large Neighborhood Search
OR	Operations Research
PDP	Pickup and Delivery Problem
PDPTW	Pickup and Delivery Problem with Time Windows
PDPTWL	Pickup and Delivery Problem with Time Windows and Last-in-First-Out Loading
PDPTWMS	Pickup and Delivery Problem with Time Windows and Multiple Stacks
PDTSP	Pickup and Delivery Traveling Salesman Problem
PDTSP	Pickup and Delivery Traveling Salesman Problem
PDVRP	Pickup and Delivery Vehicle Routing Problem
RMP	Restricted Master Problem
SCP	Stacker-Crane Problem
SVPDP	Single-Vehicle Pickup and Delivery Problem
TSP	Traveling Salesman Problem
TSPPD	Traveling Salesman Problem with Pickup and Delivery
VRP	Vehicle Routing Problem
VRPPD	Vehicle Routing Problems with Pickups and Deliveries
VRPTW	Vehicle Routing Problem with Time Windows

Symbols	
G	Network graph, $G = (V, \bar{A})$
V	Set of nodes or vertices in G including the depot
$\bar{A}$	Set of all feasible arcs in G
(i,j)	Arc from node i to node j, $(i,j) \in \bar{A}$
P	Set of all pickup nodes
D	Set of all delivery nodes
e_i	Earliest time that a service can begin at node i
l_i	Latest time that a service can begin at node i
L	Length of the planning horizon
q_i	Load or unload associated with node i
s_i	Service duration time at node i
t_{ij}	Travel time from node i to node j includes s_i
c_{ij}	Cost of traversing arc (i,j)
K	Set of all available vehicles
Q	Vehicle capacity
b_i^k	Time that vehicle k begins service at node i
w_i^k	Total load of vehicle k upon leaving node i
x_{ij}^k	Binary variable equal to 1 iff vehicle k traverses arc (i,j)
G^R	Relaxed network of fragments and empty movement arcs
Φ	Set of all fragments
σ_f	Total internal cost of fragment f
$\tau_f(t)$	Transit time, total time required for a vehicle to travel through the nodes of f
f^-	Tail node of fragment f, $f^- \in P$
f^+	Head node of fragment f, $f^+ \in D$
η_f	Earliest arrival time at f^+
ψ_f	Latest possible departure time from f^-
$\Phi(p)$	Set of all fragments that contain node $p \in P$
x_f	Binary variables, equal to one iff f is used in the solution.
$\mathscr{A}$	Set of all non-timed arcs in G^R
c_α	Cost of traversing arc α
y_α	Binary variable, equal to 1 iff α is used in the solution.
α^-	Tail node of arc $\alpha, \alpha^- \in V$
α^+	Head node of arc $\alpha, \alpha^+ \in V$
G^{RT}	Relaxed time discretised network of fragments, empty arcs and waiting arcs

N	Set of all timed nodes including non-timed depot node
i_h	Timed node, $i_h \in N$, $i \in P \cup D$
δ	Discretization parameter (length of time)
Ω	Set of timed fragments
x_ω	Binary variables, equal to one iff ω is used in the solution.
$\Omega(p)$	Set of all timed fragments that contain node $p \in P$
ω^-	Tail timed node of ω, $\omega^- \in N$
ω^+	Head timed node of ω, $\omega^+ \in N$
A	Set of timed empty arcs and waiting arcs in G^{RT}
c_a	Cost of traversing arc a
y_a	Binary variable, equal to 1 iff a is used in the solution.
a^-	Tail timed node of a, $a^- \in N$
a^+	Head timed node of a, $a^+ \in N$
S	Set of a vehicle compartments
Q_s	Capacity of compartment $s \in S$
$X_f(p)$	Pickup position of request $p \in f$, $p \in P$
$\Gamma_f(p)$	Delivery position of request $p \in f$, $p \in P$
θ	A set of requests (represented by the pickup nodes)
I	A set of stack incompatible pairs of requests
Λ_{ps}	Number of units of request $p \in P$ that are placed on stack s, $\Lambda_{ps} \leq q_p$
Υ_{ps}	Binary variables, equal to 1 iff some part of request $p \in P$ uses stack s
$O(\omega)$	Original fragment of a timed fragment, $O(\omega) = f$, $\omega \in \Omega$, and $f \in \Phi$
$O(a)$	Original arc of a timed arc, $O(a) = \alpha$, $a \in A$, and $\alpha \in \mathscr{A}$
$t(i_h)$	Time associated with the timed node, $t(i_h) = h$, $i_h \in N$
$\oplus$	Concatenation operator
$\overrightarrow{\Phi}(\alpha)$	Set of fragments that can follow an empty arc α
$\overrightarrow{\Omega}(a)$	Set of timed fragments that can follow a timed empty arc a
$\overleftarrow{\Phi}(\alpha)$	Set of fragments that can precede an empty arc α
$\overleftarrow{\Omega}(a)$	Set of timed fragments that can precede a timed empty arc a
$\overrightarrow{\mathscr{A}}(f)$	Set of arcs that can follow fragment f
$\overrightarrow{A}(\omega)$	Set of timed arcs that can follow timed fragment ω
$\overleftarrow{\mathscr{A}}(f)$	Set of arcs that can precede fragment f
$\overleftarrow{A}(\omega)$	Set of timed arcs that can precede timed fragment ω
$M(f)$	Set of all *minimal-matching* fragments to f, $M(f) \subseteq \Phi$, and $\lvert M(f) \rvert \geq 1$

Chapter 1

Introduction

1.1. Motivation

Transportation plays a major role in economic growth by linking people to resources. It has a direct impact on business activity such as productivity, employment and access to new markets. People are either customers seeking quality transportation services or providing transportation services for profit. Maximizing benefits or reducing costs is a core element in the transportation/logistic industry. According to the Australian Logistics Council, 8.6% of Australia's gross domestic product (GDP) is a direct outcome of the freight logistics industry with an estimated total employment of 1.2 million people in 2013. Furthermore, even a 1% efficiency improvement in this sector will add $2 billion to GDP (ALC 2018). According to the Commonwealth Scientific and Industrial Research Organization, logistics "is an area of research with potential for high impact". In Australia consumers pay a 10 % overhead cost on the price of goods to cover transportation expenses (CSIRO 2018).

At the same time, transportation is one of the major sources of air pollution and emission of carbon dioxide. In the United States transportation accounted for about 26% of total greenhouse gas emissions in 2014 (U.S. EPA 2018). Governments endeavor to reduce road congestion and the negative impacts of transportation activities on the environment. According to the Bureau of Infrastructure, Transport and Regional Economics (BITRE) "the avoidable cost of congestion for the Australian capital cities is estimated to be around $16.5 billion for the 2015 financial year, having grown from about $12.8 billion for 2010" (BITRE 2018).

Operations Research (OR) has been extensively used to enhance the performance of transportation applications via varieties of optimization techniques. Typically, the aims are to reduce the operation time and cost and to minimize the negative effects of the transportation on the ecosystem. Real-world transportation applications are usually very large and difficult to solve. Computational experiments showed that significant savings on times and costs of the transportation operations can be achieved using computerized solution techniques based on mathematical models.

1.2. Background

Any development in science is one step in what is a cumulative process within a series of past and present generations of research. The problem of finding a route that visits all given points in a graph, starting and ending at the same point and visiting all points only once, has been known for centuries. The challenge in this research project was how to find a best route among many, a generalization of the well-known Hamilton's puzzle named after the Irish mathematician William Rowan Hamilton (1805 – 1865). Shaping the idea, organizing the problem and giving it a name is a different story. The best term that described the problem, and which received a semi-consensus through the mathematical community in the beginning of the 20th century, was the *Traveling Salesman Problem* (TSP). It is not certain who posed the TSP first. Dantzig, Fulkerson, and Johnson (1954) stated that "The origin of this problem is somewhat obscure. It appears to have been discussed informally among mathematicians at mathematics meetings for many years". In his paper M. Flood claimed that "This problem was posed, in 1934, by Hassler Whitney in a seminar talk at Princeton University" (Flood 1956). However, A. Schrijver cast doubt on this claim (see, Schrijver 2005).

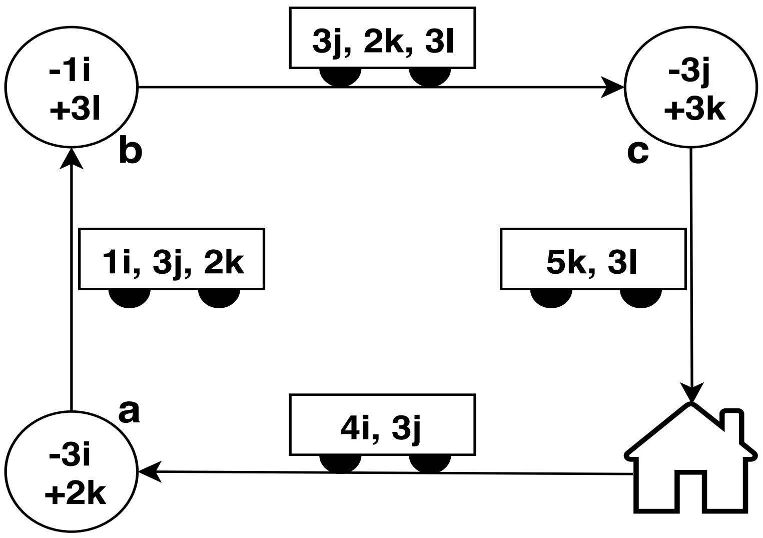
Dantzig, Fulkerson, and Johnson (1954) introduced for the first time an exact solution to the TSP. In their seminal paper they used a linear programming method to optimally solve the problem of 49 locations (48 states and Washington D. C.). Motivated by a real-world application Dantzig and Ramser (1959) introduced a realistic variant that generalized the TSP called "The Truck Dispatching Problem", known later as *Capacitated Vehicle Routing Problem* (CVRP). In the CVRP a solution may require more than one vehicle and each vehicle is restricted to a fixed capacity. Those remarkable changes opened the door for research on routing problems applying "what-if" scenarios. For example, in the case where the services at given locations must be started within given time intervals (time windows) then the CVRP will extend to the VRPTW in which the shortened form TW refers to time windows constraints. The abbreviation form VRP is usually used instead of CVRP if the problem incorporates constraints other than vehicle capacity (see, e.g., Cordeau et al. 2007).

Vehicle Routing Problems with Pickups and Deliveries (VRPPD) are very important variants of the VRPs. In the VRPPD vehicle(s) with finite capacity need to be routed to transport goods, such that item(s) with a given size need to be carried from pickup (origin) nodes to delivery (destination) nodes. In addition, the origin must precede the destination and both must be on the same route. The most common objective is to find the best routes that will minimize the number of vehicles used and the total costs (distance traveled). Other minimization objectives may arise such as route duration or customer inconvenience in passenger transportation systems.

In general the VRPPD can be classified as static (deterministic) or dynamic. If the information about all orders (requests) is known to the dispatcher in advance, before planning the vehicle(s) route(s), then the problem is static, otherwise it is dynamic. In the dynamic case the dispatcher has to either predict unknown orders before constructing the route(s) or re-optimize (update) the route(s) plan when receiving a new order. Prediction (estimation) can be done using probability distributions. However, the dynamic case is outside the scope of this thesis. In the literature, there are three common types of the VRPPD depending on the problem structure: One-to-Many-to-One, Many-to-Many and One-to-One (see, e.g., Battarra, Cordeau, and Iori 2014; Cordeau, Laporte, and Ropke 2008).

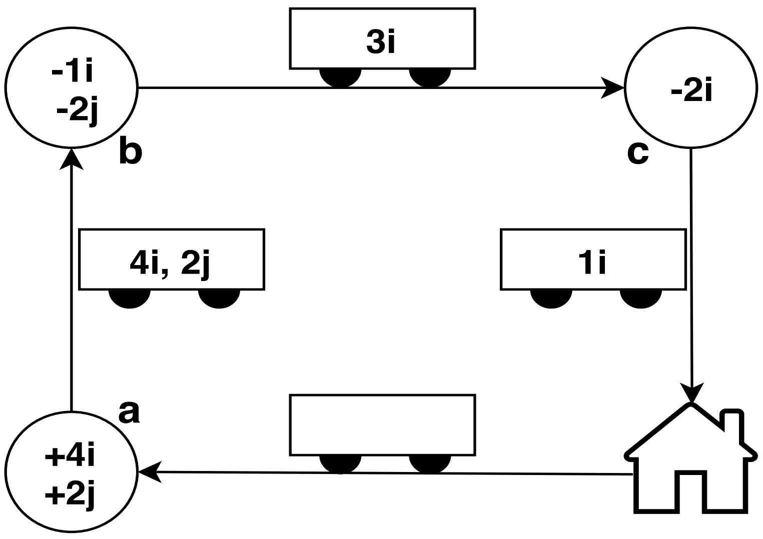
In One-to-Many-to-One there are at least two types of commodities. Goods are delivered from the depot to many locations and some other materials are collected and returned to the depot. Customer location may involve both types of services, unloading item(s) coming from the depot (One-to-Many), or loading item(s) going to the depot (Many-to-One). One-to-Many-to-One problems can be seen in transportation of beverages, home appliances and pallets, for example delivering full pallets and collecting the empty ones. Figure 1.1 illustrates a route structure of a One-to-Many-to-One problem. The route consists of a depot and three customer locations a, b and c with four types of commodities i, j, k and l.

Figure 1.1. An Example of One-to-Many-to-One Problem



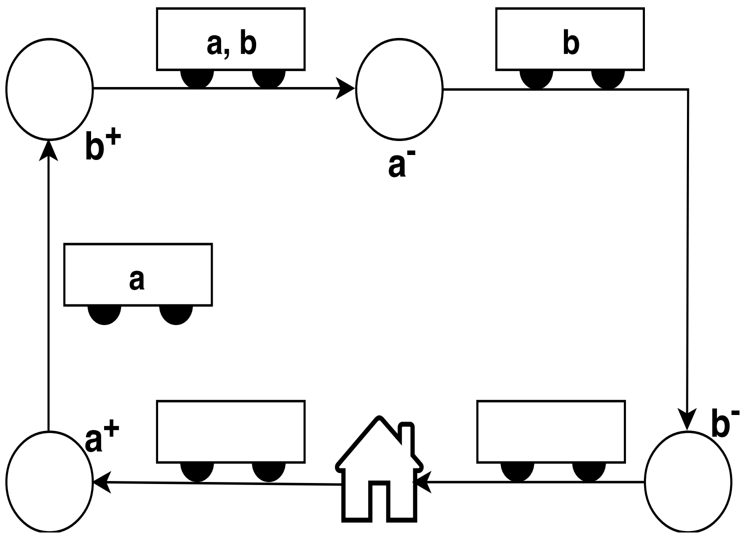
In Many-to-Many, customer locations can be sources (origins) or destinations of item(s). This problem can be seen in stock relocating and transportation of items between a chain of manufactures. Figure 1.2 illustrates a route structure of a Many-to-Many problem. The route consists of a depot and three customer locations a, b and c with two types of commodities i and j.

Figure 1.2. An Example of Many-to-Many Problem



The One-to-One problem is the classical VRPPD, denoted as the *Pickup and Delivery Problem* (PDP). In One-to-One PDPs each load (item) has one point of origin and one destination: the origin and its corresponding destination form a request. Examples of this problem occur in urban courier operations, maritime shipping and door-to-door passenger transportation. Figure 1.3 illustrates a route structure of a One-to-One problem. The route consists of a depot and two requests (four nodes) a and b.

Figure 1.3. An Example of One-to-One Problem



The VRPPD can be classified into the following (Parragh, Doerner, and Hartl 2008b):

1. In cases where there are no pairing constraints between pickup and delivery nodes the VRPPD is referred to as:
   i) The *Pickup and Delivery Traveling Salesman Problem* (PDTSP), in the single-vehicle case.
   ii) The *Pickup and Delivery Vehicle Routing Problem* (PDVRP), in the multi-vehicle case.
2. In cases where pickup and delivery nodes are paired (One-to-One) to form requests the VRPPD is referred to as:
   i) The *Single-Vehicle Pickup and Delivery Problem*(SVPDP), in the single-vehicle case.
   ii) The *Pickup and Delivery Problem* (PDP), in the multi-vehicle case.

This thesis focuses mainly on the *Pickup and Delivery Problem with Time Windows* (PDPTW). The objective is to develop algorithms for solving some complicated PDPTW variants. The complexity is due to constraints accompanying the real-life extensions. This is usually referred to as the *Rich Vehicle Routing Problem* (RVRP). In the PDPTW a vehicle route must satisfy the following constraints to be feasible:

i) Each node must be visited only once.
ii) Vehicle load never exceeds its capacity.
iii) The pickup and the delivery nodes of a request must served by the same vehicle.
iv) For each request the pickup node must be visited before its corresponding delivery node.
v) The time windows for all nodes must respected.

While the PDPTW is modeled for goods transportation, its extension, the so called *Dial-a-Ride Problem*

(DARP), is used with a passenger transportation system. In addition to the PDPTW constraints above, the route of the DARP must satisfy the following constraints to be feasible:

i) The ride (travel) time of each customer must not exceed a predefined time.

ii) Customer waiting time at any intermediate locations must be either zero or limited to a predefined time.

These additional constraints are referred to as *customer inconvenience* constraints.

The PDPTW is a rich variant of the vehicle routing problem (VRP). Considering additional real-life extensions such as using vehicles with multiple stacks and imposing loading constraints to each stack makes the problem richer.

The *Pickup and Delivery Problem with Time Windows and Last-in-First-Out (LIFO) Loading* (PDPTWL) also requires that loading and unloading an item must obey the LIFO loading policy (see, Cherkesly, Desaulniers, and Laporte 2015). The *Pickup and Delivery Problem with Time Windows and Multiple Stacks* (PDPTWMS) is a generalization of the PDPTWL for the multiple stacks case. In the PDPTWMS all vehicles are assumed to be identical, and have multiple independent compartments (independently accessible) of finite capacity. Loading and unloading an item in each compartment must obey the LIFO loading policy which is called the multi-stack policy in this context. A vehicle route is feasible for the PDPTWMS if it is feasible for the PDPTW and respects the multi-stack policy (see, Cherkesly et al. 2016). Under LIFO loading policies any loaded request must not be moved until it reaches its destination. This will avoid extra cost associated with rearranging requests inside a stack.

1.3. Summary of Contributions

This thesis presents a novel branch-and-cut method and introduces a new methodology and formulation for exactly solving the PDPTW and its variants.

Generating all routes that satisfy the PDPTW constraints, even for modest size problems, leads to a huge number of routes. As an alternative to current methods that consider entire routes, the proposed method reduces the number of variables by generating fragments. A fragment is a part of legal vehicle route such that the vehicle arrives empty at a first (pickup) node and departs empty from a last (delivery) node, but it is never empty at any intermediate node.

We construct a *relaxed network* flow model with side constraints to chain the fragments. We omit constraints to model the time windows and to eliminate sub-tours (cycles). Thus, when chaining together the fragments implied by the solution of the integer program it may not be possible to meet all time window constraints and/or sub-tours may occur. We use lazy constraints to cut off any such infeasible solutions generated while solving the integer program.

Moreover, the time windows of all nodes except for the depot can be discretized into timed nodes using a fixed-width time interval. This will produce several types of *timed* arcs: *start* arcs that are used to connect the depot to a timed pickup node, *end* arcs that connect a timed delivery node to the depot, *waiting* arcs that represent a vehicle waiting at each timed node, and *empty* arcs (empty vehicle arcs) that are used to connect a timed delivery node to a timed pickup node. Time discretization also creates multiple versions of each fragment. A *timed fragment* starts from a timed version of the first pickup node and ends at a timed version of the last delivery node. In addition, the arrival times at the end of each timed fragment and each timed empty arc are rounded down to the closest time interval.

Due to the cumulative effects of rounding down arrival times, not all paths through the relaxed network correspond to legal vehicle routes. Illegal paths will be eliminated later using cuts, implemented using the *lazy constraint* technology available in commercial IP solvers. A solver *Callback* function checks that every chain of fragments is a valid vehicle route whenever it finds a candidate improved integer feasible solution. If a solution is not feasible, then lazy constraints (cuts) will be used to cut off the combination of variables corresponding to the smallest portion of the chain that is illegal, and to eliminate any cycles that may occur.

This thesis also introduces for the first time an algorithm for solving a very important practical extension to the PDPTWMS. The work in this thesis assumes that the size of a customer's demand (order) may exceed the stack capacity and be only limited to the vehicle's capacity, and a customer demand can be split among all stacks. Applying this extension can result in a big reduction in the total traveled distance and/or the number of vehicles used. To the best of our knowledge, all previous methods in the literature that have dealt with multiple stacks routing such as: the *Single-Vehicle Pickup and Delivery Problem with Multiple Stacks*; the *Double Traveling Salesman Problem with Multiple Stacks* (DTSPMS); the *Double Vehicle Routing Problem With Multiple Stacks* (DVRPMS); the PDPTWMS, have assumed that the demand of each customer is either a single-size unit or cannot be split among different stacks and cannot exceed the stack's capacity.

Moreover, we introduced several valid inequalities and infeasible paths cuts for the PDPTW. Computational results showed the effectiveness of these constraints, in regard to the better lower bound obtained and reduced solution times.

1.4. Thesis Structure

The remainder of the thesis is organized as follows.
Chapter 2 provides a literature review on the PDPTW variants that are related to the work of this thesis and the algorithms used for solving them.
Chapter 3 presents a novel exact method and introduces a new methodology and formulation for solving the PDPTW and its variants. Also, it shows that the proposed method can applied to solve the PDPTW with last-in-first-out (LIFO) Loading (PDPTWL).
Chapter 4 shows that the proposed method can applied to solve the PDPTW with multiple stacks (PDPTWMS). Also, it introduces a new algorithm that can solve a real-life extension related to the PDPTWMS.
Chapter 5 presents valid inequalities and infeasible paths cuts for the PDPTW.
Chapter 6 provides conclusions and ideas for further work.

Chapter 2

Literature Review

2.1. Introduction

The PDP is a generalization of the TSP. The TSP is known to be an NP-hard combinatorial problem (see, e.g., Lenstra and Rinnooy Kan 1981; Garey and Johnson 1979; Papadimitriou 1977). Savelsbergh (1985) showed that the TSP with time windows (TSPTW) is NP-complete. A vast number of publications in the literature considered the PDP and many exact and heuristic algorithms have been developed to solve the problem. When the routes are designed to serve passengers the problem is called the *Dial-a-Ride Problem* (DARP). This chapter presents the most important approaches, found in the literature, for solving the PDP in the context of goods transportation with and without time windows, and devoted to the static (deterministic) case.

The PDPTW can be modeled using a directed graph $G = (V, \bar{A})$ with customers as a set of nodes or vertices V and the connection lanes between them as a set of arcs $\bar{A}$. The notations and formulation presented by Ropke and Cordeau (2009) are used in this section. Suppose the problem contains n requests. If $i, 1 \leq i \leq n$, denotes any pickup node then the corresponding delivery node will be denoted by $n+i$. Nodes 0 and $2n+1$ represent the start and end depot respectively. The set of nodes V can be characterized as $V = \{0, 2n+1\} \cup P \cup D$, where $P = \{1, \cdots, n\}$ is a set of pickup nodes, and $D = \{n+1, \cdots, 2n\}$ is a set of delivery nodes. The set of arcs $\bar{A}$ can be characterized as $\bar{A} = \{(i, j) : i, j \in V, i \neq j\}$. To each node $i \in V$ are associated a time window $[e_i(earliest), l_i(latest)]$, a service duration $s_i \geq 0$ and a load q_i, such that $s_0 = s_{2n+1} = 0$, $q_0 = q_{2n+1} = 0$, $q_i > 0$ if $i \in P$, $q_i = -q_{i-n}$ if $i \in D$. The length of the planning horizon is denoted by L. At any node, waiting is permissible if the vehicle arrives before the earliest time. Arriving after the latest time is not allowed. For some nodes $i \in P \cup D$ the time windows can be tightened, and for some arcs $(i, j) \in \bar{A}$ infeasible arcs can be eliminated using the rules presented in Dumas, Desrosiers, and Soumis (1991).

We assume that we have an unlimited supply of identical vehicles. Let K be the set of all available vehicles. Each vehicle has a capacity Q. Let b_i^k and w_i^k, $i \in V, k \in K$, be variables that denote the time that vehicle k begins service at node i and the load of vehicle k upon leaving node i respectively. The travel cost from node i to node j is denoted by c_{ij} and the travel time is denoted by t_{ij}. We assume that the travel time t_{ij} includes the service duration time s_i at node i, and that travel costs and travel times satisfy the triangle inequality. Let x^k_{ij} be a binary variable equal to 1 if and only if vehicle k traverses arc (i,j). The PDPTW can be formulated as a non-linear mixed-integer program (MIP) as follows:

$$\text{Min} \quad \sum_{k\in K}\sum_{i\in V}\sum_{j\in V} c_{ij}x^k_{ij} \tag{2.1}$$

subject to

$$\sum_{k\in K}\sum_{j\in V} x^k_{ij} = 1, \qquad \forall i \in P \tag{2.2}$$

$$\sum_{j\in V} x^k_{ij} - \sum_{j\in V} x^k_{n+i,j} = 0, \qquad \forall i \in P, k \in K \tag{2.3}$$

$$\sum_{j\in V} x^k_{0j} = 1, \qquad \forall k \in K \tag{2.4}$$

$$\sum_{j\in V} x^k_{ji} - \sum_{j\in V} x^k_{ij} = 0, \qquad \forall i \in P \cup D, k \in K \tag{2.5}$$

$$\sum_{i\in V} x^k_{i,2n+1} = 1, \qquad \forall k \in K \tag{2.6}$$

$$b^k_j \geq (b^k_i + t_{ij})x^k_{ij}, \qquad \forall i \in V, j \in V, k \in K \tag{2.7}$$

$$w^k_j \geq (w^k_i + q_j)x^k_{ij}, \qquad \forall i \in V, j \in V, k \in K \tag{2.8}$$

$$b^k_i + t_{i,n+i} \leq b^k_{n+i}, \qquad \forall i \in P, k \in K \tag{2.9}$$

$$e_i \leq b^k_i \leq l_i, \qquad \forall i \in V, k \in K \tag{2.10}$$

$$max\{0, q_i\} \leq w^k_i \leq min\{Q, Q + q_i\}, \qquad \forall i \in V, k \in K \tag{2.11}$$

$$x^k_{ij} \in \{0,1\}, \qquad \forall i \in V, j \in V, k \in K \tag{2.12}$$

The objective function (2.1) is to minimize the total routing cost. Constraints (2.2) - (2.3) guarantee that each request is served only once and the pickup and delivery nodes are served by the same vehicle. Constraints (2.4) - (2.6) guarantee that each vehicle departs from the origin depot and arrives at the destination depot. Constraints (2.7) and (2.8) guarantee the consistency of time windows and load variables respectively. Constraint (2.9) ensures that the pickup node precedes the delivery node of the same request. Constraint (2.10) ensures that time windows are satisfied. Constraint (2.11) guarantees that the load of vehicle k will not exceed its capacity at any node.

Constraints (2.7) and (2.8) can be linearized using standard big M techniques. However, this formulation has proved to be of little practical use for solving PDPTW (Ropke and Cordeau 2009). Nevertheless, a number of heuristic and exact solution techniques have been developed.

2.2. The Pickup and Delivery Problem for Goods Transportation

The focus of this thesis is One-to-One PDPs where the pickup and delivery nodes are paired. We use the same classification of (Parragh, Doerner, and Hartl 2008b) as explained in chapter 1. However, some papers in the literature use different abbreviation. For example some papers use PDTSP or TSPPD for the single-vehicle pickup and delivery problem.

2.2.1. Single-Vehicle Pickup and Delivery Problem (SVPDP)

The simplest version of the PDP is when the demand is a single unit and the vehicle can carry only a single object. The problem is called the *Stacker-Crane Problem* (SCP) as it is similar to operating a mobile crane. The problem was first formulated in the literature as an arc-routing problem. Frederickson, Hecht, and Kim (1976) presented a heuristic algorithm to solve the problem. They classified the problem into three categories depending on the node that a path terminated at: SCP if the route is a Hamiltonian circuit; SCP2 if the route terminates at a specified node, not necessarily the start node; SCP3 if the route terminates at any node. They also considered the multiple SCP (m-SCP). Moreover, they showed that the SCP and the m-SCP are NP-complete.

A local search heuristic was based on a variable-depth search for the SVPDP with time windows presented by Van Der Bruggen et al. (1993). The objective is to minimize the route duration. Their algorithm is similar to the Lin and Kernighan (1973) algorithm for the TSP. It consists of two phases. Seven types of arc-exchange procedures are used in both phases. The algorithm starts with an initial feasible solution in the first phase and improves it in the second one. Infeasible solutions were penalized in the objective function (allowed) in the first phase, then improved iteratively allowing feasible solutions only in the second phase. Renaud, Boctor, and Ouenniche (2000) have introduced a two-phase heuristic for the SVPDP without time windows. The solution is constructed in the first phase and the improvement made in the second phase. They called the first phase a Double Insertion heuristic. In the first phase a route is constructed by inserting each delivery node simultaneously to its corresponding pickup node. The second phase adapted the 4-Opt* improvement heuristic of Renaud, Boctor, and Laporte (1996). In this second phase, called the Deletion and re-Insertion heuristic, a pickup node and its corresponding delivery node are inserted or deleted at the same time. Renaud, Boctor, and Laporte (2002) have developed seven perturbation heuristics and obtained better results compared to Renaud, Boctor, and Laporte (1996).

A branch-and-bound algorithm was proposed by Kalantari, Hill, and Arora (1985) to solve a single and multiple vehicle PDP with finite or infinite capacity. Their method was designed to eliminate, in each branch at the nodes of the search tree, any arc violating the precedence condition. This algorithm was derived from Little et al. (1963). Ruland and Rodin (1997) presented a branch-and-cut algorithm similar to the Padberg and Rinaldi (1991) algorithm for the TSP. They examined the polyhedral structure of the SVPPD and introduced four types of valid inequalities (cuts): connectivity, precedence, generalized order and order-matching constraints. Their method was able to optimally solve instances with up to 15 requests. Dumitrescu et al. (2010) extended the branch-and-cut algorithm of Ruland and Rodin (1997) by introducing new valid inequalities and new polyhedral results for the SVPPD. Their algorithm was able to solve instances with up to 35 requests.

2.2.2. Multi-Vehicle Pickup and Delivery Problem (PDP)

An early metaheuristic for the PDPTW was developed by Nanry and Barnes (2000). They presented a reactive tabu search metaheuristic based on three types of move. The first move considers moving a request (a pair of pickup and delivery nodes) from one route to another. The second move involves swapping requests between routes. The third move consists of repositioning nodes within the same route. Instances with up to 50 requests were tested. Lau and Liang (2002) developed a two-phase metaheuristic. In the first phase an initial solution is constructed through a hybrid procedure, called a *partitioned insertion heuristic*, that combines an *insertion heuristic* and a *sweep heuristic*. A tabu search algorithm is used in the second phase and consists of three moves similar to those of Nanry and Barnes (2000).

Li and Lim (2003) developed a tabu-embedded simulated annealing metaheuristic for the PDPTW in which an *insertion heuristic* was used to construct an initial solution that satisfied all PDPTW constraints. Their method is based on a restart strategy from the best solution obtained after a number of non-improving iterations. To define a local neighborhood search three types of operators to pickup-delivery (PD) paired customers were used: *PD-Shift , PD-Exchange, and PD-Rearrange* operators. The PD-pairs first moved from one route to another in the PD-Shift operator. The PD-Exchange operator swaps PD-pairs of two routes. The third operator removes and then reinserts PD pairs in the same route. Infeasible insertions not allowed in all operators.

Bent and Van Hentenryck (2006) have proposed a two-stage hybrid metaheuristic. Simulated annealing (SA) is used in the first stage to reduce the number of vehicles. In the second stage a large neighborhood search (LNS) is used to minimize the total travel cost. The algorithm was tested on benchmark instances with 100, 200 and 600 customers. Ropke and Pisinger (2006) have developed a large neighborhood search (LNS) heuristic similar to the heuristic used by Shaw (1997) for the VRPTW. Their method uses several removal and insertion heuristics. Three types of request removal are applied: similar requests removal, random removal and worst request removal. They used a function called a *relatedness measure* to define the similarity between requests. Their LNS heuristic uses large moves to a current solution that can rearrange up to 30 - 40 percent of all requests in one iteration. Two types of parallel insertion were used: a basic greedy heuristic and several regret heuristics. This algorithm was tested on benchmark instances with up to 500 requests. An insertion-based heuristic was presented by Lu and Dessouky (2006) to solve the PDPTW. They defined the time difference between the time window and the service time as a slack in the time window. As a measure of the visual attraction of a current solution they consider a non-standard measure called *crossing length percentage* (CLP). In addition to the classical insertion rules, their algorithm incorporates the cost of decreasing the slack in the time window and the CLP into the insertion evaluation.

The first branch-and-price algorithm for the PDPTW was introduced by Dumas, Desrosiers, and Soumis (1991). Their algorithm is based on a set-partitioning formulation. Suppose the PDPTW consists of homogeneous fleet of vehicles. Let Ψ denote a set of all feasible routes satisfying the PDPTW and c_r denote the cost of route $r \in \Psi$. Due to the pairing conditions between the pickup and the delivery nodes, the set of pickup nodes P can also represent the set of requests. Let κ_{ir} be a binary constant equal to 1 if and only if request i is served by route r. The binary variable y_r is equal to 1 if and only if route r is used in the solution. The set-partitioning formulation to minimize the total routing cost for the PDPTW is as the follows:

$$\text{minimize} \quad \sum_{r \in \Psi} c_r y_r \tag{2.13}$$

subject to

$$\sum_{r \in \Psi} \kappa_{ir} y_r = 1, \qquad \forall i \in P \tag{2.14}$$

$$y_r \in \{0,1\}, \qquad \forall r \in \Psi \tag{2.15}$$

Constraints (2.14) guarantee that each request is served only once. The model (2.13) - (2.15) is called the *master problem*. In practical cases the cardinality of Ψ is a very large number and the model (2.13) - (2.15) is unsolvable. To solve the problem, the integrality condition is relaxed and a subset $\Psi' \subset \Psi$ is used. The resultant problem is called the *Restricted Master Problem* (RMP). The RMP initially starts with n routes $|\Psi'| = n$, one route for each request. A subproblem is used to generate new routes as needed using a forward dynamic programming algorithm. The subproblem is a shortest path problem with capacity, time windows and pickup and delivery. To make this subproblem produce good routes dominance rules were applied. At every iteration only columns (routes) with a negative reduced cost can enter the basis of the RMP. Optimizing the RMP produces dual variables associated with its constraints. The arcs costs of the subproblem are modified using the current value of the dual variables obtained. The subproblem is re-optimized to generate columns with negative marginal cost. If there is no such column to generate, then the optimal solution of the RMP is also optimal for the linear relaxation of the master problem. If that solution is not integer, then it is used as starting point (lower bound) for the branch-and-bound enumeration tree. To reduce the size of the underlying network arc elimination rules, in regard to the time window conditions, have been proposed as follows.
For $i, j \in \{1, ..., n\}$, if the travel times satisfy the triangle inequality, then the following arcs can be eliminated:

- If the path $\{j, i, j+n, i+n\}$ is infeasible, then the arc $(i, j+n)$ can be eliminated.
- If the path $\{i, i+n, j, j+n\}$ is infeasible, then the arc $(i+n, j)$ can be eliminated.
- If the paths $\{i, j, i+n, j+n\}$ and $\{i, j, j+n, i+n\}$ are both infeasible, then the arc (i, j) can be eliminated.
- If the paths $\{i, j, i+n, j+n\}$ and $\{j, i, i+n, j+n\}$ are both infeasible, then the arc $(i+n, j+n)$ can be eliminated.

Their algorithm can solve problems with multi-depot and fleets of heterogeneous vehicles. They successfully tested instances ranging from 19 to 55 requests.

Using different pricing rules, Savelsbergh and Sol (1998) presented a column generation based algorithm using a linear relaxation that allows variables to be generated by the branch-and-price algorithm. Their method used heuristics, whenever possible, to solve the pricing problem and instances of up to 30 requests were solved.

A branch-and-cut algorithm presented by Lu and Dessouky (2004) was based on relaxing the capacity and time windows. They dynamically added additional constraints to cut off solutions that do not respect the capacity or time window constraints. A two-index formulation was used and instances with up to 25 requests were solved.

A branch-and-cut algorithm was introduced by Ropke, Cordeau, and Laporte (2007) for the PDPTW. In contrast to Cordeau (2006), their new formulation has an exponential number of constraints but leads to a better solution because these constraints provide tighter bounds and contain fewer variables. They presented three new valid inequalities that were used within a branch-and-cut algorithm: strengthened capacity inequalities, strengthened infeasible path inequalities, and fork inequalities. Instances with up to 96 requests were solved to optimality within a reasonable time.

Ropke and Cordeau (2009) have presented a branch-and-cut-and-price algorithm that outperformed the previous algorithms. In their method, to compute a lower bound, a set partitioning of all feasible routes was solved by obtaining a restricted master problem, then generating variables of negative reduced cost at the current linear relaxation of the restricted master problem. This method is used with both elementary and non-elementary shortest path problems with side constraints as pricing subproblems. Five types of inequalities were used in the branch-and-cut-and-price algorithm: infeasible path, fork, reachability, rounded capacity, and two-path inequalities. Instances with up to 100 requests were solved to optimality.

Baldacci, Bartolini, and Mingozzi (2011) have introduced a new exact algorithm that took less time and solved more instances than Ropke and Cordeau (2009). In this method, a set partitioning style formulation is used. A near optimal dual solution to the linear programming relaxation is found by combining two dual ascent heuristics and a cut-and-column generation algorithm. They used that dual solution as a lower bound and then generated all variables with reduced costs smaller than the gap between the lower and upper bounds.

2.2.3. The Pickup and Delivery Problem with LIFO or FIFO Loading

This section provides an overview of the literature in regard to SVPDP and PDP with last-in-first-out (LIFO) or first-in-first-out (FIFO) loading with or without time window constraints. Applications of the pickup and delivery vehicle routing problem with LIFO loading constraints arise in the transportation of animals, heavyweight goods and hazardous materials such as livestock, cars and chemical containers where unloading vehicles requires more time and special handling. This problem can also be seen in the routing of automated guided vehicles (AGV) that are used for material transportation between workstations in manufacturing systems. In the LIFO loading system a vehicle is rear-loaded and any new loaded item(s) must be placed on the top of other item(s). To avoid unnecessary extra cost or danger related to rearranging items at any delivery location, the LIFO policy guarantees that the loaded item need not to be moved until it reaches its unload destination. An extension to this problem is when the load and unload are handled in FIFO fashion, that is, if request i is picked up before request j, then i must be delivered before j. This is very common in AGV routing problems where the items are stacked from one end and released from the other end.

Publications on pickup and delivery problems with loading constraints are quite limited. Only one exact method for the pickup and delivery problem with time windows and LIFO loading (PDPTWL) has been published (Cherkesly, Desaulniers, and Laporte 2015). According to Pollaris et al. (2015), research on combining VRPs and loading constraints is a relatively new domain and there are only seven papers in the literature that have considered pickup and delivery problems with loading constraints.

Carrabs, Cordeau, and Laporte (2007) introduced a variable neighborhood search heuristic and three new local search operators: multi-relocate, 2-opt-L, and double-bridge for solving the SVPDP with LIFO loading. Results for instances with up to 375 requests were reported. A branch-and-bound algorithm was introduced later by Carrabs, Cerulli, and Cordeau (2007) for the SVPDP with LIFO or FIFO loading. A lower bound was computed at each node of the branch-and-bound search tree by solving the assignment problem and the shortest spanning r-arborescence problem. Instances with up to 35 customers of the LIFO case and 27 customers of the FIFO case were solved to optimality.

Erdoğan, Cordeau, and Laporte (2009) proposed an integer linear programming formulation with a polynomial number of variables and constraints for the SVPDP with FIFO loading. They also showed that the problem is NP-hard. CPLEX was used to solve instances with up to 12 requests. They also proposed several heuristics, two constructive heuristics used as a starting point for two improvement heuristics: a probabilistic tabu search heuristic and an iterated local search heuristic. Cordeau et al. (2010) presented a branch-and-cut algorithm for the SVPDP with LIFO loading. Three mathematical formulations were introduced to impose the LIFO condition for the SVPDP. The first and the second formulations add an additional polynomial number of variables and constraints to the classical formulation for the SVPDP, considering situations in which the vehicle is capacitated or uncapacitated. In the third formulation the same variables of the SVPDP were used considering an uncapacitated vehicle case and an additional exponential number of constraints are added. Three new inequalities and previously known inequalities were used to strengthen the linear relaxation of these formulations. The new inequalities were: incompatible predecessor and successor inequalities, hamburger inequalities and incompatible path inequalities. A branch-and-cut algorithm for the SVPDP with FIFO loading also presented later by Cordeau, Dell'Amico, and Iori (2010).

Cherkesly, Desaulniers, and Laporte (2015) have proposed three exact branch-price-and-cut algorithms for the PDPTWL. The first algorithm includes the LIFO constraints in the master problem. The second algorithm uses an elementary shortest path pricing problem with pickup and delivery, time windows, capacity and LIFO constraints by applying a dynamic programming algorithm. A mixture of the first and the second algorithms is used by the third algorithm and instances with up to 75 requests were solved to optimality.

2.2.4. The Pickup and Delivery Problem with Multiple Stacks

In the pickup and delivery problem with multiple stacks PDPMS the vehicles are rear-loaded and the load space is divided into several compartments with fixed capacity. When an item is loaded it is placed on the top of a stack. Loading or unloading an item must obey the LIFO policy for each stack. Rearranging items is not allowed. Applications of the PDPMS can be seen in the transportation of livestock, pallets and cars.

It is worth mentioning that, to our knowledge, all papers have assumed that the demand at each customer is either a single-unit or cannot be split among different stacks. We see this as unrealistic in real-world situations. In chapter 4 we propose an exact algorithm to solve this practical case.

The only work found in the literature studying the pickup and delivery problem with time windows and multiple stacks (PDPTWMS) is the one of Cherkesly et al. (2016). They presented two branch-price-and-cut algorithms. The first algorithm solves the pricing problem by fully enforcing the multi-stack policy. The second algorithm relaxed multi-stack paths in the pricing problem. Infeasible path inequalities were added to the master problem if illegal multi-stack routes were used in a linear relaxation solution. Instances with up to 75 requests were solved to optimality within 2 hours of computation time.

A large neighborhood search has been proposed by Côté, Gendreau, and Potvin (2012) for the SVPDP with multiple stacks. Côté et al. (2012) proposed a branch-and-cut algorithm for the SVPDP with multiple stacks. New valid inequalities and three different formulations were presented and instances with up to 21 requests were solved to optimality. A branch-and-cut algorithm and a new integer programming formulation for the SVPDP with multiple stacks was proposed by Sampaio and Urrutia (2017). Also, new valid inequalities were presented. Their computational results are compatible to those obtained by Côté et al. (2012). Pereira and Urrutia (2018) have proposed new valid inequalities and two formulations with ad hoc branch-and-cut algorithms for the SVPDP with multiple stacks. Compared to the methods of Côté et al. (2012) and Sampaio and Urrutia (2017), their algorithm is faster and solved some instances that were previously unsolved.

Multiple stacks also occur in the *Double Traveling Salesman Problem with Multiple Stacks* (DTSPMS), in which a single vehicle must visit all pickup nodes in a first region and return to the depot before visiting all delivery nodes in a second region (see, e.g., Felipe, Ortuño, and Tirado 2009; Petersen and Madsen 2009; Petersen, Archetti, and Speranza 2010; Lusby et al. 2010; Carrabs, Cerulli, and Speranza 2013; Alba Martínez et al. 2013). A multi-vehicle case is the double vehicle routing problem with multiple stacks (DVRPMS) (see, Iori and Riera-Ledesma 2015).

The following publication has been incorporated as Chapter 3.

Alyasiry A, Forbes M, and Bulmer M (2019) An exact algorithm for the pickup and delivery problem with time windows and last-in-first-out loading. Transportation Science 53(6):1695–1705.

Contribution to the authorship: Ali Alyasiry and Michael Forbes conceived the main idea. Ali Alyasiry with support from Michael Forbes developed the theoretical framework. Ali Alyasiry drafted the manuscript, processed the experimental data, analyzed and interpreted the data, performed the calculations, and designed the figures. Ali Alyasiry and Michael Forbes carried out the proofreading of the manuscript. Michael Forbes and Michael Bulmer supervised the project. Two anonymous reviewers provided feedback that helped to improve the presentation of the manuscript.

Chapter 3

An Exact Algorithm for the Pickup and Delivery Problem with Time Windows and Last-in-First-out Loading

ABSTRACT

Applications of the pickup and delivery problem with time windows and last-in-first-out (LIFO) loading (PDPTWL) constraints can be found in the transportation of animals, heavyweight goods, and hazardous materials, where unloading vehicles requires more time and special handling. Examples include carrying livestock, cars, and chemical containers. Research on exact methods to solve the pickup and delivery problem with time windows (PDPTW) and its variants has mainly focused on branch-and-price-and-cut algorithms. In this chapter, we propose a novel exact approach based on *fragments* - a series of pickup and delivery requests starting and ending with an empty vehicle. We use fragments to formulate a relaxed network flow model with side constraints. Lazy constraints are used to cut off any illegal solution that may occur while solving the integer program. Extensive computational experiments show that the proposed approach is superior to the current state-of-the-art method.

3.1. Introduction

Solving the vehicle routing problem (VRP) requires designing optimal routes from one or more depots to a number of customers. This will usually involve the minimization of some combination of the number of vehicles used and the total distance traveled. The task is to determine an optimal routing plan where each customer is visited only once and the demands of each customer are completely satisfied. The classical variant of the VRP is the capacitated VRP (CVRP), in which the vehicle load cannot exceed some maximum capacity. The VRP with time windows (VRPTW) is another important extension of the CVRP, where the service at any location must begin within a prespecified time interval.

For a comprehensive review, we refer the reader to a recent book edited by Toth and Vigo (2014).

The pickup and delivery problem (PDP) is a subclass of the VRP. In the PDP, we are given requests defined by pickup locations and corresponding delivery locations. Each request must be served by just one vehicle (pairing constraint) and the pickup location must precede the delivery location (precedence constraint). The PDP with time windows (PDPTW) generalizes the VRPTW. In the PDPTW, constraints on vehicle capacity and allowed time windows at each location must be satisfied.

The problem is called the one-to-one PDPTW, the focus of this chapter, if each request consists of a single origin (pickup location) and a single destination (delivery location) (see, e.g., Berbeglia et al. 2007; Parragh, Doerner, and Hartl 2008a, b; Battarra, Cordeau, and Iori 2014). Although the PDPTW is used for modeling goods transportation, its extension, the so-called dial-a-ride problem (DARP), is used for passenger transportation. When dealing with passenger transportation, additional constraints such as ride time and waiting time constraints, are normally used to reduce customer inconvenience. Applications of the DARP can be seen in door-to-door transportation for disabled or elderly people (Cordeau and Laporte 2003).

To avoid unnecessary extra cost or danger related to rearranging items at any delivery location, the LIFO policy guarantees that the loaded item need not to be moved until it reaches its destination. Adding these complicating constraints gives extra richness to the PDPTW.

3.1.1. Novelty and Contributions

Research on exact methods to solve the PDPTW has been mostly restricted to branch-and-cut-and-price algorithms. In this chapter, we propose a novel exact method by introducing a new methodology and formulation for solving the PDPTW. This method is easy to implement and can be extended in a straightforward way to solve many variants of the PDPTW including the PDPTWL, which is the focus of this chapter.

The contributions of this chapter are the following:

- Compared with set partitioning formulations, we reduce the number of variables by using fragments of requests.
- We make multiple copies of each fragment by dividing the time window of each fragment's starting node into fixed-width intervals and rounding down the arrival time at each fragment's ending node to the closest time interval. We do the same for every possible arc connecting delivery to pickup nodes (empty vehicle arc) that may be used to connect the fragments.
- We construct a new relaxed network flow model to chain the generated fragments.
- We use a general-purpose integer programming (IP) solver to solve the relaxed model. We check that every chain of fragments is a valid vehicle route whenever we find a candidate improved integer feasible solution during the solution of the integer program. If not, lazy constraints are used to cut off the combination of variables corresponding to the smallest portion of the chain that is illegal and to eliminate any cycle that may occur.
- We present computation results that show that the proposed method is superior to the current

state-of-the-art method of Cherkesly, Desaulniers, and Laporte (2015). In addition, we have solved nine instances that were previously unsolved.

We define a fragment to be part of legal vehicle route from a pickup node to a delivery node such that the vehicle arrives empty at the pickup node and departs empty from the delivery node, but it is never empty at any intermediate node. Fragments are similar to mini-clusters mentioned in the literature (see, e.g., Desrosiers, Dumas, and Soumis 1988; Desrosiers et al. 1991; Ioachim et al. 1995; Bomdorfer 1998). However, the purpose and the role of fragments here are different from those of mini-clusters.

The remainder of the chapter is organized as follows. The next section describes the proposed method. In Section 3.3 we present the formulation, with the resulting algorithm shown in Section 3.4. In Section 3.5 we report computational results. Finally, conclusions are presented in Section 3.6.

3.2. Problem Description

The PDPTW can be modeled using a directed graph $G = (V, \bar{A})$ with customer locations as a set of nodes or vertices V and the connection lanes between them as a set of arcs $\bar{A}$. The notation presented in section 2.1 for the graph G are used in this chapter. We will refer to the graph G as network G. In the remainder of the thesis, we assume that we have a single depot represented by node 0.

3.2.1. Preprocessing

In addition to the arcs that can be eliminated using the rules presented in Dumas, Desrosiers, and Soumis (1991), the LIFO policy imposes the elimination of arcs $\{(i, j+n) : i, j \in P, i \neq j\}$. Also, time windows can be tightened as in Dumas, Desrosiers, and Soumis (1991).

3.2.2. Fragment Generation

Generating all admissible routes that satisfy the PDPTW constraints, even for modest size problems, leads to a huge number of routes. Branch-and-price approaches attempt to overcome this problem by considering all possible routes implicitly, rather than explicitly. Nonetheless, solving such problems normally requires sophisticated procedures. As an alternative to branch-and-price approaches, we reduce the number of variables by generating fragments. By definition, fragments respect all capacity, time window, pairing, and precedence constraints, as well as any other loading constraint such as those associated with the PDPTWL. A fragment starts with a pickup node and ends whenever the vehicle is empty at a delivery node.

Feasible fragments can be generated in different ways *e.g.* using (dynamic, labeling, recursive procedures *etc.*). To generate all feasible LIFO fragments we modified one of the existing algorithms used to enumerate all circuits or paths in a directed graph (see, e.g., Johnson 1975; Szwarcfiter and Lauer 1976; Sedgewick 2001). We use depth-first search to enumerate all paths by expanding the current elementary path under examination starting from node 0. The current path started at node

0 can be extended by inserting a candidate node, if that node respects all the PDPTWL constraints. Fragments can be generated in G by applying the following additional checks to one of the existing algorithms to enumerate all possible paths starting from node 0.

1. We can only extend to a delivery node if LIFO rules are respected.
2. We can only extend to a pickup node if capacity is respected.
3. We check that (in respect to time window) any partial path can be completed by delivering all the requests currently loaded. If this is not possible, do not extend the current path.
4. We terminate each individual search (stop expanding the current path) when all order(s) are delivered (a complete fragment is generated).

Each fragment is a result of removing node 0 (the depot) from the generated path. Let f denote the fragment and Φ denote the set of all fragments. Let σ_f denote the total internal cost of each fragment. Consider a fragment $f = (r_1, ..., r_m)$, then $\sigma_f = \sum_{i=1}^{m-1} c_{r_i, r_{i+1}}$. In addition, let $\tau_f(t) \in \mathbb{R}$ represent the *transit time*, the minimum elapsed time (including all waiting time) required for a vehicle to travel through the nodes of fragment f starting from node r_1 at time t.

3.2.3. Time Window Discretization

We define a fixed length of time denoted by δ. We discretized the time window of all nodes except for the depot. This will produce $\lambda_i + 1$ copies of node i, where $\lambda_i = \lceil \frac{(l_i - e_i - \delta)}{\delta} \rceil, i \in P \cup D$. Let Δ_i be the set of new versions of node i such that $\Delta_i = \{i_{e_i}, i_{e_i+\delta}, i_{e_i+2\delta}, \cdots, i_{e_i+\lambda_i\delta}\} = \{i_h : e_i \leq h \leq l_i - \delta\}, i \in P \cup D$. We will refer to nodes i_h as *timed* nodes to distinguish them from the original nodes.

A new network results from the time window discretization which consists of several types of timed arcs: *start* arcs that are used to connect the depot to a timed pickup node, *end* arcs that connect a timed delivery node to the depot, *waiting* arcs that represent the waiting at each timed node from time h to $h + \delta$, and *empty* arcs that are used to connect a timed delivery node to a timed pickup node. Note that the timed start arcs need only be created for i_{e_i} (the first timed version of node $i \in P$). Similarly, the timed end arcs need only be created for $i_{e_i+\lambda_i\delta}$ (the last timed version of node $i \in D$).

Time discretization also creates multiple versions of each fragment. We will refer to these fragments as *timed fragments* to distinguish them from the original ones. A timed fragment starts from a timed version of the first pickup location and ends at a timed version of the last delivery location. Note that timed arcs and fragments incur the same cost as the original versions, and waiting arcs incur 0 cost. Let ω denote a timed fragment and Ω denote the set of all timed fragments.

We round down the arrival time of each timed empty arc and timed fragment to the nearest timed node using the function $F(t, j) = max\{e_j, e_j + \delta \lfloor \frac{(t-e_j)}{\delta} \rfloor\}, j \in P \cup D$. In addition, we ensure that $e_j \leq h' \leq l_j$, where h' is the end time at node j of an empty timed arc or a timed fragment.

Many empty timed arcs and timed fragments can be eliminated using the following dominance criteria. An empty timed arc or a timed fragment which starts at a later time will dominate its peer if both arrive at the same timed node, and if both have the same sequence of pickup and delivery nodes.

We define a new relaxed network $G^{RT} = (N, A \cup \Omega)$. The set of nodes N can be characterized as

$N = \{0\} \cup (\cup_{i \in P \cup D} \Delta_i)$ and the set of arcs A can be characterized as $A = \{(0, i_{e_i}) : i \in P\} \cup \{(i_{e_i+\lambda_i\delta}, 0) : i \in D\} \cup \{(i_h, i_{h+\delta}) : i \in P \cup D\} \cup \{(i_h, j_{h'}) : (i, j) \in \bar{A},\ i_h, j_{h'} \in N \setminus \{0\},\ i \in D, j \in P,\ h' = F(h + t_{ij}, j)\}$. The set of timed fragments Ω can be characterized as $\Omega = \{(i_h, j_{h'}, f) : i_h, j_{h'} \in N \setminus \{0\},\ i \in P, j \in D,\ h' = F(h + \tau_f(h), j), f \in \Phi\}$.

Let $a \in A$ denote timed arcs (start, end, waiting and empty arcs), each of which has a start "tail" timed node denoted by $a^- \in N$ and an end "head" timed node denoted by $a^+ \in N$. Likewise, a timed fragment ω has a start "tail" timed node $\omega^- \in N$ and an end "head" timed node $\omega^+ \in N$. Moreover, let $T(f)$ denote the set of timed fragments that are generated from the original fragment $f \in \Phi$ such that $T(f) \subseteq \Omega$. Let $\beta \in \bar{A}$ denote the original delivery to pickup (empty) arc and let $T'(\beta)$ denote the set of timed empty arcs that generated from the original arc β such that $T'(\beta) \subseteq A$.

Every legal vehicle route for PDPTWL can be represented as a series of fragments and empty arcs. Due to the rounding down of arrival times, this series of fragments and empty arcs can in turn be represented as a path through G^{RT} where we choose the first version of each timed fragment and timed empty arc that commences at or after our arrival time at each node.

Due to the cumulative effects of rounding down arrival times, some paths through G^{RT} may not be legal vehicle routes for PDPTWL. These will be eliminated later using lazy constraints. Moreover, rounding up the arrival time to the nearest time interval would give a network which could be used for producing legal, but not necessarily optimal, solutions.

3.2.4. Numerical Example

Let $n = 5$, $Q = 22$, and $\delta = 10$. Suppose $t_{ij} = c_{ij}, \forall i, j \in V$ and the cost of each vehicle is very large. Table 3.1 shows the load, unload, and the time window of each node and provides the distance (c_{ij}) matrix between all pairs of nodes.

Table 3.1. Example Data

Node	q_i	e_i	l_i	0	1	2	3	4	5	6	7	8	9	10
0	0	0	400	0	7.39	25.69	22.47	21.50	26.28	30.93	22.51	26.20	7.22	10.26
1	7	55	95		0	21.57	15.89	15.93	26.16	23.54	15.16	18.83	4.74	10.22
2	7	95	135			0	11.50	7.38	14.31	24.83	22.31	23.17	18.65	16.37
3	8	75	115				0	4.57	24.05	14.00	10.92	11.72	15.62	17.03
4	6	125	165					0	19.48	18.56	15.01	16.21	14.30	14.26
5	11	175	215						0	38.05	33.58	35.45	21.58	16.41
6	-7	85	125							0	8.59	4.83	25.8	29.15
7	-7	125	165								0	3.76	17.97	22.05
8	-8	105	145									0	21.38	25.11
9	-6	155	195										0	5.47
10	-11	205	245											0

For this instance of the PDPTWL, there are 10 feasible fragments, 31 timed fragments, and 19 undominated timed fragments. The set of fragments Φ = $\{$(1, 3, 8, 6), (1, 6), (2, 3, 8, 7), (2, 7), (3, 1, 6, 8), (3, 2, 7, 8), (3, 8), (4, 2, 7, 9), (4, 9), (5, 10)$\}$. From the first fragment (1, 3, 8, 6), four timed fragments can be created, each one consisting of a timed version of the first pickup location, a timed version of the last delivery location, and the original fragment as follows: $T((1,3,8,6)) = \{(1_{(55)}, 6_{(105)}, (1,3,8,6)), (1_{(65)}, 6_{(105)}, (1,3,8,6)), (1_{(75)}, 6_{(105)}, (1,3,8,6)), (1_{(85)}, 6_{(115)}, (1,3,8,6))\}$. The last timed fragment starts at the timed node $1_{(85)}$, passes over node 3 and 8, and ends at time 117.53, rounded down to the timed node $6_{(115)}$. Because the first three timed fragments end at the same timed node $6_{(105)}$, the timed fragment that starts with timed node $1_{(75)}$ dominates the first two. This will produce $T((1,3,8,6)) = \{(1_{(75)}, 6_{(105)}, (1,3,8,6)), (1_{(85)}, 6_{(115)}, (1,3,8,6))\}$, the undominated timed fragments of the original fragment (1, 3, 8, 6).

Also, there are 10 feasible delivery to pickup arcs, 33 timed empty arcs, and 21 undominated timed empty arcs (note that the number of delivery to pickup arcs is equal to the number of fragments just by coincidence). The set of empty arcs β = {(6, 2), (6, 3), (6, 4), (6, 5), (7, 4), (7, 5), (8, 2), (8, 4), (8, 5), (9, 5)}. From the original empty arc (6, 4), four timed empty arcs can be created, each consisting of a timed version of the delivery location and a timed version of the pickup location as follows: $T'((6,4)) = \{(6_{(85)}, 4_{(125)}), (6_{(95)}, 4_{(125)}), (6_{(105)}, 4_{(125)}), (6_{(115)}, 4_{(125)})\}$. The last one dominates the others because they all end at the same timed node.

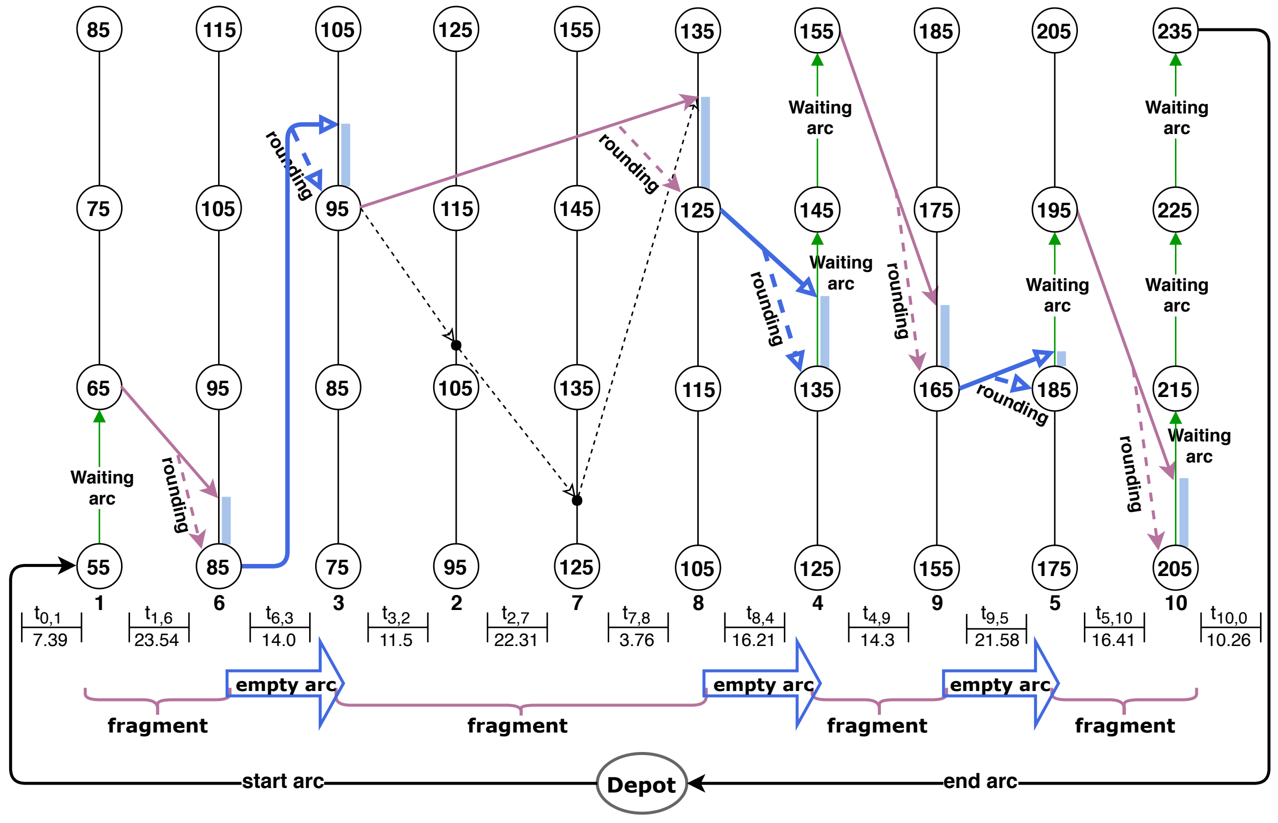
Finally, the optimal route is represented in G^{RT} as one timed start arc $(0, 1_{(55)})$, four timed fragments $\{(1_{(65)}, 6_{(85)}, (1,6)), (3_{(95)}, 8_{(125)}, (3,2,7,8)), (4_{(155)}, 9_{(165)}, (4,9)), (5_{(195)}, 10_{(205)}, (5,10))\}$, three timed empty arcs $\{(6_{(85)}, 3_{(95)}), (8_{(125)}, 4_{(135)}), (9_{(165)}, 5_{(185)})\}$, one timed end arc $(10_{(235)}, 0)$, and several waiting arcs. This route has a total cost of 161.26.

Figure 3.1 shows the optimal route of the numerical example and illustrates the time window discretization, the rounding process, and the arcs traversed through the network. To explain the flow in our network, recall that fragments presented as arcs start from a timed version of the first pickup location and end at a timed version of the last delivery location. For example, arcs (3, 2), (2, 7), (7, 8) inside fragment (3, 2, 7, 8) are not part of the relaxed network.

3.3. Problem Formulation

Let c_a denote the cost of traversing timed arc $a \in A$. In addition, a large number is added to the cost of the start timed arcs to minimize the total number of routes (vehicles) in the final solution. Let σ_ω denote the sum of all internal distances starting from the first node and ending at the last node of each timed fragment (internal cost of a timed fragment). Also, let $\Omega(p)$ denote the set of timed fragments that contain node $p \in P$ such that $\Omega(p) = \{\omega \in T(f) : p \in f, f \in \Phi\}$. We consider the problem of flowing vehicles through G^{RT} so that each request is covered. The binary variables x_ω are equal to 1 if and only if the timed fragment $\omega \in \Omega$ is used in the solution. Finally, the binary variables y_a are equal to 1 if and only if the timed arc $a \in A$ is used in the solution.

Figure 3.1. Optimal Route of the Numerical Example



3.3.1. The Relaxed Formulation of the PDPTWL (PDPTWL-R)

The problem can be formulated as an IP problem as follows:

$$\text{(PDPTWL-R)} \quad \text{minimize} \sum_{a\in A} c_a y_a + \sum_{\omega\in\Omega} \sigma_\omega x_\omega \tag{3.1}$$

subject to

$$\sum_{\omega\in\Omega(p)} x_\omega = 1, \quad \forall p \in P \tag{3.2}$$

$$\sum_{a\in A, a^+=i} y_a + \sum_{\omega\in\Omega,\omega^+=i} x_\omega = \sum_{a\in A,a^-=i} y_a + \sum_{\omega\in\Omega,\omega^-=i} x_\omega, \quad \forall i \in N \tag{3.3}$$

$$y_a, x_\omega \in \{0,1\}, \quad \forall a \in A, \forall \omega \in \Omega \tag{3.4}$$

The objective function (3.1) minimizes the total routing cost. Constraints (3.2) ensure that each request is served exactly once. Finally, constraints (3.3) guarantee the flow balance at every timed node, that is, for each timed node in the network the number of timed arcs (including timed fragments) entering a timed node equals the number leaving.

By construction, every feasible solution to PDPTWL can be represented as a feasible solution to PDPTWL-R. However, due to rounding, not every feasible solution to PDPTWL-R is a feasible solution to PDPTWL. Therefore we apply a branch-and-cut approach using lazy constraints.

3.3.2. Lazy Constraints (Cuts)

Significant efficiencies to solve integer and mixed-integer programs can be achieved by reducing the size of the original pool of active constraints, that is, using constraints only when you need them. Modern commercial solvers have made this possible by providing a "callback" facility. We apply lazy constraints with the following steps.

1. In the original formulation, relax the constraints (lazy constraints) that are less likely to be violated. Or, as in the case of our model above, simply do not include the additional lazy constraints.
2. During the optimization process, use the callback function to check any candidate IP solution. The commercial solver automatically calls the callback function every time a new candidate solution is found.
3. When checking the current candidate integer solution, if any portion of that solution violates the relaxed constraints (lazy constraints), the solution is then rejected and new constraints will be added to the original pool of constraints to cut off that portion.
4. Proceed with the branch-and-bound process.

3.3.3. Separation Problem

Due to the rounding down of time in our model, chains of fragments may result in time window-infeasible routes. Cycles (subtours) may also arise. To cut off any illegal solution that may be obtained from solving PDPTWL-R, we first need to identify the violated constraint(s) by solving a separation problem. If any violated constraint is found in any candidate integer solution, it is added as a lazy constraint during a callback.

Suppose a path R is part of a current IP candidate solution and contains at least two fragments (paths with just one fragment must be legal). The path R consists of a timed start arc, a timed end arc, one or more timed empty arcs, and two or more timed fragments, and represents the work done by one vehicle. Ignoring the time window discretization, we check the original versions of the fragments and empty arcs sequence, which the path R consists of, against the time window condition (and we also check for cycles). In other words, we check if the untimed path represented by R is a feasible route for PDPTWL. If all paths in the current IP candidate solution are legal, we accept the candidate solution. If not, one or more lazy constraints are added to cut off all "timed" copies of the infeasible paths (or cycles), which also cuts off the current candidate solution of the IP.

Suppose the path R is infeasible with respect to the time windows condition. Let $f(R)$ represent the set of original fragments $f \in \Phi$ and $\varepsilon(R)$ represent the set of original arcs $\varepsilon \in \bar{A}$ based on the timed fragments and timed arcs in the path R. Let $\cup_{\varepsilon \in \varepsilon(R)} T'(\varepsilon)$ be the timed arc set and $\cup_{f \in f(R)} T(f)$ be the timed fragment set in the path R. The lazy constraint to cut off a path R that is time window infeasible is as follows:

$$\sum_{a \in \cup_{\varepsilon \in \varepsilon(R)} T'(\varepsilon)} y_a + \sum_{\omega \in \cup_{f \in f(R)} T(f)} x_\omega \leq 2|f(R)| \tag{3.5}$$

We strengthen this constraint by omitting the start and end arcs and check the path starting from the first two fragments and onward trying to find a shortest illegal chain. Let a set $f(r) \subseteq f(R)$ be the shortest sequence of fragments that violates the time window condition in the path R. Also, let $\beta(r)$ represent the set of empty arcs β that connect the fragments in $f(r)$. Then the lazy constraint to cut off the time window infeasibility becomes

$$\sum_{a \in \cup_{\beta \in \beta(r)} T'(\beta)} y_a + \sum_{\omega \in \cup_{f \in f(r)} T(f)} x_\omega \leq 2|f(r)| - 2 \tag{3.6}$$

Similarly, suppose all paths in the solution are feasible but the solution contains one or more subtours. Then the subtour elimination lazy constraint for a cycle r is

$$\sum_{a \in \cup_{\beta \in \beta(r)} T'(\beta)} y_a + \sum_{\omega \in \cup_{f \in f(r)} T(f)} x_\omega \leq 2|f(r)| - 1 \tag{3.7}$$

Continuing with our example in Section 3.2.4, during the solution of PDPTWL-R, a candidate solution is generated that consists of one path R as follows:
$\{\cup_{\varepsilon \in \varepsilon(R)} T'(\varepsilon)\} = \{(0, 1_{(55)}), (10_{(235)}, 0), (6_{(115)}, 4_{(125)}), (9_{(185)}, 5_{(205)})\}$ and
$\{\cup_{f \in f(R)} T(f)\} = \{(1_{(85)}, 6_{(115)}, (1,3,8,6)), (4_{(125)}, 9_{(165)}, (4,2,7,9)), (5_{(205)}, 10_{(215)}, (5,10))\}$. The original versions of the timed arcs and the timed fragments sequence of the path R are [0, 1, 3, 8, 6, 4, 2, 7, 9, 5, 10, 0], which consists of three fragments ($|f(R)| = 3$), and R happened to be infeasible with respect to time windows. Clearly, constraint (3.5) guarantees that any timed version of this chain of sequence ((0, 1), (1, 3, 8, 6), (6, 4), (4, 2, 7, 9), (9, 5), (5, 10), (10, 0)) will not occur again.

Constraint (3.6) replaces constraint (3.5), ignoring starting arc (0, 1) and ending arc (10, 0). We need to check the path R, starting from the first two fragments onward, to find shortest violated constraint. The first sequence $r = [1,3,8,6,4,2,7,9]$ happened to be infeasible with respect to time windows, where $|f(r)| = 2$. Clearly, constraint (3.6) guarantees that any timed version of this sequence ((1, 3, 8, 6), (6, 4), (4, 2, 7, 9)) will not occur again.

3.3.4. Bound on the Number of Vehicles

To speed up the run time, we can obtain a lower bound on the number of vehicles by solving PDPTWL-R without the integrality condition on the variables and with the objective modified to count the number of starting arcs used. Let Z_v denote the optimal solution obtained. Let v denote the target number of vehicles. We initially set $v = \lceil Z_v \rceil$ and add the constraint $\sum_{a \in A, a^- = 0} y_a = v$. We solve this augmented version of PDPTWL-R, adding lazy constraints as required. If the resultant IP has no feasible solution, we increase the number of vehicles by setting $v = v + 1$ and solve it again.

3.4. Branch-and-Cut Algorithm

Our method can be described in the following algorithm steps.

Algorithm 1 (Pseudo Code for the Proposed Method)

```
Define the problem in a directed graph (network $G$).
Tighten the time windows of each node as much as possible.
Reduce $G$ by eliminating inadmissible arcs.
Generate fragments.
Construct the relaxed discretized network of timed fragments, timed empty arcs, and timed waiting
arcs (the resulting network $G^{RT}$).
Formulate the integer programming model PDPTWL-R.
Solve the linear programming relaxation of PDPTWL-R with the objective of minimizing $Z_v$ (the
number of vehicles).
Set $v = \lceil Z_v \rceil$ and add the constraint $\sum_{a \in A, a^- = 0} y_a = v$
loop
    Solve the IP with the objective of minimizing the total routing cost using IPCallback function
    to check integer solutions.
    if solution is feasible then
        break loop
    else
        Set $v = v + 1$
IPCallback function:
Examine the solution for each vehicle route.
if any route is illegal in $G$ then
    Add lazy constraint to cut off shortest illegal chain of fragments.
if any cycles in the solution then
    Add lazy constraints to cut off cycles
```

3.5. Computational Results

We used a computer equipped with Intel Core i7-6700 processor (3.4 GHz) running the Window 10 operating system. Python 3.5 was used to implement the proposed method with Gurobi 7.0.2 as the IP solver.

To the best of our knowledge, the only method in the literature for exactly solving the PDPTWL is the one introduced by Cherkesly, Desaulniers, and Laporte (2015). Their results were generated on a computer with Intel Core i7-3770 processor (3.4 GHz) running Linux. They considered a modified version of four groups of instances derived from Ropke and Cordeau (2009), all of which have 10 instances. These groups are called AA, BB, CC, and DD. They modified these instances by changing the vehicle capacities and delaying the time window of the delivery nodes. For AA and BB instances, the time windows were changed to $[e_i = e_i + 45, l_i = l_i + 45], \forall i \in D$. The vehicle capacity was changed from 15 to 22 and from 20 to 30 for AA and BB, respectively. For CC and DD the time windows were changed to $[e_i = e_i + 15, l_i = l_i + 15], \forall i \in D$. The vehicle capacity was changed from 15 to 18 and from 20 to 25 for CC and DD, respectively. They also added two groups called AA* and

BB* by modifying groups AA and BB; the time windows of the original instances were changed to $[e_i = e_i + 60, l_i = l_i + 60], \forall i \in D$. The vehicle capacity was changed from 15 to 26 and from 20 to 35 for AA* and BB*, respectively. We follow their terminology by referring to these instances that are modified for the PDPTWL as AA, BB, CC, DD, AA*, and BB*. The cost of each vehicle was set to 10,000. A time limit of 7200 seconds was imposed by Cherkesly, Desaulniers, and Laporte (2015) on instances AA* and BB*, and 3600 seconds on others, so we did the same.

During our testing, we discovered some minor differences due to rounding, which could have led to structurally different solutions. An email contact was made on February 1, 2017 with Cherkesly (2017), who confirmed that "distances are rounded to 4 decimals and travel times to 2 decimals." This was confirmed as travel times rounded down to two decimal digits. We obtained the same results as Cherkesly, Desaulniers, and Laporte (2015) when using the same rounding. However, for the sake of consistency in our solution and to avoid complicating subsequent research efforts, we rounded down both the distances and the travel times to the second decimal digit. All solutions are still structurally identical, but slight differences in the final objective values reported can be seen due to that rounding.

The goals of our testing in this section are to compare our method to the state-of-the-art method of Cherkesly, Desaulniers, and Laporte (2015), and to assess the impact of changing the parameter δ. In addition, we evaluate the impact of multithreading which is the default setting for the Gurobi solver (see Gurobi Optimization 2017).

In her Ph.D. thesis, Cherkesly (2015) made improvements based on some speedups in the column generation. Because of these improvements, more instances were solved to optimality within the time limit and better results in terms of solution times were obtained. In tables 3.2 and 3.3 we compare our method to these improved algorithms. Moreover, Cherkesly (2015) reports results of three different algorithms for each instance. We have used the minimum time among these algorithms for each instance.

Initially, we set the time intervals to $\delta = 5$ for all instances. No feasible solution was obtained within the time limit for six instances CC65, CC70, CC75, DD35, DD60, and DD65. When we set $\delta = 1$, two instances, DD35 and DD65, were solved to optimality and a good feasible solution upper bound obtained for CC75 and DD60. Table 3.4 shows the results for these four instances.

The columns of tables 3.2, 3.3, and 3.4 represent the following:

- *Name*: the instances name;
- Z_R: the lower bound at the root node (i.e., the linear programming relaxation of PDPTWL-R augmented with the integer lower bound on the number of vehicles);
- *Sec.*: the total computing time in seconds;
- Z^*: the optimal solution obtained, otherwise we report, in bold, the best upper bound (feasible solution) obtained or left blank if no feasible solution found within the time limit;
- Z_{LB}: the best lower bound (LB relaxation) at a child node if the instance was not solved to optimality within the time limit, left blank otherwise;
- F_N: the total number of original fragments;
- TF_N: the total number of undominated timed fragments;

- *Nodes*: the number of nodes that have been explored in the branch-and-bound tree;
- L_C: the total number of lazy constraints used in any solution.

Table 3.2. Computational Results ($\delta = 5$)

	Cherkesly et al. (2015)			Our Method							
Name	Z_R	*Sec.*	Z^*	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	Z^*
AA30	31,129.40	1.0	31,129.4	176	421	31,129.13		0	0	0.3	31,129.13
AA35	31,285.20	8.2	31,294.1	294	646	31,284.77		0	1	0.3	31,293.74
AA40	41,349.20	2.3	41,349.2	368	780	41,348.80		0	0	0.3	41,348.80
AA45	41,521.40	4.2	41,521.4	623	1,121	41,520.87		0	0	0.5	41,520.87
AA50	41,643.60	8.3	41,643.6	768	1,385	41,643.10		0	0	0.7	41,643.10
AA55	46,803.80	13.6	51,743.2	935	1,676	51,742.50		0	0	1.0	51,742.52
AA60	46,999.20	858.2	51,949.7	1,015	1,837	51,926.59		82	1	2.0	51,949.04
AA65	47,172.00	169.3	52,077.4	1,118	2,021	52,056.02		0	2	1.6	52,076.67
AA70	47,896.10	151.9	52,219.2	1,469	2,556	52,207.68		0	0	2.4	52,218.44
AA75	51,607.20	229.4	52,330.1	2,498	4,005	52,325.16		0	0	3.9	52,329.30
BB30	31,076.30	2.9	31,077.5	145	296	31,073.79		0	2	0.3	31,077.12
BB35	31,312.40	3.0	31,312.4	267	502	31,307.47		0	1	0.3	31,311.97
BB40	35,695.50	37.3	41,404.0	342	660	41,393.09		0	2	0.7	41,403.49
BB45	37,645.10	123.0	41,537.5	540	1,000	41,515.60		230	10	1.9	41,536.99
BB50	41,791.10	18.3	41,791.1	706	1,308	41,763.62		0	1	1.4	41,790.49
BB55	46,391.40	518.2	51,911.7	799	1,484	51,879.48		542	15	4.0	51,911.05
BB60	62,305.50	15.0	62,305.5	742	1,383	62,246.49		65	9	2.1	62,304.78
BB65	62,564.60	22.4	62,564.6	801	1,506	62,466.90		119	18	2.9	62,563.85
BB70	66,002.40	429.3	72,535.2	997	1,860	72,524.90		0	0	3.3	72,534.42
BB75	68,197.00	806.7	72,656.7	1,446	2,495	72,633.56		210	3	5.1	72,655.87
CC30	23,318.90	27.5	31,088.6	241	2,056	31,083.75		10	3	2.4	31,088.20
CC35	24,777.20	32.7	31,237.4	438	3,129	31,232.69		0	0	2.8	31,237.04
CC40	26,024.60	30.4	31,340.2	1,021	6,135	31,333.75		3	0	3.8	31,339.79
CC45	29,562.70			2,359	11,716	31,469.79		74,568	160	721.8	31,532.09
CC50	35,156.60	345.6	41,673.6	3,813	16,976	41,652.85		2,194	26	72.2	41,673.05
CC55	36,778.10	1,416.1	41,793.5	5,294	21,960	41,767.00		6,733	22	129.3	41,792.84
CC60	38,262.60	3,500.1	41,947.3	6,589	26,990	41,919.76		48,728	69	923.2	41,946.63
CC65				9,243	36,057	42,036.09	42,050.39	65,564	957		
CC70				13,888	50,406	42,181.24	42,197.47	24,954	84		
CC75				19,480	69,038	42,344.53	42,359.15	11,643	16		
DD30	19,153.30	790.2	21,103.2	1,437	5,709	21,071.62		1,737	24	26.3	21,102.82
DD35	21,854.00	154.4	31,127.8	3,052	11,167	21,197.42	21,292.56	246,316	140		
DD40	23,024.60	176.1	31,245.3	5,196	18,095	31,241.46		0	0	8.7	31,244.79
DD45	24,560.70	434.2	31,350.4	8,940	28,715	31,343.62		0	0	11.7	31,349.87
DD50	25,547.80	827.3	31,450.2	16,064	49,289	31,437.65		107	5	25.2	31,449.70
DD55				25,271	74,702	31,582.86		32,938	99	1,294.7	31,637.14
DD60				39,687	111,392	31,749.77	31,757.63	8,207	246		
DD65				25,579	73,362	32,054.14	32,081.48	6,693	0		
DD70				40,855	109,654	42,063.75		3,618	28	650.0	42,092.28
DD75				62,468	157,594	42,182.63	42,191.00	5,354	110	3,600.0	**42,319.61**

Table 3.5 shows comparison summary results to Cherkesly, Desaulniers, and Laporte (2015), considering the best results obtained for each instance (best of their three algorithms and best of $\delta = 5$ and $\delta = 1$ for our method) and only for instances that solved by both methods including the improvements in Cherkesly (2015). The columns of Table 3.5 represent the following: instances by group (*Group*); number of instances in each group that are solved to optimality (*Solved*); and average CPU time in seconds (*Seconds*) of each group.

Clearly, if the length of δ is decreased, then the number of variables will be larger and the solution time of the linear programming relaxation will increase in most cases. However, the number of violated constraints (lazy constraints) in respect of time windows will be fewer and the lower bound at the root node will be tighter. To evaluate the impact of the parameter δ, we provide computational results for all instances setting $\delta = 1$ and time limit to 3,600 seconds (see Appendix A).

Table 3.3. Computational Results ($\delta = 5$)

	Cherkesly et al. (2015)			Our method							
Name	Z_R	*Sec.*	*Z**	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	*Z**
AA*30	31,051.4	3.4	31,051.6	271	271	31,051.08		0	0	0.3	31,051.20
AA*35	31,231.7	242.8	31,244.8	415	416	31,228.14		0	0	0.5	31,244.37
AA*40	36,364.4	16.5	41,331.4	569	570	41,318.01		0	0	0.5	41,330.99
AA*45	36,590.3	355.9	41,515.4	882	884	41,498.70		0	2	1.0	41,514.90
AA*50	38,375.0	1,284.0	41,637.5	1,073	1,075	41,603.52		56	5	1.7	41,636.97
AA*55	41,843.5	356.5	41,880.2	1,283	1,285	41,835.40		127	5	2.2	41,879.62
AA*60	45,280.4	52.4	51,808.2	1,381	1,383	51,803.30		0	2	1.7	51,807.57
AA*65	47,048.8	242.7	51,961.9	1,522	1,524	51,940.37		0	7	2.1	51,961.24
AA*70				1,939	1,942	52,122.85		1,064	20	7.2	52,170.77
AA*75				3,116	3,119	52,249.71		1,672	35	13.0	52,298.84
BB*30	36,144.2	3.6	41,111.0	169	169	41,102.54		0	1	0.3	41,110.65
BB*35	37,122.8	26.2	41,332.9	307	307	41,326.29		0	1	0.5	41,332.47
BB*40	39,337.7	108.7	41,477.1	381	381	41,462.42		17	4	0.8	41,476.65
BB*45	41,646.1	50.9	41,699.5	596	596	41,601.71		933	33	3.1	41,698.98
BB*50	46,505.4	1,245.0	51,719.1	773	774	51,678.64		70	3	2.2	51,718.58
BB*55	49,891.9			926	927	51,916.55		4,755	93	16.5	52,033.90
BB*60	65,265.4	74.0	72,184.3	1,025	1,026	62,362.56		0	6	2.8	72,183.56
BB*65	66,055.7	1,719.5	72,394.5	1,111	1,112	72,331.24		581	21	4.5	72,393.77
BB*70				1,383	1,385	72,523.97		1,387	11	9.9	72,604.07
BB*75				1,930	1,932	72,614.71		1,929	19	16.2	72,746.85

Table 3.4. Computational Results ($\delta = 1$)

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	*Z**
CC75	19,480	302,256	52,283.07	52,294.88	1,951	5	3,600.0	**52,316.09**
DD35	3,052	49,033	21,340.89		31	0	123.3	31,127.43
DD60	39,687	466,993	31,768.22	31,778.29	1,031	4	3,600.0	**31,814.13**
DD65	25,579	306,298	41,967.35		446	0	1,381.1	41,984.09

Table 3.5. Comparison Summary Results

	Cherkesly et al. (2015)		Our method	
Group	*Solved*	*Seconds*	*Solved*	*Seconds*
AA	10	144.6	10	1.3
BB	10	197.6	10	2.2
CC	6	892.1	7	189.0
DD	5	476.4	8	39.0
AA*	8	319.3	10	1.3
BB*	7	461.1	10	2.0
Total	46		55	

One major advantage of using our algorithm is that it can use all available threads in the computer. All computations were performed using all eight available threads (the default solver setting) unless otherwise stated. Although we would usually expect slower solutions using a single thread, a faster solution may be obtained for some instances (see Gurobi Optimization 2017). We note that a feasible solution for CC65 was found with a single thread where none had been found with eight threads. To evaluate the impact of the thread count parameter, we provide in Table 3.6 summary results for all instances using a single thread and a time limit of 3,600 seconds. The columns in Table 3.6 represent the following: instances by group (*Group*); number of instances in each group that are solved to optimality (*Solved*); and average elapsed time in seconds (*Seconds*) over all solved instances of each group.

Table 3.6. Impact of the Number of Threads

	One thread				Eight threads			
	$\delta = 1$		$\delta = 5$		$\delta = 1$		$\delta = 5$	
Group	*Solved*	*Seconds*	*Solved*	*Seconds*	*Solved*	*Seconds*	*Solved*	*Seconds*
AA	10	6.4	10	1.4	10	4.2	10	1.3
BB	10	15.3	10	2.5	10	7.8	10	2.2
CC	6	812.3	7	398.1	6	632.4	7	265.1
DD	7	1150	6	595.1	7	652.4	6	336.1
AA*	10	5.3	10	5.2	10	4.2	10	1.3
BB*	10	10.6	10	24.6	10	5.3	10	2.0
Total	53		53		53		53	

Instances AA, BB, AA* and BB* solved very quickly using the proposed method. Therefore, we decided to enlarge these instances by adding more requests. To do that we created instances (AA80, ..., AA125) and (BB80, ..., BB125) in a similar fashion to Ropke and Cordeau (2009). We provide a description of the characteristics of Ropke and Cordeau (2009) instances in Appendix A. We applied the same modification used by Cherkesly, Desaulniers, and Laporte (2015) for the pickup and delivery vehicle routing problem with time windows and last-in-first-out (LIFO) loading (PDPTWL) on these instances. We imposed a one hour time limit to solve these instances. Table 3.7 shows the results obtained by our method for these instances setting $\delta = 5$. In addition, we provide computational results for these instances setting $\delta = 1$ (see Appendix A).

It is worth mentioning that we never needed to add a subtour elimination lazy constraint. This is due to the characteristics of the instances, and to the time window discretization procedure. Thus, the lazy constraints reported in all tables are of time window infeasible paths type. Also, the number of nodes reported in all tables are the total number of nodes over all values of the number of vehicles v tried.

Table 3.8 analyzes the results of CC group instances. The columns represent the following: Sec_f and Sec_{sol} represent fragment generation time, and the time used by the solver to solve the MIP, respectively; $Sec.$ is the total time; $\%Sec_f$ and $\%Sec_{sol.}$ represent the percentage of fragment generation time, and the percentage of MIP time (solver time), in relation to the total time, respectively. If the total time is blank then the problem was not solved to optimality in one hour. We see that, the fragment generation time is relatively small compared to the MIP time, dropping to less than one percent on the most difficult instances.

Table 3.7. Computational Results ($\delta = 5$)

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	*Z**
AA80	1,826	3,172	62,486.36		2,378	41	15.5	62,541.33
AA85	2,508	4,266	72,691.72		0	3	6.9	72,703.23
AA90	2,492	4,623	63,055.70		65,482	145	556.6	72,924.18
AA95	3,117	5,542	63,112.05		72,102	218	520.3	63,260.65
AA100	2,747	4,876	73,370.37		2,536	28	25.9	73,424.04
AA105	4,194	6,959	73,633.23		1,520	25	63.0	83,522.13
AA110	3,561	6,047	83,244.65		624	11	16.2	83,268.80
AA115	4,729	7,893	93,623.99		1,778	57	78.0	93,690.44
AA120	3,790	6,772	103,937.94		1,645	12	18.7	103,963.13
AA125	4,829	8,280	83,962.12		2,898	91	167.1	93,840.61
BB80	2,356	4,041	62,668.86		2,000	78	42.4	62,743.40
BB85	2,372	4,064	62,636.40		3,586	17	40.7	62,688.02
BB90	2,773	4,708	82,963.94		725	8	10.4	82,985.83
BB95	3,016	5,457	63,092.31		18,298	75	588.2	73,005.02
BB100	3,334	5,958	73,013.92		1,974	41	52.4	73,059.83
BB105	8,852	14,299	72,966.80		11,460	92	290.3	73,032.72
BB110	5,087	8,298	73,421.31		25,370	85	594.6	73,543.43
BB115	4,811	8,183	83,725.62		32,237	210	751.4	83,812.37
BB120	9,500	15,095	73,818.52		7,199	49	330.4	83,700.24
BB125	6,480	10,675	93,808.07		3,840	61	166.4	93,868.50
AA*80	2,531	2,533	62,509.41		1,949	66	25.9	62,588.19
AA*85	3,565	3,572	72,574.99		3,923	65	41.6	72,633.39
AA*90	3,413	3,418	72,839.78		12,871	109	148.5	72,923.07
AA*95	4,447	4,455	63,268.26		3,820	18	73.1	72,996.92
AA*100	4,076	4,079	83,225.92		492	23	14.4	83,254.93
AA*105	5,638	5,646	83,239.95		4,969	67	107.2	83,349.25
AA*110	5,315	5,323	83,166.33		8,879	121	173.8	83,232.45
AA*115	7,171	7,182	93,483.71		6,408	75	209.1	93,577.39
AA*120	5,668	5,674	93,774.18		21,047	230	447.3	93,899.67
AA*125	7,273	7,279	83,683.97		158,263	411	3,573.4	83,844.43
BB*80	2,803	2,808	62,708.64		38,737	27	433.6	72,556.05
BB*85	2,769	2,776	62,596.86		6,159	40	90.9	62,718.69
BB*90	3,439	3,447	72,931.00		15,988	89	211.9	73,075.64
BB*95	3,856	3,863	72,955.75		1,829	21	26.4	73,001.11
BB*100	4,385	4,393	73,084.31		5,888	80	99.3	73,210.37
BB*105	10,206	10,219	72,988.78		159,992	182	3,313.1	73,132.54
BB*110	6,110	6,118	83,401.03		73,891	186	1,054.7	83,540.28
BB*115	6,699	6,712	83,734.50		10,648	89	247.2	93,619.79
BB*120	11,417	11,431	83,528.11	83,603.08	56,636	413	3,600.0	**83,812.66**
BB*125	8,334	8,348	93,736.86		138,508	297	3,459.7	93,883.89

Table 3.8. Analyzing the Results for CC Group Instances

Name	T_N	TF_N	*Nodes*	*LC*	Sec_f	$Sec_{sol.}$	*Sec.*	$\%Sec_f$	$\%Sec_{sol.}$
CC30	241	2,056	10	3	0.1	2.3	2.4	4.2	95.8
CC35	438	3,129	0	0	0.2	2.6	2.8	7.1	92.9
CC40	1,021	6,135	3	0	0.5	3.3	3.8	13.2	86.8
CC45	2,359	11,716	74,568	160	1.5	720.3	721.8	0.2	99.8
CC50	3,813	16,976	2,194	26	2.9	69.3	72.2	4.0	96.0
CC55	5,294	21,960	6,733	22	4.2	125.1	129.3	3.2	96.8
CC60	6,589	26,990	48,728	69	5.3	917.9	923.2	0.6	99.4
CC65	9,243	36,057	65,564	957	7.8				
CC70	13,888	50,406	24,954	84	12.0				
CC75	19,480	69,038	11,643	16	20.9				

3.6. Conclusions

In this chapter, we have proposed a novel exact algorithm and introduced a new formulation for the PDPTW. We generated fragments that can be chained together to form routes. We constructed an integer program formulation to solve a relaxed space-time network flow model to chain the fragments. In addition, lazy constraints were used to cut off any illegal solution that resulted from the relaxation. The computational results show that the proposed method is efficient compared with the current state-of-the-art method for the PDPTWL, and we solve nine instances that were previously unsolved.

Moreover, our algorithm is an alternative to branch-and-price-and-cut methods and a new direction for future research for the PDPTW and other VRP variants. There is likely to be great scope for improvement by adding other techniques such as problem-specific cuts.

The following publication has been incorporated as Chapter 4.

Alyasiry A, and Forbes M (2019) An exact algorithm for the pickup and delivery problem with time windows and multiple stacks, submitted to *Computers and Operations Research* on September 26, 2019.

Contribution to the authorship: Ali Alyasiry and Michael Forbes conceived the main idea. Ali Alyasiry with support from Michael Forbes developed the theoretical framework. Ali Alyasiry drafted the manuscript, processed the experimental data, analyzed and interpreted the data, performed the calculations, and designed the figures. Ali Alyasiry and Michael Forbes carried out the proofreading of the manuscript.

Chapter 4

An Exact Algorithm for the Pickup and Delivery Problem with Time Windows and Multiple Stacks

ABSTRACT

This chapter addresses an extension of the pickup and delivery problem with time windows and multiple stacks (PDPTWMS). We call the new extension the PDPTWMS with stack splitting (PDPTWMS-S). Applications of the PDPTWMS-S arise in less-than-truckload transportation systems where loads are packed as pallets/boxes or other standardized containers. In the PDPTWMS-S, the vehicle's loading space is divided into compartments of finite capacity. Loading and unloading an item in each compartment must obey the last-in-first-out policy. Moreover, a customer's demand can be split among all compartments, and a customer's demand can also exceed a single compartment's capacity. Thus, a customer's demand is only limited to the vehicle's capacity. Computational experiments show that applying this extension can result in a big routing cost reduction (the total distance traveled and/or the number of vehicles used). We compose the pickup and delivery vertices into fragments of requests. A fragment is a sequence of correctly paired and scheduled pickup and delivery vertices starting and ending with an empty vehicle. We then use a branch-and-cut algorithm to chain these fragments into routes and solve the PDPTWMS and PDPTWMS-S to optimality. Results confirm that this approach outperforms the current state-of-the-art method for the PDPTWMS. We solve to optimality 660 instances, defined for the PDPTWMS, that were previously unsolved. For the PDPTWMS-S, we introduce an integer programming formulation to determine a feasible loading plan.

4.1. Introduction

In a one-to-one pickup and delivery problem with time windows (PDPTW) vehicles are routed to serve given requests each of which consists of a pickup and a corresponding delivery. For each request, the pickup must precede the delivery (precedence) and both must be performed by the same vehicle (pairing). Routes must respect precedence, pairing, vehicle-capacity, and time-window constraints. A set of feasible routes is a solution to the PDPTW if each request is performed by exactly one of the routes.

A further specialization of the PDPTW is the pickup and delivery problem with time windows and last-in-first-out (LIFO) loading (PDPTWL). In this problem loading and unloading an item must obey the LIFO loading policy, meaning loaded items are treated as if they are placed on a stack in the vehicle (see, Cherkesly, Desaulniers, and Laporte 2015). This will avoid extra cost associated with rearranging items inside the vehicle's loading compartment. LIFO loading is very common in the transportation of pallets, animals, heavyweight goods and hazardous materials. The problem is also seen in *Automated Guided Vehicle* systems that are often used in sea ports and warehouses.

The pickup and delivery problem with time windows and multiple stacks (PDPTWMS) is a generalization of the PDPTWL for the multiple stacks case. In the PDPTWMS, loading and unloading an item in each compartment must obey the LIFO loading policy. This is called the multi-stack policy. A vehicle route is feasible for the PDPTWMS if it feasible for the PDPTW and respects the multi-stack policy. The objective of the PDPTWMS is to satisfy all requests with a least-cost routing. Each route incurs a travel cost and a fixed cost (vehicle cost). The fixed cost is relatively high. Therefore, the priority is to minimize the number of routes (the number of vehicles used) in the solution. We assume an unlimited supply of identical vehicles. Each vehicle consists of $k = |S|$ independent compartments (independently accessible) of finite capacity Q_s, where $S = \{1, \dots, k\}$ is the set of the compartments in any vehicle. The total vehicle capacity is $Q = k \times Q_s$.

The only work in the literature studying the PDPTWMS is that by Cherkesly et al. (2016). They present two branch-price-and-cut algorithms. The first algorithm solves the pricing problem by fully enforcing the multi-stack policy. The second algorithm relaxes the definition of a feasible route in the pricing problem and infeasible path inequalities are added to the relaxed master problem if infeasible routes are used in a linear relaxation solution. Instances with up to 75 requests are solved to optimality within two hours of computation time.

In the literature several papers considered the single-vehicle pickup and delivery problem with LIFO loading or first-in-first-out (FIFO) loading (see, e.g., Carrabs, Cerulli, and Cordeau 2007; Carrabs, Cordeau, and Laporte 2007; Erdoğan, Cordeau, and Laporte 2009; Cordeau, Dell'Amico, and Iori 2010; Cordeau et al. 2010; Li et al. 2011), and the pickup and delivery problem with time windows and LIFO loading (PDPTWL) (see, e.g., Cherkesly, Desaulniers, and Laporte 2015; Alyasiry, Forbes, and Bulmer 2019), for the single-stack case.

For the multiple stacks case there are papers on the single-vehicle pickup and delivery problem with multiple stacks (see, e.g., Côté et al. 2012; Côté, Gendreau, and Potvin 2012; Sampaio and Urrutia 2017; Pereira and Urrutia 2018), the double traveling salesman problem with multiple stacks (DTSPMS) (see, e.g., Petersen and Madsen 2009; Felipe, Ortuño, and Tirado 2009; Petersen, Archetti, and Speranza 2010; Lusby et al. 2010; Carrabs, Cerulli, and Speranza 2013; Alba Martínez et al. 2013), and the double vehicle routing problem with multiple stacks (DVRPMS) (Iori and Riera-Ledesma

2015).

To the best of our knowledge, all previous methods in the literature that have dealt with multiple stacks routing have assumed that the demand of each customer cannot be split among different stacks and cannot exceed the stack capacity.

In this chapter we introduce an extension to the PDPTWMS. We allow the demand of a customer to exceed the stack capacity and be only limited to the vehicle capacity, and we allow the demand of a customer to be split among all stacks. In the vehicle routing problem literature splitting often refers to splitting a request to transport it on two or more vehicles (for more details, see Archetti and Speranza 2012). In contrast, the splitting we consider is among the compartments of one vehicle. Therefore, we denote the new variant the PDPTWMS with stack splitting (PDPTWMS-S). Computational experiments show a reduction in the total traveled distance and/or a reduction in the number of vehicles used in many instances when applying the proposed variant.

In this chapter, we extend the work of Alyasiry, Forbes, and Bulmer (2019) for the PDPTWL and adapt that method to the PDPTWMS and the PDPTWMS-S. Alyasiry, Forbes, and Bulmer (2019) introduced a branch-and-cut approach based on fragments of requests. A fragment is a sequence of correctly paired and scheduled pickup and delivery vertices starting and ending with an empty vehicle. Using fragments, they formulate a relaxed network flow model with side constraints.

For the PDPTWMS and the PDPTWMS-S we generate all fragments that are feasible for the PDPTW while ensuring that we can assign pickups to compartments so as to respect the multi-stack policy. Such an assignment of pickups is called a loading plan. We then chain the fragments to form routes using a branch-and-cut approach similar to that of Alyasiry, Forbes, and Bulmer (2019). Computational results show that our method outperforms the current state-of-the-art method for the PDPTWMS and solves to optimality 660 instances, defined for the PDPTWMS, that were previously unsolved.

For the PDPTWMS-S we introduce an integer programming (IP) formulation to assign requests to stacks to produce a feasible multi-stack loading plan. In addition, we use a number of techniques to minimize the number of stack assignment IPs solved.

The remainder of this chapter is organized as follows. The next section provides notation. Section 4.3 formally defines the PDPTWMS and PDPTWMS-S and describes the difference between them. In section 4.4 we define the fragments and we present procedures for generating loading plans (assigning pickups to compartments) including a proposed IP formulation to generate loading plans for the PDPTWMS-S. In Section 4.5 we present the mathematical formulation for the PDPTWMS and the PDPTWMS-S. In Section 4.6 we report computational results for the PDPTWMS and the PDPTWMS-S. Conclusions presented in Section 4.7.

4.2. Notation

The PDPTWMS and the PDPTWMS-S can be modeled using a directed graph (network) $G = (V, \bar{A})$. V represents the set of nodes or vertices and $\bar{A}$ represents the set of arcs. The set of vertices V can be characterized as $V = \{0\} \cup P \cup D$, where vertex 0 represents the depot; $P = \{1, \cdots, n\}$ is a set of pickup vertices; and $D = \{n+1, \cdots, 2n\}$ is a set of delivery vertices. The set of arcs $\bar{A}$ can be characterized as $\bar{A} = \{(i, j) : i, j \in V, i \neq j\}$. Infeasible arcs are eliminated and time windows are tightened using the rules presented in Dumas, Desrosiers, and Soumis (1991). The notations used in section 2.1 for the graph G are used in this chapter. Due to the pairing conditions between the pickup and the delivery vertices, the set of pickup vertices P can also represent the set of requests.

4.3. The PDPTWMS-S VS the PDPTWMS

Using multiple stacks normally increases the loading options compared to a single stack and may lead to a significant saving. Because of this loading flexibility, many infeasible loading pattern with a single stack may become feasible when using extra stacks. For example, suppose the problem consists of two requests 1 and 2 with pickup vertices p_1, p_2 and delivery vertices d_1 and d_2. The loading and unloading pattern of $\langle p_1, p_2, d_1, d_2 \rangle$ is feasible for a multi-stack case, but it is infeasible for a single-stack case. When the pickup and deliveries of two requests interact in this way, regardless of other pickups or deliveries, the requests are said be stack incompatible. However, using a vehicle, with the same capacity and more compartments, is not always the best choice. For example, suppose the vehicle capacity is 12 and the demand at each request is four. Ignoring the time windows, the path $\langle p_1, p_2, p_3, d_1, d_2, d_3 \rangle$ is feasible for three stacks case (the capacity of each compartment is four). Whereas, it is infeasible for four stacks case (the capacity of each compartment is three).

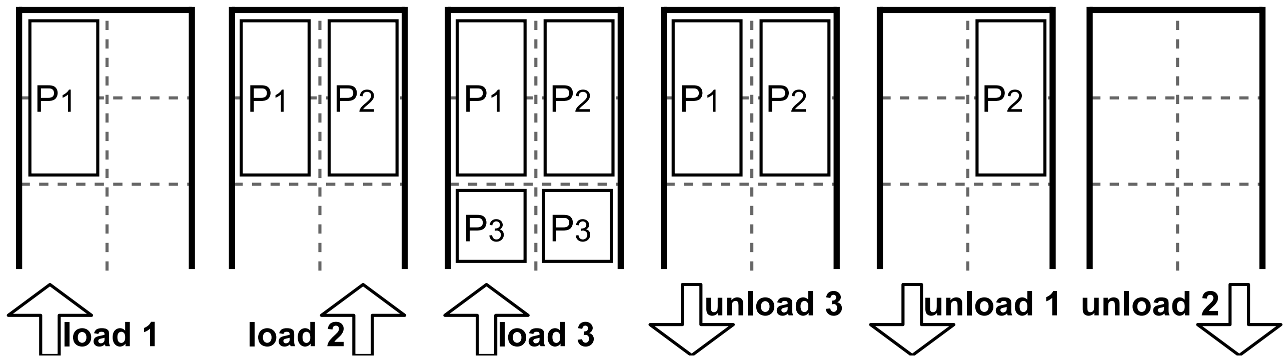
In the PDPTWMS for any route to be feasible it must be feasible for the PDPTW and it must satisfy the capacity and LIFO constraints for each stack (Cherkesly et al. 2016). Also, the demand at each customer cannot be split among stacks and cannot exceed the stack capacity. However, in real-world situations a customer's demand often consists of pallet loads which may be split among all stacks, and the size of any request may exceed the stack capacity and be only limited to the vehicle's capacity. To illustrate, we provide a simple example as follows.

Example 4.3.1. Suppose we have a vehicle contains two compartments each compartment has a capacity of three. Also, suppose the demand at each request is two. Clearly, when using the PDPTWMS assumption, paths that start with any combination of three pickups are infeasible, but many of them are feasible for the PDPTWMS-S.

Moreover, suppose a request's demand is more than three. In the PDPTWMS this request must be rejected because its demand exceeds the vehicle's compartment capacity.

Figure 4.1 shows the following path $\langle p_1, p_2, p_3, d_3, d_1, d_2 \rangle$ that is feasible for the PDPTWMS-S but infeasible for the PDPTWMS. The demand at each request is two and the vehicle contains two compartments each compartment has a capacity of three.

Figure 4.1. A Feasible Loading Plan for the PDPTWMS-S That Is Infeasible for the PDPTWMS



4.4. Fragments

As an alternative to route based approaches, we reduce the number of variables by considering fragments. We generate all possible fragments of requests (sub-paths). Generated fragments must satisfy pairing, precedence, time window, and vehicle capacity constraints. In addition, fragments must satisfy the multi-stack policy. That is, there must exist an assignment of requests to stacks that respects the capacity of each stack and the LIFO loading policy for each stack. Multi-stack fragments, especially with the proposed variant, are a non-trivial extension of the PDPTWL fragments introduced by Alyasiry, Forbes, and Bulmer (2019).

For a fragment to be feasible for the PDPTWMS or for the PDPTWMS-S it must first be feasible for the PDPTW and there must exist a loading plan for the fragment that assigns requests to stacks so as to respect both stack capacity and the LIFO policy. More formally, a feasible loading plan for a fragment specifies the quantity of each request assigned to each stack such that:

- Every request is completely assigned to the stacks.
- No pair of stack incompatible requests are assigned to the same stack.
- The total quantity in any stack at any time is less than the stack capacity.

For the PDPTWMS, each request is assigned to exactly one stack.

Recall that we denote a fragment by f and the set of all fragments by Φ. We denote by $X_f(p)$ the pickup position of request p in f, and $\Gamma_f(p)$ the delivery position of request p in f, where $p \in P$. For example, if $f = \langle p_c, p_b, d_c, p_a, d_a, d_b \rangle$, then $X_f(c) = 1$, $X_f(b) = 2$, $X_f(a) = 4$, $\Gamma_f(c) = 3$, $\Gamma_f(a) = 5$, and $\Gamma_f(b) = 6$. Note that a,b and $c \in P$.

Using $X_f(p)$ and $\Gamma_f(p)$ we can compute the set of stack incompatible pairs of requests for fragment f. That is the requests that cannot be placed on the same stack because of the LIFO loading policy.

Definition 4.4.1. We say that the requests $p' \in P$ and $\tilde{p} \in P$ are *stack incompatible* in fragment f if either of the following conditions are met:

- $X_f(p') < X_f(\tilde{p})$, $\Gamma_f(p') < \Gamma_f(\tilde{p})$ and $X_f(\tilde{p}) < \Gamma_f(p')$;
- $X_f(\tilde{p}) < X_f(p')$, $\Gamma_f(\tilde{p}) < \Gamma_f(p')$ and $X_f(p') < \Gamma_f(\tilde{p})$.

4.4.1. Fragments for the PDPTWMS

We generate all fragments using an efficient recursive algorithm. To determine if there is a feasible loading plan for the PDPTWMS we also use a recursive algorithm to implicitly enumerate all possible loading plans. The loading plan algorithm uses a number of acceleration techniques including:

- A fragment with a number of requests less than or equal the number of the stacks must be feasible.
- Stack incompatible requests cannot go on the same stack.
- If we have multiple empty stacks available for a request, we avoid equivalent solutions by considering only the first empty stack.

Implicit enumeration turns out to be very efficient in our computational experiments because the maximum number of requests in each fragment is relatively small.

The procedure for generating all feasible fragments for the PDPTWMS is shown in pseudo-code in Algorithm 2. The procedure *Extend* produces a sequence of vertices to generate a candidate fragment f, then the procedure *CanLoad* checks whether the generated fragment f can be feasibly assigned to the vehicle compartments.

Extend is a recursive function which takes a partial fragment, the current time and the current load. It first determines if the partial fragment can be completed (i.e. all requests delivered) and, if it can be completed, then tries all fragment extensions to pickups. The construction process dynamically enforces all PDPTW rules - pairing, precedence, time windows and vehicle capacity. Immediately before a candidate fragment is added to the set of feasible fragments we check for a feasible loading plan. Note that we never add pickups to a partial fragment unless the fragment can be feasibly completed, where feasibility includes checking for a loading plan.

CanLoad is a recursive function which attempts to load a partial fragment f into the stacks which have current contents defined by state S. The function repeatedly processes the next request in f by removing it from its stack if it is a delivery or adding it to a stack if it is a pickup. The procedure uses information on the delivery position of each request in the fragment to avoid placing stack incompatible requests on the same stack. If multiple stacks are empty, only the first empty stack encountered is considered.

Algorithm 2 Generate all PDPTWMS feasible fragments

```
1: function CanLoad($f, S, DPos$)
2:     if $f$ is empty then
3:         return true
4:     $r \leftarrow$ first vertex in $f$
5:     $f' \leftarrow f$ with $r$ removed
6:     if $r \in D$ then                                ▷ Remove delivery and check rest of fragment
7:         $s \leftarrow$ stack that has $r - n$ on top
8:         pop $s$
9:         if CanLoad($f', S, DPos$) then
10:            return true
11:        push $r$ onto $s$                                               ▷ Restore the state of $S$
12:    else
13:        DoneEmpty $\leftarrow$ false
14:        for $s \in S$ do
15:            if $s$ is empty then                                  ▷ Only try the first empty stack
16:                if not DoneEmpty then
17:                    push $r$ into $s$
18:                    if CanLoad($f', S, DPos$) then
19:                        return true
20:                    pop $s$                                                       ▷ Restore the state
21:                    DoneEmpty $\leftarrow$ true
22:            else                          ▷ Only try this stack if there is space and LIFO applies
23:                $r' \leftarrow$ current top request in $s$
24:                $q_s \leftarrow$ current total load of $s$
25:                if $DPos_r < DPos_{r'}$ and $q_s + q_r \leq Q_s$ then
26:                    push $r$ into $s$
27:                    if CanLoad($f', S, DPos$) then
28:                        return true
29:                    pop $s$                                                       ▷ restore the state
30:    return false

31: $\Phi \leftarrow \emptyset$                                        ▷ Initialise global set of fragments
```

```
32: function EXTEND($f, t', q'$)
33:     if $q' = 0$ then
34:         for $p \in P \cap f$ do
35:             $DPos_p \leftarrow \Gamma_f(p)$                ▷ Delivery position of pickup request $p$
36:         $S \leftarrow ((), \dots, ())$                     ▷ Empty all stacks
37:         if ($|f| \leq 2 \times NumStacks$) or (CANLOAD($f', S, DPos$)) then
38:             $\Phi \leftarrow \Phi \cup \{f\}$
39:             return true
40:         else
41:             return false
42:     CanComplete ← false
43:     $r_m \leftarrow$ the last vertex in $f$
44:     for $r \in P \cap f$ do                            ▷ Try to completely deliver everything first
45:         if $(r+n) \notin f$ then
46:             $t'' \leftarrow min(t' + t_{r_m,(r+n)}, e_{r+n})$       ▷ Arrival time at $(r+n)$
47:             if $t'' \leq l_{r+n}$ and EXTEND($f + (r+n), t'', q' - q_r$) then
48:                 CanComplete ← true
49:     if not CanComplete then
50:         return false
51:     for $r \in P$ do                                   ▷ Try every possible pickup
52:         if $r \notin f$ then
53:             $t'' \leftarrow min(t' + t_{r_m,r}, e_r)$              ▷ Arrival time at $r$
54:             if $t'' \leq l_r$ and $(q' + q_r) \leq Q$ then
55:                 EXTEND($f + r, t'', q' + q_r$)                ▷ Extend with pickup $r$
56:     return true                                        ▷ We can complete $f$

57: for $i \in P$ do
58:     EXTEND($i, e_i, q_i$)
```

4.4.2. Fragments for the PDPTWMS-S

Clearly, if the maximum request size is no larger than a stack capacity and the fragment is feasible for the PDPTWMS, then it is feasible for the PDPTWMS-S. If the fragment respects all PDPTW rules but does not have a legal loading plan with respect to the PDPTWMS rules, then we attempt to find a loading plan that splits requests across stacks.

If a fragment is PDPTW feasible we can remove any internal sequence of two vertices where the delivery of a request immediately follows the pickup. This is because we know that the maximum vehicle capacity is never exceeded, so there must be enough room across all stacks to load the request. This reduction can be applied repeatedly. For example, consider the following fragment $\langle p_1, p_2, d_1, p_3, p_4, p_5, d_5, d_4, d_2, d_3 \rangle$, we can remove the internal sequence $\langle p_4, p_5, d_5, d_4 \rangle$. Now we only need to check if the remaining sequence $\langle p_1, p_2, d_1, p_3, d_2, d_3 \rangle$ has a feasible loading plan.

In respect to multi-stack loading feasibility, different fragments may share a similar loading configuration (*Loading Equivalent*) if their pickups and deliveries share similar positions and sizes and are sequenced in the same pattern. We store fragments that have already been checked and if possible just lookup the results for similar configuration fragments. To illustrate, suppose requests 1, 2 and 3 are the same size as requests 4, 5 and 6 (respectively), $q_1 = q_4, q_2 = q_5$ and $q_3 = q_6$, then if $\langle p_1, p_2, p_3, d_1, d_2, d_3 \rangle$ has a feasible loading plan, so does $\langle p_4, p_5, p_6, d_4, d_5, d_6 \rangle$.

Remark 1. If fragments f' and $\tilde{f}$ are *Loading Equivalent* then f' has a feasible PDPTWMS-S loading plan if and only if $\tilde{f}$ has a feasible PDPTWMS-S loading plan.

When considering a candidate loading plan we do not need to check stack capacity is respected every time we load a request (pickup). It is only necessary to check stack capacity at those points in the fragment when a pickup is followed by a delivery. If the stack capacity constraints are respected at these points then they are respected for the whole fragment. Let Π denote a set of sets, where each set represents undelivered requests that appear on board every time the vehicle travel from pickups to deliveries. For example, from the fragment $\langle p_1, p_2, d_1, p_3, d_2, d_3 \rangle$, $\Pi = \{\{p_1, p_2\}, \{p_2, p_3\}\}$. We make use of Π to check that the loads for the sets in Π do not exceed the stack capacities.

Assume a fragment f is PDPTW feasible and not PDPTWMS feasible. We introduce an IP formulation to check that f can be loaded for the PDPTWMS-S. This IP has a constant objective function of 0, so it simply determines if a feasible solution exists.

Let θ denote the set of requests (represented by the pickup vertices) that appear in f. We denote by I the set of stack incompatible pairs of requests in θ. Let the binary variables Υ_{ps} equal 1 if some part of request $p \in P$ uses stack s. Also, let the integer variables Λ_{ps} denote the number of units of request p that are placed on stack s such that, $\Lambda_{ps} \leq q_p$. The stack assignment problem can be formulated as follows (we omit the constant objective function).

$$\Lambda_{ps} \leq (min\{Q_s, q_p\})\Upsilon_{ps}, \qquad \forall p \in \theta, \forall s \in S \tag{4.1}$$

$$\sum_{s \in S} \Lambda_{ps} = q_p, \qquad \forall p \in \theta \tag{4.2}$$

$$\sum_{p \in \zeta} \Lambda_{ps} \leq Q_s, \qquad \forall s \in S, \forall \zeta \in \Pi \tag{4.3}$$

$$\Upsilon_{is} + \Upsilon_{js} \leq 1, \qquad \forall (i,j) \in I, \forall s \in S \tag{4.4}$$

$$\Upsilon_{ps} \in \{0,1\}, \Lambda_{ps} \in \mathbb{Z}^+ \qquad \forall p \in \theta, \forall s \in S \tag{4.5}$$

Constraints (4.1) ensure that the number of units of request p placed on any stack will never exceed the stack capacity. Constraints (4.2) guarantee that all units of request p will be loaded into stacks. Constraints (4.3) guarantee that the stack capacity is respected. Constraints (4.4) ensure that stack incompatible requests will not be loaded on the same stack.

The overall algorithm to check the PDPTWMS-S feasibility, for a fragment that is already known to be PDPTW feasible, can be described in the following steps:

1. Remove internal sequences.
2. Check if a *Loading Equivalent* fragment has a cached result and return cached result if one is found.
3. Check PDPTWMS loading feasibility. If feasible, cache result and return.

4. Solve stack assignment IP. Cache result and return.

The aim of this algorithm is to minimize the number of stack assignment IPs solved. It is very effective as around 90 percent of fragments checked do not require an IP to be solved.

4.4.3. Fragment Dominance Rules

Consider a fragment $f \in \Phi$ that consists of a sequence of pickup and delivery vertices. Let σ_f denote the cumulative distances starting from the first vertex and ending at the last vertex of f (the cost of fragment f). Let $f^- \in P$ and $f^+ \in D$ denote the first pickup vertex and the last delivery vertex of fragment f respectively. Let η_f denote the earliest arrival time at f^+; η_f can be obtained by calculating the arrival time at each vertex working forward from the earliest departure time (e) of f^-. Also, let ψ_f denote the latest possible departure time from f^-; ψ_f can be obtained by calculating the departure time at each vertex working backward from the latest arrival time (l) of f^+. Finally, if $\theta_{f_1} = \theta_{f_2}$, $f_1^- = f_2^-$, $f_1^+ = f_2^+$, $\sigma_{f_1} \leq \sigma_{f_2}$, $\eta_{f_1} \leq \eta_{f_2}$, and $\psi_{f_1} \geq \psi_{f_2}$, and at least one of the inequalities is strict, then fragment f_1 dominates fragment f_2. If all the comparisons are equal then f_1 dominates f_2 if it appears earlier in lexicographical order. All dominated fragments are disregarded.

4.5. Chaining the Fragments in a Relaxed Model

To chain the generated fragments we formulate a network flow model with side constraints. The models to chain the fragments for PDPTWMS or for PDPTWMS-S are the same. We call this model the relaxed PDPTWMS (PDPTWMS-R). The notation and formulation in this section are similar to those presented in Section 3.3 except that we do not apply time discretization.

Let $\mathscr{A}$ denote the set of all non-timed arcs. The set of arcs $\mathscr{A}$ can be characterized as $\mathscr{A} = \{(0,i) : i \in P\} \cup \{(i,0) : i \in D\} \cup \{(i,j) : (i,j) \in \bar{A}, i \in D, j \in P\}$. Let $\alpha \in \mathscr{A}$ denote arcs (start, end, and empty arcs), each of which has a start “tail” vertex denoted by $\alpha^- \in V$ and an end “head” vertex denoted by $\alpha^+ \in V$. Likewise a fragment $f \in \Phi$ has a start “tail” vertex $f^- \in P$ and an end “head” vertex $f^+ \in D$. Let c_α denote the cost of traversing arc $\alpha \in \mathscr{A}$. Let σ_f denotes the cost of fragment f. Also, let $\Phi(p)$ denote the set of fragments that contain vertex p such that $\Phi(p) = \{f \in \Phi, p \in f\}, \forall p \in P$. The binary variables x_f are equal to 1 if and only if the fragment $f \in \Phi$ is used in the solution. Finally, the binary variables y_α are equal to 1 if and only if the arc $\alpha \in \mathscr{A}$ is used in the solution.

The PDPTWMS-R can be formulated as an integer programming problem as follows.

$$\text{minimize} \sum_{\alpha \in \mathscr{A}} c_\alpha y_\alpha + \sum_{f \in \Phi} \sigma_f x_f \tag{4.6}$$

subject to

$$\sum_{f \in \Phi(p)} x_f = 1, \qquad \forall p \in P \tag{4.7}$$

$$\sum_{\alpha \in \mathscr{A}, \alpha^+ = i} y_\alpha + \sum_{f \in \Phi, f^+ = i} x_f = \sum_{\alpha \in \mathscr{A}, \alpha^- = i} y_\alpha + \sum_{f \in \Phi, f^- = i} x_f, \quad \forall i \in V \tag{4.8}$$

$$y_\alpha, x_f \in \{0,1\}, \qquad \forall \alpha \in \mathscr{A}, \forall f \in \Phi \tag{4.9}$$

The objective function (4.6) minimizes the total routing cost. Constraints (4.7) ensure that each request is served exactly once. Finally, constraints (4.8) ensure that for each vertex in the network the number of arcs (including fragments) entering a vertex equals the number leaving (flow balance).

Suppose a path R is infeasible with respect to the time windows condition. Let $f(R)$ represent the set of fragments $f \in \Phi$ and $\alpha(R)$ represent the set of arcs $\alpha \in \mathscr{A}$ in the path R. Omitting the start and the end arcs, we can search the path for the shortest infeasible chain. Let $f(r) \subseteq f(R)$ be the set of fragments in the shortest sequence of fragments that violates the time window condition in the path R. Also, let $\beta(r) \subseteq \alpha(R)$ represent the set of empty arcs β that connect the fragments in $f(r)$. The lazy constraint to cut off the time window infeasibility is as follows:

$$\sum_{\alpha \in \beta(r)} y_\alpha + \sum_{f \in f(r)} x_f \leq 2|f(r)| - 2 \tag{4.10}$$

Similarly, the lazy constraint to cut off a cycle r is:

$$\sum_{\alpha \in \beta(r)} y_\alpha + \sum_{f \in f(r)} x_f \leq 2|f(r)| - 1 \tag{4.11}$$

4.6. Computational Results

We used a computer equipped with Intel Core i7-6700 processor (3.4 GHz) running the Window 10 operating system. Gurobi 7.0.2 and Python 3.5 were used to implement our methods. A time limit of 7200 seconds imposed for all testing. For all instances in our tests the traveling costs c_{ij} and the travel times t_{ij} are rounded down to the second decimal digit.

The only work in the literature for exactly solving the PDPTWMS is the one introduced by Cherkesly et al. (2016). Their results were generated on a computer with Intel Core i7-3770 processor (3.4 GHz), CPLEX 12.4.0.0 solver used to solve the restricted master problem. To minimize the total number of vehicles used a big cost of 100,000 was imposed on each start arc. They assume that $c_{ij} = t_{ij}$, $\forall (i, j) \in A$ for all instances. A time limit of 7200 seconds was imposed for the solution of each instance. They generated two classes of instances C1 and C2. In the C1 class each request has a demand of single-unit size and the total capacity of each vehicle is six. In the C2 class the demand of each request is a random number between three and nine. The total capacity of the vehicle is 24 if the number of the vehicle compartments is one or two, and 27 if the vehicle has three compartments. They tested these instances with the number of requests ranging from 25 to 75 in increments of 5. These PDPTWMS instances are classified as follows:

- "a280" instances derived from the a280 instances of the traveling salesman problem library (TSPLIB). There is a total of 198 instances, 99 of each class (C1 and C2). Classified into nine groups, each group consists of 11 instances depend on the number of requests. For each group the earliest time, at which service at vertex $i \in V \setminus \{0\}$ can begin, is randomly generated in three different setting, that is, 500–1000, 1000–1200, and 1500–2000. For each setting three different time windows lengths were assigned, that is, 15, 30, and 45 as follows.

1. $e_i \leq 500, \forall i \in P$ and $e_i \leq 1000, \forall i \in D$, denoted by: w15–500–1000, w30–500–1000 and w45–500–1000.
2. $e_i \leq 1000, \forall i \in P$ and $e_i \leq 1200, \forall i \in D$, denoted by: w15–1000–1200, w30–1000–1200 and w45–1000–1200.
3. $e_i \leq 1500, \forall i \in P$ and $e_i \leq 2000, \forall i \in D$, denoted by: w15–1500–2000, w30–1500–2000 and w45–1500–2000.

To illustrate, for group w15–500–1000, the length of the time window at each vertex $i \in V \setminus \{0\}$ is 15, and the maximum earliest time, that is randomly generated, is 500 for each pickup vertex and 1000 for each delivery vertex.

- "brd14051", "d18512", "fnl4461", and "nrw1379" instances, 55 instances each, derived from the brd14051, d18512, fnl4461, and nrw1379 instances of the TSPLIB. The total is 440 instances, 220 of each class (C1 and C2). Classified into five groups, each group consists of 11 instances depend on the number of requests. For each group the earliest time, at which service at vertex $i \in V \setminus \{0\}$ can begin, is randomly generated, and five different time windows lengths were assigned, that is 45, 60, 75, 90, and 120 as follows.
  $e_i \leq 3000, \forall i \in P$ and $e_i \leq 4000, \forall i \in D$, denoted by:
  w45–3000–4000, w60–3000–4000, w75–3000–4000, w90–3000–4000, and w120–3000–4000.

Each class was tested for one, two and three stacks variants. The total number of instances is 1,914 $(3 \times (198 + 440))$.

4.6.1. The PDPTWMS

In our tests all instances defined for the PDPTWMS (1914 instances) were solved to optimality except seven of a280-c1 type. Feasible upper bounds were obtained for three of them a280-c1-131-w45-1500-2000 and a280-c1-141-w45-1500-2000 of two stacks and a280-c1-141-w30-1500-2000 of three stacks. No feasible solution was found within the time limit for the instances a280-c1-121-w45-1500-2000, a280-c1-131-w45-1500-2000, a280-c1-141-w45-1500-2000, and a280-c1-151-w45-1500-20001 of three stacks. More detailed computational results for all instances are provided in Appendix B.

Tables 4.1 and 4.2 show comparison summary results to Cherkesly et al. (2016) for each group of a280 instances, and for each group of the combined brd14051, d18512, fnl4461, and nrw1379 instances. The average times are calculated considering only instances solved to optimality by both methods.

Table 4.3 presents the average CPU time for each group for all instances that solved to optimality by our method. Table 4.4 shows brief comparison summary results where the instances are grouped according to the class (C) type and the number of stacks. The average times are calculated considering only instances solved to optimality by both methods.

The columns in tables 4.1-4.4 are labelled as follows: instances by group (*Group*), the number of instances in each group that are solved to optimality (*Solved*), and the average CPU time in seconds (*Sec.*). Note that *Sec.* also includes the CPU time used to generate the non-dominated fragments.

Table 4.1. PDPTWMS Comparison Summary Results for the a280 Groups of C1 and C2 Classes

	Cherkesly et al. (2016)						Our method					
	1 stack		2 stacks		3 stacks		1 stack		2 stacks		3 stacks	
Group	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*
C1 class												
w15-500-1000	11	59.9	7	547.8	6	1,654.1	11	0.1	11	0.2	11	0.2
w15-1000-1200	11	60.6	5	2,036.2	3	634.7	11	0.1	11	0.3	11	0.1
w15-1500-2000	7	98.6	2	563.9	2	3,009.4	11	0.3	11	0.5	11	0.9
w30-500-1000	10	1,783.1	3	523.7	3	1,944.1	11	0.1	11	0.3	11	0.4
w30-1000-1200	10	971.8	3	2,396.4	2	1,069.6	11	0.2	11	2.4	11	0.4
w30-1500-2000	8	654.3	1	162.1	1	653.3	11	0.2	11	0.5	10	0.5
w45-500-1000	7	2,130.4	2	2,956.5	1	2,262.0	11	0.2	11	0.4	11	0.3
w45-1000-1200	7	772.2	3	3,480.1	2	6,293.6	11	0.1	11	1.2	11	2.0
w45-1500-2000	6	785.1	1	17.4	1	102.6	11	0.3	9	0.6	7	0.6
C2 class												
w15-500-1000	11	3.7	11	541.4	10	1,308.4	11	0.1	11	0.4	11	0.5
w15-1000-1200	11	3.8	9	939.1	8	726.5	11	0.1	11	0.8	11	0.8
w15-1500-2000	11	64.8	6	1,391.0	5	1,764.7	11	0.4	11	2.7	11	2.1
w30-500-1000	11	49.5	6	124.0	5	838.1	11	0.1	11	0.3	11	0.4
w30-1000-1200	11	21.5	9	1,070.6	7	825.2	11	0.2	11	2.5	11	1.2
w30-1500-2000	11	834.9	6	1,410.6	4	554.0	11	0.8	11	6.4	11	3.7
w45-500-1000	10	310.5	4	419.7	4	3,033.5	11	0.3	11	0.8	11	2.0
w45-1000-1200	11	267.3	6	1,037.0	5	287.3	11	0.9	11	2.5	11	5.4
w45-1500-2000	9	478.3	6	1,234.6	4	1,965.9	11	1.0	11	17.8	11	20.1

Table 4.2. PDPTWMS Comparison Summary Results for the brd14051, d18512, fnl4461, and nrw1379 Groups of C1 and C2 Classes

	Cherkesly et al. (2016)						Our method					
	1 stack		2 stacks		3 stacks		1 stack		2 stacks		3 stacks	
Group	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*	*Solved*	*Sec.*
C1 class												
w45-3000-4000	44	57.7	26	1,070.7	19	895.2	44	0.1	44	0.2	44	0.1
w60-3000-4000	44	223.0	18	1,659.7	14	965.0	44	0.1	44	0.1	44	0.1
w75-3000-4000	44	281.3	16	1,306.7	11	804.0	44	0.1	44	0.2	44	0.1
w90-3000-4000	43	325.8	12	843.3	12	1,057.3	44	0.1	44	0.1	44	0.1
w120-3000-4000	39	703.6	10	1,231.0	7	1,233.5	44	0.1	44	0.2	44	0.2
C2 class												
w45-3000-4000	44	2.1	42	601.9	37	710.9	44	0.1	44	0.3	44	0.3
w60-3000-4000	44	5.3	38	500.2	33	576.3	44	0.1	44	0.3	44	0.3
w75-3000-4000	44	4.8	38	637.0	35	795.4	44	0.1	44	0.3	44	0.4
w90-3000-4000	44	21.4	30	707.3	25	614.0	44	0.1	44	0.3	44	0.3
w120-3000-4000	44	41.0	28	266.8	26	922.5	44	0.1	44	0.5	44	0.6

The results for the PDPTWMS instances indicate that our method outperforms the branch-price-and-cut algorithms of Cherkesly et al. (2016). Table 4.4 shows that our method is at least more than 100 times faster and on average more than 1,000 faster and solves to optimality 660 instances previously unsolved.

Table 4.3. Average Time for Solved PDPTWMS Instances for All Groups of C1 and C2 Classes

	C1 class			C2 class		
	1 stack	2 stacks	3 stacks	1 stacks	2 stacks	3 stacks
Group	*Sec.*	*Sec.*	*Sec.*	*Sec.*	*Sec.*	*Sec.*
(a280)						
w15-500-1000	0.1	0.6	0.8	0.1	0.4	0.6
w15-1000-1200	0.1	1.9	2.5	0.1	2.2	2.3
w15-1500-2000	1.0	106.1	395.1	0.4	13.6	72.9
w30-500-1000	0.2	2.5	4.9	0.1	1.9	2.2
w30-1000-1200	0.3	28.7	36.5	0.2	3.2	8.2
w30-1500-2000	0.9	470.5	367.2	0.8	41.2	147.7
w45-500-1000	0.4	13.6	32.0	0.3	5.8	9.2
w45-1000-1200	0.5	172.1	441.8	0.9	26.8	78.3
w45-1500-2000	3.3	1,126.2	470.8	1.3	164.9	784.0
(brd14051, d18512, fnl4461, and nrw1379)						
w45-3000-4000	0.1	0.5	0.5	0.1	0.4	0.5
w60-3000-4000	0.1	0.7	0.9	0.1	0.7	0.9
w75-3000-4000	0.1	1.3	1.7	0.1	0.7	1.1
w90-3000-4000	0.1	1.9	3.1	0.1	1.4	2.0
w120-3000-4000	0.2	3.2	7.6	0.1	2.3	4.5

Table 4.4. PDPTWMS Comparison Summary Results

	Cherkesly et al. (2016)		Our method	
Group	*Solved*	*Sec.*	*Solved*	*Sec.*
C1 class				
1 stack	291	433.0	319	0.1
2 stacks	109	1,298.0	317	0.3
3 stacks	84	1,215.6	314	0.2
C2 class				
1 stack	316	77.2	319	0.2
2 stacks	239	644.7	319	1.2
3 stacks	208	835.7	319	1.1
Total	1247		1907	

4.6.2. The PDPTWMS-S

The goals of our testing in this section are to compare the PDPTWMS-S to the PDPTWMS and to measure the impact of using more stacks in the PDPTWMS-S. Also, we show how the techniques described in section 4.4.2 help reduce the fragment generation time and the number of stack assignment IPs solved (model (4.1)–(4.5)).

We have tested the algorithm of the PDPTWMS-S using the a280 instances. Because the number of instances is large, we limit our test to instances of 70 and 75 requests (141 and 151 vertices). We only considered instances of C2 class because in the instances of C1 class each request has a demand of single-unit size.

The columns of the tables in this section represent the following: *Instance*: the instances name; F_N: the total number of non-dominated fragments; *IPs*: the number of stack assignment IPs solved; Sec_{IP} the CPU time in seconds used to solve the stack assignment IPs, rounded to the nearest tenth; Sec_f: the CPU time in seconds used to generate non-dominated fragments, including Sec_{IP} time, rounded to the nearest tenth; *Sec.*: the total CPU time in seconds used to solve the instance, including Sec_f time, rounded to the nearest tenth; Z^*: the optimal solution obtained within the time limit or left blank if no feasible solution found. If only a feasible solution was obtained within the time limit, then it is reported in bold in column Z^* as a best upper bound found; *Gap*: The percentage gap computed as $100(Z^* - Z_R)/Z^*$, where Z_R is the lower bound at the root node.

4.6.2.1 The Added Value of Stack Splitting

Tables 4.5 and 4.6 provide computational results for the instances of two and three stacks respectively. The columns S_{Veh} and S_{Dist} show the percentage amount of saving, comparing to the PDPTWMS, in the number of vehicles and in the total traveled distance respectively. For example, if the number of vehicles used for the PDPTWMS and the PDPTWMS-S are Veh_{MS} and Veh_{MS-S} respectively, then the saving $S_{Veh} = 100(Veh_{MS-S} - Veh_{MS})/Veh_{MS}$. Similarly, $S_{Dist} = 100(Dist_{MS-S} - Dist_{MS})/Dist_{MS}$.

Analyzing the results in Tables 4.5 and 4.6, it can be seen that a reduction in cost was obtained for all instances. The distance increases for some instances when the number of vehicles is reduced. This is because the vehicle cost is relatively high, and the priority is to minimize the number of vehicles. Moreover, we see that a reduction of the number of vehicles was more common in the three stack case (13 out of 17 instances with a feasible solution) than in the two stack case (3 out of 18 instances). This suggests that splitting customer requests is more effective as the number of stacks increases from 2 to 3, which may be because the stack capacity has reduced - in this case from 12 to 9.

The stack splitting extension usually leads to increased solution time. This is due to the flexibility in the loading configuration, thus more fragments are generated. For example, in the most difficult instance - 151-w45-1500-2000 with 3 stacks and vehicle capacity of 27 - the PDPTWMS had 3,448,372 non-dominated fragments whereas the PDPTWMS-S had 5,543,220.

Table 4.5. PDPTWMS-S for Two Stacks Vehicle with Capacity 24

Instance(a280-C2)	F_N	*IPs*	*Gap*	Sec_{IP}	Sec_f	*Sec.*	Z^*	S_{Veh}	S_{Dist}
141-w15-500-1000	4,237	511	4.942	0.8	1.5	2.7	2,013,834.80	0.00	-2.01
151-w15-500-1000	3,191	424	0.005	0.5	1.3	1.5	2,116,832.91	-4.55	7.89
141-w15-1000-1200	7,238	739	0.006	1.0	2.5	3.9	1,813,158.30	-5.26	-3.96
151-w15-1000-1200	10,475	1,176	0.032	1.7	4.1	10.7	1,714,862.84	0.00	-4.99
141-w15-1500-2000	59,529	10,912	0.032	14.6	35.1	63.0	1,312,428.12	0.00	-2.07
151-w15-1500-2000	160,856	18,415	0.020	24.5	68.5	119.8	1,413,193.17	0.00	-3.87
141-w30-500-1000	11,203	1,753	0.016	2.2	5.1	8.5	1,713,208.06	0.00	-0.64
151-w30-500-1000	13,669	1,305	0.022	1.6	4.7	9.9	1,713,336.68	0.00	-0.31
141-w30-1000-1200	25,327	3,458	0.009	4.7	11.9	17.6	1,512,481.92	0.00	-3.36
151-w30-1000-1200	24,892	4,677	0.013	6.1	16.0	22.8	1,612,839.00	0.00	-3.21
141-w30-1500-2000	343,312	47,498	0.042	70.1	182.8	288.8	1,212,488.17	0.00	-9.59
151-w30-1500-2000	350,584	38,766	0.012	56.8	162.5	253.7	1,312,366.58	0.00	-5.34
141-w45-500-1000	16,002	3,880	0.021	5.3	11.8	16.0	1,713,586.08	0.00	-1.56
151-w45-500-1000	44,914	5,442	0.034	6.9	19.9	44.5	1,713,456.88	0.00	-0.59
141-w45-1000-1200	52,755	10,389	0.065	15.0	35.6	218.5	1,312,423.10	-7.14	6.06
151-w45-1000-1200	113,509	15,596	0.013	24.5	63.4	92.7	1,412,202.13	0.00	-0.45
141-w45-1500-2000	408,990	113,153	0.029	168.1	405.1	578.4	1,311,960.23	0.00	-0.76
151-w45-1500-2000	1,536,140	133,097	0.038	212.9	688.7	3,150.8	1,212,235.66	0.00	-9.02

Table 4.6. PDPTWMS-S for Three Stacks Vehicle with Capacity 27

Instance(a280-C2)	F_N	*IPs*	*Gap*	Sec_{IP}	Sec_f	*Sec.*	Z^*	S_{Veh}	S_{Dist}
141-w15-500-1000	6,173	775	0.008	1.3	2.7	2.9	1,813,023.92	-5.26	-2.88
151-w15-500-1000	4,971	657	0.015	1.0	2.2	3.3	1,914,804.80	-5.00	-3.60
141-w15-1000-1200	13,113	835	0.004	1.7	4.1	5.7	1,712,573.55	0.00	-3.74
151-w15-1000-1200	16,493	1,553	0.022	3.0	6.6	11.0	1,613,921.40	0.00	-3.12
141-w15-1500-2000	171,713	27,468	0.061	70.4	131.3	189.8	1,112,208.19	-8.33	2.70
151-w15-1500-2000	448,430	50,235	0.030	138.9	275.7	1,240.6	1,211,882.19	0.00	-9.02
141-w30-500-1000	19,032	3,878	0.024	6.4	12.6	17.4	1,512,574.03	-6.25	-2.41
151-w30-500-1000	23,053	2,377	0.036	5.5	11.3	49.9	1,512,797.21	-6.25	0.44
141-w30-1000-1200	56,605	6,744	0.028	15.7	32.4	43.8	1,311,867.72	-7.14	1.44
151-w30-1000-1200	56,160	9,314	0.027	21.4	41.2	67.7	1,412,149.87	-6.67	0.34
141-w30-1500-2000	1,207,062	167,816	8.877	614.1	1,062.1	4,260.2	1,110,698.84	-8.33	-1.98
151-w30-1500-2000	1,045,034	134,544	0.071	395.2	797.5	7,200.0	**1,112,556.51**	-8.33	1.79
141-w45-500-1000	35,882	10,660	0.011	29.3	49.4	57.2	1,511,974.05	-6.25	-3.81
151-w45-500-1000	94,147	14,754	0.018	35.1	73.3	95.0	1,512,158.54	-6.25	-0.43
141-w45-1000-1200	111,147	30,319	0.029	68.7	132.7	210.4	1,210,993.08	-7.69	-4.21
151-w45-1000-1200	231,639	33,939	0.022	84.1	183.0	245.5	1,311,844.47	0.00	-8.46
141-w45-1500-2000	1,886,234	625,479	0.022	2449.3	4,629.4	6,714.4	1,110,746.34	-8.33	-4.35
151-w45-1500-2000	5,543,220	860,975		3221.1	6,350.0				

4.6.2.2 The Impact of Increasing the Number of Stacks with Stack Splitting

To measure the impact of increasing the number of stacks when using vehicles with different stacks and similar capacity, we modify the instances by changing the vehicle capacity from 27 to 24 for three stacks. Table 4.7 presents the results obtained for these instances.

One of the PDPTWMS assumptions is that a customer's demand may not exceed the stack capacity. We modified the a280 instances of C2 class (used in this section) by assuming the number of stacks is

4, 6 or 24 and fixing the vehicle capacity to 24 (24 stacks corresponds to the PDPTW solution). This allows a customer demand to exceed the stack capacity. In Appendix B we provide computational results for these instances. We also provide a table that shows more detail of the impact of using more stacks on the cost.

Table 4.7. PDPTWMS-S for Three Stacks Vehicle with Capacity 24

Instance(a280-C2)	F_N	*Gap*	Sec_f	*Sec.*	Z^*
141-w15-500-1000	5,467	5.205	2.1	2.5	1,913,172.11
151-w15-500-1000	4,483	0.014	2.0	3.0	2,014,859.28
141-w15-1000-1200	11,933	0.008	3.4	5.6	1,712,822.23
151-w15-1000-1200	14,965	0.032	5.1	10.3	1,614,344.71
141-w15-1500-2000	141,466	0.042	85.6	352.2	1,211,811.03
151-w15-1500-2000	350,269	0.042	174.9	666.7	1,212,956.38
141-w30-500-1000	15,298	0.014	9.3	13.5	1,612,629.93
151-w30-500-1000	19,525	0.016	9.2	14.0	1,612,876.31
141-w30-1000-1200	48,082	0.010	19.0	26.7	1,411,488.39
151-w30-1000-1200	47,166	6.566	33.6	51.6	1,512,183.54
141-w30-1500-2000	802,994	0.023	553.5	880.7	1,210,968.58
151-w30-1500-2000	771,978	0.024	492.2	1,291.7	1,212,115.75
141-w45-500-1000	27,885	0.029	25.6	37.2	1,613,025.20
151-w45-500-1000	68,949	0.020	43.1	64.0	1,612,494.48
141-w45-1000-1200	87,734	7.576	80.0	118.7	1,311,115.12
151-w45-1000-1200	197,058	0.039	133.8	374.7	1,312,379.84
141-w45-1500-2000	1,178,158	0.033	1,778.7	4,474.7	1,211,452.14
151-w45-1500-2000	3,618,888		2,710.0		

Table 4.8 presents results showing the impact of increasing the number of stacks from two to three has on the number of vehicles, the total traveled distance, the number of generated fragments, and the total computational time.

The columns in table 4.8 represent the following: Δ_{F_N}: the percentage difference in the number of non-dominated fragments, computed as $100(F_{N_3} - F_{N_2})/F_{N_2}$, where F_{N_2} and F_{N_3} are the number of non-dominated fragments with two and three stacks respectively; Δ_{Sec}: the percentage difference in the total CPU time in seconds used to solve the instance, computed as $100(Sec_3 - Sec_2)/Sec_2$, where Sec_2 and Sec_3 are the total CPU time in seconds used to solve the instance with two and three stacks respectively; Δ_{Dist}: the percentage difference in the total traveled distance, computed as $100(Dist_3 - Dist_2)/Dist_2$, where $Dist_2$ and $Dist_3$ are the total traveled distance with two and three stacks respectively; Δ_{Veh}: the percentage difference in the number of vehicles, computed as $100(Veh_3 - Veh_2)/Veh_2$, where Veh_2 and Veh_3 are the number of vehicles with two and three stacks respectively.

Table 4.8 shows the positive impact on cost of increasing the number of stacks. In particular, the number of vehicles has decreased in nearly every instance. However, due to the loading flexibility we observe a sharp increase in the number of fragments. This make the problem harder to solve and usually increases the solution time. The time to generate fragments Sec_f is higher for all instances.

Table 4.8. Two Stacks (Capacity 24) - VS. - Three Stacks (Capacity 24)

Instance (a280-C2)	Δ_{F_N}	Δ_{Sec}	Δ_{Dist}	Δ_{Veh}
141-w15-500-1000	29.0	-7.4	-4.8	-5.0
151-w15-500-1000	40.5	100.0	-11.7	-4.8
141-w15-1000-1200	64.9	43.6	-2.6	-5.6
151-w15-1000-1200	42.9	-3.7	-3.5	-5.9
141-w15-1500-2000	137.6	459.0	-5.0	-7.7
151-w15-1500-2000	117.8	456.5	-1.8	-14.3
141-w30-500-1000	36.6	58.8	-4.4	-5.9
151-w30-500-1000	42.8	41.4	-3.5	-5.9
141-w30-1000-1200	89.8	51.7	-8.0	-6.7
151-w30-1000-1200	89.5	126.3	-5.1	-6.3
141-w30-1500-2000	133.9	205.0	-12.2	0.0
151-w30-1500-2000	120.2	409.1	-2.0	-7.7
141-w45-500-1000	74.3	132.5	-4.1	-5.9
151-w45-500-1000	53.5	43.8	-7.2	-5.9
141-w45-1000-1200	66.3	-45.7	-10.5	0.0
151-w45-1000-1200	73.6	304.2	1.5	-7.1
141-w45-1500-2000	188.1	673.6	-4.2	-7.7
151-w45-1500-2000	135.6			

4.6.2.3 The Effectiveness of the Techniques Described in Section 4.4.2

Recall that the overall algorithm to find a PDPTWMS-S loading plan uses the following steps:

1. Remove internal sequences.
2. Check if a *Loading Equivalent* fragment has a cached result and return cached result if one is found.
3. Check PDPTWMS loading feasibility. If feasible, cache result and return.
4. Solve stack assignment IP. Cache result and return.

In order to see the effectiveness of these techniques we use the data of Tables 4.5 and 4.6 for two stacks case with capacity 24 and three stacks case with capacity 27. In these tables we see that the number of stack assignment IPs solved is much smaller than the number of non-dominated fragments generated. Across all the two stack instances the number of IPs solved is 13 percent of the number of non-dominated fragments and for three stacks it is 18 percent of the number of non-dominated fragments. While it is not shown in the tables, our tests also show that the stack assignment IP is infeasible (i.e. no loading plan exists) 96 percent of the time it is used in two stack instances and 55 percent of the time it is used in three stack instances. This again shows how increasing the number of stacks increases the prevalence of splitting.

We also ran an experiment where we disabled the check for Load Equivalent fragments. These results are shown in Table 4.9. In this table, column Δ_{IP} shows the percentage reduction in the number of stack assignment IPs solved when using the check for Load Equivalent fragments. It is computed as $100(IP_W - IP_O)/IP_O$, where IP_O is the number of IPs without the check, and IP_W is the number of IPs

Table 4.9. The Reduction in the IPs and the Fragment Generation Time

	2 stacks (capacity 24)		3 stacks (capacity 27)	
Instance (a280-C2)	Δ_{IP}	Δ_{Sec_f}	Δ_{IP}	Δ_{Sec_f}
141-w15-500-1000	-41.4	-34.8	-24.3	-43.8
151-w15-500-1000	-51.3	-27.8	-33.9	-26.7
141-w15-1000-1200	-61.8	-46.8	-54.5	-43.1
151-w15-1000-1200	-58.2	-40.6	-42.2	-37.7
141-w15-1500-2000	-68.1	-55.8	-56.2	-47.4
151-w15-1500-2000	-74.9	-60.1	-69.3	-56.6
141-w30-500-1000	-49.3	-35.4	-41.2	-28.4
151-w30-500-1000	-42.5	-36.5	-43.8	-31.5
141-w30-1000-1200	-67.6	-54.2	-57.9	-47.0
151-w30-1000-1200	-55.7	-35.5	-47.0	-37.5
141-w30-1500-2000	-78.4	-65.7	-71.3	-59.7
151-w30-1500-2000	-81.6	-68.6	-77.0	-68.3
141-w45-500-1000	-57.0	-44.9	-50.5	-45.1
151-w45-500-1000	-66.4	-48.2	-61.8	-50.4
141-w45-1000-1200	-63.1	-46.2	-58.7	-43.6
151-w45-1000-1200	-63.7	-46.6	-61.0	-44.1
141-w45-1500-2000	-71.7	-59.1	-63.8	-49.7
151-w45-1500-2000	-76.7	-59.3	-73.4	-63.1

with the check. Likewise, column Δ_{Sec_f} shows the percentage reduction in the fragment generation time. It is computed as $100(Sec_{f_W} - Sec_{f_O})/Sec_{f_O}$, where Sec_{f_O} is the CPU time in seconds used to generate non-dominated fragments without the check, and Sec_{f_W} is the CPU time in seconds used to generate non-dominated fragments with the check. It is clear that checking for Load Equivalent fragments provides a significant speedup, both in terms of reducing the number of stack assignment IPs solved and reducing overall fragment generation time. This effect is particularly significant for the more difficult problems towards the bottom of the table.

4.7. Conclusions

In this chapter, we introduce a new practical variant of the PDPTWMS which we denote the PDPTWMS-S. In this variant we allow the size of a customer's demand to exceed the stack capacity and be only limited to the vehicle capacity, and we allow a customer's demand to be split among all stacks. This flexibility is very important for the transportation of pallets or other standardized containers and is essential when the size of any single request exceeds the size of a stack. Computational experiments show that relaxing the loading rules in this way results in a significant cost reduction, especially as the number of stacks increases.

We present a new IP formulation to model the assignment of requests to stacks when the requests can be split across multiple stacks. We propose an algorithm to produce loading plans for the PDPTWMS-S which takes advantage of the problem characteristics to minimize the number of stack assignment IPs solved.

Computational results show that our method outperforms the current state-of-the-art method with in excess of 100 fold improvements in run time. We solve 660 instances of the PDPTWMS that were previously unsolved.

Chapter 5

Valid Inequalities and Infeasible Paths Cuts

5.1. Introduction

In this chapter, we propose new valid inequalities for the PDPTW exploiting the nature of the network presented in previous chapters. The aim is to improve the lower bound obtained by strengthening the linear programming (LP) relaxation of the PDPTW formulation. Valid inequalities close the gap between the LP and the integer program (IP) solutions, and reduce the size of the branch-and-bound search tree. We generate theses valid inequalities only at the root node, and we repeat the procedure until no more violated valid inequalities are found. Valid inequalities should not cut off any feasible integer solutions. Moreover, we propose cuts that can strengthen the lazy constraints proposed in the previous chapters to cut infeasible paths with respect to time windows condition.

For a complete overview of the theory of valid inequalities, we refer the reader to the textbooks by (e.g., Wolsey 1998; Nemhauser and Wolsey 1999; Pochet and Wolsey 2006).

5.2. Valid Inequalities

We are interested in refining the feasible region (the search space) in the linear programming formulation (3.1)–(3.4) presented in chapter 3 by relaxing the integrality condition of the variables x_ω and y_a as follows:

$$y_a, x_\omega \geq 0, \quad \forall a \in A, \forall \omega \in \Omega \tag{5.1}$$

The notation and formulation used in this chapter are similar to those in previous chapters. Recall that $f = (f^-, \ldots, f^+) \in \Phi$, where $f^-, f^+ \in V$, denotes the original fragments, and $\omega = (\omega^-, \omega^+, f) \in \Omega$, where $\omega^-, \omega^+ \in N$, denotes *timed* fragments. Also, recall that $\alpha = (\alpha^-, \alpha^+) \in \mathscr{A}$, where $\alpha^-, \alpha^+ \in V$, denote the original arcs (*start*, *end*, and *empty* arcs) and $a = (a^-, a^+) \in A$, where $a^-, a^+ \in N$, denote the *timed* arcs. Note that α^- and a^- of a *start* arc and α^+ and a^+ of an *end* arc is 0 (the depot).

Let $O(\omega) = f$, where $\omega \in \Omega$ and $f \in \Phi$, denote the original fragment of a *timed* fragment. Similarly, let $O(a) = \alpha$, where $a \in A$ and $\alpha \in \mathscr{A}$, denote the original arc of a *timed* arc. Let $t(i_h) = h$ be the time associated with the *timed* node $i_h \in N$. Let $\oplus$ denote the concatenation operator.

Recall that our network preprocessing means an empty arc (α^-, α^+) with $\alpha^- \geq n+1$ and $\alpha^+ \leq n$ only exists in the network if the route $(\alpha^- - n, \alpha^-, \alpha^+, \alpha^+ + n)$ is legal. This route is made up of two fragments. We can extend this reasoning to the case where one of the two fragments is a longer fragment. Thus, we can derive several valid inequalities for the PDPTW as follows.

5.2.1. Fragments After Empty Arcs

We define the set of fragments that can follow an *empty* arc as follows:
$\overrightarrow{\Phi}(\alpha) = \{f \in \Phi \colon f^- = \alpha^+ \wedge [(\alpha^- - n) \oplus \alpha^- \oplus f] \text{ is feasible}\}$.

Similarly, we define the set of *timed* fragments that can follow a *timed empty* arc as follows: $\overrightarrow{\Omega}(a) = \{\omega \in \Omega \colon O(\omega) \in \overrightarrow{\Phi}(O(a)) \wedge t(\omega^-) \geq t(a^+)\}$.

Let $\overrightarrow{T}(a) = \{a' \in A \colon (O(a') = O(a)) \wedge t(a'^+) \geq t(a^+)\}$ denote all *timed* arcs between the same locations as a, at or later than a.

The following inequalities are valid for the PDPTW:

$$y_\alpha \leq \sum_{f \in \overrightarrow{\Phi}(\alpha)} x_f \qquad \forall \alpha \in \mathscr{A}, \alpha^- \neq 0 \tag{5.2}$$

$$\sum_{a' \in \overrightarrow{T}(a)} y_{a'} \leq \sum_{\omega \in \overrightarrow{\Omega}(a)} x_\omega \qquad \forall a \in A, a^- \neq 0 \tag{5.3}$$

5.2.2. Fragments Before Empty Arcs

We define the set of fragments that can precede an *empty* arc as follows:
$\overleftarrow{\Phi}(\alpha) = \{f \in \Phi \colon (f^+ = \alpha^-) \wedge ([f \oplus \alpha^+ \oplus (\alpha^+ + n)] \text{ is feasible})\}$.

Similarly, we define the set of *timed* fragments that can precede a *timed empty* arc as follows: $\overleftarrow{\Omega}(a) = \{\omega \in \Omega \colon O(\omega) \in \overleftarrow{\Phi}(O(a)) \wedge t(\omega^+) \leq t(a^-)\}$.

Let $\overleftarrow{T}(a) = \{a' \in A \colon (O(a') = O(a)) \wedge (t(a'^-) \leq t(a^-)\})$ denote all *timed* arcs between the same locations as a, at or before a.

The following inequalities are valid for the PDPTW:

$$y_\alpha \leq \sum_{f \in \overleftarrow{\Phi}(\alpha)} x_f \qquad \forall \alpha \in \mathscr{A}, \alpha^+ \neq 0 \tag{5.4}$$

$$\sum_{a' \in \overleftarrow{T}(a)} y_{a'} \leq \sum_{\omega \in \overleftarrow{\Omega}(a)} x_\omega \qquad \forall a \in A, a^+ \neq 0 \tag{5.5}$$

5.2.3. Arcs After Fragments

Arcs after fragments are either *empty* arcs or *end* arcs. We define the set of arcs that can follow fragment f as follows:

$\overrightarrow{\mathscr{A}}(f) = \{\alpha \in \mathscr{A} \colon ((f^+ = \alpha^-) \wedge (\alpha^+ = 0)) \vee ((f^+ = \alpha^-) \wedge ([f \oplus \alpha^+ \oplus (\alpha^+ + n)]$ is feasible$) \wedge (\alpha^+ \neq 0))\}$.

Similarly, we define the set of *timed* arcs that can follow *timed* fragment as follows: $\overrightarrow{A}(\omega) = \{a \in A \colon O(a) \in \overrightarrow{\mathscr{A}}(O(\omega)) \wedge t(a^-) \geq t(\omega^+)\}$.

Let $\overrightarrow{T}(\omega) = \{\omega' \in \Omega \colon O(\omega') = O(\omega) \wedge t(\omega'^-) \geq t(\omega^-)\}$ denote all *timed* versions of the same fragment at or later than ω.

The following inequalities are valid for the PDPTW:

$$x_f \leq \sum_{\alpha \in \overrightarrow{\mathscr{A}}(f)} y_\alpha \qquad \forall f \in \Phi \tag{5.6}$$

$$\sum_{\omega' \in \overrightarrow{T}(\omega)} x_{\omega'} \leq \sum_{a \in \overrightarrow{A}(\omega)} y_a \qquad \forall \omega \in \Omega \tag{5.7}$$

5.2.4. Arcs Before Fragments

Arcs before fragments are either *empty* arcs or *start* arcs. We define the set of arcs that can precede fragment f as follows:
$\overleftarrow{\mathscr{A}}(f) = \{\alpha \in \mathscr{A} \colon ((\alpha^+ = f^-) \wedge (\alpha^- = 0)) \vee ((\alpha^+ = f^-) \wedge ([(\alpha^- - n) \oplus \alpha^- \oplus f]$ is feasible$) \wedge (\alpha^- \neq 0))\}$.

Similarly, we define the set of *timed* arcs that can precede *timed* fragment as follows: $\overleftarrow{A}(\omega) = \{a \in A \colon O(a) \in \overleftarrow{\mathscr{A}}(O(\omega)) \wedge t(a^+) \leq t(\omega^-)\}$.

Let $\overleftarrow{T}(\omega) = \{\omega' \in \Omega \colon (O(\omega') = O(\omega)) \wedge (t(\omega'^-) \leq t(\omega^-))\}$ denote all *timed* versions of the same fragment at or before ω.

The following inequalities are valid for the PDPTW:

$$x_f \leq \sum_{\alpha \in \overleftarrow{\mathscr{A}}(f)} y_\alpha \qquad \forall f \in \Phi \tag{5.8}$$

$$\sum_{\omega' \in \overleftarrow{T}(\omega)} x_{\omega'} \leq \sum_{a \in \overleftarrow{A}(\omega)} y_a \qquad \forall \omega \in \Omega \tag{5.9}$$

5.2.5. Subset-Row Inequalities

We can apply the *subset-row* (SR) inequalities, that were introduced by Jepsen et al. (2008) for the VRPTW, to the PDPTW as follows.

Let J represents a subset of customers such that $J \subseteq V$. We are only interested in subsets consisting of three requests. Let $\Phi(J) \subseteq \Phi$ denote the subset of fragments that visit at least two requests in J. The following inequalities are valid for the PDPTW:

$$\sum_{f \in \Phi(J)} x_f \leq 1 \qquad \forall J \subseteq V, |J| = 3 \tag{5.10}$$

5.2.6. Minimal-Matching Set

Recall that θ denote the set of requests (represented by the pickup vertices) that appear in fragment f; η_f denote the earliest arrival time at f^+; and ψ_f denote the latest possible departure time from f^-.

Definition 5.2.1. We say that the fragment $\tilde{f} \in \Phi$ is *minimal-matching* to $f' \in \Phi$ if:

i) $f'^- = \tilde{f}^-$;

ii) $f'^+ = \tilde{f}^+$;

iii) $\theta_{f'} \subseteq \theta_{\tilde{f}}$;

iv) $\eta_{f'} \leq \eta_{\tilde{f}}$; and

v) $\psi_{f'} \geq \psi_{\tilde{f}}$.

Let $M(f) \subseteq \Phi$, where $|M(f)| \geq 1$, denote the set of all *minimal-matching* fragments to f. For the valid inequalities (5.6)–(5.9) we can replace the fragment f by all fragments in $M(f)$ in any cuts.

5.3. Infeasible Paths Cuts

In this section, we show how to strengthen the lazy constraints proposed in the previous chapters to cut time window infeasible paths.

Suppose a path R is part of an IP candidate solution and contains at least two fragments. Suppose the path R is infeasible with respect to the time windows condition. Let $f(R)$ represent the set of original fragments $f \in \Phi$ and $\varepsilon(R)$ represent the set of original arcs $\varepsilon \in E'$ based on the timed fragments and timed arcs in the path R.

Let r be a subpath that represents the shortest chain of fragments sequence found in the path R that violates the time window condition. Let $f(r) \subseteq f(R)$, where $f(r) = (f_1, ..., f_m), m \geq 2$, denote the set of fragments sequenced in r. To obtain r we use a different procedure from the one in previous chapters. To illustrate, suppose $f(R) = (f^a, f^b, f^c, f^d)$. We start a loop to check the subpaths (f^a, f^b), (f^b, f^c), and (f^c, f^d) against time window condition. We extend the check to each of (f^a, f^b, f^c), (f^b, f^c, f^d), and (f^a, f^b, f^c, f^d). We break the loop as soon as find an infeasible chain. As a result of this procedure any truncation of r is feasible.

Let $\beta(r)$ denote the set of empty arcs β that connect the fragments in r. Recall that $\cup_{f \in f(r)} T(f)$ denotes the timed fragment set and $\cup_{\beta \in \beta(r)} T'(\beta)$ denotes the timed arc set in r.

We extend $\beta(r)$ to include all forward empty arcs linking non adjacent fragments in the subpath r. Let $\beta^c(r)$ represent a transitive tournament, the set of empty arcs β that connect fragments in r in a forward direction. We need to use a portion of r that is feasible. Let $\bar{r} = r \setminus f_m$ be the feasible portion of r without the last fragment. Also, let $\hat{r} = r \setminus f_1$ be the feasible portion of r without the first fragment. Let $\beta^c(\bar{r})$ represent a transitive tournament, the set of empty arcs β that connect the fragments in $\bar{r}$. Also, let $\beta^c(\hat{r})$ represent a transitive tournament, the set of empty arcs β that connect the fragments in $\hat{r}$.

We can strengthen the lazy constraints (3.6) used to cut off the sequence of fragments corresponding to the smallest portion of the chain that is time window infeasible in a path R as follows:

- Augment the first (last) fragment in the chain with all alternative first (last) fragments that make the chain infeasible.
- Instead of first (last) fragments that make the chain infeasible, we consider alternative first (last) fragments that ensure the chain is feasible.
- Using the set $\beta^c(r)$ instead of $\beta(r)$.
- Replace the set $f(r)$ by the *minimal-matching* set $M(f(r))$.

To illustrate, Figure 5.1(a) depicts an infeasible subpath r contains three fragments ($|f(r)| = 3$) and two empty arcs ($|\beta(r)| = 2$). To eliminate the subpath r, it is sufficient to use the constraint (3.6) as it represented in Figure 5.1(a). This constraint can be further lifted by augmenting the so called tournament constraints (see, e.g., Ascheuer, Fischetti, and Grötschel 2000) as depicted in Figures 5.1(b) and 5.2(b). In addition, further lifting can be achieved by augmenting similar concepts to those of *outfork* and *infork* inequalities that introduced by Ropke, Cordeau, and Laporte (2007) as depicted in Figures 5.1(c) and 5.2(c) respectively. Finally, tighter conditions can be achieve by looking at options for feasible fragments at the end or start of the chain as depicted in Figures 5.1(d) and 5.2(d) respectively.

Moreover, there are special cases when f_m or f_1 consists of a single request. The last rule can be utilized to check for feasible empty arcs at the end or start of the chain as depicted in Figures 5.1(e) and 5.2(e) respectively. Note that the subpaths r in figures 5.1(e) and 5.2(e) contain two fragments.

We proposed infeasible paths cuts that replace the lazy constraints introduced in the previous chapters as follows:

5.3.1. End Subpath Cuts

Let $\overrightarrow{\Phi}(f(\bar{r})) = \{f \in \Phi \colon (f^- = f_m^-) \wedge ([f(\bar{r}) \oplus f] \text{ is feasible})\}$ denote the set of all fragments $f \in \Phi$ that can follow $f(\bar{r})$, such that $(f_1, ..., f_{m-1}, f)$ are node-disjoint fragments. The following constraints are valid for the PDPTW:

$$\sum_{\alpha \in \cup_{\beta \in \beta^c(r)} T'(\beta)} y_\alpha + \sum_{\omega \in \cup_{f \in M(f(\bar{r}))} T(f)} x_\omega \leq 2|f(r)| - 3 + \sum_{\omega \in \cup_{f \in \overrightarrow{\Phi}(f(\bar{r}))} T(f)} x_\omega \tag{5.11}$$

5.3.1.1 Last Fragment Consists of Single Request

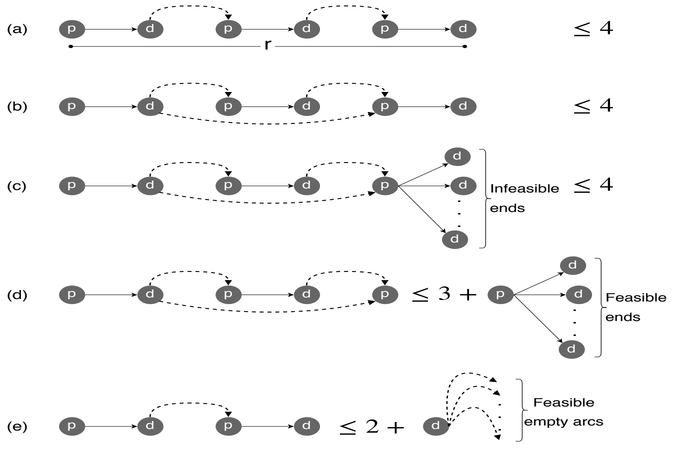
Let $\overrightarrow{\mathscr{A}}(f(\bar{r})) = \{\alpha \in \mathscr{A} \colon (\alpha^- = f_{m-1}^+) \wedge ([f(\bar{r}) \oplus \alpha] \text{ is feasible})\}$ denote the set of arcs $\alpha \in \mathscr{A}$ that can follow $f(\bar{r})$. The following constraints are valid for the PDPTW:

$$\sum_{\alpha \in \cup_{\beta \in \beta^c(\bar{r})} T'(\beta)} y_\alpha + \sum_{\omega \in \cup_{f \in M(f(\bar{r}))} T(f)} x_\omega \leq 2|f(r)| - 4 + \sum_{\alpha \in \cup_{\beta \in \overleftarrow{\mathscr{A}}(f(\bar{r}))} T'(\beta)} y_\alpha + \sum_{\forall \alpha^+ = 0, \alpha^- = f_{m-1}^+} y_\alpha \tag{5.12}$$

5.3.2. Start Subpath Cuts

Let $\overleftarrow{\Phi}(f(\hat{r})) = \{f \in \Phi \colon (f^+ = f_1^+) \wedge ([f \oplus f(\hat{r})] \text{ is feasible})\}$denote the set of fragments $f \in \Phi$ that can precede $f(\hat{r})$, such that $(f, f_2, ..., f_m)$ are node-disjoint fragments. The following constraints are

Figure 5.1. End Subpath Cuts



valid for the PDPTW:

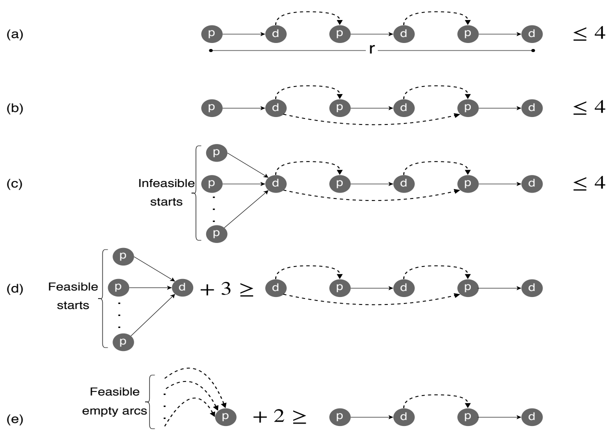
$$\sum_{\alpha\in\cup_{\beta\in\beta^c(r)}T'(\beta)} y_\alpha + \sum_{\omega\in\cup_{f\in M(f(\hat{r}))}T(f)} x_\omega \le 2|f(r)|-3+\sum_{\omega\in\cup_{f\in\overleftarrow{\Phi}(f(\hat{r}))}T(f)} x_\omega \tag{5.13}$$

5.3.2.1 First Fragment Consists of Single Request

Let $\overleftarrow{\mathscr{A}}(f(\hat{r})) = \{\alpha\in\mathscr{A} : (\alpha^+ = f_2^-)\wedge([\alpha\oplus f(\hat{r})]$ is feasible$)\}$ denote the set of arcs $\alpha\in\mathscr{A}$ that can precede $f(\hat{r})$. The following constraints are valid for the PDPTW:

$$\sum_{\alpha\in\cup_{\beta\in\beta^c(\hat{r})}T'(\beta)} y_\alpha + \sum_{\omega\in\cup_{f\in M(f(\hat{r}))}T(f)} x_\omega \le 2|f(r)|-4+\sum_{\alpha\in\cup_{\beta\in\overleftarrow{\mathscr{A}}(f(\hat{r}))}T'(\beta)} y_\alpha + \sum_{\forall\alpha^-=0,\alpha^+=f_2^-} y_\alpha \tag{5.14}$$

Figure 5.2. Start Subpath Cuts



5.4. Bound on the Number of Vehicles

Similarly to what we present in section 3.3.4, we solve the model (3.1)–(3.3) to minimize the sum of fixed costs (vehicles costs) by setting the objective coefficients of all variables to zero except the *start* arcs and relax the constraints (3.4)–(3.7). We also apply the inequalities (5.2)–(5.10) to generate the first set of cuts.

5.5. Overall Solution Process

We use a number of acceleration techniques in the overall solution process which can involve solving multiple LPs and IPs. We solve LPs to bound the number of vehicles required and to generate valid inequalities. We solve IPs to find solutions to the problem with a specified target number of vehicles. Every time we solve an LP we repeatedly apply the inequalities (5.2)–(5.10) until no more are found or the LP is infeasible. Every time we solve an IP we store any infeasible path cuts generated in the callback function and when that IP solve is finished we add them to the model permanently, as by default the cuts generated in a callback are only added to the model for the duration of the IP solve.

We initially set the objective function to minimize the number of vehicles used. We solve the LP and set the target number of vehicles as in Section 5.4. We restore the original objective function and iterate over increasing values of v until a feasible solution is found.

At each iteration we solve the LP and add valid inequalities. Let Z_{LB} denote the updated lower bound obtained when we solve the LP model. Also, let $\iota = 0.05 \times Z_{LB}$ denote an initial tolerance for the gap between the optimal IP and LP solutions. We solve an IP restricted to those variables whose reduced cost at the LP optimal solution is no more than ι. We stop if the optimal solution of the IP is no more than $Z_{LB} + \iota$. Otherwise we solve the IP with no restrictions on the variables used and stop unless this IP has no feasible solutions.

Note that setting variables to zero based on their reduced cost takes advantage of the following well known proposition (see Desaulniers, Gschwind, and Irnich 2020).

Proposition 1. Assume $\mathscr{M}$ is a minimization integer programming problem with variables x_u for $u \in \mathscr{U}$ where $\mathscr{U}$ is the index set. Let UB be an upper bound on the optimal value of problem $\mathscr{M}$ and let $\mathscr{D}$ be a dual solution to the linear relaxation of $\mathscr{M}$ providing a lower bound $LB(\mathscr{D})$. If an integer variable x_u for some $u \in \mathscr{U}$ having reduced cost $\bar{c}_u(\mathscr{D})$ with respect to $\mathscr{D}$ fulfills $\bar{c}_u(\mathscr{D}) > UB - LB(\mathscr{D})$, then $x_u = 0$ holds true for every optimal solution to $\mathscr{M}$.

Looking for a solution within 5% of the lower bound was found to be effective both in terms of the number of variables set to zero and the likelihood of finding a solution in that gap and avoiding solving another IP.

In summary, the overall solution process can be described in the following algorithm steps.

Algorithm 3 Pseudo Code for Overall Solution Process

```
Set the objective function to minimize the number of vehicles and calculate $v$
Solve an LP applying the inequalities (5.2)–(5.10) to the model
Restore the real objective function
loop
    Solve an LP applying the inequalities (5.2)–(5.10) to the model
    Set $\iota = 0.05 \times Z_{LB}$
    Remove variables with reduced cost $\geq \iota$ from the model
    Solve the IP
    if the IP is feasible and has optimal solution $\leq Z_{LB} + \iota$ then
        Exit loop with optimal solution
    Restore all variables to the model
    Add currently generated infeasible paths cuts (5.11)–(5.14) to the model
    Solve the IP
    if the IP is feasible then
        Exit loop with optimal solution
    Add currently generated infeasible paths cuts (5.11)–(5.14)
    Set $v = v + 1$
```

5.6. Computational Results

All experiments were performed on a computer equipped with Intel Core i7-6700 processor (3.4 GHz) running the Window 10 operating system. Gurobi 7.0.2 was used as an IP solver with Python 3.5.

To assess the impact of using the proposed cuts and the valid inequalities we test the same instances used in Chapter 3. We apply the fragments dominance rules that introduced in Section 4.4 when generating the fragments. We set $\delta = 5$ for all instances.

Table 5.1 shows the computational results for the AA, BB, CC, and DD instance groups, and Table 5.2 shows the computational results for the AA* and BB* instance groups, and their columns represent the following:

- *Name*: the instances name;
- F_N: the total number of non-dominated fragments;
- TF_N: the total number of non-dominated timed fragments;
- Z_R: the lower bound at the root node after valid inequalities are added;
- Z_{LB}: the best lower bound (LB relaxation) at a child node if the instance was not solved to optimality within the time limit, left blank otherwise;
- *Nodes*: the number of nodes that have been explored in the branch-and-bound tree;
- *LC*: the number of lazy constraints added;
- Δ_{Sec}: The difference in the total computing time in seconds, compared to the results reported in Chapter 3, computed as $Sec_W - Sec_O$, where Sec_W and Sec_O are the total time with and without valid inequalities, respectively;
- *VI*: the total number of valid inequalities added;
- Z^*: the optimal solution obtained, otherwise we report, in bold, the best upper bound (feasible solution) obtained or left blank if no feasible solution found within the 3600 seconds time limit.

Table 5.1. Computational Results ($\delta = 5$)

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	Δ_{Sec}	*VI*	Z*
AA30	171	448	31,129.13		0	0	-0.4	3	31,129.13
AA35	275	661	31,291.82		0	0	0.1	11	31,293.74
AA40	346	795	41,348.80		0	0	0.1	2	41,348.80
AA45	578	1,121	41,520.87		0	0	0.1	5	41,520.87
AA50	715	1,377	41,643.10		0	0	0.0	10	41,643.10
AA55	879	1,670	51,742.52		0	0	0.1	16	51,742.52
AA60	955	1,835	51,944.82		0	0	0.0	24	51,949.04
AA65	1,058	2,024	52,075.32		0	2	1.2	42	52,076.67
AA70	1,384	2,543	52,218.04		0	0	0.7	42	52,218.44
AA75	2,159	3,676	52,328.94		0	0	-0.3	58	52,329.30
AA80	1,719	3,159	62,511.90		451	42	-5.9	49	62,541.33
AA85	2,364	4,212	72,694.59		0	6	-3.0	26	72,703.23
AA90	2,380	4,619	63,097.73		4,757	64	-461.9	120	72,924.18
AA95	2,953	5,452	63,147.27		66,052	286	163.4	96	63,260.65
AA100	2,644	4,880	73,382.41		1,021	44	-7.8	69	73,424.04
AA105	3,729	6,530	83,481.70		1,196	36	-28.8	112	83,522.13
AA110	3,444	6,066	83,252.56		437	30	4.0	57	83,268.80
AA115	4,451	7,696	93,648.44		1,748	90	-25.8	83	93,690.44
AA120	3,690	6,821	103,947.46		412	10	-4.0	45	103,963.13
AA125	4,600	8,198	84,063.39		422	32	-115.7	194	93,840.61
BB30	144	329	31,074.08		0	2	0.0	2	31,077.12
BB35	253	526	31,311.97		0	0	0.0	3	31,311.97
BB40	325	682	41,393.60		0	4	0.3	8	41,403.49
BB45	501	997	41,517.63		441	18	0.2	27	41,536.99
BB50	660	1,303	41,787.55		0	2	-0.5	14	41,790.49
BB55	751	1,485	51,893.55		237	18	-0.6	36	51,911.05
BB60	702	1,402	62,246.49		16	14	0.1	2	62,304.78
BB65	761	1,532	62,466.90		54	22	-1.4	2	62,563.85
BB70	949	1,885	72,528.02		45	2	0.8	21	72,534.42
BB75	1,336	2,454	72,637.68		75	8	0.0	28	72,655.87
BB80	1,967	3,593	62,693.19		1,217	160	-22.7	86	62,743.40
BB85	2,207	3,964	62,653.41		4,457	38	-5.1	82	62,688.02
BB90	2,386	4,323	82,974.04		446	4	-1.4	20	82,985.83
BB95	2,816	5,287	63,131.81		9,864	60	-271.1	102	73,005.02
BB100	3,172	5,874	73,025.94		1,729	68	-20.9	88	73,059.83
BB105	7,251	12,275	72,985.89		6,176	68	-177.1	135	73,032.72

Continued Table 5.1 – *Computational Results ($\delta = 5$)*

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	Δ_{Sec}	*VI*	*Z**
BB110	4,764	8,019	73,447.34		17,190	306	-78.5	142	73,543.43
BB115	4,558	8,019	83,730.32		37,874	196	-41.3	85	83,812.37
BB120	8,425	13,890	73,875.01		7,289	262	42.2	145	83,700.24
BB125	5,983	10,165	93,827.82		8,259	94	-13.2	95	93,868.50
CC30	207	1,954	31,084.29		0	2	-1.3	0	31,088.20
CC35	359	2,889	31,233.61		0	0	-0.5	12	31,237.04
CC40	790	5,431	31,333.75		0	0	1.0	0	31,339.79
CC45	1,747	9,850	31,469.78		207,728	372	1070.3	14	31,532.09
CC50	2,745	13,848	41,652.97		4,719	40	-31.3	8	41,673.05
CC55	3,980	18,196	41,775.45		7,447	124	47.1	10	41,792.84
CC60	5,065	22,645	41,925.89		9,255	106	-714.4	22	41,946.63
CC65	7,270	30,472	42,039.61	42,060.17	27,276	1,248	0.0	22	
CC70	10,681	41,945	42,185.01	42,215.98	14,124	332	0.0	28	
CC75	14,516	55,990	42,346.66	42,394.41	10,733	136	0.0	32	
DD30	1,163	4,998	21,073.34		6,290	120	-1.3	38	21,102.82
DD35	2,312	9,244	21,197.54		26,285	142	-3174.2	35	31,127.43
DD40	3,794	14,505	31,243.02		0	0	-2.1	28	31,244.79
DD45	6,433	22,613	31,345.00		0	0	-1.2	41	31,349.87
DD50	11,108	37,571	31,438.47		141	6	0.2	52	31,449.70
DD55	16,727	54,302	31,591.04		32,203	230	144.8	89	31,637.14
DD60	25,413	78,073	31,753.81	31,775.45	10,209	640	0.0	110	
DD65	17,102	55,079	32,068.22		13,332	122	-1654.0	96	41,984.09
DD70	27,340	81,578	42,064.62		13,196	166	446.5	76	42,092.28
DD75	39,711	111,657	42,186.56		7,062	548	0.0	70	**42,271.50**

Table 5.2. Computational Results ($\delta = 5$)

Name	F_N	TF_N	Z_R	*Nodes*	*LC*	Δ_{Sec}	*VI*	*Z**
AA*30	263	263	31,051.08	0	0	0.0	12	31,051.20
AA*35	386	386	31,237.01	0	0	0.0	12	31,244.37
AA*40	532	532	41,330.99	0	0	0.0	18	41,330.99
AA*45	820	821	41,509.24	0	2	0.4	41	41,514.90
AA*50	996	997	41,619.29	0	4	0.1	40	41,636.97
AA*55	1,195	1,196	41,854.49	49	0	-0.4	31	41,879.62
AA*60	1,288	1,289	51,805.06	0	2	-0.6	25	51,807.57
AA*65	1,424	1,425	51,955.23	0	8	-0.2	51	51,961.24
AA*70	1,802	1,804	52,150.22	210	14	-2.4	84	52,170.77

Continued Table 5.2 – *Computational Results* ($\delta = 5$)

Name	F_N	TF_N	Z_R	*Nodes*	*LC*	Δ_{Sec}	*VI*	*Z**
AA*75	2,770	2,772	52,268.65	259	24	-4.6	123	52,298.84
AA*80	2,381	2,382	62,520.20	515	34	-15.7	94	62,588.19
AA*85	3,283	3,290	72,582.55	5,912	42	-2.1	93	72,633.39
AA*90	3,232	3,237	72,860.60	4,562	84	-101.0	99	72,923.07
AA*95	4,111	4,120	63,402.24	9,588	18	-0.8	164	72,996.92
AA*100	3,816	3,819	83,240.86	332	14	1.4	188	83,254.93
AA*105	5,121	5,127	83,275.42	6,207	58	-26.1	199	83,349.25
AA*110	5,044	5,051	83,176.07	6,136	72	-69.3	109	83,232.45
AA*115	6,709	6,719	93,511.08	885	34	-165.7	172	93,577.39
AA*120	5,443	5,449	93,818.84	3,037	106	-363.3	148	93,899.67
AA*125	6,832	6,838	83,725.00	103,249	152	-897.5	237	83,844.43
BB*30	166	166	41,110.65	0	0	0.0	10	41,110.65
BB*35	290	290	41,330.00	0	0	0.0	24	41,332.47
BB*40	363	363	41,466.37	0	2	0.1	19	41,476.65
BB*45	566	566	41,605.29	83	18	-1.8	39	41,698.98
BB*50	739	740	51,688.09	112	4	0.9	46	51,718.58
BB*55	883	884	51,926.19	6,461	92	17.9	67	52,033.90
BB*60	968	969	72,171.99	0	2	-1.2	47	72,183.56
BB*65	1,054	1,055	72,354.29	299	18	0.3	69	72,393.77
BB*70	1,310	1,312	72,544.85	438	20	-2.3	61	72,604.07
BB*75	1,768	1,770	72,663.60	340	8	-8.9	71	72,746.85
BB*80	2,489	2,495	62,750.58	5,789	16	-368.4	120	72,556.05
BB*85	2,609	2,616	62,633.49	3,176	40	-45.9	140	62,718.69
BB*90	3,121	3,130	72,960.45	5,706	40	-91.9	124	73,075.64
BB*95	3,581	3,586	72,975.00	267	10	-8.7	108	73,001.11
BB*100	4,146	4,153	73,115.82	2,240	76	-55.5	125	73,210.37
BB*105	8,746	8,760	73,019.37	40,881	166	-2,324.7	225	73,132.54
BB*110	5,737	5,745	83,455.29	27,676	94	-574.4	198	83,540.28
BB*115	6,343	6,358	93,564.80	5,256	48	-135.2	163	93,619.79
BB*120	10,234	10,248	83,553.58	52,575	456	0.0	217	**83,714.11**
BB*125	7,734	7,748	93,789.00	28,029	120	-2,775.7	300	93,883.89

Comparing to the results that reported in Chapter 3, we observe an improvement to the lower bound obtained for almost all test instances. Two instances DD35 and DD65 solved to optimality (setting $\delta = 5$) that were previously unsolved, and better feasible upper bounds were obtained for DD75 and BB*120. A significant improvement in the solution time can be seen for most test instances, specially the big AA* and BB* instance groups problems where the cuts and the valid inequalities make a huge difference. For example, the total computing time to solve AA*120 reduced from 447 seconds to only 84 seconds, and the total computing time to solve BB*125 reduced from 3,459 seconds to 684 seconds.

As mentioned in Section 3.5 all lazy constraints reported in all tables are of time window infeasible paths type. No cycles occurred in any instances.

In column VI of Tables 5.1 and 5.2 we provide the overall total number of the valid inequalities that have been added. To give some insight into the relative frequency of the different valid inequalities, we provide the number of the valid inequalities of each type that have been generated in the results of instance AA125. The number of valid inequalities of fragments after empty arcs, fragments before empty arcs, arcs after fragments, and arcs before fragments inequalities are 19, 19, 34 and 30, respectively. The number of valid inequalities of timed fragments after timed empty arcs, timed fragments before timed empty arcs, timed arcs after timed fragments, and timed arcs before timed fragments inequalities are 6, 1, 31 and 10, respectively. Finally, the number of valid inequalities of subset-row type are 44.

5.7. Conclusions

In this chapter, we propose new valid inequalities and infeasible paths cuts that leads to a better solution for the PDPTW. The computational results obtained in this chapter are a substantial improvement on the results reported in Chapter 3.

Chapter 6

Conclusions and Future Research

6.1. Summary of Work Undertaken

In this thesis, we proposed an efficient exact algorithm to solve the *Pickup and Delivery Problem with Time Window* (PDPTW) and its variants. We introduced new methodology and formulation for solving the problem. We generated fragments that can be chained together to form routes that satisfy the PDPTW. Our goal was to develop an alternative method to the classical and widely used method in the literature namely *Branch-and-Cut-and-Price* (BCP). The BCP method is based on a set-partitioning formulation that uses column generation to solve a continuous relaxation of the problem. Integrality should be obtained at the final stage using the branch-and-bound procedure. Typically, this method is very difficult to implement and adding even minor extensions to the original problem can break all solution components (cutting, pricing, and branching). Dealing with richer variants (more realistic problem constraints) of the PDPTW will make the solution harder. The extensive computational experiments that we performed have shown that the proposed method is efficient compared with the BCP method.

Moreover, adding more extensions (richer variants) will add more restrictions on the generated fragments and this will make the problem size smaller (fewer fragments) resulting in a faster solution.

In Chapter 3, we have utilized techniques such as *Time Window Discretization* and *Lazy Constraints* to construct an integer program formulation for a relaxed space-time network flow model to chain the fragments. In Chapter 4, we extend our algorithm presented in Chapter 3 to solve the classical PDPTWMS. We used a recursive procedure to implicitly enumerate all possible loading plans. Moreover, we introduced for the first time a new extension to the PDPTWMS. We call it PDPTWMS with Stack Splitting (PDPTWMS-S). In this extension the size of a customer's demand was allowed to exceed the stack capacity and be only limited to the vehicle's capacity, and can be split among all stacks. For the PDPTWMS-S we introduced an integer programming formulation to model the assignment of requests to stack to produce a feasible multi-stack loading plan.

In Chapter 5, we have introduced several valid inequalities that improved the lower bounds of the LP relaxations of the problem. Moreover, we introduced several infeasible paths cuts that significantly improved the solution time.

6.2. Future Work

For the fragments-based method a great deal of work remains. A mixture of this method and the BCP method may be advantageous. The crucial question is what if an enormous number of fragments must be generated. Further research could also look to the use of dynamic time discretization instead of fixed interval. In addition, considering using valid cuts could be very useful.

The possibilities for future work using our method in more practical PDPTW variants are numerous. For example, using our method for dynamic requests, especially with dial-a-ride problem.

Moreover, the flexibility in the process of generating the fragments for LIFO or FIFO loading would be of great benefit to industry practitioners for Automated Guided Vehicle (AGV) systems.

References

Alba Martínez M. A, Cordeau J-F, Dell'Amico M, and Iori M (2013) A branch-and-cut algorithm for the double traveling salesman problem with multiple stacks. *INFORMS Journal on Computing* 25(1):41–55.

Alyasiry A, Forbes M, Bulmer M (2019) An exact algorithm for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science* 53(6):1695–1705.

Australian Logistics Council (2016) *Annual report of Australian Logistics Council*. Accessed August, 2018, `http://austlogistics.com.au/wp-content/uploads/2016/03/ALC-Annual-Report-15-16.pdf`.

Archetti C, Speranza M.G. (2012) Vehicle routing problems with split deliveries. *International Transactions in Operational Research* 19: 3–22.

Ascheuer N, Fischetti M, Grötschel M (2000) A polyhedral study of the asymmetric traveling salesman problem with time windows. *Networks* 36(2):69–79.

Baldacci R, Bartolini E, Mingozzi A (2011) An exact algorithm for the pickup and delivery problem with time windows. *Operations Research* 59(2):414–426.

Battarra M, Cordeau J-F, Iori M (2014) Pickup-and-delivery problems for goods transportation. Toth P, Vigo D, eds. *Vehicle Routing: Problems, Methods, and Applications* (SIAM, Philadelphia), 161–191.

Bent R, Van Hentenryck P (2006) A two-stage hybrid algorithm for pickup and delivery vehicle routing problems with time windows. *Computers & Operations Research* 33(4):875–893.

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

Bomdorfer R (1998) Aspects of set packing, partitioning, and covering. Ph.D. thesis, Technical University of Berlin, Berlin.

Bureau of Infrastructure, Transport and Regional Economics (2015) *Traffic and congestion cost trends for Australian capital cities*. Accessed August, 2018, `https://bitre.gov.au/publications/2015/files/is_074.pdf`.

Carrabs F, Cerulli R, Cordeau J-F (2007) An additive branch-and-bound algorithm for the pickup and delivery traveling salesman problem with LIFO or FIFO loading. *INFOR* 45(4):223–238.

Carrabs F, Cerulli R, Speranza M (2013) A branch-and-bound algorithm for the double traveling salesman problem with two stacks. *Networks* 61(1):58–75.

Carrabs F, Cordeau J-F, Laporte G (2007) Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing* 19(4):618–632.

Casazza M, Ceselli A, Nunkesser M (2012) Efficient algorithms for the double traveling salesman problem with multiple stacks. *Computers & Operations Research* 39(5):1044–1053.

Cherkesly M (2015) Le problème de tournées de véhicules avec cueillettes, livraisons, fenêtes de temps en contraintes de manutention. Ph.D. thesis, École Polytechnique de Montréal, Montréal.

Cherkesly M (2017) Discussion via email of the use of rounding for distances and travel times, February 1.

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

Cherkesly M, Desaulniers G, Laporte G (2015) Branch-price-and-cut algorithms for the pickup and delivery problem with time windows and last-in-first-out loading. *Transportation Science* 49(4):752–766.

Commonwealth Scientific and Industrial Research Organisation (2018) *Logistics, supply chains, and transportation.* Accessed August, 2018, `https://research.csiro.au`.

Cordeau J-F (2006) A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research* 54(3):573–586.

Cordeau J-F, Dell'Amico M, Iori M (2010) Branch-and-cut for the pickup and delivery traveling salesman problem with FIFO loading. *Computers & Operations Research* 37(5):970–980.

Cordeau J-F, Iori M, Laporte G, Salazar González J (2010) A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with LIFO loading. *Networks* 55(1):46–59.

Cordeau J-F, Laporte G (2003) The dial-a-ride problem (DARP): Variants, modeling issues and algorithms. *Quart. J. Belgian French Italian Oper. Res. Soc.* 1(2):89–101.

Cordeau J-F, Laporte G, Ropke S (2008) Recent models and algorithms for one-to-one pickup and delivery problems. Golden B. L, Raghavan S, Wasil E. A, eds, *The Vehicle Routing Problem: Latest Advances and New Challenges* (Springer, New York), 327–357.

Cordeau J-F, Laporte G, Savelsbergh M, Vigo D (2007) Vehicle Routing. Barnhart C, Laporte G, eds, *Transportation, Handbooks in Operations Research and Management Science* (Elsevier, Netherlands) 367–428.

Côté J-F, Archetti C, Speranza M, Gendreau M, Potvin J-Y (2012) A branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks. *Networks* 60(4):212–226.

Côté J-F, Gendreau M, Potvin J-Y (2012) Large neighborhood search for the pickup and delivery traveling salesman problem with multiple stacks. *Networks* 60(1):19–30.

Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America* 2(4):393–410.

Dantzig G, Ramser J (1959) The truck dispatching problem. *Management Science* 6(1):80–91.

Desaulniers G, Gschwind T, Irnich S (2020) Variable fixing for two-arc sequences in branch-price-and-cut algorithms on path-based models. *Transportation Science*, Published online in Articles in Advance 4 June. 2020. https://doi.org/10.1287/trsc.2020.0988.

Desrosiers J, Dumas Y, Soumis F (1988) The multiple vehicle dial-a-ride problem. Daduna JR, Wren A, eds. *Computer-Aided Transit Scheduling. Lecture Notes in Economics and Mathematical Systems, vol. 308* (Springer, Berlin), 15–27.

Desrosiers J, Dumas Y, Soumis F, Taillefer S, Villeneuve D (1991) An algorithm for mini-clustering in handicapped transport. Working paper, *Les Cahiers du GERAD, HEC Montréal.*

Dumas Y, Desrosiers J, Soumis F (1991) The pickup and delivery problem with time windows. *Eur. J. Oper. Res.* 54(1):7–22.

Dumitrescu I, Ropke S, Cordeau J-F, Laporte G (2010) The traveling salesman problem with pickup and delivery: polyhedral results and a branch-and-cut algorithm. *Mathematical Programming* 121(2):269–305.

Erdoğan G, Cordeau J-F, Laporte G (2009) The pickup and delivery traveling salesman problem with first-in-first-out loading. *Computers & Operations Research* 36(6):1800–1808.

Felipe A, Ortuño M, Tirado G (2009) New neighborhood structures for the double traveling salesman problem with multiple stacks. *Top* 17(1):190–213.

Flood M (1956) The traveling-salesman problem. *Operations Research* 4(1):61–75.

Frederickson G, Hecht M, Kim C (1976) Approximation algorithms for some routing problems. *17th Annual Symposium on Foundations of Computer Science*(Houston, Texas), 216–227.

Garey M, Johnson D (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness.* (W. H. Freeman and Company, New York).

Gurobi Optimization (2017) *Gurobi Optimizer Reference Manual*, version 7.0. `https://www.gurobi.com/documentation/7.0/refman/index.html`.

Ioachim I, Desrosiers J, Dumas Y, Solomon M, Villeneuve D (1995) A request clustering algorithm for door-to-door handicapped transportation. *Transportation Sci.* 29(1):63–78.

Iori M, Martello S (2010) Routing problems with loading constraints. *TOP* 18(1):4–27.

Iori M, Riera-Ledesma J (2015) Exact algorithms for the double vehicle routing problem with multiple stacks. *Computers & Operations Research* 63:83–101.

Jepsen M, Petersen B, Spoorendonk S, Pisinger D (2008) *Subset-row inequalities applied to the vehicle-routing problem with time windows*. *Operations Research* 56(2):497–511.

Johnson DB (1975) Finding all the elementary circuits of a directed graph. *SIAM J. Comput.* 4(1):77–84

Kalantari B, Hill A, Arora S (1985) An algorithm for the traveling salesman problem with pickup and delivery customers. *European Journal of Operational Research* 22(3):377 – 386.

Lau H, Liang Z (2002) Pickup and delivery with time windows: Algorithms and test case generation. *International Journal on Artificial Intelligence Tools* 11(03):455–472.

Lenstra J, Rinnooy Kan A (1981) Complexity of vehicle routing and scheduling problems. *Networks* 11(2):221–227.

Li H, Lim A (2003) A metaheuristic for the pickup and delivery problem with time windows. *International Journal on Artificial Intelligence Tools* 12(02):173–186.

Li Y, Lim A, Oon W, Qin H, Tu D (2011) The tree representation for the pickup and delivery traveling salesman problem with LIFO loading. *European Journal of Operational Research* 212(3):482–496.

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

Little J, Murty K, Sweeney D, Karel C (1963) An algorithm for the traveling salesman problem. *Operations Research* 11(6):972–989.

Lu Q, Dessouky M (2004) An exact algorithm for the multiple vehicle pickup and delivery problem. *Transportation Science* 38(4):503–514.

Lu Q, Dessouky M (2006) A new insertion-based construction heuristic for solving the pickup and delivery problem with time windows. *European Journal of Operational Research* 175(2):672–687.

Lusby R, Larsen J, Ehrgott M, Ryan D (2010) An exact method for the double TSP with multiple stacks. *International Transactions in Operational Research* 17(5):637–652.

Nanry W, Barnes J (2000) Solving the pickup and delivery problem with time windows using reactive tabu search. *Transportation Research Part B: Methodological* 34(2):107–121.

Nemhauser G, Wolsey L (1999) *Integer and Combinatorial Optimization*. Wiley-Interscience series in discrete mathematics and optimization (Wiley-Interscience publication, New York).

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

Papadimitriou C (1977) The euclidean travelling salesman problem is np-complete. *Theoretical Computer Science* 4(3):237 – 244.

Parragh SN, Doerner KF, Hartl RF (2008a) A survey on pickup and delivery problems, part I: Transportation between customers and depot. *J. für Betriebswirtschaft* 58(1):21–51.

Parragh SN, Doerner KF, Hartl RF (2008b) A survey on pickup and delivery problems, part II: Transportation between pickup and delivery locations. *J. für Betriebswirtschaft* 58(2):81–117.

Pereira A, Urrutia S (2018) Formulations and algorithms for the pickup and delivery traveling salesman problem with multiple stacks. *Computers & Operations Research* 93:1–14.

Petersen H, Archetti C, Speranza M (2010) Exact solutions to the double travelling salesman problem with multiple stacks. *Networks* 56(4):229–243.

Petersen H, Madsen O (2009) The double travelling salesman problem with multiple stacks–formulation and heuristic solution approaches. *European Journal of Operational Research* 198(1):139–147.

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

Pollaris H, Braekers K, Caris A, Janssens G.K., Limbourg S (2015) Vehicle routing problems with loading constraints: state-of-the-art and future directions. *OR Spectrum* 37(2):297–330.

Renaud J, Boctor F, Laporte G (1996) A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing* 8(2):134–143.

Renaud J, Boctor F, Laporte G (2002) Perturbation heuristics for the pickup and delivery traveling salesman problem. *Computers & Operations Research* 29(9):1129–1141.

Renaud J, Boctor F, Ouenniche J (2000) A heuristic for the pickup and delivery traveling salesman problem. *Computers & Operations Research* 27(9):905–916.

Ropke S ,Cordeau J-F (2009) Branch-and-cut-and-price for the pickup and delivery problem with time windows. *Transportation Science* 43(3):267–286.

Ropke S, Cordeau J-F, Laporte G (2007) Models and branch-and-cut algorithms for pickup and delivery problems with time windows. *Networks* 49(4):258–272.

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.

Ruland K, Rodin E (1997) The pickup and delivery problem: Faces and branch-and-cut algorithm. *Computers and Mathematics with Applications* 33(12):1–13.

Sampaio A, Urrutia S (2017) New formulation and branch-and-cut algorithm for the pickup and delivery traveling salesman problem with multiple stacks. *International Transactions in Operational Research* 24(1-2):77–98.

Savelsbergh M (1985) Local search in routing problems with time windows. *Annals of Operations Research* 4(1):285–305.

Savelsbergh M, Sol M (1998) Drive: Dynamic routing of independent vehicles. *Operations Research* 46(4):474–490.

Schrijver A (2005) On the history of combinatorial optimization (till 1960). *Handbooks in operations research and management science* 12:1–68.

Sedgewick R (2001) *Algorithms in C, Part 5: Graph Algorithms*, 3rd ed. (Addison Wesley Professional, Reading, MA).

Shaw P (1997) A new local search algorithm providing high quality solutions to vehicle routing problems. Technical report, Department of Computer Science, University of Strathclyde, Glasgow, Scotland, UK.

Szwarcfiter JL, Lauer PE (1976) A search strategy for the elementary cycles of a directed graph. *BIT Numer. Math.* 16(2):192–204.

Toth P, Vigo D (2014) *Vehicle Routing: Problems, Methods, and Applications*, 2nd ed. (SIAM, Philadelphia).

U.S. Environmental Protection Agency (2016) *U.S. Greenhouse gas inventory report: 1990-2014*. Accessed August, 2018, `https://www.epa.gov/ghgemissions/us-greenhouse-gas-inventory-report-1990-2014`.

Van Der Bruggen L, Lenstra J, Schuur P (1993) Variable-depth search for the single-vehicle pickup and delivery problem with time windows. *Transportation Science* 27(3):298–311.

Wolsey L (1998) *Integer Programming*. Wiley-Interscience series in discrete mathematics and optimization (Wiley-Interscience publication, New York).

Appendix A

Appendix A: The PDPTWL Computational Results

A.1. The PDPTWL Computational Results Setting $\delta = 1$

To evaluate the impact of the parameter δ we provide computational results setting $\delta = 1$ and time limit to 3,600 seconds. The columns of tables A.1, A.2, A.3 and A.4 represent the following: *Name*: the instances name; Z_R: the lower bound at the root node (i.e., the LP relaxation of PDPTWL-R augmented with the integer lower bound on the number of vehicles); *Sec.*: the total computing time in seconds; Z^*: the optimal solution obtained, otherwise we report, in bold, the best upper bound (feasible solution) obtained or left blank if no feasible solution found within the time limit; Z_{LB}: the best lower bound (LB relaxation) at a child node if the instance was not solved to optimality within the time limit, left blank otherwise; F_N: the total number of original fragments; TF_N: the total number of undominated timed fragments; *Nodes*: the number of nodes that have been explored in the branch and bound tree; L_C: the total number of lazy constraints used in any solution.

Table A.1. Computational Results ($\delta = 1$)

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	Z*
AA30	176	1,782	31,129.13		0	0	1.3	31,129.13
AA35	294	2,690	31,284.77		0	0	1.2	31,293.74
AA40	368	3,159	41,348.80		0	0	1.0	41,348.80
AA45	623	4,115	41,520.87		0	0	1.5	41,520.87
AA50	768	5,112	41,643.10		0	0	2.1	41,643.10
AA55	935	6,160	51,742.52		0	0	2.6	51,742.52
AA60	1,015	6,759	51,927.76		0	0	6.8	51,949.04
AA65	1,118	7,464	52,059.30		0	1	6.9	52,076.67
AA70	1,469	9,151	52,210.63		0	0	7.2	52,218.44
AA75	2,498	13,337	52,326.92		0	0	11.4	52,329.30
BB30	145	1,158	31,074.09		0	1	1.2	31,077.12
BB35	267	1,936	31,310.77		0	1	1.0	31,311.97
BB40	342	2,564	41,397.36		0	0	2.4	41,403.49
BB45	540	3,780	41,519.72		90	0	6.8	41,536.99
BB50	706	4,925	41,774.53		0	1	3.5	41,790.49
BB55	799	5,565	51,894.39		178	1	11.2	51,911.05
BB60	742	5,143	62,280.82		0	3	4.0	62,304.78
BB65	801	5,606	62,514.38		13	2	9.8	62,563.85
BB70	997	6,855	72,528.06		0	0	15.4	72,534.42
BB75	1,446	8,805	72,644.69		34	0	22.2	72,655.87
CC30	241	9,814	31,086.67		0	0	14.2	31,088.20
CC35	438	14,833	31,232.69		0	0	24.9	31,237.04
CC40	1,021	28,693	31,335.62		82	0	93.8	31,339.79
CC45	2,359	53,865	31,487.28	31,525.87	23,735	13	3,600.0	**31,532.09**
CC50	3,813	76,858	41,662.26		1,172	1	1,113.6	41,673.05
CC55	5,294	98,541	41,776.14		2,056	4	1,267.4	41,792.84
CC60	6,589	120,878	41,934.61		2,163	0	1,280.6	41,946.63
CC65	9,243	160,111	42,056.52	42,058.98	2,749	0		
CC70	13,888	221,355	42,243.43	42,244.32	1,433	0		
CC75	19,480	302,256	52,283.07	52,294.88	1,951	5	3,600.0	**52,316.09**
DD30	1,437	25,383	21,079.16		2,464	1	406.7	21,102.82
DD35	3,052	49,033	21,340.89		31	0	123.3	31,127.43
DD40	5,196	79,021	31,242.14		0	0	23.9	31,244.79
DD45	8,940	124,040	31,344.15		0	0	50.1	31,349.87
DD50	16,064	211,385	31,438.30		119	2	194.4	31,449.70
DD55	25,271	316,750	31,590.32	31,596.03	1,440	11		
DD60	39,687	466,993	31,768.22	31,778.29	1,031	4	3,600.0	**31,814.13**
DD65	25,579	306,298	41,967.35		446	0	1,381.1	41,984.09
DD70	40,855	451,227	42,067.83		1,744	2	2,387.4	42,092.28
DD75	62,468	640,506	42,190.34	42,190.34	172	2		

Table A.2. Computational Results ($\delta = 1$)

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	Z*
AA*30	271	287	31,051.08		0	0	1.2	31,051.20
AA*35	415	448	31,231.30		0	0	0.9	31,244.37
AA*40	569	604	41,318.38		0	0	0.9	41,330.99
AA*45	882	922	41,505.21		0	0	2.1	41,514.90
AA*50	1,073	1,124	41,612.97		25	0	2.3	41,636.97
AA*55	1,283	1,342	41,842.35		0	0	3.9	41,879.62
AA*60	1,381	1,447	51,807.57		0	0	2.3	51,807.57
AA*65	1,522	1,591	51,952.25		0	1	3.9	51,961.24
AA*70	1,939	2,027	52,137.83		268	3	10.0	52,170.77
AA*75	3,116	3,217	52,265.70		311	6	14.5	52,298.84
BB*30	169	180	41,105.43		0	0	0.6	41,110.65
BB*35	307	327	41,329.09		11	1	1.0	41,332.47
BB*40	381	404	41,463.74		0	0	1.4	41,476.65
BB*45	596	629	41,645.10		21	2	2.3	41,698.98
BB*50	773	814	51,681.97		133	0	3.4	51,718.58
BB*55	926	971	51,937.74		1,395	5	16.8	52,033.90
BB*60	1,025	1,070	72,168.09		0	2	4.0	72,183.56
BB*65	1,111	1,158	72,346.54		174	3	5.0	72,393.77
BB*70	1,383	1,444	72,555.38		175	2	7.4	72,604.07
BB*75	1,930	1,997	72,675.02		180	6	10.6	72,746.85

Table A.3. Computational Results ($\delta = 1$)

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	Z*
AA80	1,826	11,260	62,520.77		134	3	27.4	62,541.33
AA85	2,508	15,060	72,696.41		31	0	24.9	72,703.23
AA90	2,492	17,136	63,154.47		1,422	9	192.3	72,924.18
AA95	3,117	19,874	63,147.64		7,061	37	1,171.0	63,260.65
AA100	2,747	17,529	73,384.93		1,436	10	153.2	73,424.04
AA105	4,194	23,189	83,495.18		1,323	0	128.5	83,522.13
AA110	3,561	20,724	83,253.67		436	3	140.1	83,268.80
AA115	4,729	26,380	93,664.54		354	0	89.8	93,690.44
AA120	3,790	24,368	103,951.74		232	5	61.2	103,963.13
AA125	4,829	29,041	93,827.53		441	3	116.2	93,840.61
BB80	2,356	14,143	62,709.36		1,210	7	240.5	62,743.40
BB85	2,372	14,328	62,651.09		1,396	2	159.1	62,688.02
BB90	2,773	16,251	82,969.78		341	1	36.7	82,985.83
BB95	3,016	19,509	63,237.19		1,153	3	266.8	73,005.02
BB100	3,334	21,613	73,026.31		761	2	112.7	73,059.83
BB105	8,852	47,421	72,979.73		3,669	20	858.4	73,032.72
BB110	5,087	27,332	73,463.63		2,970	5	722.3	73,543.43
BB115	4,811	28,277	83,750.81		3,255	25	1,086.8	83,812.37
BB120	9,500	48,836	83,650.85		2,631	14	1,335.4	83,700.24
BB125	6,480	36,196	93,825.12		2,258	3	606.4	93,868.50

Table A.4. Computational Results ($\delta = 1$)

Name	F_N	TF_N	Z_R	Z_{LB}	*Nodes*	*LC*	*Sec.*	Z*
AA*80	2,531	2,617	62,549.18		616	10	33.9	62,588.19
AA*85	3,565	3,700	72,589.79		2,385	3	40.1	72,633.39
AA*90	3,413	3,681	72,869.41		1,237	23	124.9	72,923.07
AA*95	4,447	4,673	72,923.12		2,072	5	175.7	72,996.92
AA*100	4,076	4,234	83,240.77		30	0	17.9	83,254.93
AA*105	5,638	5,874	83,270.80		1,351	12	101.9	83,349.25
AA*110	5,315	5,575	83,173.48		1,436	7	103.0	83,232.45
AA*115	7,171	7,445	93,511.66		889	9	124.1	93,577.39
AA*120	5,668	5,927	93,857.13		1,173	4	120.2	93,899.67
AA*125	7,273	7,530	83,738.36		7,254	13	926.8	83,844.43
BB*80	2,803	2,967	62,803.09		466	4	36.8	72,556.05
BB*85	2,769	2,892	62,633.73		1,332	3	52.6	62,718.69
BB*90	3,439	3,596	72,970.80		806	3	68.5	73,075.64
BB*95	3,856	4,081	72,972.30		511	0	30.1	73,001.11
BB*100	4,385	4,621	73,121.12		500	4	64.4	73,210.37
BB*105	10,206	10,537	73,014.55		5,450	21	838.2	73,132.54
BB*110	6,110	6,381	83,421.67		6,940	35	737.9	83,540.28
BB*115	6,699	6,944	93,581.56		1,135	0	107.5	93,619.79
BB*120	11,417	11,984	83,564.85	83,686.01	14,864	14	3,600.0	**83,714.11**
BB*125	8,334	8,682	93,783.97		3,946	8	473.6	93,883.89

A.2. The Characteristics of the PDPTW Instances

Here we describe the characteristics of the PDPTW instances of Ropke and Cordeau (2009). In these instances the objective is to first minimize the number of vehicles, and then the total distance. Therefore, a relatively large fixed cost of 10,000 is imposed on each route. Four groups of instances were generated with up to 75 requests (n). These groups are called AA, BB, CC, and DD. Each group has 10 instances, with n varying from 30 to 75. Each group has different values of time windows $\mathscr{W}$, and vehicle capacity Q. The time windows for groups AA and BB were set to $\mathscr{W} = 60$, and for CC and DD were set to $\mathscr{W} = 120$. The vehicle capacity for groups AA and CC were set to $Q = 15$, and for BB and DD were set to $Q = 20$.

The coordinates of each pickup and delivery location were randomly chosen according to a uniform distribution over a $[0,50] \times [0,50]$ square, with a single depot located at the middle of the square. The demand q_i of request i is selected randomly from the interval $[5,Q]$. The planning horizon is set to $L = 600$, and the time windows were randomly generated by firstly choosing e_i in the interval of $[0, L - t_{i,n+i}]$, and then setting $l_i = e_i + \mathscr{W}$, $e_{n+i} = ei + t_{i,n+i}$ and $l_{n+i} = e_{n+i} + \mathscr{W}$.

Appendix B

Appendix B: The PDPTWMS and the PDPTWMS-S Computational Results

B.1. The PDPTWMS Computational Results

In our tests all instances defined for the PDPTWMS (1914 instances) were solved to optimality except seven of a280-c1 type. Feasible upper bounds were obtained for three of them a280-c1-131-w45-1500-2000 and a280-c1-141-w45-1500-2000 of two stacks and a280-c1-141-w30-1500-2000 of three stacks. These upper bounds are reported in bold in Z^* column. No feasible solution was found within the time limit for the instances a280-c1-121-w45-1500-2000, a280-c1-131-w45-1500-2000, a280-c1-141-w45-1500-2000, and a280-c1-151-w45-1500-20001 of 3 stacks.

The columns in Tables B.1–B.6 represent the following. *Instance*: the instance name; F_N: the total number of non-dominated fragments; $Avg._{Req.}$: the average number of request per fragment; Sec_f: the CPU time in seconds used to generate non-dominated fragments, rounded to the nearest tenth; *Sec.*: the total CPU time in seconds used to solve the instance, including Sec_f time, rounded to the nearest tenth; Z^*: the optimal solution obtained within the time limit or left blank if no feasible solution found. If only a feasible solution was obtained within the time limit, then it is reported in bold in column Z^* as a best upper bound found; *Gap*: The percentage gap computed as $100(Z^* - Z_R)/Z^*$, where Z_R is the lower bound at the root node.

Table B.1. One Stack Vehicle of C1 Class

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-51-c1-w15-500-1000	102	2.0	0.000	0.0	0.1	1,105,646.72
a280-61-c1-w15-500-1000	93	1.7	0.000	0.0	0.0	1,808,284.25
a280-71-c1-w15-500-1000	278	2.5	0.000	0.0	0.1	1,709,058.06
a280-81-c1-w15-500-1000	267	2.2	0.000	0.0	0.1	2,212,101.51
a280-91-c1-w15-500-1000	203	2.1	0.000	0.0	0.1	2,514,105.46
a280-101-c1-w15-500-1000	366	2.2	0.000	0.1	0.1	2,514,862.73

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-111-c1-w15-500-1000	379	2.3	0.000	0.1	0.1	2,516,451.65
a280-121-c1-w15-500-1000	1,136	2.9	0.000	0.1	0.2	2,917,966.15
a280-131-c1-w15-500-1000	639	2.4	0.000	0.1	0.1	2,917,266.76
a280-141-c1-w15-500-1000	657	2.4	0.000	0.1	0.2	3,419,879.77
a280-151-c1-w15-500-1000	1,571	2.9	0.000	0.2	0.3	3,118,716.46
a280-51-c1-w15-1000-1200	68	1.8	0.001	0.0	0.0	1,105,936.91
a280-61-c1-w15-1000-1200	173	2.5	0.000	0.0	0.0	1,207,346.83
a280-71-c1-w15-1000-1200	270	2.4	0.000	0.0	0.0	1,308,215.76
a280-81-c1-w15-1000-1200	344	2.5	0.000	0.0	0.1	1,509,621.15
a280-91-c1-w15-1000-1200	435	2.7	0.000	0.1	0.1	2,112,198.13
a280-101-c1-w15-1000-1200	284	2.2	0.027	0.0	0.1	1,713,264.58
a280-111-c1-w15-1000-1200	444	2.5	0.000	0.1	0.2	2,113,947.31
a280-121-c1-w15-1000-1200	1,045	2.7	0.000	0.1	0.2	2,315,419.30
a280-131-c1-w15-1000-1200	2,395	3.3	0.000	0.3	0.3	2,114,162.98
a280-141-c1-w15-1000-1200	1,319	3.1	0.000	0.2	0.3	2,919,613.96
a280-151-c1-w15-1000-1200	1,880	3.2	0.000	0.3	0.3	2,817,811.96
a280-51-c1-w15-1500-2000	106	2.1	0.000	0.0	0.0	1,105,889.17
a280-61-c1-w15-1500-2000	563	3.2	0.000	0.0	0.1	1,106,952.12
a280-71-c1-w15-1500-2000	1,591	3.6	0.000	0.1	0.2	1,006,669.68
a280-81-c1-w15-1500-2000	2,184	3.7	0.000	0.2	0.3	1,008,841.66
a280-91-c1-w15-1500-2000	2,702	3.6	0.000	0.2	0.3	1,409,819.71
a280-101-c1-w15-1500-2000	1,707	3.5	0.000	0.2	0.3	1,510,935.39
a280-111-c1-w15-1500-2000	4,566	4.1	0.000	0.4	0.6	1,813,021.37
a280-121-c1-w15-1500-2000	4,883	3.8	0.000	0.4	0.6	1,612,519.57
a280-131-c1-w15-1500-2000	47,776	5.2	0.000	4.2	6.0	1,613,173.05
a280-141-c1-w15-1500-2000	5,121	3.7	0.000	0.6	0.8	2,315,477.04
a280-151-c1-w15-1500-2000	13,922	4.3	0.007	1.4	2.3	1,915,403.68
a280-51-c1-w30-500-1000	221	2.6	0.000	0.0	0.1	1,004,669.26
a280-61-c1-w30-500-1000	312	2.7	0.000	0.0	0.0	1,407,454.51
a280-71-c1-w30-500-1000	423	2.7	0.000	0.0	0.1	1,608,457.15
a280-81-c1-w30-500-1000	577	2.8	0.000	0.0	0.1	1,710,398.27
a280-91-c1-w30-500-1000	530	2.6	0.000	0.1	0.1	1,811,246.10
a280-101-c1-w30-500-1000	464	2.4	0.000	0.1	0.1	2,213,758.46
a280-111-c1-w30-500-1000	824	2.7	0.002	0.1	0.2	2,515,457.27
a280-121-c1-w30-500-1000	1,360	3.0	0.000	0.1	0.2	2,515,844.40
a280-131-c1-w30-500-1000	1,227	2.7	0.001	0.2	0.2	2,415,710.52
a280-141-c1-w30-500-1000	3,355	3.2	0.000	0.3	0.5	2,515,940.93

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-151-c1-w30-500-1000	1,499	2.8	0.004	0.2	0.3	2,918,602.34
a280-51-c1-w30-1000-1200	309	3.0	0.004	0.0	0.1	1,004,965.82
a280-61-c1-w30-1000-1200	243	2.6	0.000	0.0	0.1	1,206,358.26
a280-71-c1-w30-1000-1200	278	2.6	0.000	0.0	0.1	1,307,418.77
a280-81-c1-w30-1000-1200	336	2.5	0.000	0.1	0.1	1,409,119.05
a280-91-c1-w30-1000-1200	736	3.1	0.000	0.1	0.1	1,509,755.80
a280-101-c1-w30-1000-1200	1,247	3.6	6.210	0.1	0.3	1,612,186.44
a280-111-c1-w30-1000-1200	1,416	3.3	0.000	0.2	0.2	1,913,683.06
a280-121-c1-w30-1000-1200	846	2.8	0.000	0.1	0.2	2,215,870.47
a280-131-c1-w30-1000-1200	2,559	3.4	0.000	0.3	0.4	2,114,134.68
a280-141-c1-w30-1000-1200	2,535	3.3	0.008	0.3	0.6	2,215,547.28
a280-151-c1-w30-1000-1200	4,016	3.9	0.000	0.5	0.6	2,415,923.46
a280-51-c1-w30-1500-2000	148	2.4	0.000	0.0	0.0	804,750.80
a280-61-c1-w30-1500-2000	665	3.1	0.000	0.1	0.1	805,473.10
a280-71-c1-w30-1500-2000	658	3.1	0.000	0.1	0.1	1,307,372.01
a280-81-c1-w30-1500-2000	827	3.1	0.000	0.1	0.1	1,208,313.82
a280-91-c1-w30-1500-2000	4,894	3.9	0.000	0.3	0.5	1,309,938.16
a280-101-c1-w30-1500-2000	2,406	3.4	0.000	0.2	0.3	1,411,244.75
a280-111-c1-w30-1500-2000	1,706	3.4	0.000	0.2	0.2	1,512,519.77
a280-121-c1-w30-1500-2000	3,251	3.5	0.000	0.3	0.4	1,813,293.51
a280-131-c1-w30-1500-2000	7,403	3.9	0.003	0.8	1.3	1,613,038.34
a280-141-c1-w30-1500-2000	20,621	4.6	0.000	2.3	3.0	1,914,734.39
a280-151-c1-w30-1500-2000	31,383	4.8	0.000	3.1	4.2	2,015,630.76
a280-51-c1-w45-500-1000	343	3.3	0.000	0.0	0.1	1,105,692.83
a280-61-c1-w45-500-1000	326	2.6	0.000	0.0	0.0	1,306,848.99
a280-71-c1-w45-500-1000	993	3.2	0.017	0.1	0.1	1,107,077.16
a280-81-c1-w45-500-1000	432	2.6	0.003	0.0	0.1	1,509,170.31
a280-91-c1-w45-500-1000	664	2.8	0.000	0.1	0.1	1,609,893.19
a280-101-c1-w45-500-1000	3,781	3.8	0.000	0.3	0.5	2,012,804.77
a280-111-c1-w45-500-1000	3,189	3.6	0.000	0.3	0.4	1,911,893.72
a280-121-c1-w45-500-1000	2,205	3.3	0.005	0.2	0.4	1,812,416.09
a280-131-c1-w45-500-1000	3,598	3.3	0.007	0.3	0.6	2,014,610.21
a280-141-c1-w45-500-1000	2,503	3.2	4.146	0.3	0.6	2,415,605.11
a280-151-c1-w45-500-1000	5,046	3.6	3.833	0.5	1.0	2,617,091.11
a280-51-c1-w45-1000-1200	200	2.6	0.000	0.0	0.0	804,648.73
a280-61-c1-w45-1000-1200	360	3.1	0.000	0.0	0.1	905,443.29
a280-71-c1-w45-1000-1200	243	2.5	0.000	0.0	0.1	1,207,358.59

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-81-c1-w45-1000-1200	862	3.5	0.015	0.1	0.2	1,409,059.94
a280-91-c1-w45-1000-1200	631	3.0	0.013	0.1	0.1	1,509,808.94
a280-101-c1-w45-1000-1200	1,079	3.4	0.002	0.1	0.2	1,711,812.06
a280-111-c1-w45-1000-1200	1,198	3.4	0.000	0.2	0.2	1,713,894.78
a280-121-c1-w45-1000-1200	6,006	4.5	0.010	0.6	1.0	1,612,713.35
a280-131-c1-w45-1000-1200	3,163	3.7	0.022	0.4	1.8	1,712,514.10
a280-141-c1-w45-1000-1200	2,412	3.4	0.004	0.3	0.5	2,015,001.01
a280-151-c1-w45-1000-1200	4,825	3.8	0.002	0.5	0.9	2,216,159.31
a280-51-c1-w45-1500-2000	122	2.4	0.000	0.0	0.0	805,142.15
a280-61-c1-w45-1500-2000	640	3.4	0.000	0.0	0.1	1,006,153.99
a280-71-c1-w45-1500-2000	3,488	4.3	0.000	0.3	0.4	1,007,200.46
a280-81-c1-w45-1500-2000	262	2.4	0.001	0.0	0.1	1,308,730.26
a280-91-c1-w45-1500-2000	7,041	4.8	0.000	0.5	0.8	1,209,904.94
a280-101-c1-w45-1500-2000	5,222	4.1	0.000	0.5	0.6	1,411,697.10
a280-111-c1-w45-1500-2000	9,536	4.9	0.000	0.8	1.2	1,411,906.46
a280-121-c1-w45-1500-2000	46,454	5.4	0.000	4.2	5.8	1,311,269.63
a280-131-c1-w45-1500-2000	52,113	5.0	6.542	6.5	15.7	1,511,980.74
a280-141-c1-w45-1500-2000	25,238	4.5	6.180	2.6	9.5	1,613,121.67
a280-151-c1-w45-1500-2000	10,208	4.0	0.001	1.2	1.7	1,814,790.03
brd14051-51-c12-w45-3000-4000	52	1.7	0.000	0.0	0.0	1,430,063.68
brd14051-61-c1-w45-3000-4000	70	1.7	0.000	0.0	0.0	1,739,174.22
brd14051-71-c1-w45-3000-4000	70	1.6	0.000	0.0	0.0	2,045,429.77
brd14051-81-c1-w45-3000-4000	186	2.4	0.000	0.0	0.1	2,148,189.30
brd14051-91-c1-w45-3000-4000	118	1.8	0.000	0.0	0.0	2,458,657.74
brd14051-101-c1-w45-3000-4000	292	2.5	0.000	0.1	0.1	2,565,876.04
brd14051-111-c1-w45-3000-4000	177	2.0	0.000	0.1	0.1	3,166,358.71
brd14051-121-c1-w45-3000-4000	140	1.7	0.000	0.1	0.1	3,383,634.91
brd14051-131-c1-w45-3000-4000	313	2.5	0.000	0.1	0.1	3,697,981.41
brd14051-141-c1-w45-3000-4000	381	2.3	0.000	0.1	0.1	3,582,507.61
brd14051-151-c1-w45-3000-4000	322	2.2	0.000	0.1	0.2	3,484,638.26
brd14051-51-c1-w60-3000-4000	71	1.9	0.000	0.0	0.0	1,531,264.52
brd14051-61-c1-w60-3000-4000	90	1.9	0.000	0.0	0.0	1,641,770.07
brd14051-71-c1-w60-3000-4000	105	2.0	0.000	0.0	0.0	1,944,094.94
brd14051-81-c1-w60-3000-4000	117	1.9	0.000	0.0	0.0	2,147,086.02
brd14051-91-c1-w60-3000-4000	144	1.9	0.000	0.1	0.1	2,255,371.41
brd14051-101-c1-w60-3000-4000	123	1.7	0.000	0.0	0.1	2,869,174.67
brd14051-111-c1-w60-3000-4000	251	2.3	0.000	0.1	0.1	2,961,520.20

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-121-c1-w60-3000-4000	177	1.9	0.000	0.1	0.1	3,177,713.57
brd14051-131-c1-w60-3000-4000	294	2.3	0.000	0.1	0.2	3,495,769.23
brd14051-141-c1-w60-3000-4000	291	2.1	0.000	0.1	0.1	3,078,797.43
brd14051-151-c1-w60-3000-4000	544	2.6	2.873	0.1	0.2	3,481,032.16
brd14051-51-c1-w75-3000-4000	67	2.1	0.000	0.0	0.0	1,429,624.70
brd14051-61-c1-w75-3000-4000	84	1.9	0.000	0.0	0.0	1,639,092.04
brd14051-71-c1-w75-3000-4000	119	2.0	0.000	0.0	0.0	1,950,724.97
brd14051-81-c1-w75-3000-4000	169	2.2	0.000	0.0	0.0	2,252,801.05
brd14051-91-c1-w75-3000-4000	185	2.2	0.000	0.0	0.1	2,866,669.39
brd14051-101-c1-w75-3000-4000	419	2.9	3.982	0.1	0.1	2,560,263.52
brd14051-111-c1-w75-3000-4000	319	2.3	0.000	0.1	0.1	2,457,826.78
brd14051-121-c1-w75-3000-4000	826	3.4	0.000	0.1	0.2	2,872,805.82
brd14051-131-c1-w75-3000-4000	345	2.3	3.163	0.1	0.1	3,186,953.77
brd14051-141-c1-w75-3000-4000	404	2.5	0.000	0.1	0.1	3,783,602.98
brd14051-151-c1-w75-3000-4000	592	2.6	0.000	0.1	0.2	3,588,719.92
brd14051-51-c1-w90-3000-4000	31	1.2	0.000	0.0	0.0	1,534,070.58
brd14051-61-c1-w90-3000-4000	59	1.5	0.000	0.0	0.0	1,741,906.96
brd14051-71-c1-w90-3000-4000	104	1.8	0.000	0.0	0.0	1,943,981.01
brd14051-81-c1-w90-3000-4000	142	2.0	0.000	0.0	0.0	2,354,770.02
brd14051-91-c1-w90-3000-4000	241	2.5	0.000	0.0	0.1	2,358,198.79
brd14051-101-c1-w90-3000-4000	332	2.5	0.000	0.0	0.1	2,462,483.25
brd14051-111-c1-w90-3000-4000	249	2.2	0.000	0.1	0.1	2,764,737.56
brd14051-121-c1-w90-3000-4000	360	2.5	0.000	0.1	0.1	3,070,805.37
brd14051-131-c1-w90-3000-4000	356	2.4	0.000	0.1	0.1	3,183,391.95
brd14051-141-c1-w90-3000-4000	604	2.7	0.000	0.1	0.2	2,670,075.08
brd14051-151-c1-w90-3000-4000	1,002	3.2	0.000	0.2	0.3	3,277,185.81
brd14051-51-c1-w120-3000-4000	74	1.9	0.000	0.0	0.0	1,329,122.85
brd14051-61-c1-w120-3000-4000	110	2.1	0.051	0.0	0.0	1,743,100.93
brd14051-71-c1-w120-3000-4000	162	2.3	0.000	0.0	0.0	1,740,161.37
brd14051-81-c1-w120-3000-4000	165	2.1	0.000	0.0	0.1	1,946,659.38
brd14051-91-c1-w120-3000-4000	157	2.0	0.000	0.0	0.1	2,558,976.63
brd14051-101-c1-w120-3000-4000	340	2.6	0.001	0.1	0.1	2,360,175.41
brd14051-111-c1-w120-3000-4000	497	2.8	4.243	0.1	0.1	2,356,870.57
brd14051-121-c1-w120-3000-4000	345	2.2	0.003	0.1	0.1	2,673,182.84
brd14051-131-c1-w120-3000-4000	247	2.0	0.003	0.1	0.2	3,687,838.00
brd14051-141-c1-w120-3000-4000	662	2.8	0.000	0.1	0.2	2,875,812.68
brd14051-151-c1-w120-3000-4000	2,368	4.2	0.000	0.3	0.4	3,077,929.74

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-51-c1-w45-3000-4000	64	1.8	0.000	0.0	0.0	1,430,762.23
d18512-61-c1-w45-3000-4000	57	1.5	0.000	0.0	0.0	1,839,047.52
d18512-71-c1-w45-3000-4000	90	1.8	0.000	0.0	0.0	1,841,359.14
d18512-81-c1-w45-3000-4000	68	1.4	0.000	0.0	0.0	2,455,897.77
d18512-91-c1-w45-3000-4000	228	2.3	0.000	0.0	0.1	2,355,253.83
d18512-101-c1-w45-3000-4000	177	2.1	0.000	0.1	0.1	2,966,343.28
d18512-111-c1-w45-3000-4000	147	1.8	0.000	0.1	0.1	3,074,662.19
d18512-121-c1-w45-3000-4000	162	1.8	0.000	0.1	0.1	3,690,878.74
d18512-131-c1-w45-3000-4000	429	2.4	0.000	0.1	0.1	3,180,497.02
d18512-141-c1-w45-3000-4000	433	2.5	0.000	0.1	0.1	3,782,460.23
d18512-151-c1-w45-3000-4000	315	2.1	0.000	0.1	0.1	3,996,403.97
d18512-51-c1-w60-3000-4000	50	1.6	0.000	0.0	0.0	1,637,326.58
d18512-61-c1-w60-3000-4000	77	1.9	0.000	0.0	0.0	1,538,322.35
d18512-71-c1-w60-3000-4000	85	1.8	0.000	0.0	0.0	1,841,852.54
d18512-81-c1-w60-3000-4000	100	1.8	0.000	0.0	0.0	2,354,878.87
d18512-91-c1-w60-3000-4000	146	2.2	0.000	0.1	0.1	2,153,551.03
d18512-101-c1-w60-3000-4000	260	2.3	0.009	0.1	0.1	2,659,619.35
d18512-111-c1-w60-3000-4000	230	2.2	0.000	0.1	0.1	2,662,824.36
d18512-121-c1-w60-3000-4000	203	2.0	0.000	0.1	0.1	3,388,801.22
d18512-131-c1-w60-3000-4000	567	2.7	0.000	0.1	0.1	3,277,933.88
d18512-141-c1-w60-3000-4000	341	2.3	0.000	0.1	0.1	3,480,007.99
d18512-151-c1-w60-3000-4000	372	2.3	0.003	0.1	0.2	4,513,529.57
d18512-51-c1-w75-3000-4000	40	1.4	0.000	0.0	0.0	1,637,219.52
d18512-61-c1-w75-3000-4000	59	1.6	0.000	0.0	0.0	1,842,875.92
d18512-71-c1-w75-3000-4000	107	1.9	0.000	0.0	0.0	2,146,282.89
d18512-81-c1-w75-3000-4000	113	1.9	0.000	0.0	0.1	2,452,691.66
d18512-91-c1-w75-3000-4000	106	1.7	0.000	0.0	0.0	2,660,205.22
d18512-101-c1-w75-3000-4000	232	2.2	0.000	0.0	0.1	2,357,028.87
d18512-111-c1-w75-3000-4000	192	2.0	3.494	0.1	0.1	2,870,625.19
d18512-121-c1-w75-3000-4000	172	1.8	0.000	0.1	0.1	3,487,218.82
d18512-131-c1-w75-3000-4000	263	2.3	0.006	0.1	0.1	2,976,778.35
d18512-141-c1-w75-3000-4000	440	2.4	2.791	0.2	0.2	3,582,938.47
d18512-151-c1-w75-3000-4000	244	1.9	0.000	0.1	0.1	4,299,609.29
d18512-51-c1-w90-3000-4000	94	2.1	0.000	0.0	0.0	1,331,361.95
d18512-61-c1-w90-3000-4000	117	2.2	0.072	0.0	0.0	1,539,336.03
d18512-71-c1-w90-3000-4000	98	1.8	0.000	0.0	0.0	1,842,067.38
d18512-81-c1-w90-3000-4000	86	1.7	0.000	0.0	0.0	2,249,587.08

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-91-c1-w90-3000-4000	102	1.7	0.000	0.0	0.1	2,659,392.95
d18512-101-c1-w90-3000-4000	258	2.4	0.000	0.1	0.1	2,663,249.37
d18512-111-c1-w90-3000-4000	271	2.2	0.018	0.1	0.1	3,175,300.22
d18512-121-c1-w90-3000-4000	216	2.0	0.000	0.1	0.1	2,875,829.39
d18512-131-c1-w90-3000-4000	254	2.1	0.000	0.1	0.1	3,785,990.41
d18512-141-c1-w90-3000-4000	445	2.6	0.010	0.1	0.1	3,477,904.28
d18512-151-c1-w90-3000-4000	875	3.1	0.008	0.2	0.2	3,796,807.47
d18512-51-c1-w120-3000-4000	45	1.4	0.000	0.0	0.0	1,638,120.27
d18512-61-c1-w120-3000-4000	80	1.8	0.000	0.0	0.0	1,539,452.99
d18512-71-c1-w120-3000-4000	155	2.3	0.000	0.0	0.0	1,740,600.05
d18512-81-c1-w120-3000-4000	125	2.0	0.000	0.0	0.0	1,847,152.74
d18512-91-c1-w120-3000-4000	112	1.8	0.000	0.1	0.1	2,354,620.34
d18512-101-c1-w120-3000-4000	250	2.4	0.000	0.1	0.1	2,761,278.47
d18512-111-c1-w120-3000-4000	660	2.9	0.000	0.1	0.1	2,466,794.64
d18512-121-c1-w120-3000-4000	258	2.1	0.023	0.1	0.1	2,771,706.35
d18512-131-c1-w120-3000-4000	1,162	3.5	0.021	0.2	0.4	2,668,527.44
d18512-141-c1-w120-3000-4000	1,382	3.6	0.000	0.2	0.3	3,178,889.45
d18512-151-c1-w120-3000-4000	1,012	3.1	0.011	0.2	0.3	3,086,189.04
fnl4461-51-c12-w45-3000-4000	172	2.4	0.000	0.0	0.0	1,117,964.24
fnl4461-61-c1-w45-3000-4000	121	2.1	0.000	0.0	0.0	1,423,094.44
fnl4461-71-c1-w45-3000-4000	359	2.8	0.000	0.0	0.1	1,323,916.38
fnl4461-81-c1-w45-3000-4000	143	2.0	0.000	0.0	0.1	1,929,573.15
fnl4461-91-c1-w45-3000-4000	292	2.2	0.000	0.0	0.1	1,934,919.92
fnl4461-101-c1-w45-3000-4000	467	2.5	0.000	0.1	0.1	2,139,165.34
fnl4461-111-c1-w45-3000-4000	380	2.4	0.000	0.1	0.1	2,541,782.98
fnl4461-121-c1-w45-3000-4000	1,063	3.0	0.000	0.2	0.2	2,139,318.82
fnl4461-131-c1-w45-3000-4000	468	2.4	0.000	0.1	0.1	2,853,453.19
fnl4461-141-c1-w45-3000-4000	894	2.6	0.003	0.1	0.2	2,750,231.85
fnl4461-151-c1-w45-3000-4000	489	2.2	0.007	0.1	0.2	3,357,971.60
fnl4461-51-c1-w60-3000-4000	57	1.7	0.000	0.0	0.0	1,220,042.57
fnl4461-61-c1-w60-3000-4000	162	2.4	0.000	0.0	0.0	1,422,654.64
fnl4461-71-c1-w60-3000-4000	294	2.6	0.000	0.0	0.1	1,324,348.52
fnl4461-81-c1-w60-3000-4000	457	2.8	0.000	0.0	0.1	1,628,834.91
fnl4461-91-c1-w60-3000-4000	308	2.4	0.000	0.0	0.1	1,833,175.22
fnl4461-101-c1-w60-3000-4000	585	2.7	0.000	0.1	0.1	2,035,950.33
fnl4461-111-c1-w60-3000-4000	1,670	3.2	0.000	0.2	0.2	1,938,698.22
fnl4461-121-c1-w60-3000-4000	1,261	3.0	0.003	0.1	0.2	2,238,485.74

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-131-c1-w60-3000-4000	885	2.6	0.000	0.1	0.2	2,952,409.65
fnl4461-141-c1-w60-3000-4000	1,685	3.2	0.000	0.2	0.3	2,349,280.87
fnl4461-151-c1-w60-3000-4000	1,234	2.9	0.000	0.2	0.3	3,059,037.66
fnl4461-51-c1-w75-3000-4000	85	2.0	0.000	0.0	0.0	1,120,102.41
fnl4461-61-c1-w75-3000-4000	404	3.1	0.000	0.0	0.1	1,220,416.31
fnl4461-71-c1-w75-3000-4000	296	2.7	0.000	0.0	0.1	1,727,824.49
fnl4461-81-c1-w75-3000-4000	288	2.5	0.000	0.0	0.1	1,930,407.68
fnl4461-91-c1-w75-3000-4000	445	2.7	0.000	0.1	0.1	1,527,260.01
fnl4461-101-c1-w75-3000-4000	621	2.7	0.000	0.1	0.1	2,137,532.90
fnl4461-111-c1-w75-3000-4000	680	2.8	0.000	0.1	0.1	1,937,665.70
fnl4461-121-c1-w75-3000-4000	1,414	3.0	0.006	0.2	0.3	1,837,549.11
fnl4461-131-c1-w75-3000-4000	587	2.6	0.000	0.1	0.1	2,748,937.06
fnl4461-141-c1-w75-3000-4000	3,212	3.5	0.000	0.3	0.4	2,347,843.38
fnl4461-151-c1-w75-3000-4000	1,650	2.9	0.000	0.2	0.3	2,853,472.47
fnl4461-51-c1-w90-3000-4000	52	1.6	0.000	0.0	0.0	1,118,648.46
fnl4461-61-c1-w90-3000-4000	295	2.8	0.000	0.0	0.1	1,123,510.30
fnl4461-71-c1-w90-3000-4000	140	2.0	0.000	0.0	0.0	1,525,761.87
fnl4461-81-c1-w90-3000-4000	695	3.1	0.000	0.1	0.1	1,627,684.57
fnl4461-91-c1-w90-3000-4000	367	2.7	0.000	0.1	0.1	1,729,160.30
fnl4461-101-c1-w90-3000-4000	482	2.5	0.000	0.1	0.1	1,833,621.07
fnl4461-111-c1-w90-3000-4000	808	2.8	0.000	0.1	0.1	1,936,761.90
fnl4461-121-c1-w90-3000-4000	877	3.1	0.005	0.1	0.2	2,340,776.93
fnl4461-131-c1-w90-3000-4000	962	3.0	0.009	0.1	0.2	2,345,475.60
fnl4461-141-c1-w90-3000-4000	757	2.6	0.018	0.1	0.2	2,650,703.84
fnl4461-151-c1-w90-3000-4000	2,816	3.4	0.000	0.3	0.4	2,655,128.53
fnl4461-51-c1-w120-3000-4000	187	2.6	0.000	0.0	0.0	1,117,125.50
fnl4461-61-c1-w120-3000-4000	384	2.9	0.000	0.0	0.1	1,120,288.33
fnl4461-71-c1-w120-3000-4000	1,031	3.4	0.111	0.1	0.2	1,223,101.43
fnl4461-81-c1-w120-3000-4000	784	3.1	0.000	0.1	0.1	1,527,188.75
fnl4461-91-c1-w120-3000-4000	1,427	3.4	0.000	0.1	0.2	1,426,553.18
fnl4461-101-c1-w120-3000-4000	862	3.3	0.000	0.1	0.1	1,632,738.67
fnl4461-111-c1-w120-3000-4000	404	2.4	0.000	0.1	0.1	2,241,027.01
fnl4461-121-c1-w120-3000-4000	1,370	3.1	0.000	0.2	0.2	1,937,243.01
fnl4461-131-c1-w120-3000-4000	2,279	3.3	0.000	0.3	0.3	2,244,553.28
fnl4461-141-c1-w120-3000-4000	5,354	4.1	0.013	0.6	0.8	2,346,898.86
fnl4461-151-c1-w120-3000-4000	1,455	3.0	0.000	0.3	0.3	2,549,195.10
nrw1379-51-c1-w45-3000-4000	62	1.7	0.000	0.0	0.0	1,218,667.61

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-61-c1-w45-3000-4000	106	2.1	0.000	0.0	0.0	1,626,551.60
nrw1379-71-c1-w45-3000-4000	183	2.3	0.000	0.0	0.0	1,728,483.42
nrw1379-81-c1-w45-3000-4000	318	2.5	0.000	0.0	0.1	1,932,446.25
nrw1379-91-c1-w45-3000-4000	231	2.4	0.000	0.1	0.1	2,135,772.03
nrw1379-101-c1-w45-3000-4000	216	2.2	0.003	0.1	0.1	2,139,489.32
nrw1379-111-c1-w45-3000-4000	357	2.6	0.000	0.1	0.1	3,148,953.74
nrw1379-121-c1-w45-3000-4000	454	2.4	0.000	0.1	0.1	2,648,182.04
nrw1379-131-c1-w45-3000-4000	1,438	3.0	0.000	0.2	0.2	2,647,470.79
nrw1379-141-c1-w45-3000-4000	1,018	3.2	0.000	0.2	0.2	2,853,849.03
nrw1379-151-c1-w45-3000-4000	1,218	2.8	0.000	0.2	0.3	2,857,131.53
nrw1379-51-c1-w60-3000-4000	143	2.5	0.000	0.0	0.0	1,221,629.25
nrw1379-61-c1-w60-3000-4000	92	1.9	0.000	0.0	0.0	2,026,983.26
nrw1379-71-c1-w60-3000-4000	266	2.5	0.000	0.0	0.1	1,523,900.19
nrw1379-81-c1-w60-3000-4000	410	2.7	0.000	0.1	0.1	2,239,369.66
nrw1379-91-c1-w60-3000-4000	219	2.2	0.000	0.1	0.1	2,229,853.13
nrw1379-101-c1-w60-3000-4000	261	2.3	0.000	0.0	0.1	2,342,193.59
nrw1379-111-c1-w60-3000-4000	364	2.5	0.000	0.1	0.1	2,442,702.48
nrw1379-121-c1-w60-3000-4000	732	2.7	0.000	0.1	0.2	2,850,869.91
nrw1379-131-c1-w60-3000-4000	962	2.8	3.402	0.1	0.2	2,951,764.01
nrw1379-141-c1-w60-3000-4000	748	2.6	0.000	0.1	0.2	2,650,586.32
nrw1379-151-c1-w60-3000-4000	798	2.5	0.011	0.2	0.2	3,158,950.59
nrw1379-51-c1-w75-3000-4000	108	2.4	0.000	0.0	0.0	1,119,357.02
nrw1379-61-c1-w75-3000-4000	376	3.1	0.000	0.0	0.1	1,324,740.29
nrw1379-71-c1-w75-3000-4000	320	2.6	0.000	0.1	0.1	1,526,142.94
nrw1379-81-c1-w75-3000-4000	142	2.0	0.000	0.0	0.0	2,035,141.05
nrw1379-91-c1-w75-3000-4000	416	2.8	0.000	0.1	0.1	2,136,407.22
nrw1379-101-c1-w75-3000-4000	193	2.0	0.000	0.0	0.1	2,741,433.94
nrw1379-111-c1-w75-3000-4000	349	2.3	0.000	0.1	0.1	2,543,213.41
nrw1379-121-c1-w75-3000-4000	332	2.2	0.000	0.1	0.1	2,946,659.23
nrw1379-131-c1-w75-3000-4000	2,134	3.4	0.000	0.3	0.3	3,049,943.23
nrw1379-141-c1-w75-3000-4000	2,077	3.4	0.000	0.2	0.3	2,752,365.46
nrw1379-151-c1-w75-3000-4000	1,863	3.3	0.000	0.2	0.3	2,653,363.63
nrw1379-51-c1-w90-3000-4000	102	2.1	0.000	0.0	0.0	1,318,674.62
nrw1379-61-c1-w90-3000-4000	253	2.8	0.000	0.0	0.0	1,426,317.00
nrw1379-71-c1-w90-3000-4000	388	2.8	0.000	0.0	0.1	1,323,467.55
nrw1379-81-c1-w90-3000-4000	337	2.6	0.000	0.0	0.1	1,833,175.97
nrw1379-91-c1-w90-3000-4000	406	2.7	0.000	0.1	0.1	1,833,619.20

Continued Table B.1 – *One stack vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-101-c1-w90-3000-4000	575	2.7	0.000	0.1	0.1	2,239,191.79
nrw1379-111-c1-w90-3000-4000	629	2.7	0.000	0.1	0.1	2,440,399.02
nrw1379-121-c1-w90-3000-4000	481	2.5	0.000	0.1	0.1	2,744,229.89
nrw1379-131-c1-w90-3000-4000	1,137	3.1	0.000	0.2	0.2	2,451,232.31
nrw1379-141-c1-w90-3000-4000	1,088	2.9	0.002	0.2	0.2	2,654,201.76
nrw1379-151-c1-w90-3000-4000	3,358	3.6	0.000	0.4	0.5	2,853,162.17
nrw1379-51-c1-w120-3000-4000	104	2.1	0.000	0.0	0.0	1,220,231.02
nrw1379-61-c1-w120-3000-4000	243	2.8	0.000	0.0	0.1	1,423,786.64
nrw1379-71-c1-w120-3000-4000	842	3.7	0.000	0.1	0.1	1,627,704.20
nrw1379-81-c1-w120-3000-4000	250	2.3	0.000	0.1	0.1	2,032,198.11
nrw1379-91-c1-w120-3000-4000	797	2.9	0.000	0.1	0.1	1,731,363.13
nrw1379-101-c1-w120-3000-4000	629	2.9	0.006	0.1	0.2	2,138,486.04
nrw1379-111-c1-w120-3000-4000	803	3.1	0.000	0.1	0.2	2,641,759.78
nrw1379-121-c1-w120-3000-4000	1,470	3.4	0.000	0.2	0.2	2,240,028.01
nrw1379-131-c1-w120-3000-4000	829	2.9	0.000	0.1	0.2	2,447,280.64
nrw1379-141-c1-w120-3000-4000	512	2.4	0.000	0.1	0.2	2,654,713.69
nrw1379-151-c1-w120-3000-4000	1,510	3.0	0.000	0.2	0.3	2,754,097.44

Table B.2. One Stack Vehicle of C2 Class

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-51-c2-w15-500-1000	74	1.8	0.000	0.0	0.1	1,506,584.91
a280-61-c2-w15-500-1000	163	2.1	0.000	0.0	0.0	1,507,162.57
a280-71-c2-w15-500-1000	230	2.2	0.000	0.0	0.0	1,508,562.51
a280-81-c2-w15-500-1000	352	2.5	0.000	0.1	0.1	1,809,824.45
a280-91-c2-w15-500-1000	336	2.3	0.000	0.0	0.1	1,911,166.60
a280-101-c2-w15-500-1000	211	2.0	0.000	0.0	0.1	2,716,188.99
a280-111-c2-w15-500-1000	239	2.0	0.000	0.1	0.1	2,817,076.58
a280-121-c2-w15-500-1000	706	2.4	0.000	0.1	0.1	2,616,705.37
a280-131-c2-w15-500-1000	992	2.7	0.000	0.1	0.2	3,418,752.76
a280-141-c2-w15-500-1000	1,108	2.5	0.000	0.1	0.2	3,118,587.39
a280-151-c2-w15-500-1000	743	2.3	0.000	0.1	0.2	3,721,380.54
a280-51-c2-w15-1000-1200	80	2.0	0.000	0.0	0.0	1,206,089.74
a280-61-c2-w15-1000-1200	99	2.0	0.000	0.0	0.0	1,608,096.07
a280-71-c2-w15-1000-1200	173	2.3	0.000	0.0	0.0	1,407,621.60
a280-81-c2-w15-1000-1200	262	2.3	0.000	0.0	0.1	1,811,170.76
a280-91-c2-w15-1000-1200	321	2.5	0.000	0.0	0.1	1,811,569.59

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-101-c2-w15-1000-1200	717	2.7	0.000	0.1	0.1	1,712,399.59
a280-111-c2-w15-1000-1200	805	2.9	0.000	0.1	0.1	1,913,405.77
a280-121-c2-w15-1000-1200	1,313	3.3	0.000	0.2	0.3	1,812,663.16
a280-131-c2-w15-1000-1200	1,456	2.9	0.000	0.2	0.2	2,215,076.44
a280-141-c2-w15-1000-1200	852	2.7	0.000	0.2	0.2	3,017,912.08
a280-151-c2-w15-1000-1200	1,427	2.9	0.000	0.2	0.3	2,718,034.78
a280-51-c2-w15-1500-2000	631	3.6	0.000	0.0	0.1	1,105,556.44
a280-61-c2-w15-1500-2000	303	2.8	0.000	0.0	0.1	1,006,239.49
a280-71-c2-w15-1500-2000	476	2.8	0.000	0.0	0.1	1,208,385.69
a280-81-c2-w15-1500-2000	607	3.0	0.000	0.1	0.1	1,609,497.17
a280-91-c2-w15-1500-2000	3,844	4.2	0.000	0.3	0.4	1,309,653.04
a280-101-c2-w15-1500-2000	1,039	3.0	0.000	0.1	0.2	1,511,316.43
a280-111-c2-w15-1500-2000	2,808	3.4	0.000	0.2	0.3	1,913,588.38
a280-121-c2-w15-1500-2000	7,381	4.1	0.000	0.7	0.9	1,613,194.16
a280-131-c2-w15-1500-2000	2,155	3.4	0.000	0.3	0.3	2,214,745.28
a280-141-c2-w15-1500-2000	1,666	3.0	0.000	0.2	0.3	2,415,843.15
a280-151-c2-w15-1500-2000	9,940	3.9	0.000	1.0	1.3	2,317,318.03
a280-51-c2-w30-500-1000	92	2.0	0.000	0.0	0.0	1,306,410.46
a280-61-c2-w30-500-1000	273	2.7	0.000	0.0	0.1	1,607,489.75
a280-71-c2-w30-500-1000	237	2.5	0.000	0.0	0.1	1,809,177.84
a280-81-c2-w30-500-1000	590	2.7	0.000	0.0	0.1	1,910,466.24
a280-91-c2-w30-500-1000	563	2.6	0.002	0.1	0.1	2,212,923.14
a280-101-c2-w30-500-1000	455	2.5	0.000	0.1	0.1	2,414,302.78
a280-111-c2-w30-500-1000	563	2.5	0.000	0.1	0.1	2,314,434.43
a280-121-c2-w30-500-1000	978	2.8	0.000	0.1	0.2	2,415,379.98
a280-131-c2-w30-500-1000	1,311	2.7	0.000	0.2	0.2	2,616,225.82
a280-141-c2-w30-500-1000	2,165	3.1	0.000	0.2	0.3	2,716,711.99
a280-151-c2-w30-500-1000	2,379	3.0	0.004	0.3	0.3	2,616,773.52
a280-51-c2-w30-1000-1200	58	1.7	0.000	0.0	0.0	904,649.30
a280-61-c2-w30-1000-1200	193	2.3	0.000	0.0	0.1	1,005,436.75
a280-71-c2-w30-1000-1200	562	3.1	0.000	0.0	0.1	1,407,506.61
a280-81-c2-w30-1000-1200	350	2.5	0.000	0.0	0.1	1,409,498.46
a280-91-c2-w30-1000-1200	374	2.5	0.000	0.1	0.1	1,811,138.80
a280-101-c2-w30-1000-1200	899	3.1	0.000	0.1	0.1	2,013,637.36
a280-111-c2-w30-1000-1200	2,019	3.9	5.239	0.2	0.9	1,913,242.72
a280-121-c2-w30-1000-1200	863	2.6	0.000	0.1	0.1	2,215,279.87
a280-131-c2-w30-1000-1200	3,775	3.8	0.000	0.3	0.5	2,013,900.06

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-141-c2-w30-1000-1200	1,669	3.2	0.000	0.3	0.3	2,616,708.07
a280-151-c2-w30-1000-1200	1,654	3.0	0.000	0.2	0.3	2,616,558.97
a280-51-c2-w30-1500-2000	257	2.6	0.000	0.0	0.1	904,923.13
a280-61-c2-w30-1500-2000	1,613	4.1	0.000	0.1	0.2	706,463.88
a280-71-c2-w30-1500-2000	848	3.2	0.000	0.1	0.1	1,307,457.94
a280-81-c2-w30-1500-2000	535	2.9	0.000	0.1	0.1	1,208,838.83
a280-91-c2-w30-1500-2000	3,044	3.9	0.000	0.3	0.4	1,109,237.26
a280-101-c2-w30-1500-2000	5,607	4.1	0.008	0.4	1.0	1,311,313.61
a280-111-c2-w30-1500-2000	3,824	3.8	0.000	0.3	0.4	1,411,639.99
a280-121-c2-w30-1500-2000	4,077	3.6	0.000	0.4	0.5	1,512,718.13
a280-131-c2-w30-1500-2000	9,810	3.8	0.000	0.9	1.2	1,713,720.71
a280-141-c2-w30-1500-2000	19,585	4.3	0.002	1.9	2.7	1,914,754.68
a280-151-c2-w30-1500-2000	15,373	4.4	0.000	1.5	2.0	2,216,234.28
a280-51-c2-w45-500-1000	168	2.1	0.000	0.0	0.1	1,005,077.13
a280-61-c2-w45-500-1000	531	3.0	0.003	0.0	0.1	1,005,434.53
a280-71-c2-w45-500-1000	866	3.1	0.004	0.1	0.1	1,307,743.89
a280-81-c2-w45-500-1000	573	2.7	0.000	0.1	0.1	1,509,460.20
a280-91-c2-w45-500-1000	486	2.6	0.011	0.1	0.1	1,409,458.59
a280-101-c2-w45-500-1000	1,592	3.4	5.223	0.1	0.3	1,912,081.42
a280-111-c2-w45-500-1000	1,806	3.1	0.000	0.1	0.2	1,913,492.52
a280-121-c2-w45-500-1000	4,130	3.4	0.030	0.3	1.1	1,812,268.03
a280-131-c2-w45-500-1000	2,143	3.2	0.000	0.3	0.4	2,214,733.45
a280-141-c2-w45-500-1000	2,296	3.1	0.000	0.2	0.3	2,716,907.37
a280-151-c2-w45-500-1000	7,039	3.6	0.000	0.6	0.8	2,716,694.14
a280-51-c2-w45-1000-1200	150	2.4	0.000	0.1	0.1	1,005,254.99
a280-61-c2-w45-1000-1200	322	3.1	0.000	0.0	0.1	1,106,345.24
a280-71-c2-w45-1000-1200	538	3.3	0.006	0.1	0.1	1,206,858.45
a280-81-c2-w45-1000-1200	1,063	3.5	0.000	0.1	0.1	1,207,776.99
a280-91-c2-w45-1000-1200	294	2.3	0.021	0.0	0.1	1,410,414.16
a280-101-c2-w45-1000-1200	602	2.7	0.007	0.1	0.1	1,712,006.62
a280-111-c2-w45-1000-1200	1,973	3.4	0.001	0.2	0.3	1,612,506.27
a280-121-c2-w45-1000-1200	4,123	3.9	5.507	0.4	2.0	1,813,011.75
a280-131-c2-w45-1000-1200	3,909	3.7	5.205	0.4	4.0	1,913,258.67
a280-141-c2-w45-1000-1200	4,826	3.6	0.015	0.5	1.9	1,814,371.02
a280-151-c2-w45-1000-1200	2,773	3.4	0.024	0.3	0.9	2,116,463.00
a280-51-c2-w45-1500-2000	681	3.6	0.000	0.0	0.1	705,046.80
a280-61-c2-w45-1500-2000	280	2.6	0.000	0.0	0.0	1,106,181.71

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-71-c2-w45-1500-2000	1,164	3.5	0.001	0.1	0.2	1,106,349.55
a280-81-c2-w45-1500-2000	1,614	3.6	0.032	0.1	0.2	908,466.36
a280-91-c2-w45-1500-2000	1,136	3.5	0.000	0.1	0.2	1,208,947.65
a280-101-c2-w45-1500-2000	3,819	3.9	0.004	0.3	0.4	1,211,214.67
a280-111-c2-w45-1500-2000	9,641	4.4	0.003	0.7	1.1	1,210,968.84
a280-121-c2-w45-1500-2000	19,563	5.1	0.003	1.7	2.8	1,413,206.45
a280-131-c2-w45-1500-2000	22,486	4.5	0.005	2.0	3.9	1,613,127.77
a280-141-c2-w45-1500-2000	13,781	4.3	0.001	1.3	1.9	1,815,066.87
a280-151-c2-w45-1500-2000	26,167	4.4	0.000	2.4	3.3	1,814,218.95
brd14051-51-c2-w45-3000-4000	59	1.9	0.000	0.0	0.1	1,632,010.96
brd14051-61-c2-w45-3000-4000	55	1.6	0.000	0.0	0.0	2,148,151.57
brd14051-71-c2-w45-3000-4000	127	2.2	0.000	0.0	0.0	1,843,594.43
brd14051-81-c2-w45-3000-4000	173	2.2	0.000	0.0	0.0	2,352,222.77
brd14051-91-c2-w45-3000-4000	78	1.5	0.000	0.0	0.0	2,967,828.94
brd14051-101-c2-w45-3000-4000	134	1.8	0.000	0.0	0.1	2,770,372.02
brd14051-111-c2-w45-3000-4000	249	2.5	0.000	0.1	0.1	3,066,799.53
brd14051-121-c2-w45-3000-4000	165	1.8	0.000	0.1	0.1	3,581,211.38
brd14051-131-c2-w45-3000-4000	236	2.1	0.000	0.1	0.1	3,795,762.28
brd14051-141-c2-w45-3000-4000	242	2.0	0.000	0.1	0.1	3,587,896.60
brd14051-151-c2-w45-3000-4000	364	2.3	0.000	0.1	0.1	3,989,689.54
brd14051-51-c2-w60-3000-4000	38	1.4	0.000	0.0	0.0	1,936,516.46
brd14051-61-c2-w60-3000-4000	38	1.2	0.000	0.0	0.0	1,942,441.84
brd14051-71-c2-w60-3000-4000	81	1.7	0.000	0.1	0.1	1,940,595.65
brd14051-81-c2-w60-3000-4000	87	1.7	0.000	0.0	0.0	2,653,805.23
brd14051-91-c2-w60-3000-4000	148	2.0	0.000	0.0	0.1	2,354,925.55
brd14051-101-c2-w60-3000-4000	89	1.5	0.000	0.0	0.1	2,968,843.20
brd14051-111-c2-w60-3000-4000	321	2.5	0.000	0.1	0.1	2,661,684.05
brd14051-121-c2-w60-3000-4000	222	2.1	0.000	0.1	0.1	3,786,215.00
brd14051-131-c2-w60-3000-4000	117	1.5	0.000	0.1	0.1	4,204,699.55
brd14051-141-c2-w60-3000-4000	462	2.4	0.000	0.1	0.1	3,380,410.32
brd14051-151-c2-w60-3000-4000	584	2.7	0.000	0.1	0.2	3,488,277.25
brd14051-51-c2-w75-3000-4000	62	1.8	0.000	0.0	0.0	1,330,342.35
brd14051-61-c2-w75-3000-4000	54	1.5	0.000	0.0	0.0	2,149,707.69
brd14051-71-c2-w75-3000-4000	117	2.0	0.000	0.0	0.1	2,247,939.35
brd14051-81-c2-w75-3000-4000	119	1.8	0.000	0.0	0.0	2,352,192.85
brd14051-91-c2-w75-3000-4000	180	2.1	0.000	0.0	0.1	2,459,635.11
brd14051-101-c2-w75-3000-4000	363	2.8	0.003	0.1	0.1	2,361,319.35

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-111-c2-w75-3000-4000	201	2.2	0.000	0.1	0.1	3,071,440.39
brd14051-121-c2-w75-3000-4000	207	2.2	0.000	0.1	0.1	3,283,005.05
brd14051-131-c2-w75-3000-4000	396	2.4	0.000	0.1	0.1	3,393,041.14
brd14051-141-c2-w75-3000-4000	370	2.2	0.000	0.1	0.1	3,377,651.80
brd14051-151-c2-w75-3000-4000	1,114	3.3	0.000	0.2	0.3	3,682,816.44
brd14051-51-c2-w90-3000-4000	67	1.9	0.000	0.0	0.0	1,633,558.51
brd14051-61-c2-w90-3000-4000	131	2.3	0.000	0.0	0.0	1,741,857.27
brd14051-71-c2-w90-3000-4000	133	2.2	0.003	0.0	0.0	1,838,437.96
brd14051-81-c2-w90-3000-4000	143	2.1	0.000	0.1	0.1	2,352,990.72
brd14051-91-c2-w90-3000-4000	457	2.9	0.000	0.1	0.1	2,148,124.74
brd14051-101-c2-w90-3000-4000	151	1.9	0.000	0.0	0.1	3,173,487.93
brd14051-111-c2-w90-3000-4000	420	2.6	0.000	0.1	0.1	2,561,599.80
brd14051-121-c2-w90-3000-4000	285	2.2	0.029	0.1	0.1	3,481,668.64
brd14051-131-c2-w90-3000-4000	264	2.1	0.000	0.1	0.1	3,693,605.75
brd14051-141-c2-w90-3000-4000	266	2.0	0.000	0.1	0.1	3,579,551.19
brd14051-151-c2-w90-3000-4000	938	3.0	0.021	0.2	0.2	3,177,470.86
brd14051-51-c2-w120-3000-4000	73	1.9	0.000	0.0	0.0	1,835,649.40
brd14051-61-c2-w120-3000-4000	53	1.5	0.000	0.0	0.0	1,944,654.06
brd14051-71-c2-w120-3000-4000	106	2.0	0.000	0.0	0.0	1,845,071.42
brd14051-81-c2-w120-3000-4000	278	2.6	0.000	0.0	0.0	2,047,558.37
brd14051-91-c2-w120-3000-4000	117	1.8	0.000	0.0	0.1	2,562,636.64
brd14051-101-c2-w120-3000-4000	168	2.0	0.000	0.1	0.1	2,867,210.05
brd14051-111-c2-w120-3000-4000	332	2.3	0.017	0.1	0.1	2,354,006.63
brd14051-121-c2-w120-3000-4000	345	2.3	0.000	0.1	0.1	3,076,860.26
brd14051-131-c2-w120-3000-4000	205	1.9	0.000	0.1	0.1	3,285,691.25
brd14051-141-c2-w120-3000-4000	247	2.0	0.000	0.1	0.1	3,379,567.65
brd14051-151-c2-w120-3000-4000	539	2.5	0.000	0.1	0.2	3,276,158.76
d18512-51-c2-w45-3000-4000	56	1.6	0.000	0.0	0.0	1,637,529.54
d18512-61-c2-w45-3000-4000	66	1.8	0.000	0.0	0.0	1,741,480.41
d18512-71-c2-w45-3000-4000	72	1.6	0.000	0.0	0.1	2,045,680.51
d18512-81-c2-w45-3000-4000	101	1.8	0.000	0.0	0.0	2,457,575.41
d18512-91-c2-w45-3000-4000	93	1.6	0.000	0.0	0.0	2,460,339.34
d18512-101-c2-w45-3000-4000	277	2.4	0.000	0.1	0.1	2,255,639.18
d18512-111-c2-w45-3000-4000	205	2.1	0.000	0.1	0.1	3,178,389.28
d18512-121-c2-w45-3000-4000	156	1.8	0.000	0.1	0.1	3,389,218.98
d18512-131-c2-w45-3000-4000	221	2.1	0.000	0.1	0.1	3,587,413.39
d18512-141-c2-w45-3000-4000	298	2.3	0.000	0.1	0.1	3,683,599.30

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-151-c2-w45-3000-4000	199	1.7	0.000	0.1	0.1	4,095,719.03
d18512-51-c2-w60-3000-4000	53	1.8	0.000	0.0	0.0	1,538,174.71
d18512-61-c2-w60-3000-4000	82	1.8	0.045	0.0	0.0	1,637,856.70
d18512-71-c2-w60-3000-4000	118	2.0	0.000	0.0	0.0	1,842,056.85
d18512-81-c2-w60-3000-4000	110	1.8	0.000	0.0	0.0	2,151,721.49
d18512-91-c2-w60-3000-4000	126	1.8	0.000	0.0	0.0	2,560,228.09
d18512-101-c2-w60-3000-4000	141	1.9	0.032	0.1	0.1	3,171,890.90
d18512-111-c2-w60-3000-4000	140	1.8	0.000	0.1	0.1	3,175,319.30
d18512-121-c2-w60-3000-4000	192	1.9	0.000	0.1	0.1	3,590,961.22
d18512-131-c2-w60-3000-4000	467	2.7	0.000	0.1	0.1	3,279,992.56
d18512-141-c2-w60-3000-4000	692	2.9	0.000	0.1	0.2	3,181,107.15
d18512-151-c2-w60-3000-4000	245	2.0	2.442	0.1	0.1	4,094,347.30
d18512-51-c2-w75-3000-4000	41	1.4	0.000	0.0	0.0	1,638,598.56
d18512-61-c2-w75-3000-4000	111	2.1	0.000	0.0	0.0	1,537,294.95
d18512-71-c2-w75-3000-4000	107	1.9	0.000	0.0	0.0	1,943,961.69
d18512-81-c2-w75-3000-4000	119	2.0	0.000	0.0	0.0	2,454,590.07
d18512-91-c2-w75-3000-4000	184	2.2	0.000	0.0	0.1	2,452,927.43
d18512-101-c2-w75-3000-4000	334	2.6	0.000	0.1	0.1	2,356,044.57
d18512-111-c2-w75-3000-4000	239	2.2	0.000	0.1	0.1	2,562,983.72
d18512-121-c2-w75-3000-4000	141	1.7	0.000	0.1	0.1	3,181,712.72
d18512-131-c2-w75-3000-4000	182	1.8	0.000	0.1	0.1	3,284,314.01
d18512-141-c2-w75-3000-4000	376	2.2	0.000	0.1	0.1	3,784,577.37
d18512-151-c2-w75-3000-4000	349	2.2	0.000	0.1	0.1	3,794,529.45
d18512-51-c2-w90-3000-4000	74	2.0	0.000	0.0	0.0	1,432,921.92
d18512-61-c2-w90-3000-4000	64	1.7	0.000	0.0	0.0	1,638,532.35
d18512-71-c2-w90-3000-4000	136	2.1	0.000	0.0	0.0	1,839,226.37
d18512-81-c2-w90-3000-4000	84	1.6	0.000	0.0	0.0	1,948,010.96
d18512-91-c2-w90-3000-4000	240	2.6	0.000	0.0	0.1	2,659,208.13
d18512-101-c2-w90-3000-4000	200	2.2	0.003	0.0	0.1	2,459,997.34
d18512-111-c2-w90-3000-4000	125	1.6	0.000	0.1	0.1	3,174,754.19
d18512-121-c2-w90-3000-4000	357	2.4	0.030	0.1	0.1	3,081,488.39
d18512-131-c2-w90-3000-4000	320	2.3	0.000	0.1	0.1	3,278,730.49
d18512-141-c2-w90-3000-4000	419	2.6	0.000	0.1	0.2	3,179,371.24
d18512-151-c2-w90-3000-4000	383	2.3	0.000	0.1	0.2	4,099,959.93
d18512-51-c2-w120-3000-4000	37	1.4	0.000	0.0	0.0	1,740,698.41
d18512-61-c2-w120-3000-4000	76	1.8	0.000	0.0	0.0	1,537,858.31
d18512-71-c2-w120-3000-4000	159	2.1	0.000	0.0	0.0	1,637,987.52

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-81-c2-w120-3000-4000	173	2.3	0.000	0.1	0.1	1,947,456.93
d18512-91-c2-w120-3000-4000	110	1.8	0.000	0.0	0.1	2,253,777.87
d18512-101-c2-w120-3000-4000	263	2.5	0.000	0.0	0.1	2,153,648.86
d18512-111-c2-w120-3000-4000	289	2.2	0.000	0.1	0.1	2,365,177.04
d18512-121-c2-w120-3000-4000	170	1.8	0.000	0.1	0.1	3,183,370.63
d18512-131-c2-w120-3000-4000	609	2.8	0.000	0.1	0.1	2,973,755.58
d18512-141-c2-w120-3000-4000	396	2.4	0.000	0.1	0.2	3,279,504.17
d18512-151-c2-w120-3000-4000	831	3.0	0.000	0.1	0.2	3,687,562.90
fnl4461-51-c2-w45-3000-4000	41	1.4	0.000	0.0	0.0	1,421,985.11
fnl4461-61-c2-w45-3000-4000	164	2.5	0.000	0.0	0.0	1,622,912.14
fnl4461-71-c2-w45-3000-4000	126	2.0	0.000	0.0	0.0	1,826,245.96
fnl4461-81-c2-w45-3000-4000	422	2.9	0.000	0.1	0.1	1,526,218.78
fnl4461-91-c2-w45-3000-4000	680	2.8	0.000	0.1	0.1	1,728,595.79
fnl4461-101-c2-w45-3000-4000	595	2.7	0.000	0.1	0.1	2,239,486.67
fnl4461-111-c2-w45-3000-4000	327	2.2	0.003	0.1	0.1	2,239,791.98
fnl4461-121-c2-w45-3000-4000	709	2.6	0.000	0.1	0.1	2,542,020.45
fnl4461-131-c2-w45-3000-4000	816	2.7	0.000	0.1	0.2	2,654,219.34
fnl4461-141-c2-w45-3000-4000	1,260	3.0	0.000	0.2	0.2	2,955,710.53
fnl4461-151-c2-w45-3000-4000	1,028	2.8	0.000	0.2	0.2	2,652,550.29
fnl4461-51-c2-w60-3000-4000	83	2.0	0.000	0.0	0.0	1,020,443.84
fnl4461-61-c2-w60-3000-4000	107	2.0	0.000	0.0	0.0	1,422,755.92
fnl4461-71-c2-w60-3000-4000	194	2.3	0.000	0.0	0.0	1,524,536.40
fnl4461-81-c2-w60-3000-4000	374	2.5	0.000	0.0	0.1	1,830,083.29
fnl4461-91-c2-w60-3000-4000	496	2.8	0.000	0.0	0.1	1,628,120.42
fnl4461-101-c2-w60-3000-4000	463	2.5	0.000	0.1	0.1	2,037,262.00
fnl4461-111-c2-w60-3000-4000	395	2.4	0.000	0.1	0.1	2,138,070.35
fnl4461-121-c2-w60-3000-4000	643	2.6	0.000	0.1	0.2	2,240,899.69
fnl4461-131-c2-w60-3000-4000	640	2.5	0.000	0.1	0.2	2,549,814.38
fnl4461-141-c2-w60-3000-4000	465	2.3	0.000	0.1	0.2	2,750,696.41
fnl4461-151-c2-w60-3000-4000	2,735	3.3	0.000	0.3	0.5	2,450,965.69
fnl4461-51-c2-w75-3000-4000	286	3.0	0.000	0.0	0.0	1,319,218.32
fnl4461-61-c2-w75-3000-4000	169	2.2	0.000	0.0	0.0	1,322,394.85
fnl4461-71-c2-w75-3000-4000	235	2.5	0.000	0.0	0.0	1,526,576.56
fnl4461-81-c2-w75-3000-4000	457	2.9	0.002	0.0	0.1	1,626,284.93
fnl4461-91-c2-w75-3000-4000	292	2.4	0.013	0.1	0.1	1,829,980.48
fnl4461-101-c2-w75-3000-4000	307	2.4	0.000	0.1	0.1	1,832,148.59
fnl4461-111-c2-w75-3000-4000	466	2.5	0.004	0.1	0.1	1,936,961.62

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-121-c2-w75-3000-4000	894	2.8	0.000	0.1	0.2	2,643,523.58
fnl4461-131-c2-w75-3000-4000	763	2.7	0.008	0.1	0.2	2,344,218.81
fnl4461-141-c2-w75-3000-4000	1,076	2.6	0.000	0.2	0.2	2,449,451.86
fnl4461-151-c2-w75-3000-4000	732	2.5	0.000	0.1	0.2	2,856,585.38
fnl4461-51-c2-w90-3000-4000	95	2.0	0.000	0.0	0.0	1,117,349.30
fnl4461-61-c2-w90-3000-4000	122	2.2	0.000	0.0	0.1	1,320,502.24
fnl4461-71-c2-w90-3000-4000	174	2.3	0.000	0.0	0.0	1,623,002.58
fnl4461-81-c2-w90-3000-4000	195	2.3	0.000	0.0	0.1	1,730,303.70
fnl4461-91-c2-w90-3000-4000	431	2.7	0.000	0.0	0.1	1,631,008.30
fnl4461-101-c2-w90-3000-4000	474	2.4	0.000	0.1	0.1	2,135,796.00
fnl4461-111-c2-w90-3000-4000	617	2.7	0.009	0.1	0.1	2,342,343.82
fnl4461-121-c2-w90-3000-4000	1,063	2.9	0.000	0.1	0.2	2,343,815.93
fnl4461-131-c2-w90-3000-4000	2,060	3.3	0.003	0.2	0.3	1,939,065.32
fnl4461-141-c2-w90-3000-4000	1,331	3.1	0.001	0.2	0.3	2,549,659.03
fnl4461-151-c2-w90-3000-4000	2,369	3.2	0.010	0.3	0.4	2,549,838.11
fnl4461-51-c2-w120-3000-4000	207	2.6	0.000	0.0	0.0	1,017,080.41
fnl4461-61-c2-w120-3000-4000	246	2.6	0.000	0.0	0.0	1,119,735.18
fnl4461-71-c2-w120-3000-4000	587	3.3	0.014	0.0	0.1	1,120,386.91
fnl4461-81-c2-w120-3000-4000	788	3.3	0.022	0.1	0.1	1,327,257.32
fnl4461-91-c2-w120-3000-4000	1,155	3.0	0.000	0.1	0.1	1,527,039.56
fnl4461-101-c2-w120-3000-4000	354	2.4	0.000	0.1	0.1	1,937,037.62
fnl4461-111-c2-w120-3000-4000	1,074	3.0	0.000	0.1	0.2	2,036,035.67
fnl4461-121-c2-w120-3000-4000	4,408	4.0	0.000	0.4	0.6	2,141,632.56
fnl4461-131-c2-w120-3000-4000	898	2.8	0.000	0.2	0.2	2,344,844.04
fnl4461-141-c2-w120-3000-4000	1,340	2.8	0.014	0.2	0.4	2,350,524.25
fnl4461-151-c2-w120-3000-4000	2,106	3.3	0.000	0.3	0.4	2,250,465.47
nrw1379-51-c2-w45-3000-4000	67	1.8	0.000	0.0	0.0	1,118,333.70
nrw1379-61-c2-w45-3000-4000	107	2.0	0.000	0.0	0.0	1,623,819.29
nrw1379-71-c2-w45-3000-4000	117	2.0	0.000	0.0	0.0	2,131,097.53
nrw1379-81-c2-w45-3000-4000	110	1.8	0.000	0.0	0.0	1,935,804.32
nrw1379-91-c2-w45-3000-4000	279	2.3	0.000	0.1	0.1	2,135,258.43
nrw1379-101-c2-w45-3000-4000	344	2.3	0.000	0.1	0.1	1,936,205.84
nrw1379-111-c2-w45-3000-4000	510	2.8	0.000	0.1	0.1	2,438,858.02
nrw1379-121-c2-w45-3000-4000	628	2.7	0.000	0.1	0.1	2,545,336.85
nrw1379-131-c2-w45-3000-4000	333	2.2	0.000	0.1	0.1	2,847,996.13
nrw1379-141-c2-w45-3000-4000	1,037	3.0	0.000	0.1	0.2	3,560,770.05
nrw1379-151-c2-w45-3000-4000	320	2.2	0.000	0.1	0.2	3,565,548.09

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-51-c2-w60-3000-4000	152	2.7	0.000	0.0	0.0	1,319,826.23
nrw1379-61-c2-w60-3000-4000	157	2.4	0.000	0.0	0.0	1,221,162.51
nrw1379-71-c2-w60-3000-4000	141	2.2	0.000	0.0	0.0	1,827,939.45
nrw1379-81-c2-w60-3000-4000	183	2.1	0.000	0.0	0.1	1,834,217.35
nrw1379-91-c2-w60-3000-4000	170	2.0	0.000	0.0	0.1	1,934,655.93
nrw1379-101-c2-w60-3000-4000	296	2.3	0.000	0.1	0.1	2,238,218.02
nrw1379-111-c2-w60-3000-4000	504	2.6	0.000	0.1	0.1	2,545,198.80
nrw1379-121-c2-w60-3000-4000	615	2.8	0.000	0.1	0.1	2,753,347.01
nrw1379-131-c2-w60-3000-4000	1,258	3.1	0.000	0.1	0.2	2,550,504.88
nrw1379-141-c2-w60-3000-4000	984	2.9	0.000	0.1	0.2	3,361,125.03
nrw1379-151-c2-w60-3000-4000	1,947	3.3	0.000	0.2	0.3	2,957,354.83
nrw1379-51-c2-w75-3000-4000	65	2.1	0.000	0.0	0.0	1,421,078.26
nrw1379-61-c2-w75-3000-4000	113	2.0	0.000	0.0	0.0	1,321,910.06
nrw1379-71-c2-w75-3000-4000	106	1.8	0.000	0.0	0.0	1,829,015.20
nrw1379-81-c2-w75-3000-4000	257	2.4	0.000	0.1	0.1	1,832,753.16
nrw1379-91-c2-w75-3000-4000	419	2.8	0.000	0.1	0.1	2,035,677.80
nrw1379-101-c2-w75-3000-4000	303	2.3	0.000	0.1	0.1	2,035,343.91
nrw1379-111-c2-w75-3000-4000	230	2.0	0.000	0.1	0.1	2,443,375.49
nrw1379-121-c2-w75-3000-4000	847	2.8	0.000	0.1	0.2	2,442,453.79
nrw1379-131-c2-w75-3000-4000	456	2.3	0.000	0.1	0.1	3,151,219.33
nrw1379-141-c2-w75-3000-4000	510	2.3	0.000	0.1	0.2	2,851,078.60
nrw1379-151-c2-w75-3000-4000	794	2.8	0.002	0.2	0.2	2,756,853.03
nrw1379-51-c2-w90-3000-4000	120	2.4	0.000	0.0	0.0	1,121,122.21
nrw1379-61-c2-w90-3000-4000	143	2.4	0.000	0.0	0.0	1,828,234.24
nrw1379-71-c2-w90-3000-4000	707	3.2	0.000	0.1	0.1	1,628,758.36
nrw1379-81-c2-w90-3000-4000	540	2.9	0.000	0.0	0.1	1,632,006.31
nrw1379-91-c2-w90-3000-4000	395	2.6	0.000	0.1	0.1	2,031,348.66
nrw1379-101-c2-w90-3000-4000	416	2.5	0.000	0.1	0.1	2,235,677.25
nrw1379-111-c2-w90-3000-4000	248	2.0	0.000	0.1	0.1	2,339,747.26
nrw1379-121-c2-w90-3000-4000	628	2.9	0.000	0.1	0.1	2,345,671.24
nrw1379-131-c2-w90-3000-4000	1,762	3.4	0.000	0.3	0.3	2,442,281.94
nrw1379-141-c2-w90-3000-4000	2,242	3.5	0.000	0.3	0.3	3,057,575.67
nrw1379-151-c2-w90-3000-4000	700	2.6	0.000	0.2	0.2	2,656,194.76
nrw1379-51-c2-w120-3000-4000	256	3.1	0.000	0.0	0.0	1,019,971.67
nrw1379-61-c2-w120-3000-4000	287	2.8	0.000	0.0	0.0	1,121,192.57
nrw1379-71-c2-w120-3000-4000	132	2.1	0.000	0.0	0.0	1,729,845.53
nrw1379-81-c2-w120-3000-4000	518	3.0	0.000	0.0	0.1	1,532,388.64

Continued Table B.2 – *One stack vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-91-c2-w120-3000-4000	510	2.7	0.000	0.1	0.1	1,632,981.80
nrw1379-101-c2-w120-3000-4000	317	2.4	0.000	0.1	0.1	2,239,448.66
nrw1379-111-c2-w120-3000-4000	485	2.6	0.000	0.1	0.1	1,940,213.27
nrw1379-121-c2-w120-3000-4000	1,769	3.4	0.000	0.2	0.3	2,142,170.22
nrw1379-131-c2-w120-3000-4000	958	2.7	0.000	0.2	0.2	2,445,740.83
nrw1379-141-c2-w120-3000-4000	1,227	3.0	3.783	0.2	0.3	2,655,737.15
nrw1379-151-c2-w120-3000-4000	1,280	2.9	0.000	0.2	0.3	2,853,063.09

Table B.3. Two Stacks Vehicle of C1 Class

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-51-c1-w15-500-1000	388	2.8	0.000	0.0	0.1	704,139.82
a280-61-c1-w15-500-1000	626	2.8	0.000	0.1	0.1	1,005,105.04
a280-71-c1-w15-500-1000	859	3.1	0.002	0.1	0.2	1,106,782.67
a280-81-c1-w15-500-1000	1,380	3.2	0.000	0.1	0.2	1,207,975.16
a280-91-c1-w15-500-1000	1,088	2.9	0.002	0.1	0.2	1,309,503.67
a280-101-c1-w15-500-1000	1,423	2.9	0.000	0.1	0.2	1,409,612.74
a280-111-c1-w15-500-1000	2,279	3.3	0.002	0.2	0.4	1,611,219.84
a280-121-c1-w15-500-1000	4,007	3.4	0.017	0.5	0.9	1,712,274.46
a280-131-c1-w15-500-1000	4,666	3.5	0.007	0.6	0.9	1,612,743.53
a280-141-c1-w15-500-1000	3,864	3.2	4.961	0.5	1.5	2,013,842.53
a280-151-c1-w15-500-1000	6,503	3.5	4.968	0.9	1.9	2,013,645.83
a280-51-c1-w15-1000-1200	344	3.0	0.000	0.0	0.1	704,241.41
a280-61-c1-w15-1000-1200	825	3.5	0.000	0.1	0.1	805,276.20
a280-71-c1-w15-1000-1200	1,405	3.4	0.008	0.1	0.2	906,162.44
a280-81-c1-w15-1000-1200	1,671	3.4	0.017	0.1	0.4	1,107,905.51
a280-91-c1-w15-1000-1200	2,311	3.6	0.000	0.3	0.3	1,209,114.82
a280-101-c1-w15-1000-1200	3,279	3.7	8.258	0.4	0.9	1,209,237.52
a280-111-c1-w15-1000-1200	6,172	4.3	0.000	0.6	0.9	1,310,817.53
a280-121-c1-w15-1000-1200	6,990	3.8	0.019	0.8	1.8	1,311,476.37
a280-131-c1-w15-1000-1200	22,504	4.4	0.009	2.6	8.5	1,411,613.96
a280-141-c1-w15-1000-1200	16,787	4.3	0.005	2.1	3.8	1,713,145.91
a280-151-c1-w15-1000-1200	13,059	4.1	0.010	1.5	4.1	1,913,896.23
a280-51-c1-w15-1500-2000	1,448	3.7	0.000	0.1	0.2	604,017.15
a280-61-c1-w15-1500-2000	6,183	4.3	0.000	0.6	0.8	604,750.89
a280-71-c1-w15-1500-2000	19,401	5.1	0.016	1.9	3.2	605,106.84
a280-81-c1-w15-1500-2000	22,965	4.9	0.023	2.0	3.6	706,656.23

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-91-c1-w15-1500-2000	32,926	4.8	0.020	3.7	8.2	807,191.11
a280-101-c1-w15-1500-2000	30,306	4.7	0.024	3.3	9.9	908,288.31
a280-111-c1-w15-1500-2000	68,413	5.1	0.016	9.2	20.5	1,009,818.33
a280-121-c1-w15-1500-2000	218,381	5.6	0.059	38.9	122.7	910,181.09
a280-131-c1-w15-1500-2000	533,402	6.2	0.039	82.4	261.2	910,594.60
a280-141-c1-w15-1500-2000	159,718	5.3	0.023	24.9	64.1	1,211,335.28
a280-151-c1-w15-1500-2000	435,079	5.6	8.188	105.9	672.5	1,211,961.13
a280-51-c1-w30-500-1000	1,547	3.7	0.004	0.3	0.4	603,546.84
a280-61-c1-w30-500-1000	1,381	3.4	0.002	0.1	0.2	804,772.95
a280-71-c1-w30-500-1000	2,151	3.7	0.005	0.2	0.4	1,005,905.57
a280-81-c1-w30-500-1000	2,548	3.5	0.003	0.3	0.4	1,007,178.13
a280-91-c1-w30-500-1000	2,715	3.4	8.270	0.3	1.0	1,208,355.05
a280-101-c1-w30-500-1000	4,190	3.8	0.001	0.5	0.7	1,310,185.00
a280-111-c1-w30-500-1000	8,938	4.0	0.002	1.1	1.5	1,409,993.00
a280-121-c1-w30-500-1000	9,293	3.8	0.020	1.3	1.8	1,411,517.78
a280-131-c1-w30-500-1000	14,950	3.8	0.009	2.4	5.0	1,410,644.05
a280-141-c1-w30-500-1000	25,363	4.1	0.016	3.7	9.1	1,511,954.79
a280-151-c1-w30-500-1000	14,617	3.9	0.018	2.2	7.4	1,713,830.47
a280-51-c1-w30-1000-1200	1,212	3.7	0.006	0.1	0.2	703,715.86
a280-61-c1-w30-1000-1200	2,770	4.2	0.000	0.3	0.3	704,727.51
a280-71-c1-w30-1000-1200	3,816	4.2	0.008	0.4	0.6	805,230.82
a280-81-c1-w30-1000-1200	5,842	4.4	0.022	0.5	0.8	907,608.98
a280-91-c1-w30-1000-1200	11,293	4.9	0.003	1.2	1.8	1,007,772.30
a280-101-c1-w30-1000-1200	11,858	4.6	9.001	1.3	6.7	1,109,220.22
a280-111-c1-w30-1000-1200	14,119	4.6	0.005	1.8	2.6	1,210,409.83
a280-121-c1-w30-1000-1200	20,160	4.5	0.021	2.8	7.8	1,211,247.88
a280-131-c1-w30-1000-1200	71,264	4.9	0.027	11.9	36.6	1,210,866.68
a280-141-c1-w30-1000-1200	50,133	4.8	0.063	10.5	210.5	1,312,519.73
a280-151-c1-w30-1000-1200	118,209	5.5	0.022	19.4	48.0	1,412,251.82
a280-51-c1-w30-1500-2000	1,990	4.2	0.008	0.3	0.5	503,324.73
a280-61-c1-w30-1500-2000	31,049	5.5	0.014	3.7	5.6	503,966.23
a280-71-c1-w30-1500-2000	27,892	5.4	0.001	3.5	5.1	705,184.07
a280-81-c1-w30-1500-2000	24,878	5.0	0.017	2.7	4.6	705,792.26
a280-91-c1-w30-1500-2000	71,498	5.5	0.029	10.9	28.2	807,610.06
a280-101-c1-w30-1500-2000	51,007	5.0	0.011	7.0	11.5	808,699.28
a280-111-c1-w30-1500-2000	131,615	5.5	0.049	18.5	51.2	909,697.59
a280-121-c1-w30-1500-2000	254,629	5.7	0.030	47.2	159.4	910,463.74

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-131-c1-w30-1500-2000	524,729	5.9	0.039	98.3	968.7	1,010,125.31
a280-141-c1-w30-1500-2000	762,343	5.9	8.896	168.1	3,287.3	1,110,862.27
a280-151-c1-w30-1500-2000	954,555	6.1	0.018	186.7	653.4	1,112,209.32
a280-51-c1-w45-500-1000	1,517	3.8	0.011	0.2	0.3	704,103.61
a280-61-c1-w45-500-1000	1,703	3.5	0.008	0.2	0.4	805,047.64
a280-71-c1-w45-500-1000	7,629	4.3	0.011	1.4	1.8	705,234.84
a280-81-c1-w45-500-1000	3,296	3.5	0.018	0.4	0.8	906,554.04
a280-91-c1-w45-500-1000	8,213	4.0	0.034	1.5	2.0	907,835.31
a280-101-c1-w45-500-1000	29,803	4.7	0.005	6.3	10.1	1,108,781.98
a280-111-c1-w45-500-1000	35,271	4.7	0.007	8.0	12.3	1,108,277.26
a280-121-c1-w45-500-1000	43,271	4.7	0.011	10.1	19.3	1,209,445.47
a280-131-c1-w45-500-1000	47,060	4.5	7.632	10.2	68.1	1,310,051.87
a280-141-c1-w45-500-1000	33,770	4.3	0.000	8.4	10.2	1,511,653.25
a280-151-c1-w45-500-1000	43,464	4.5	0.013	11.6	24.3	1,712,626.81
a280-51-c1-w45-1000-1200	4,687	4.6	0.000	0.6	0.9	503,405.10
a280-61-c1-w45-1000-1200	6,290	4.8	0.015	0.9	1.5	603,928.14
a280-71-c1-w45-1000-1200	7,223	4.6	12.374	0.7	1.2	805,414.83
a280-81-c1-w45-1000-1200	37,117	5.9	0.010	9.8	12.1	806,155.19
a280-91-c1-w45-1000-1200	13,017	4.7	0.015	1.7	3.9	907,256.86
a280-101-c1-w45-1000-1200	11,755	4.6	0.028	1.8	5.2	1,008,691.09
a280-111-c1-w45-1000-1200	26,081	4.8	0.020	5.1	12.0	1,209,321.08
a280-121-c1-w45-1000-1200	157,789	5.9	8.979	42.2	172.5	1,108,969.53
a280-131-c1-w45-1000-1200	189,355	5.7	0.055	50.4	1,289.6	1,109,472.94
a280-141-c1-w45-1000-1200	169,611	5.4	0.027	48.3	133.1	1,210,701.87
a280-151-c1-w45-1000-1200	109,724	5.1	6.592	24.0	260.7	1,511,966.79
a280-51-c1-w45-1500-2000	2,651	4.8	0.000	0.4	0.6	403,913.88
a280-61-c1-w45-1500-2000	14,059	5.0	0.000	1.8	2.3	504,191.41
a280-71-c1-w45-1500-2000	52,021	5.7	0.014	10.5	13.8	605,257.19
a280-81-c1-w45-1500-2000	53,779	5.5	0.003	7.4	11.6	706,058.30
a280-91-c1-w45-1500-2000	249,801	6.5	0.009	44.2	70.9	707,448.54
a280-101-c1-w45-1500-2000	180,234	5.6	0.074	34.4	72.9	710,046.75
a280-111-c1-w45-1500-2000	1,492,775	7.4	0.042	344.9	753.1	809,244.27
a280-121-c1-w45-1500-2000	3,658,829	7.5	0.114	1,440.0	6,905.4	709,387.20
a280-131-c1-w45-1500-2000	3,396,830	6.9	0.060	1,586.8	7,200.0	**909,660.93**
a280-141-c1-w45-1500-2000	2,183,015	6.3	0.069	603.5	7,200.0	**910,699.03**
a280-151-c1-w45-1500-2000	2,921,295	6.7	0.032	946.3	2,305.0	1,011,672.88
brd14051-51-c12-w45-3000-4000	113	2.4	0.000	0.0	0.0	1,023,786.44

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-61-c1-w45-3000-4000	137	2.2	0.001	0.0	0.1	1,334,094.14
brd14051-71-c1-w45-3000-4000	231	2.6	0.000	0.0	0.1	1,433,071.08
brd14051-81-c1-w45-3000-4000	615	3.2	0.000	0.1	0.1	1,539,880.28
brd14051-91-c1-w45-3000-4000	579	3.2	0.000	0.1	0.1	1,744,132.10
brd14051-101-c1-w45-3000-4000	1,239	3.5	0.000	0.1	0.2	1,950,436.02
brd14051-111-c1-w45-3000-4000	922	3.1	0.004	0.1	0.2	1,847,289.86
brd14051-121-c1-w45-3000-4000	647	3.0	0.000	0.1	0.1	2,566,423.89
brd14051-131-c1-w45-3000-4000	1,127	3.4	0.000	0.2	0.2	2,575,419.64
brd14051-141-c1-w45-3000-4000	1,764	3.2	0.004	0.2	0.3	2,564,574.49
brd14051-151-c1-w45-3000-4000	4,298	4.2	4.210	0.6	1.4	2,365,180.57
brd14051-51-c1-w60-3000-4000	152	2.7	0.000	0.0	0.0	1,126,673.51
brd14051-61-c1-w60-3000-4000	204	2.6	7.557	0.0	0.1	1,332,842.39
brd14051-71-c1-w60-3000-4000	312	3.1	0.001	0.0	0.1	1,640,124.84
brd14051-81-c1-w60-3000-4000	280	2.5	0.003	0.0	0.1	1,539,699.26
brd14051-91-c1-w60-3000-4000	402	2.6	0.211	0.0	0.1	1,644,014.27
brd14051-101-c1-w60-3000-4000	583	2.8	0.030	0.1	0.1	1,951,340.72
brd14051-111-c1-w60-3000-4000	1,173	3.6	0.053	0.1	0.2	1,848,816.44
brd14051-121-c1-w60-3000-4000	1,046	3.1	0.000	0.2	0.2	2,159,017.05
brd14051-131-c1-w60-3000-4000	1,108	3.1	0.036	0.1	0.2	2,371,003.09
brd14051-141-c1-w60-3000-4000	2,003	3.4	0.002	0.3	0.4	2,058,945.67
brd14051-151-c1-w60-3000-4000	7,055	4.1	0.001	1.0	1.3	2,160,333.38
brd14051-51-c1-w75-3000-4000	157	2.7	0.000	0.0	0.0	1,023,309.53
brd14051-61-c1-w75-3000-4000	273	2.8	0.000	0.0	0.1	1,228,441.43
brd14051-71-c1-w75-3000-4000	580	3.2	0.000	0.1	0.1	1,336,372.33
brd14051-81-c1-w75-3000-4000	372	2.6	0.060	0.0	0.1	1,539,578.33
brd14051-91-c1-w75-3000-4000	749	3.1	0.007	0.1	0.1	1,951,567.30
brd14051-101-c1-w75-3000-4000	1,204	3.5	5.683	0.1	0.2	1,745,372.97
brd14051-111-c1-w75-3000-4000	2,420	4.0	0.000	0.3	0.4	1,645,096.90
brd14051-121-c1-w75-3000-4000	5,709	4.7	0.050	0.9	1.3	1,854,638.50
brd14051-131-c1-w75-3000-4000	1,850	3.4	0.124	0.2	0.4	2,064,768.47
brd14051-141-c1-w75-3000-4000	4,256	4.0	0.077	0.5	0.7	2,164,067.00
brd14051-151-c1-w75-3000-4000	5,824	4.0	4.407	0.9	2.9	2,266,762.96
brd14051-51-c1-w90-3000-4000	93	2.3	0.089	0.0	0.0	1,025,016.53
brd14051-61-c1-w90-3000-4000	218	2.6	0.195	0.0	0.0	1,029,009.71
brd14051-71-c1-w90-3000-4000	381	2.8	0.050	0.0	0.1	1,332,066.84
brd14051-81-c1-w90-3000-4000	643	2.9	0.000	0.1	0.1	1,439,958.79
brd14051-91-c1-w90-3000-4000	1,050	3.6	0.000	0.1	0.2	1,544,458.72

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-101-c1-w90-3000-4000	1,084	3.3	0.000	0.1	0.2	1,748,150.30
brd14051-111-c1-w90-3000-4000	1,962	3.6	0.017	0.2	0.3	1,647,855.40
brd14051-121-c1-w90-3000-4000	3,323	4.0	0.004	0.4	0.6	1,854,367.88
brd14051-131-c1-w90-3000-4000	3,562	3.9	0.167	0.5	0.7	1,859,956.42
brd14051-141-c1-w90-3000-4000	3,822	3.9	0.000	0.5	0.6	1,951,854.89
brd14051-151-c1-w90-3000-4000	23,875	5.3	0.001	4.2	5.2	2,158,254.77
brd14051-51-c1-w120-3000-4000	318	3.0	0.000	0.0	0.1	1,024,561.29
brd14051-61-c1-w120-3000-4000	199	2.4	0.123	0.0	0.0	1,334,521.00
brd14051-71-c1-w120-3000-4000	623	3.2	0.012	0.1	0.1	1,332,700.91
brd14051-81-c1-w120-3000-4000	931	3.3	0.000	0.1	0.2	1,232,889.81
brd14051-91-c1-w120-3000-4000	1,164	3.5	0.000	0.1	0.2	1,542,405.06
brd14051-101-c1-w120-3000-4000	1,938	3.7	0.037	0.3	0.4	1,748,508.45
brd14051-111-c1-w120-3000-4000	4,203	4.3	0.059	0.9	2.7	1,541,632.88
brd14051-121-c1-w120-3000-4000	2,583	3.4	0.005	0.4	0.5	1,646,981.59
brd14051-131-c1-w120-3000-4000	1,342	3.0	0.019	0.2	0.3	2,062,324.71
brd14051-141-c1-w120-3000-4000	11,064	4.7	0.016	2.1	2.7	1,858,210.29
brd14051-151-c1-w120-3000-4000	35,365	5.7	0.008	10.7	12.9	1,958,173.15
d18512-51-c1-w45-3000-4000	118	2.3	0.000	0.1	0.1	1,126,858.36
d18512-61-c1-w45-3000-4000	181	2.6	0.000	0.0	0.0	1,435,888.50
d18512-71-c1-w45-3000-4000	568	3.4	0.003	0.1	0.1	1,332,279.00
d18512-81-c1-w45-3000-4000	148	2.0	0.001	0.0	0.0	2,050,363.62
d18512-91-c1-w45-3000-4000	1,074	3.5	0.015	0.1	0.2	1,747,062.75
d18512-101-c1-w45-3000-4000	968	3.3	0.001	0.1	0.2	2,152,190.62
d18512-111-c1-w45-3000-4000	656	2.9	0.001	0.1	0.1	2,262,287.94
d18512-121-c1-w45-3000-4000	678	3.0	0.000	0.1	0.1	2,465,908.44
d18512-131-c1-w45-3000-4000	2,126	3.5	0.000	0.3	0.4	2,060,430.82
d18512-141-c1-w45-3000-4000	1,812	3.6	0.000	0.3	0.3	2,361,396.18
d18512-151-c1-w45-3000-4000	2,809	3.5	0.000	0.4	0.5	2,471,700.76
d18512-51-c1-w60-3000-4000	84	1.9	0.082	0.0	0.0	1,129,250.38
d18512-61-c1-w60-3000-4000	264	2.8	0.070	0.0	0.1	1,131,454.07
d18512-71-c1-w60-3000-4000	170	2.3	0.000	0.0	0.0	1,434,563.14
d18512-81-c1-w60-3000-4000	491	2.9	0.000	0.1	0.1	1,539,518.62
d18512-91-c1-w60-3000-4000	568	3.3	0.037	0.1	0.2	1,743,920.19
d18512-101-c1-w60-3000-4000	1,482	3.7	0.006	0.2	0.2	1,846,521.53
d18512-111-c1-w60-3000-4000	2,334	4.1	0.008	0.3	0.4	1,749,428.07
d18512-121-c1-w60-3000-4000	1,794	3.7	0.070	0.2	0.3	2,371,870.71
d18512-131-c1-w60-3000-4000	2,075	3.5	0.041	0.3	0.8	2,364,580.03

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-141-c1-w60-3000-4000	2,316	3.5	4.266	0.3	0.6	2,359,359.53
d18512-151-c1-w60-3000-4000	2,461	3.5	0.023	0.3	0.5	2,679,551.07
d18512-51-c1-w75-3000-4000	78	1.9	0.000	0.0	0.0	1,230,296.89
d18512-61-c1-w75-3000-4000	179	2.5	0.025	0.0	0.1	1,334,715.19
d18512-71-c1-w75-3000-4000	265	2.6	0.000	0.0	0.1	1,435,567.84
d18512-81-c1-w75-3000-4000	418	2.9	6.222	0.1	0.1	1,641,246.51
d18512-91-c1-w75-3000-4000	806	3.4	0.046	0.1	0.2	1,540,257.67
d18512-101-c1-w75-3000-4000	1,186	3.3	0.032	0.1	0.2	1,641,701.04
d18512-111-c1-w75-3000-4000	1,009	3.0	4.908	0.1	0.3	2,053,482.38
d18512-121-c1-w75-3000-4000	695	2.7	0.000	0.1	0.2	2,162,585.37
d18512-131-c1-w75-3000-4000	4,218	4.6	0.000	0.6	0.7	1,952,661.24
d18512-141-c1-w75-3000-4000	3,534	3.8	0.002	0.5	0.7	2,261,956.75
d18512-151-c1-w75-3000-4000	3,985	4.1	0.003	0.6	0.8	2,574,655.65
d18512-51-c1-w90-3000-4000	216	2.7	0.000	0.0	0.0	1,025,581.53
d18512-61-c1-w90-3000-4000	450	3.1	8.922	0.0	0.1	1,130,604.48
d18512-71-c1-w90-3000-4000	602	3.2	0.133	0.1	0.1	1,436,412.59
d18512-81-c1-w90-3000-4000	240	2.5	5.774	0.0	0.1	1,742,525.68
d18512-91-c1-w90-3000-4000	767	3.4	5.681	0.1	0.2	1,745,045.57
d18512-101-c1-w90-3000-4000	1,822	3.8	0.000	0.2	0.3	1,543,572.23
d18512-111-c1-w90-3000-4000	1,501	3.4	0.001	0.2	0.3	1,850,903.07
d18512-121-c1-w90-3000-4000	1,044	3.1	0.009	0.1	0.2	1,958,561.31
d18512-131-c1-w90-3000-4000	1,850	3.5	0.008	0.3	0.4	2,258,833.07
d18512-141-c1-w90-3000-4000	3,919	4.1	4.600	0.7	1.1	2,155,876.66
d18512-151-c1-w90-3000-4000	9,061	4.9	0.033	1.8	3.8	2,572,549.35
d18512-51-c1-w120-3000-4000	169	2.6	0.001	0.0	0.1	1,026,795.72
d18512-61-c1-w120-3000-4000	467	3.2	0.000	0.0	0.1	1,232,042.19
d18512-71-c1-w120-3000-4000	833	3.4	8.155	0.1	0.2	1,232,314.41
d18512-81-c1-w120-3000-4000	506	3.2	0.000	0.1	0.1	1,334,739.37
d18512-91-c1-w120-3000-4000	944	3.3	0.031	0.1	0.2	1,440,439.28
d18512-101-c1-w120-3000-4000	4,399	4.4	0.004	0.8	1.0	1,543,494.16
d18512-111-c1-w120-3000-4000	3,317	3.8	0.052	0.6	0.8	1,647,514.93
d18512-121-c1-w120-3000-4000	1,689	3.3	0.040	0.3	0.5	1,856,066.43
d18512-131-c1-w120-3000-4000	11,215	5.0	0.044	2.5	3.7	1,853,802.49
d18512-141-c1-w120-3000-4000	13,601	4.9	4.565	2.8	5.1	2,154,684.74
d18512-151-c1-w120-3000-4000	11,525	4.7	0.033	2.8	4.1	1,961,847.13
fnl4461-51-c12-w45-3000-4000	607	3.2	0.000	0.0	0.1	713,414.73
fnl4461-61-c1-w45-3000-4000	645	3.2	0.000	0.1	0.1	816,600.24

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-71-c1-w45-3000-4000	1,265	3.6	0.000	0.1	0.2	918,673.81
fnl4461-81-c1-w45-3000-4000	1,236	3.4	0.000	0.1	0.2	1,122,721.45
fnl4461-91-c1-w45-3000-4000	2,301	3.4	0.000	0.2	0.3	1,226,120.82
fnl4461-101-c1-w45-3000-4000	3,075	3.5	0.008	0.3	0.5	1,229,960.80
fnl4461-111-c1-w45-3000-4000	4,627	3.7	0.025	0.4	0.9	1,331,168.06
fnl4461-121-c1-w45-3000-4000	6,913	3.8	0.035	0.7	2.0	1,331,779.58
fnl4461-131-c1-w45-3000-4000	4,244	3.6	0.012	0.5	1.2	1,839,540.02
fnl4461-141-c1-w45-3000-4000	9,752	3.9	6.047	1.0	1.9	1,637,696.11
fnl4461-151-c1-w45-3000-4000	9,845	4.3	0.086	1.1	3.3	1,846,050.07
fnl4461-51-c1-w60-3000-4000	464	3.2	0.010	0.0	0.1	814,616.60
fnl4461-61-c1-w60-3000-4000	546	3.1	0.000	0.1	0.1	917,149.85
fnl4461-71-c1-w60-3000-4000	1,365	3.5	0.014	0.1	0.2	919,358.07
fnl4461-81-c1-w60-3000-4000	3,401	4.0	0.036	0.3	0.7	1,021,010.03
fnl4461-91-c1-w60-3000-4000	3,520	4.0	0.000	0.3	0.4	1,225,208.57
fnl4461-101-c1-w60-3000-4000	4,281	3.7	0.025	0.4	0.9	1,229,828.86
fnl4461-111-c1-w60-3000-4000	7,977	3.9	7.446	0.8	2.7	1,329,600.30
fnl4461-121-c1-w60-3000-4000	8,737	4.0	0.052	0.9	1.8	1,332,109.08
fnl4461-131-c1-w60-3000-4000	9,815	3.8	0.028	1.3	3.2	1,540,478.85
fnl4461-141-c1-w60-3000-4000	13,272	4.1	0.027	1.6	3.7	1,537,306.09
fnl4461-151-c1-w60-3000-4000	9,792	3.9	0.013	1.4	1.9	1,845,025.92
fnl4461-51-c1-w75-3000-4000	457	3.0	0.047	0.0	0.1	714,602.86
fnl4461-61-c1-w75-3000-4000	1,159	3.7	0.000	0.1	0.1	915,861.57
fnl4461-71-c1-w75-3000-4000	1,364	3.4	0.011	0.1	0.2	919,272.84
fnl4461-81-c1-w75-3000-4000	2,187	3.7	0.016	0.2	0.3	1,123,128.72
fnl4461-91-c1-w75-3000-4000	4,865	4.2	0.023	0.4	1.0	1,022,908.16
fnl4461-101-c1-w75-3000-4000	4,047	3.6	7.988	0.5	1.1	1,226,103.49
fnl4461-111-c1-w75-3000-4000	11,338	4.5	0.046	1.2	3.8	1,128,019.23
fnl4461-121-c1-w75-3000-4000	16,262	4.4	8.082	2.0	10.7	1,228,438.14
fnl4461-131-c1-w75-3000-4000	9,699	4.0	0.069	1.1	3.4	1,536,084.54
fnl4461-141-c1-w75-3000-4000	23,374	4.3	0.021	3.2	5.7	1,436,287.81
fnl4461-151-c1-w75-3000-4000	15,372	3.9	0.066	1.9	6.3	1,639,996.08
fnl4461-51-c1-w90-3000-4000	384	3.0	0.000	0.0	0.1	614,203.13
fnl4461-61-c1-w90-3000-4000	1,842	3.9	0.043	0.2	0.3	815,726.75
fnl4461-71-c1-w90-3000-4000	1,231	3.3	10.751	0.2	0.3	918,433.76
fnl4461-81-c1-w90-3000-4000	5,832	4.2	0.010	0.7	0.9	820,638.98
fnl4461-91-c1-w90-3000-4000	3,811	4.2	0.044	0.4	1.0	1,121,634.78
fnl4461-101-c1-w90-3000-4000	4,440	3.8	0.057	0.5	2.3	1,125,259.60

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-111-c1-w90-3000-4000	6,519	3.8	0.046	0.9	2.8	1,227,532.86
fnl4461-121-c1-w90-3000-4000	12,452	4.4	0.126	1.9	5.0	1,333,519.42
fnl4461-131-c1-w90-3000-4000	13,934	4.3	0.022	1.8	3.3	1,334,315.21
fnl4461-141-c1-w90-3000-4000	17,715	4.2	0.077	2.6	9.4	1,538,061.18
fnl4461-151-c1-w90-3000-4000	36,652	4.5	0.174	5.9	20.2	1,442,087.22
fnl4461-51-c1-w120-3000-4000	1,119	3.6	0.000	0.1	0.2	612,542.60
fnl4461-61-c1-w120-3000-4000	2,542	4.0	0.036	0.3	0.4	714,771.89
fnl4461-71-c1-w120-3000-4000	7,077	4.5	0.294	0.9	2.0	718,312.49
fnl4461-81-c1-w120-3000-4000	13,442	4.7	0.025	2.2	2.8	820,566.22
fnl4461-91-c1-w120-3000-4000	7,391	4.4	0.020	0.7	1.6	920,467.48
fnl4461-101-c1-w120-3000-4000	21,487	5.0	0.038	3.2	4.9	1,023,380.69
fnl4461-111-c1-w120-3000-4000	10,607	4.1	0.042	1.4	2.0	1,130,615.38
fnl4461-121-c1-w120-3000-4000	15,832	4.2	0.042	2.1	3.9	1,229,696.18
fnl4461-131-c1-w120-3000-4000	21,260	4.2	0.055	3.2	7.3	1,234,133.03
fnl4461-141-c1-w120-3000-4000	47,087	4.9	6.874	8.6	28.9	1,434,660.58
fnl4461-151-c1-w120-3000-4000	32,488	4.4	6.372	5.6	17.1	1,537,555.86
nrw1379-51-c1-w45-3000-4000	299	2.8	0.000	0.1	0.1	917,274.64
nrw1379-61-c1-w45-3000-4000	463	3.2	0.000	0.0	0.1	1,022,425.70
nrw1379-71-c1-w45-3000-4000	1,015	3.4	0.002	0.1	0.2	1,222,474.94
nrw1379-81-c1-w45-3000-4000	1,699	3.8	0.000	0.2	0.2	1,124,093.92
nrw1379-91-c1-w45-3000-4000	1,162	3.5	0.000	0.1	0.2	1,326,649.77
nrw1379-101-c1-w45-3000-4000	2,405	3.6	0.047	0.3	1.0	1,331,620.61
nrw1379-111-c1-w45-3000-4000	2,335	3.6	0.000	0.3	0.3	1,738,815.59
nrw1379-121-c1-w45-3000-4000	2,898	3.4	0.123	0.3	0.7	1,538,184.52
nrw1379-131-c1-w45-3000-4000	6,471	3.8	0.026	0.8	1.1	1,739,363.91
nrw1379-141-c1-w45-3000-4000	7,776	4.2	0.001	0.8	1.2	1,947,369.66
nrw1379-151-c1-w45-3000-4000	7,791	3.7	0.000	0.9	1.2	1,844,310.59
nrw1379-51-c1-w60-3000-4000	513	3.3	0.000	0.0	0.1	716,125.76
nrw1379-61-c1-w60-3000-4000	476	2.9	0.000	0.0	0.1	1,221,290.03
nrw1379-71-c1-w60-3000-4000	1,579	3.8	0.000	0.1	0.2	1,020,784.41
nrw1379-81-c1-w60-3000-4000	1,350	3.5	0.013	0.1	0.2	1,431,303.31
nrw1379-91-c1-w60-3000-4000	1,286	3.4	0.000	0.2	0.2	1,427,543.21
nrw1379-101-c1-w60-3000-4000	2,746	3.8	0.029	0.3	0.4	1,431,001.16
nrw1379-111-c1-w60-3000-4000	2,213	3.6	0.031	0.2	0.5	1,534,724.99
nrw1379-121-c1-w60-3000-4000	3,641	3.5	0.026	0.4	0.6	1,637,462.86
nrw1379-131-c1-w60-3000-4000	5,784	3.7	5.684	0.7	1.1	1,737,360.55
nrw1379-141-c1-w60-3000-4000	7,697	4.0	0.054	0.9	1.9	1,742,631.42

Continued Table B.3 – *Two stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-151-c1-w60-3000-4000	9,479	3.9	4.958	1.2	2.7	1,945,155.19
nrw1379-51-c1-w75-3000-4000	812	3.9	0.004	0.1	0.1	719,045.87
nrw1379-61-c1-w75-3000-4000	1,048	3.9	0.000	0.1	0.1	1,021,850.51
nrw1379-71-c1-w75-3000-4000	1,722	3.8	0.000	0.2	0.2	1,120,635.35
nrw1379-81-c1-w75-3000-4000	1,256	3.7	0.006	0.1	0.2	1,326,073.38
nrw1379-91-c1-w75-3000-4000	3,262	4.1	0.052	0.4	0.6	1,327,419.28
nrw1379-101-c1-w75-3000-4000	2,172	3.3	0.027	0.2	0.4	1,532,106.39
nrw1379-111-c1-w75-3000-4000	3,160	3.7	0.001	0.4	0.5	1,536,262.84
nrw1379-121-c1-w75-3000-4000	4,237	3.7	0.010	0.5	0.7	1,634,632.39
nrw1379-131-c1-w75-3000-4000	13,029	4.3	0.011	1.8	2.5	1,636,521.98
nrw1379-141-c1-w75-3000-4000	13,740	4.4	0.021	1.8	3.1	1,844,367.89
nrw1379-151-c1-w75-3000-4000	14,060	4.3	5.332	1.9	7.3	1,842,168.38
nrw1379-51-c1-w90-3000-4000	554	3.3	0.018	0.0	0.1	915,646.51
nrw1379-61-c1-w90-3000-4000	1,151	3.8	0.046	0.1	0.2	1,022,166.00
nrw1379-71-c1-w90-3000-4000	1,904	3.9	0.000	0.2	0.3	918,710.46
nrw1379-81-c1-w90-3000-4000	2,015	3.7	0.000	0.3	0.3	1,126,199.60
nrw1379-91-c1-w90-3000-4000	3,935	4.0	0.000	0.4	0.5	1,227,908.61
nrw1379-101-c1-w90-3000-4000	5,898	4.0	7.478	0.7	1.9	1,328,254.51
nrw1379-111-c1-w90-3000-4000	4,400	3.7	0.063	0.5	1.1	1,431,277.63
nrw1379-121-c1-w90-3000-4000	4,864	3.7	0.046	0.6	1.9	1,434,576.05
nrw1379-131-c1-w90-3000-4000	15,999	4.5	0.048	2.4	5.6	1,537,884.66
nrw1379-141-c1-w90-3000-4000	14,023	4.5	0.015	1.6	4.1	1,639,629.13
nrw1379-151-c1-w90-3000-4000	23,950	4.5	0.011	3.4	6.4	1,741,329.37
nrw1379-51-c1-w120-3000-4000	947	3.7	0.000	0.1	0.2	713,518.39
nrw1379-61-c1-w120-3000-4000	1,811	4.1	0.000	0.2	0.2	820,396.80
nrw1379-71-c1-w120-3000-4000	5,856	4.8	0.000	0.9	1.1	1,023,671.52
nrw1379-81-c1-w120-3000-4000	2,066	3.6	0.025	0.3	0.4	1,124,427.25
nrw1379-91-c1-w120-3000-4000	8,237	4.3	0.031	1.0	1.4	1,126,317.57
nrw1379-101-c1-w120-3000-4000	7,720	4.3	0.166	1.0	6.1	1,229,454.40
nrw1379-111-c1-w120-3000-4000	8,625	4.3	0.040	1.0	1.9	1,331,060.69
nrw1379-121-c1-w120-3000-4000	12,204	4.3	0.001	1.7	2.2	1,330,192.96
nrw1379-131-c1-w120-3000-4000	14,747	4.3	0.024	2.3	3.2	1,435,845.10
nrw1379-141-c1-w120-3000-4000	22,800	4.6	0.120	3.9	9.1	1,544,681.67
nrw1379-151-c1-w120-3000-4000	32,044	4.5	0.056	4.6	9.3	1,540,809.49

Table B.4. Two Stacks Vehicle of C2 Class

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-51-c2-w15-500-1000	351	2.7	0.016	0.0	0.2	804,505.75
a280-61-c2-w15-500-1000	547	2.8	0.000	0.1	0.1	905,317.48
a280-71-c2-w15-500-1000	984	3.0	0.000	0.1	0.1	1,006,319.61
a280-81-c2-w15-500-1000	765	2.8	0.007	0.1	0.1	1,307,680.63
a280-91-c2-w15-500-1000	1,298	2.9	0.009	0.1	0.2	1,308,468.87
a280-101-c2-w15-500-1000	796	2.6	0.013	0.1	0.2	1,611,816.31
a280-111-c2-w15-500-1000	1,436	2.9	0.001	0.2	0.3	1,812,795.65
a280-121-c2-w15-500-1000	3,208	3.1	0.007	0.3	0.5	1,712,383.39
a280-131-c2-w15-500-1000	4,221	3.2	0.027	0.5	1.1	1,813,248.46
a280-141-c2-w15-500-1000	3,795	3.1	0.004	0.4	0.8	2,014,118.29
a280-151-c2-w15-500-1000	2,907	2.8	0.001	0.4	0.5	2,215,601.60
a280-51-c2-w15-1000-1200	272	2.7	0.001	0.0	0.1	904,616.44
a280-61-c2-w15-1000-1200	690	3.3	0.000	0.1	0.1	905,304.99
a280-71-c2-w15-1000-1200	1,312	3.8	0.003	0.1	0.2	1,005,844.73
a280-81-c2-w15-1000-1200	1,082	3.0	0.012	0.1	0.2	1,208,065.26
a280-91-c2-w15-1000-1200	1,737	3.4	0.016	0.2	0.4	1,209,278.75
a280-101-c2-w15-1000-1200	3,385	3.5	0.016	0.3	0.9	1,310,248.25
a280-111-c2-w15-1000-1200	4,742	3.8	0.009	0.5	1.6	1,311,125.72
a280-121-c2-w15-1000-1200	11,288	4.3	0.003	1.2	2.2	1,209,964.92
a280-131-c2-w15-1000-1200	10,769	3.8	0.022	1.3	4.0	1,512,876.87
a280-141-c2-w15-1000-1200	6,342	3.6	0.003	0.7	1.6	1,913,701.55
a280-151-c2-w15-1000-1200	9,823	3.9	0.047	1.3	12.7	1,715,643.50
a280-51-c2-w15-1500-2000	3,359	4.4	0.009	0.3	0.5	603,995.21
a280-61-c2-w15-1500-2000	2,787	4.1	0.050	0.3	0.4	605,569.68
a280-71-c2-w15-1500-2000	6,680	4.1	0.003	0.7	1.0	805,775.04
a280-81-c2-w15-1500-2000	4,994	3.9	0.000	0.5	0.7	906,453.57
a280-91-c2-w15-1500-2000	18,483	4.7	0.021	2.0	5.4	908,308.28
a280-101-c2-w15-1500-2000	23,309	4.5	0.044	2.7	8.1	909,507.21
a280-111-c2-w15-1500-2000	26,995	4.2	0.028	3.5	9.6	1,211,176.62
a280-121-c2-w15-1500-2000	70,492	5.0	8.188	8.7	20.2	1,210,361.65
a280-131-c2-w15-1500-2000	123,292	5.3	0.013	17.5	30.3	1,311,113.43
a280-141-c2-w15-1500-2000	57,699	4.5	0.037	8.1	32.6	1,312,690.77
a280-151-c2-w15-1500-2000	148,616	4.9	0.023	21.4	41.1	1,413,724.29
a280-51-c2-w30-500-1000	449	2.8	0.000	0.2	0.4	704,319.20
a280-61-c2-w30-500-1000	1,430	3.5	0.018	0.1	0.2	905,161.88
a280-71-c2-w30-500-1000	1,434	3.2	0.000	0.2	0.2	1,106,008.14

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-81-c2-w30-500-1000	2,696	3.3	0.000	0.3	0.4	1,107,899.36
a280-91-c2-w30-500-1000	1,615	3.0	0.029	0.2	0.3	1,309,578.83
a280-101-c2-w30-500-1000	2,024	3.0	0.002	0.2	0.3	1,410,051.58
a280-111-c2-w30-500-1000	5,007	3.6	0.012	0.7	1.9	1,510,445.38
a280-121-c2-w30-500-1000	6,448	3.5	6.168	0.9	4.1	1,611,447.41
a280-131-c2-w30-500-1000	8,573	3.4	0.026	1.2	2.9	1,613,265.56
a280-141-c2-w30-500-1000	10,128	3.6	0.018	1.3	4.1	1,713,293.67
a280-151-c2-w30-500-1000	12,958	3.6	0.023	1.6	6.3	1,713,377.70
a280-51-c2-w30-1000-1200	253	2.8	0.001	0.0	0.1	704,323.82
a280-61-c2-w30-1000-1200	1,134	3.6	0.017	0.1	0.2	704,686.65
a280-71-c2-w30-1000-1200	2,418	3.8	0.009	0.3	0.4	906,094.32
a280-81-c2-w30-1000-1200	1,453	3.2	0.037	0.1	0.2	1,008,097.97
a280-91-c2-w30-1000-1200	3,475	3.7	0.005	0.4	0.6	1,208,357.02
a280-101-c2-w30-1000-1200	4,908	3.7	0.000	0.5	0.7	1,210,281.85
a280-111-c2-w30-1000-1200	8,995	4.4	0.017	0.9	3.1	1,311,047.57
a280-121-c2-w30-1000-1200	9,221	3.9	0.022	1.3	3.7	1,512,233.67
a280-131-c2-w30-1000-1200	31,937	4.7	0.011	4.0	12.9	1,310,716.89
a280-141-c2-w30-1000-1200	22,691	4.2	0.001	3.0	4.4	1,512,916.34
a280-151-c2-w30-1000-1200	22,762	4.2	0.014	3.1	8.8	1,613,265.18
a280-51-c2-w30-1500-2000	2,552	4.1	0.001	0.3	0.4	503,914.39
a280-61-c2-w30-1500-2000	13,828	4.9	0.000	2.0	2.5	505,504.19
a280-71-c2-w30-1500-2000	5,380	3.9	0.018	0.6	0.8	805,654.39
a280-81-c2-w30-1500-2000	11,344	4.8	0.015	1.3	1.9	707,568.45
a280-91-c2-w30-1500-2000	52,403	5.2	0.017	6.0	15.0	807,306.80
a280-101-c2-w30-1500-2000	56,729	5.0	0.026	7.4	16.9	909,324.18
a280-111-c2-w30-1500-2000	62,159	4.9	0.036	7.7	17.9	910,182.49
a280-121-c2-w30-1500-2000	118,954	5.1	0.008	18.6	31.4	1,110,197.52
a280-131-c2-w30-1500-2000	249,580	5.2	0.031	40.5	93.0	1,111,854.38
a280-141-c2-w30-1500-2000	308,254	5.2	0.020	53.9	131.5	1,213,812.88
a280-151-c2-w30-1500-2000	311,100	5.3	0.030	55.2	141.4	1,313,064.12
a280-51-c2-w45-500-1000	842	3.0	0.003	0.1	0.2	703,927.15
a280-61-c2-w45-500-1000	4,391	3.9	0.002	0.7	1.0	804,638.87
a280-71-c2-w45-500-1000	3,629	3.6	0.000	0.5	0.6	806,421.06
a280-81-c2-w45-500-1000	3,531	3.4	0.020	0.4	1.4	1,007,185.67
a280-91-c2-w45-500-1000	5,893	3.9	0.045	0.9	3.1	1,007,875.32
a280-101-c2-w45-500-1000	10,465	4.1	0.012	1.8	3.1	1,309,506.18
a280-111-c2-w45-500-1000	10,626	3.7	0.009	1.5	3.7	1,410,225.86

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-121-c2-w45-500-1000	27,356	4.0	7.064	4.2	15.8	1,410,218.35
a280-131-c2-w45-500-1000	21,328	4.1	0.017	3.7	11.0	1,511,177.77
a280-141-c2-w45-500-1000	15,015	3.6	0.003	2.4	3.4	1,713,801.31
a280-151-c2-w45-500-1000	41,124	4.1	0.028	7.0	20.9	1,713,536.45
a280-51-c2-w45-1000-1200	1,580	3.7	0.000	0.2	0.3	703,894.44
a280-61-c2-w45-1000-1200	7,817	5.0	0.006	0.9	1.4	704,350.19
a280-71-c2-w45-1000-1200	6,467	4.5	0.035	0.9	1.4	705,808.85
a280-81-c2-w45-1000-1200	11,538	4.7	0.029	1.4	2.7	806,929.37
a280-91-c2-w45-1000-1200	17,830	5.2	0.009	2.8	5.0	1,007,972.95
a280-101-c2-w45-1000-1200	11,807	4.2	0.016	1.6	4.3	1,209,617.99
a280-111-c2-w45-1000-1200	32,860	4.6	0.043	4.6	44.0	1,110,248.08
a280-121-c2-w45-1000-1200	40,731	4.7	0.041	6.8	18.6	1,210,906.98
a280-131-c2-w45-1000-1200	85,765	5.0	0.085	14.3	77.6	1,212,083.17
a280-141-c2-w45-1000-1200	48,351	4.4	7.064	7.7	107.2	1,411,713.50
a280-151-c2-w45-1000-1200	110,900	5.0	0.009	18.5	32.1	1,412,256.71
a280-51-c2-w45-1500-2000	5,050	4.8	0.025	0.7	1.2	504,154.93
a280-61-c2-w45-1500-2000	8,281	4.5	0.017	0.9	1.7	704,677.52
a280-71-c2-w45-1500-2000	24,477	4.9	0.005	3.2	4.7	705,136.45
a280-81-c2-w45-1500-2000	43,197	5.3	0.024	5.1	11.9	706,827.73
a280-91-c2-w45-1500-2000	37,661	5.1	0.001	5.4	7.2	809,324.84
a280-101-c2-w45-1500-2000	59,888	5.0	0.027	9.2	19.9	908,896.81
a280-111-c2-w45-1500-2000	210,759	5.9	0.012	36.4	80.3	909,513.98
a280-121-c2-w45-1500-2000	575,246	6.4	0.008	120.6	222.8	1,210,352.23
a280-131-c2-w45-1500-2000	633,432	5.8	0.028	124.3	465.4	1,110,714.87
a280-141-c2-w45-1500-2000	376,911	5.3	0.021	82.3	175.9	1,312,052.36
a280-151-c2-w45-1500-2000	1,352,275	6.1	0.042	261.3	822.4	1,213,449.07
brd14051-51-c2-w45-3000-4000	105	2.3	0.056	0.0	0.0	1,531,410.64
brd14051-61-c2-w45-3000-4000	104	2.1	0.000	0.0	0.0	1,639,735.15
brd14051-71-c2-w45-3000-4000	298	2.9	0.001	0.0	0.1	1,537,543.79
brd14051-81-c2-w45-3000-4000	559	2.8	0.100	0.1	0.1	1,540,031.32
brd14051-91-c2-w45-3000-4000	341	2.5	0.075	0.1	0.1	1,955,016.84
brd14051-101-c2-w45-3000-4000	550	3.1	0.047	0.1	0.1	1,851,371.34
brd14051-111-c2-w45-3000-4000	1,150	3.5	0.000	0.1	0.2	2,052,597.24
brd14051-121-c2-w45-3000-4000	989	3.1	0.012	0.2	0.3	2,159,806.69
brd14051-131-c2-w45-3000-4000	799	2.9	0.000	0.1	0.2	2,777,927.68
brd14051-141-c2-w45-3000-4000	1,545	3.2	0.052	0.2	0.3	2,572,917.59
brd14051-151-c2-w45-3000-4000	3,864	3.9	0.009	0.5	0.8	2,771,634.10

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-51-c2-w60-3000-4000	87	2.0	0.000	0.0	0.0	1,329,050.90
brd14051-61-c2-w60-3000-4000	141	2.3	0.036	0.0	0.0	1,332,066.18
brd14051-71-c2-w60-3000-4000	394	3.3	0.000	0.1	0.1	1,438,022.37
brd14051-81-c2-w60-3000-4000	311	2.6	5.456	0.0	0.1	1,840,954.78
brd14051-91-c2-w60-3000-4000	476	2.7	0.000	0.1	0.1	1,641,814.97
brd14051-101-c2-w60-3000-4000	414	2.7	0.027	0.1	0.1	2,049,699.60
brd14051-111-c2-w60-3000-4000	1,772	3.5	0.017	0.2	0.5	1,748,146.32
brd14051-121-c2-w60-3000-4000	814	2.7	0.000	0.1	0.2	2,672,735.89
brd14051-131-c2-w60-3000-4000	462	2.3	0.004	0.1	0.2	2,573,302.22
brd14051-141-c2-w60-3000-4000	2,854	3.5	0.010	0.4	0.7	2,365,316.49
brd14051-151-c2-w60-3000-4000	3,381	3.6	0.004	0.5	0.7	2,571,220.39
brd14051-51-c2-w75-3000-4000	146	2.4	0.000	0.0	0.0	1,125,898.45
brd14051-61-c2-w75-3000-4000	159	2.2	0.004	0.0	0.1	1,437,287.81
brd14051-71-c2-w75-3000-4000	311	2.5	0.000	0.0	0.1	1,436,361.04
brd14051-81-c2-w75-3000-4000	392	2.6	0.000	0.0	0.1	1,640,266.81
brd14051-91-c2-w75-3000-4000	602	2.8	0.025	0.1	0.1	1,642,764.08
brd14051-101-c2-w75-3000-4000	1,308	3.6	0.000	0.2	0.2	1,954,257.09
brd14051-111-c2-w75-3000-4000	1,072	3.3	0.070	0.1	0.3	2,056,417.42
brd14051-121-c2-w75-3000-4000	1,115	3.5	0.003	0.2	0.2	2,160,327.67
brd14051-131-c2-w75-3000-4000	895	2.8	0.025	0.2	0.2	2,371,433.14
brd14051-141-c2-w75-3000-4000	2,366	3.5	0.033	0.3	0.8	2,157,009.39
brd14051-151-c2-w75-3000-4000	5,425	4.0	0.136	0.8	1.3	2,264,207.25
brd14051-51-c2-w90-3000-4000	148	2.4	0.000	0.0	0.0	1,227,739.73
brd14051-61-c2-w90-3000-4000	339	3.0	0.000	0.0	0.1	1,233,436.69
brd14051-71-c2-w90-3000-4000	321	2.8	0.000	0.0	0.1	1,431,667.91
brd14051-81-c2-w90-3000-4000	407	2.7	0.000	0.1	0.1	1,539,377.55
brd14051-91-c2-w90-3000-4000	1,620	3.8	0.000	0.2	0.3	1,842,455.86
brd14051-101-c2-w90-3000-4000	946	3.1	0.000	0.1	0.2	1,853,548.59
brd14051-111-c2-w90-3000-4000	2,737	3.8	0.030	0.3	0.5	1,646,219.98
brd14051-121-c2-w90-3000-4000	1,085	2.9	0.048	0.2	0.4	2,159,958.63
brd14051-131-c2-w90-3000-4000	1,142	3.0	0.232	0.2	0.3	2,376,334.44
brd14051-141-c2-w90-3000-4000	2,967	3.6	0.028	0.4	1.1	2,461,503.11
brd14051-151-c2-w90-3000-4000	4,820	3.8	0.010	0.7	0.9	2,060,144.02
brd14051-51-c2-w120-3000-4000	209	2.5	0.052	0.0	0.0	1,226,569.19
brd14051-61-c2-w120-3000-4000	162	2.2	0.039	0.0	0.0	1,233,118.91
brd14051-71-c2-w120-3000-4000	304	2.6	0.071	0.0	0.1	1,335,646.12
brd14051-81-c2-w120-3000-4000	843	3.1	0.023	0.1	0.2	1,435,156.54

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-91-c2-w120-3000-4000	459	2.7	0.000	0.1	0.1	1,645,373.15
brd14051-101-c2-w120-3000-4000	739	2.8	0.017	0.1	0.2	1,850,880.99
brd14051-111-c2-w120-3000-4000	2,210	3.4	0.079	0.3	0.9	1,643,122.11
brd14051-121-c2-w120-3000-4000	1,164	3.0	0.041	0.2	0.2	1,855,649.16
brd14051-131-c2-w120-3000-4000	1,003	2.8	0.006	0.2	0.2	2,161,617.39
brd14051-141-c2-w120-3000-4000	3,391	3.8	0.016	0.5	0.7	2,055,945.10
brd14051-151-c2-w120-3000-4000	7,005	4.2	4.644	1.6	2.4	2,156,503.74
d18512-51-c2-w45-3000-4000	92	1.9	0.000	0.0	0.1	1,335,180.49
d18512-61-c2-w45-3000-4000	136	2.4	0.000	0.0	0.0	1,434,165.75
d18512-71-c2-w45-3000-4000	236	2.6	0.082	0.0	0.1	1,538,729.58
d18512-81-c2-w45-3000-4000	335	2.6	0.033	0.1	0.1	1,438,441.92
d18512-91-c2-w45-3000-4000	337	2.5	5.457	0.1	0.1	1,847,276.34
d18512-101-c2-w45-3000-4000	1,013	3.3	0.000	0.1	0.2	1,847,941.86
d18512-111-c2-w45-3000-4000	502	2.6	0.000	0.1	0.1	2,060,958.06
d18512-121-c2-w45-3000-4000	539	2.7	0.024	0.1	0.1	2,569,659.30
d18512-131-c2-w45-3000-4000	1,368	3.4	0.004	0.2	0.3	2,365,416.14
d18512-141-c2-w45-3000-4000	1,266	3.2	3.734	0.2	0.3	2,670,087.11
d18512-151-c2-w45-3000-4000	1,057	2.8	3.834	0.2	0.3	2,568,450.88
d18512-51-c2-w60-3000-4000	105	2.4	0.000	0.0	0.0	1,334,087.60
d18512-61-c2-w60-3000-4000	206	2.5	0.059	0.0	0.0	1,231,573.25
d18512-71-c2-w60-3000-4000	286	2.6	0.057	0.0	0.1	1,434,784.42
d18512-81-c2-w60-3000-4000	296	2.4	0.000	0.0	0.1	1,540,641.86
d18512-91-c2-w60-3000-4000	431	2.6	0.042	0.0	0.1	1,848,431.86
d18512-101-c2-w60-3000-4000	695	3.1	0.003	0.1	0.2	1,951,554.07
d18512-111-c2-w60-3000-4000	771	3.1	0.000	0.1	0.2	2,161,619.55
d18512-121-c2-w60-3000-4000	974	3.0	0.000	0.1	0.2	2,470,228.73
d18512-131-c2-w60-3000-4000	2,259	3.5	0.000	0.3	0.4	2,263,992.53
d18512-141-c2-w60-3000-4000	3,950	3.8	4.413	0.6	1.2	2,262,079.50
d18512-151-c2-w60-3000-4000	2,421	3.7	0.006	0.4	0.5	2,573,354.35
d18512-51-c2-w75-3000-4000	97	2.0	0.073	0.0	0.0	1,231,920.85
d18512-61-c2-w75-3000-4000	275	2.7	0.000	0.0	0.0	1,131,867.14
d18512-71-c2-w75-3000-4000	257	2.6	0.000	0.0	0.1	1,436,176.43
d18512-81-c2-w75-3000-4000	329	2.6	0.000	0.0	0.1	1,842,955.64
d18512-91-c2-w75-3000-4000	1,076	3.4	0.012	0.1	0.2	1,638,060.36
d18512-101-c2-w75-3000-4000	1,199	3.3	0.033	0.2	0.3	1,643,809.84
d18512-111-c2-w75-3000-4000	1,130	3.4	0.022	0.1	0.3	1,856,505.11
d18512-121-c2-w75-3000-4000	728	3.0	0.036	0.1	0.2	2,267,465.08

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-131-c2-w75-3000-4000	714	2.7	0.109	0.1	0.2	2,260,568.96
d18512-141-c2-w75-3000-4000	2,735	3.4	4.067	0.4	0.5	2,462,355.65
d18512-151-c2-w75-3000-4000	2,286	3.3	0.081	0.3	0.6	2,267,619.29
d18512-51-c2-w90-3000-4000	144	2.3	0.037	0.0	0.0	1,027,666.58
d18512-61-c2-w90-3000-4000	254	2.7	0.076	0.0	0.1	1,230,407.03
d18512-71-c2-w90-3000-4000	406	2.9	0.016	0.0	0.1	1,231,286.09
d18512-81-c2-w90-3000-4000	393	2.7	0.000	0.1	0.1	1,438,432.19
d18512-91-c2-w90-3000-4000	763	3.4	0.000	0.1	0.1	1,844,768.78
d18512-101-c2-w90-3000-4000	1,189	3.4	0.007	0.2	0.2	1,645,105.50
d18512-111-c2-w90-3000-4000	697	2.8	0.017	0.1	0.1	2,054,159.53
d18512-121-c2-w90-3000-4000	1,086	3.0	4.487	0.2	0.3	2,261,567.89
d18512-131-c2-w90-3000-4000	1,426	3.2	0.000	0.2	0.3	2,158,285.22
d18512-141-c2-w90-3000-4000	3,191	3.8	0.036	0.5	1.2	2,059,391.75
d18512-151-c2-w90-3000-4000	3,578	3.7	0.033	0.6	0.8	2,671,688.00
d18512-51-c2-w120-3000-4000	98	2.0	0.046	0.0	0.0	1,130,987.30
d18512-61-c2-w120-3000-4000	187	2.5	0.000	0.0	0.0	1,133,544.38
d18512-71-c2-w120-3000-4000	428	2.7	0.047	0.0	0.1	1,332,740.16
d18512-81-c2-w120-3000-4000	832	3.2	0.003	0.1	0.2	1,335,303.13
d18512-91-c2-w120-3000-4000	670	3.1	0.000	0.1	0.1	1,541,946.33
d18512-101-c2-w120-3000-4000	1,694	3.8	0.034	0.3	0.4	1,646,728.20
d18512-111-c2-w120-3000-4000	1,074	3.0	0.078	0.2	0.3	1,753,358.30
d18512-121-c2-w120-3000-4000	832	2.8	0.016	0.1	0.2	2,268,283.26
d18512-131-c2-w120-3000-4000	3,677	3.8	0.000	0.7	0.8	1,854,150.76
d18512-141-c2-w120-3000-4000	3,409	3.6	4.631	0.5	1.5	2,158,232.58
d18512-151-c2-w120-3000-4000	5,847	3.9	0.050	1.0	2.0	2,364,479.59
fnl4461-51-c2-w45-3000-4000	144	2.3	0.036	0.0	0.0	916,175.75
fnl4461-61-c2-w45-3000-4000	422	3.0	0.000	0.1	0.1	1,219,706.37
fnl4461-71-c2-w45-3000-4000	873	3.0	0.014	0.1	0.1	1,018,739.39
fnl4461-81-c2-w45-3000-4000	2,519	3.9	0.049	0.3	0.6	1,021,741.32
fnl4461-91-c2-w45-3000-4000	3,767	3.6	0.039	0.4	1.2	1,123,735.41
fnl4461-101-c2-w45-3000-4000	2,104	3.2	0.019	0.2	0.5	1,429,373.24
fnl4461-111-c2-w45-3000-4000	2,226	3.3	0.026	0.2	0.5	1,531,580.15
fnl4461-121-c2-w45-3000-4000	3,434	3.4	0.045	0.4	1.5	1,634,708.79
fnl4461-131-c2-w45-3000-4000	4,101	3.5	0.032	0.5	1.3	1,741,088.78
fnl4461-141-c2-w45-3000-4000	5,092	3.4	0.023	0.6	1.0	1,942,579.03
fnl4461-151-c2-w45-3000-4000	7,559	3.6	0.021	0.9	1.7	1,841,308.92
fnl4461-51-c2-w60-3000-4000	319	2.9	0.000	0.0	0.1	716,979.95

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-61-c2-w60-3000-4000	556	3.0	0.003	0.0	0.1	817,360.46
fnl4461-71-c2-w60-3000-4000	1,855	3.7	0.018	0.1	0.5	1,019,522.24
fnl4461-81-c2-w60-3000-4000	1,465	3.0	0.011	0.1	0.2	1,023,363.95
fnl4461-91-c2-w60-3000-4000	3,433	3.7	0.005	0.3	0.5	1,122,368.65
fnl4461-101-c2-w60-3000-4000	4,748	3.8	0.000	0.5	0.7	1,227,529.15
fnl4461-111-c2-w60-3000-4000	3,048	3.5	0.000	0.3	0.5	1,430,500.08
fnl4461-121-c2-w60-3000-4000	4,174	3.5	0.026	0.5	1.3	1,432,345.96
fnl4461-131-c2-w60-3000-4000	4,493	3.5	0.189	0.5	1.2	1,644,564.72
fnl4461-141-c2-w60-3000-4000	6,738	3.8	0.008	0.7	1.1	1,640,162.74
fnl4461-151-c2-w60-3000-4000	13,547	3.9	0.122	1.5	7.9	1,644,140.91
fnl4461-51-c2-w75-3000-4000	601	3.4	0.000	0.1	0.1	814,584.01
fnl4461-61-c2-w75-3000-4000	639	2.9	0.037	0.1	0.1	916,909.01
fnl4461-71-c2-w75-3000-4000	887	3.1	0.001	0.1	0.1	919,610.36
fnl4461-81-c2-w75-3000-4000	1,578	3.4	0.019	0.2	0.3	1,021,032.56
fnl4461-91-c2-w75-3000-4000	2,569	3.6	0.046	0.5	1.0	1,123,468.77
fnl4461-101-c2-w75-3000-4000	1,872	3.2	0.000	0.2	0.3	1,327,240.99
fnl4461-111-c2-w75-3000-4000	3,991	3.6	0.080	0.4	1.3	1,230,246.83
fnl4461-121-c2-w75-3000-4000	5,506	3.6	0.044	0.6	1.8	1,532,829.32
fnl4461-131-c2-w75-3000-4000	9,987	4.1	0.032	1.1	3.6	1,535,333.78
fnl4461-141-c2-w75-3000-4000	5,707	3.4	5.608	0.7	1.8	1,739,059.42
fnl4461-151-c2-w75-3000-4000	13,471	4.0	0.072	1.6	10.0	1,743,936.98
fnl4461-51-c2-w90-3000-4000	470	3.4	0.000	0.0	0.1	813,822.37
fnl4461-61-c2-w90-3000-4000	785	3.5	0.009	0.1	0.2	916,416.04
fnl4461-71-c2-w90-3000-4000	1,646	3.5	9.681	0.2	0.3	1,016,343.65
fnl4461-81-c2-w90-3000-4000	1,137	3.4	0.029	0.1	0.3	1,022,016.59
fnl4461-91-c2-w90-3000-4000	3,714	3.9	0.087	0.4	0.9	1,026,535.00
fnl4461-101-c2-w90-3000-4000	3,259	3.3	0.018	0.4	0.6	1,326,283.47
fnl4461-111-c2-w90-3000-4000	7,473	3.8	0.052	0.8	1.5	1,332,454.81
fnl4461-121-c2-w90-3000-4000	5,953	3.7	0.000	0.7	0.9	1,532,455.95
fnl4461-131-c2-w90-3000-4000	14,326	4.1	0.064	1.7	5.2	1,332,941.14
fnl4461-141-c2-w90-3000-4000	17,084	4.4	0.049	2.1	5.8	1,537,556.51
fnl4461-151-c2-w90-3000-4000	22,932	4.2	0.150	2.8	21.5	1,643,253.38
fnl4461-51-c2-w120-3000-4000	813	3.2	0.000	0.1	0.2	611,962.38
fnl4461-61-c2-w120-3000-4000	1,360	3.4	0.034	0.1	0.2	716,421.41
fnl4461-71-c2-w120-3000-4000	7,274	4.8	0.050	1.0	1.4	716,215.42
fnl4461-81-c2-w120-3000-4000	6,573	4.3	10.687	0.7	3.1	920,071.49
fnl4461-91-c2-w120-3000-4000	10,727	4.2	0.291	1.4	7.0	923,358.75

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-101-c2-w120-3000-4000	5,791	3.7	0.028	0.7	1.4	1,227,427.95
fnl4461-111-c2-w120-3000-4000	13,885	4.2	0.147	1.8	5.0	1,129,857.32
fnl4461-121-c2-w120-3000-4000	16,286	4.3	0.016	2.5	3.9	1,330,946.90
fnl4461-131-c2-w120-3000-4000	22,894	4.4	7.370	2.8	7.6	1,332,993.05
fnl4461-141-c2-w120-3000-4000	35,213	4.5	6.384	5.7	23.8	1,536,715.96
fnl4461-151-c2-w120-3000-4000	35,871	4.4	0.041	5.4	9.7	1,538,153.76
nrw1379-51-c2-w45-3000-4000	325	3.0	0.000	0.1	0.1	816,666.70
nrw1379-61-c2-w45-3000-4000	425	2.9	0.000	0.1	0.1	1,018,847.73
nrw1379-71-c2-w45-3000-4000	861	3.4	0.046	0.1	0.1	1,222,895.64
nrw1379-81-c2-w45-3000-4000	894	3.3	0.001	0.1	0.1	1,328,428.96
nrw1379-91-c2-w45-3000-4000	956	3.0	0.000	0.1	0.2	1,428,364.58
nrw1379-101-c2-w45-3000-4000	2,027	3.3	0.009	0.2	0.3	1,429,559.68
nrw1379-111-c2-w45-3000-4000	2,421	3.5	0.029	0.3	0.4	1,536,936.73
nrw1379-121-c2-w45-3000-4000	2,550	3.3	5.638	0.3	0.8	1,738,929.50
nrw1379-131-c2-w45-3000-4000	1,810	3.1	0.017	0.2	0.4	2,040,919.99
nrw1379-141-c2-w45-3000-4000	3,834	3.5	0.032	0.5	0.9	2,446,300.22
nrw1379-151-c2-w45-3000-4000	2,475	3.5	0.035	0.3	0.5	2,354,334.19
nrw1379-51-c2-w60-3000-4000	377	3.2	0.000	0.0	0.1	814,733.92
nrw1379-61-c2-w60-3000-4000	712	3.4	0.135	0.1	0.2	818,866.13
nrw1379-71-c2-w60-3000-4000	927	3.6	0.021	0.1	0.2	1,122,340.55
nrw1379-81-c2-w60-3000-4000	993	3.0	0.000	0.1	0.2	1,327,118.94
nrw1379-91-c2-w60-3000-4000	1,202	3.1	0.019	0.1	0.3	1,328,776.14
nrw1379-101-c2-w60-3000-4000	1,742	3.4	0.000	0.2	0.3	1,432,098.18
nrw1379-111-c2-w60-3000-4000	2,160	3.3	0.033	0.2	0.3	1,637,182.94
nrw1379-121-c2-w60-3000-4000	3,392	3.5	5.659	0.4	1.1	1,740,399.17
nrw1379-131-c2-w60-3000-4000	7,056	3.9	0.060	0.8	2.3	1,740,491.27
nrw1379-141-c2-w60-3000-4000	6,818	3.8	0.025	0.9	1.9	2,149,936.88
nrw1379-151-c2-w60-3000-4000	11,173	3.9	0.035	1.4	2.6	1,948,485.30
nrw1379-51-c2-w75-3000-4000	497	3.5	0.064	0.1	0.1	1,019,776.94
nrw1379-61-c2-w75-3000-4000	482	3.0	0.000	0.0	0.1	1,019,800.39
nrw1379-71-c2-w75-3000-4000	704	3.2	0.011	0.1	0.1	1,123,342.18
nrw1379-81-c2-w75-3000-4000	1,149	3.0	0.004	0.1	0.2	1,225,530.94
nrw1379-91-c2-w75-3000-4000	2,972	3.9	0.010	0.3	0.4	1,329,567.67
nrw1379-101-c2-w75-3000-4000	2,753	3.8	0.000	0.3	0.4	1,327,013.78
nrw1379-111-c2-w75-3000-4000	1,833	3.2	0.017	0.2	0.4	1,634,908.24
nrw1379-121-c2-w75-3000-4000	3,544	3.4	0.000	0.4	0.5	1,634,544.89
nrw1379-131-c2-w75-3000-4000	4,240	3.5	0.002	0.5	0.7	1,838,469.19

Continued Table B.4 – *Two stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-141-c2-w75-3000-4000	6,100	3.5	0.041	0.8	1.8	1,943,428.31
nrw1379-151-c2-w75-3000-4000	6,185	3.8	0.013	0.7	1.2	1,844,772.90
nrw1379-51-c2-w90-3000-4000	447	3.1	0.000	0.0	0.1	817,338.97
nrw1379-61-c2-w90-3000-4000	909	3.5	0.035	0.1	0.1	1,323,842.54
nrw1379-71-c2-w90-3000-4000	2,383	3.8	0.000	0.3	0.4	1,121,492.11
nrw1379-81-c2-w90-3000-4000	3,795	4.1	8.813	0.4	0.7	1,125,978.57
nrw1379-91-c2-w90-3000-4000	2,589	3.7	0.001	0.3	0.4	1,325,284.56
nrw1379-101-c2-w90-3000-4000	2,247	3.3	0.010	0.3	0.4	1,429,649.92
nrw1379-111-c2-w90-3000-4000	2,996	3.6	0.018	0.3	0.5	1,431,758.81
nrw1379-121-c2-w90-3000-4000	6,670	4.1	0.031	1.0	1.9	1,435,103.85
nrw1379-131-c2-w90-3000-4000	11,398	4.2	0.159	1.5	2.3	1,437,540.33
nrw1379-141-c2-w90-3000-4000	15,659	4.3	0.108	2.1	5.4	1,742,817.25
nrw1379-151-c2-w90-3000-4000	6,470	3.8	5.609	0.8	3.1	1,741,603.29
nrw1379-51-c2-w120-3000-4000	1,564	4.2	0.000	0.2	0.3	716,037.83
nrw1379-61-c2-w120-3000-4000	826	3.4	0.014	0.1	0.1	919,057.60
nrw1379-71-c2-w120-3000-4000	2,681	4.4	0.002	0.3	0.4	1,122,920.38
nrw1379-81-c2-w120-3000-4000	2,396	3.7	0.187	0.3	0.5	1,027,318.36
nrw1379-91-c2-w120-3000-4000	3,365	3.7	0.025	0.5	0.6	1,124,584.34
nrw1379-101-c2-w120-3000-4000	3,234	3.6	0.017	0.4	0.6	1,331,193.97
nrw1379-111-c2-w120-3000-4000	5,686	4.1	0.073	0.6	2.7	1,334,745.56
nrw1379-121-c2-w120-3000-4000	8,447	4.1	0.072	1.0	2.6	1,435,025.36
nrw1379-131-c2-w120-3000-4000	13,407	4.1	0.041	1.7	2.7	1,636,791.80
nrw1379-141-c2-w120-3000-4000	13,417	4.1	0.079	1.7	4.3	1,743,347.75
nrw1379-151-c2-w120-3000-4000	19,013	4.2	0.059	2.6	12.7	1,742,133.52

Table B.5. Three Stacks Vehicle of C1 Class

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-51-c1-w15-500-1000	496	3.0	0.004	0.0	0.1	704,126.49
a280-61-c1-w15-500-1000	990	3.2	0.002	0.1	0.2	904,794.30
a280-71-c1-w15-500-1000	967	3.2	0.002	0.1	0.1	1,006,601.33
a280-81-c1-w15-500-1000	1,961	3.6	0.000	0.2	0.3	1,106,939.15
a280-91-c1-w15-500-1000	1,893	3.4	0.006	0.2	0.3	1,208,607.87
a280-101-c1-w15-500-1000	1,847	3.1	0.000	0.2	0.2	1,309,044.73
a280-111-c1-w15-500-1000	3,207	3.7	0.001	0.4	0.5	1,511,496.48
a280-121-c1-w15-500-1000	5,182	3.6	0.018	0.6	1.6	1,611,497.85
a280-131-c1-w15-500-1000	7,279	3.8	0.007	0.8	1.5	1,511,310.92

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-141-c1-w15-500-1000	5,031	3.4	0.037	0.6	1.6	1,814,033.52
a280-151-c1-w15-500-1000	8,589	3.7	0.013	1.2	1.9	1,813,209.13
a280-51-c1-w15-1000-1200	439	3.2	0.002	0.0	0.1	704,092.54
a280-61-c1-w15-1000-1200	998	3.7	0.000	0.1	0.1	805,244.53
a280-71-c1-w15-1000-1200	1,813	3.6	0.000	0.2	0.2	905,617.31
a280-81-c1-w15-1000-1200	1,998	3.6	0.000	0.2	0.3	1,007,980.71
a280-91-c1-w15-1000-1200	2,717	3.7	0.000	0.2	0.4	1,208,491.24
a280-101-c1-w15-1000-1200	4,987	4.0	0.000	0.5	0.7	1,108,703.68
a280-111-c1-w15-1000-1200	8,223	4.5	0.009	0.9	1.7	1,310,744.34
a280-121-c1-w15-1000-1200	8,940	4.0	0.005	1.0	2.4	1,311,039.99
a280-131-c1-w15-1000-1200	32,575	4.7	0.016	4.0	10.4	1,311,758.85
a280-141-c1-w15-1000-1200	22,354	4.5	0.017	2.9	8.6	1,612,670.64
a280-151-c1-w15-1000-1200	15,420	4.3	0.003	1.8	2.6	1,813,421.14
a280-51-c1-w15-1500-2000	3,228	4.2	0.003	0.3	0.4	503,698.37
a280-61-c1-w15-1500-2000	10,765	4.7	0.000	1.1	1.4	604,539.54
a280-71-c1-w15-1500-2000	27,260	5.3	0.077	2.9	7.1	505,530.63
a280-81-c1-w15-1500-2000	30,252	5.1	0.000	2.8	4.0	607,006.78
a280-91-c1-w15-1500-2000	58,483	5.2	0.024	7.0	15.0	706,696.27
a280-101-c1-w15-1500-2000	65,628	5.3	0.003	7.9	11.5	808,671.44
a280-111-c1-w15-1500-2000	115,902	5.5	0.014	15.6	38.5	1,009,208.80
a280-121-c1-w15-1500-2000	458,108	6.1	10.930	86.2	506.6	908,904.37
a280-131-c1-w15-1500-2000	951,379	6.5	0.068	159.5	2,370.3	810,191.26
a280-141-c1-w15-1500-2000	367,560	5.8	0.008	62.4	183.8	1,210,259.19
a280-151-c1-w15-1500-2000	965,126	6.1	0.024	230.1	1,208.0	1,111,276.51
a280-51-c1-w30-500-1000	2,138	4.0	0.003	0.4	0.5	603,345.01
a280-61-c1-w30-500-1000	1,891	3.7	0.006	0.2	0.3	604,452.79
a280-71-c1-w30-500-1000	2,555	3.9	0.000	0.3	0.4	906,014.28
a280-81-c1-w30-500-1000	4,183	3.9	0.011	0.5	1.0	906,790.70
a280-91-c1-w30-500-1000	3,989	3.7	9.035	0.4	1.9	1,107,838.77
a280-101-c1-w30-500-1000	5,165	4.0	0.004	0.7	0.9	1,309,586.07
a280-111-c1-w30-500-1000	16,792	4.5	7.635	2.3	4.3	1,309,260.67
a280-121-c1-w30-500-1000	13,839	4.1	0.011	2.1	3.7	1,310,518.19
a280-131-c1-w30-500-1000	30,567	4.3	0.010	5.2	9.9	1,309,958.98
a280-141-c1-w30-500-1000	43,155	4.5	0.005	7.2	10.4	1,411,432.72
a280-151-c1-w30-500-1000	27,080	4.3	6.193	4.9	20.3	1,612,842.77
a280-51-c1-w30-1000-1200	1,427	3.8	0.000	0.2	0.2	603,500.24
a280-61-c1-w30-1000-1200	3,902	4.5	0.004	0.4	0.5	704,292.51

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-71-c1-w30-1000-1200	7,601	4.7	0.015	0.9	1.4	704,928.70
a280-81-c1-w30-1000-1200	7,968	4.6	0.008	0.8	1.2	906,790.89
a280-91-c1-w30-1000-1200	15,487	5.2	0.000	1.8	2.4	1,007,627.95
a280-101-c1-w30-1000-1200	21,094	5.1	0.029	2.5	6.7	1,008,968.84
a280-111-c1-w30-1000-1200	21,284	5.1	0.008	3.2	6.4	1,210,408.72
a280-121-c1-w30-1000-1200	39,945	4.9	0.018	5.9	15.3	1,110,281.52
a280-131-c1-w30-1000-1200	175,721	5.5	0.021	35.0	90.1	1,110,238.04
a280-141-c1-w30-1000-1200	86,964	5.2	0.018	17.2	41.4	1,211,218.18
a280-151-c1-w30-1000-1200	214,420	6.0	0.031	45.2	235.6	1,312,075.67
a280-51-c1-w30-1500-2000	2,963	4.6	0.012	0.3	0.5	503,193.29
a280-61-c1-w30-1500-2000	89,927	6.4	0.012	11.6	16.4	403,860.79
a280-71-c1-w30-1500-2000	73,302	6.1	0.016	13.0	19.3	604,930.65
a280-81-c1-w30-1500-2000	54,529	5.6	0.012	7.0	10.4	605,414.51
a280-91-c1-w30-1500-2000	105,977	5.8	0.051	17.0	34.4	707,804.03
a280-101-c1-w30-1500-2000	96,744	5.4	0.069	15.3	40.0	708,935.31
a280-111-c1-w30-1500-2000	449,669	6.3	0.037	80.2	468.2	808,792.64
a280-121-c1-w30-1500-2000	609,718	6.3	0.059	129.9	403.2	809,717.71
a280-131-c1-w30-1500-2000	1,646,113	6.6	0.015	373.3	1,477.4	909,863.81
a280-141-c1-w30-1500-2000	1,839,331	6.3	0.117	449.9	7,200.0	**911,862.33**
a280-151-c1-w30-1500-2000	2,565,809	6.6	0.030	576.1	1,202.6	1,011,282.29
a280-51-c1-w45-500-1000	2,153	4.1	0.000	0.2	0.3	604,293.45
a280-61-c1-w45-500-1000	2,698	3.9	0.035	0.4	0.7	704,823.28
a280-71-c1-w45-500-1000	12,231	4.6	0.005	2.6	3.4	704,915.79
a280-81-c1-w45-500-1000	6,870	4.1	0.020	1.1	3.2	906,144.45
a280-91-c1-w45-500-1000	17,095	4.4	0.008	3.2	4.0	807,154.32
a280-101-c1-w45-500-1000	49,409	4.9	0.007	9.9	16.1	1,008,209.58
a280-111-c1-w45-500-1000	57,412	4.9	0.033	14.8	33.5	1,008,033.33
a280-121-c1-w45-500-1000	65,613	5.0	0.012	18.0	25.8	1,108,961.22
a280-131-c1-w45-500-1000	90,098	4.8	0.056	21.3	153.3	1,109,975.51
a280-141-c1-w45-500-1000	72,678	4.8	0.042	20.5	83.2	1,411,449.89
a280-151-c1-w45-500-1000	67,672	4.8	0.014	19.6	28.9	1,512,836.71
a280-51-c1-w45-1000-1200	8,511	5.1	0.000	1.5	1.8	503,149.41
a280-61-c1-w45-1000-1200	7,962	5.0	0.027	1.5	2.7	503,806.83
a280-71-c1-w45-1000-1200	14,078	5.2	0.000	1.8	2.2	705,302.85
a280-81-c1-w45-1000-1200	61,170	6.4	0.011	21.8	29.5	706,029.07
a280-91-c1-w45-1000-1200	20,158	5.0	0.010	2.9	4.5	906,784.78
a280-101-c1-w45-1000-1200	16,503	4.9	0.011	2.6	4.5	908,327.16

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-111-c1-w45-1000-1200	52,902	5.3	0.018	11.6	19.0	1,108,844.50
a280-121-c1-w45-1000-1200	248,537	6.2	0.013	86.6	205.4	1,008,516.89
a280-131-c1-w45-1000-1200	394,436	6.1	0.095	117.0	2,015.0	1,009,708.51
a280-141-c1-w45-1000-1200	518,889	6.3	0.027	212.3	1,355.6	1,110,078.35
a280-151-c1-w45-1000-1200	277,781	5.7	7.072	79.6	1,219.2	1,411,678.66
a280-51-c1-w45-1500-2000	4,349	5.3	0.000	0.4	0.6	403,634.06
a280-61-c1-w45-1500-2000	31,885	5.6	0.000	4.7	5.9	404,374.55
a280-71-c1-w45-1500-2000	99,416	6.2	0.000	20.1	24.3	505,054.96
a280-81-c1-w45-1500-2000	205,517	6.4	0.018	37.7	63.8	605,645.72
a280-91-c1-w45-1500-2000	472,066	7.0	0.029	99.5	170.8	607,275.00
a280-101-c1-w45-1500-2000	536,463	6.3	13.996	125.5	311.2	707,687.88
a280-111-c1-w45-1500-2000	4,690,589	8.1	0.008	1,550.8	2,719.0	708,451.17
a280-121-c1-w45-1500-2000	12,266,511	8.3		6,447.3		
a280-131-c1-w45-1500-2000	8,860,473	7.4		5,020.1		
a280-141-c1-w45-1500-2000	9,562,163	7.1		3,182.4		
a280-151-c1-w45-1500-2000	15,112,639	7.7		6,335.9		
brd14051-51-c12-w45-3000-4000	122	2.6	0.000	0.0	0.0	1,023,631.37
brd14051-61-c1-w45-3000-4000	138	2.2	0.001	0.0	0.0	1,334,094.14
brd14051-71-c1-w45-3000-4000	250	2.7	0.023	0.0	0.1	1,432,886.05
brd14051-81-c1-w45-3000-4000	754	3.4	0.000	0.1	0.1	1,539,739.22
brd14051-91-c1-w45-3000-4000	649	3.3	0.000	0.1	0.1	1,744,129.16
brd14051-101-c1-w45-3000-4000	1,482	3.7	0.000	0.2	0.2	1,950,418.17
brd14051-111-c1-w45-3000-4000	1,188	3.3	0.000	0.1	0.2	1,745,005.96
brd14051-121-c1-w45-3000-4000	797	3.4	0.000	0.1	0.2	2,566,416.45
brd14051-131-c1-w45-3000-4000	1,260	3.5	0.000	0.2	0.2	2,476,464.16
brd14051-141-c1-w45-3000-4000	2,675	3.6	0.000	0.4	0.5	2,463,685.19
brd14051-151-c1-w45-3000-4000	7,153	4.6	0.000	1.0	1.3	2,263,195.09
brd14051-51-c1-w60-3000-4000	158	2.7	0.000	0.0	0.0	1,126,673.51
brd14051-61-c1-w60-3000-4000	226	2.7	0.060	0.0	0.1	1,232,774.45
brd14051-71-c1-w60-3000-4000	315	3.1	0.001	0.0	0.1	1,639,848.58
brd14051-81-c1-w60-3000-4000	299	2.6	0.003	0.1	0.1	1,539,566.79
brd14051-91-c1-w60-3000-4000	483	2.8	0.000	0.1	0.1	1,538,230.48
brd14051-101-c1-w60-3000-4000	810	3.1	0.000	0.1	0.1	1,849,587.02
brd14051-111-c1-w60-3000-4000	1,523	3.9	0.024	0.2	0.3	1,847,057.56
brd14051-121-c1-w60-3000-4000	1,391	3.4	0.000	0.2	0.2	2,057,588.94
brd14051-131-c1-w60-3000-4000	1,364	3.3	0.000	0.2	0.2	2,368,434.77
brd14051-141-c1-w60-3000-4000	3,325	3.9	0.000	0.5	0.6	1,958,090.51

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-151-c1-w60-3000-4000	16,593	4.9	0.005	2.8	3.7	2,158,342.82
brd14051-51-c1-w75-3000-4000	173	2.8	0.000	0.0	0.1	1,023,080.49
brd14051-61-c1-w75-3000-4000	287	2.9	0.000	0.0	0.1	1,228,411.84
brd14051-71-c1-w75-3000-4000	891	3.8	0.094	0.1	0.2	1,335,893.03
brd14051-81-c1-w75-3000-4000	418	2.8	0.000	0.0	0.1	1,537,656.98
brd14051-91-c1-w75-3000-4000	1,050	3.5	0.000	0.1	0.2	1,850,021.02
brd14051-101-c1-w75-3000-4000	1,457	3.7	5.748	0.2	0.3	1,745,039.34
brd14051-111-c1-w75-3000-4000	3,230	4.3	0.006	0.4	0.6	1,644,781.44
brd14051-121-c1-w75-3000-4000	6,424	4.8	0.000	1.1	1.3	1,752,053.26
brd14051-131-c1-w75-3000-4000	2,391	3.7	0.000	0.3	0.4	2,059,399.20
brd14051-141-c1-w75-3000-4000	6,378	4.3	0.004	0.9	1.2	2,058,474.38
brd14051-151-c1-w75-3000-4000	10,878	4.5	0.045	2.0	3.2	2,064,801.33
brd14051-51-c1-w90-3000-4000	95	2.4	0.000	0.0	0.0	923,025.11
brd14051-61-c1-w90-3000-4000	254	2.8	0.000	0.0	0.0	927,059.02
brd14051-71-c1-w90-3000-4000	471	3.0	0.000	0.1	0.1	1,330,560.09
brd14051-81-c1-w90-3000-4000	977	3.4	0.000	0.1	0.2	1,438,893.77
brd14051-91-c1-w90-3000-4000	1,452	4.0	0.000	0.2	0.3	1,442,607.66
brd14051-101-c1-w90-3000-4000	1,210	3.4	0.001	0.2	0.2	1,747,453.81
brd14051-111-c1-w90-3000-4000	3,607	4.2	0.006	0.5	0.6	1,547,439.06
brd14051-121-c1-w90-3000-4000	5,526	4.5	0.024	0.9	1.2	1,851,346.33
brd14051-131-c1-w90-3000-4000	6,446	4.5	0.009	1.1	1.6	1,652,967.71
brd14051-141-c1-w90-3000-4000	5,442	4.3	0.001	0.7	1.2	1,951,676.83
brd14051-151-c1-w90-3000-4000	50,005	5.9	0.003	11.6	15.0	2,056,637.59
brd14051-51-c1-w120-3000-4000	432	3.4	0.000	0.1	0.1	1,024,500.73
brd14051-61-c1-w120-3000-4000	208	2.5	0.065	0.0	0.0	1,332,671.35
brd14051-71-c1-w120-3000-4000	763	3.5	0.011	0.1	0.2	1,332,592.00
brd14051-81-c1-w120-3000-4000	1,222	3.7	0.000	0.2	0.3	1,232,860.96
brd14051-91-c1-w120-3000-4000	1,674	3.9	0.000	0.2	0.3	1,541,072.89
brd14051-101-c1-w120-3000-4000	2,830	4.1	0.007	0.4	0.5	1,645,645.36
brd14051-111-c1-w120-3000-4000	6,874	4.8	0.006	1.7	2.0	1,439,905.45
brd14051-121-c1-w120-3000-4000	4,033	3.9	0.044	0.6	1.1	1,545,940.26
brd14051-131-c1-w120-3000-4000	1,746	3.2	0.032	0.3	0.5	1,956,564.36
brd14051-141-c1-w120-3000-4000	21,524	5.3	0.002	4.8	5.7	1,756,762.47
brd14051-151-c1-w120-3000-4000	56,575	6.1	0.039	22.1	28.1	1,853,683.10
d18512-51-c1-w45-3000-4000	120	2.3	0.000	0.0	0.1	1,126,858.36
d18512-61-c1-w45-3000-4000	203	2.8	0.000	0.0	0.0	1,435,850.25
d18512-71-c1-w45-3000-4000	1,039	4.1	0.000	0.1	0.1	1,231,980.78

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-81-c1-w45-3000-4000	159	2.1	0.000	0.0	0.0	1,950,054.94
d18512-91-c1-w45-3000-4000	1,601	4.1	0.017	0.2	0.3	1,746,809.70
d18512-101-c1-w45-3000-4000	1,352	3.7	0.000	0.2	0.2	2,051,913.32
d18512-111-c1-w45-3000-4000	799	3.1	0.000	0.1	0.1	2,261,744.74
d18512-121-c1-w45-3000-4000	779	3.1	0.000	0.1	0.2	2,465,550.56
d18512-131-c1-w45-3000-4000	3,765	4.1	0.001	0.5	0.7	2,059,455.69
d18512-141-c1-w45-3000-4000	2,354	3.9	0.000	0.3	0.4	2,261,120.80
d18512-151-c1-w45-3000-4000	4,010	3.9	0.000	0.5	0.7	2,372,298.17
d18512-51-c1-w60-3000-4000	87	1.9	0.000	0.0	0.0	1,026,569.61
d18512-61-c1-w60-3000-4000	297	3.0	0.075	0.0	0.1	1,131,334.24
d18512-71-c1-w60-3000-4000	175	2.3	0.000	0.0	0.0	1,434,380.41
d18512-81-c1-w60-3000-4000	748	3.4	0.000	0.1	0.1	1,438,133.33
d18512-91-c1-w60-3000-4000	659	3.4	0.001	0.1	0.1	1,643,206.95
d18512-101-c1-w60-3000-4000	1,859	4.0	0.008	0.2	0.3	1,846,388.52
d18512-111-c1-w60-3000-4000	2,661	4.3	0.002	0.3	0.4	1,749,092.63
d18512-121-c1-w60-3000-4000	2,963	4.3	4.263	0.4	0.9	2,368,277.31
d18512-131-c1-w60-3000-4000	2,723	3.8	0.005	0.4	0.6	2,263,807.29
d18512-141-c1-w60-3000-4000	4,010	4.1	0.032	0.5	0.8	2,154,498.96
d18512-151-c1-w60-3000-4000	3,686	4.0	0.048	0.5	1.4	2,575,328.53
d18512-51-c1-w75-3000-4000	82	2.0	0.000	0.0	0.0	1,130,349.15
d18512-61-c1-w75-3000-4000	216	2.6	0.001	0.0	0.1	1,234,345.40
d18512-71-c1-w75-3000-4000	287	2.7	0.000	0.0	0.1	1,435,566.01
d18512-81-c1-w75-3000-4000	509	3.2	6.158	0.1	0.1	1,639,246.02
d18512-91-c1-w75-3000-4000	1,369	4.1	0.002	0.2	0.3	1,438,700.49
d18512-101-c1-w75-3000-4000	1,593	3.6	0.000	0.2	0.2	1,539,582.55
d18512-111-c1-w75-3000-4000	1,345	3.3	0.129	0.2	0.4	1,852,005.83
d18512-121-c1-w75-3000-4000	838	2.9	0.004	0.1	0.2	2,161,951.66
d18512-131-c1-w75-3000-4000	5,036	4.7	0.000	0.7	0.9	1,851,496.61
d18512-141-c1-w75-3000-4000	5,069	4.2	0.002	0.9	1.2	2,161,209.80
d18512-151-c1-w75-3000-4000	7,432	4.7	0.004	1.2	1.5	2,472,592.49
d18512-51-c1-w90-3000-4000	234	2.8	0.000	0.0	0.0	1,025,238.06
d18512-61-c1-w90-3000-4000	570	3.3	0.096	0.1	0.1	1,029,283.73
d18512-71-c1-w90-3000-4000	1,158	3.8	0.000	0.2	0.2	1,334,811.82
d18512-81-c1-w90-3000-4000	270	2.6	0.000	0.0	0.1	1,540,612.04
d18512-91-c1-w90-3000-4000	1,029	3.7	0.000	0.2	0.2	1,540,856.17
d18512-101-c1-w90-3000-4000	2,825	4.4	0.000	0.4	0.5	1,441,400.31
d18512-111-c1-w90-3000-4000	2,307	3.9	0.000	0.3	0.4	1,648,144.95

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-121-c1-w90-3000-4000	1,252	3.3	0.165	0.2	0.3	1,861,295.39
d18512-131-c1-w90-3000-4000	3,187	4.0	0.015	0.5	0.6	1,955,259.65
d18512-141-c1-w90-3000-4000	5,818	4.4	0.069	1.1	1.9	1,955,028.92
d18512-151-c1-w90-3000-4000	15,409	5.3	0.009	3.6	4.6	2,470,181.18
d18512-51-c1-w120-3000-4000	219	3.0	0.000	0.0	0.1	1,026,410.26
d18512-61-c1-w120-3000-4000	578	3.4	0.005	0.0	0.1	1,232,024.32
d18512-71-c1-w120-3000-4000	1,317	3.9	0.002	0.2	0.3	1,029,355.40
d18512-81-c1-w120-3000-4000	532	3.3	0.000	0.1	0.1	1,334,200.47
d18512-91-c1-w120-3000-4000	1,434	3.7	0.023	0.2	0.2	1,439,386.49
d18512-101-c1-w120-3000-4000	8,796	4.9	0.002	2.2	2.7	1,543,148.31
d18512-111-c1-w120-3000-4000	4,785	4.2	0.057	0.8	1.5	1,546,439.45
d18512-121-c1-w120-3000-4000	2,513	3.7	0.066	0.4	0.9	1,754,421.26
d18512-131-c1-w120-3000-4000	15,266	5.3	0.000	5.1	5.6	1,751,648.22
d18512-141-c1-w120-3000-4000	20,475	5.3	0.003	4.6	6.0	2,051,539.57
d18512-151-c1-w120-3000-4000	16,735	5.0	0.002	4.5	5.3	1,959,112.67
fnl4461-51-c12-w45-3000-4000	751	3.5	0.000	0.1	0.1	613,103.10
fnl4461-61-c1-w45-3000-4000	947	3.5	0.027	0.1	0.1	816,088.89
fnl4461-71-c1-w45-3000-4000	1,407	3.7	0.000	0.1	0.2	918,673.81
fnl4461-81-c1-w45-3000-4000	1,685	3.7	0.001	0.1	0.2	1,121,615.82
fnl4461-91-c1-w45-3000-4000	3,904	3.9	0.000	0.4	0.5	1,125,690.31
fnl4461-101-c1-w45-3000-4000	4,646	3.8	0.025	0.5	1.0	1,127,897.39
fnl4461-111-c1-w45-3000-4000	6,428	4.0	0.000	0.6	0.9	1,328,094.39
fnl4461-121-c1-w45-3000-4000	9,853	4.1	0.024	1.0	1.8	1,231,684.54
fnl4461-131-c1-w45-3000-4000	6,287	4.0	0.009	0.7	1.4	1,738,981.08
fnl4461-141-c1-w45-3000-4000	12,957	4.2	0.060	1.3	2.0	1,437,750.49
fnl4461-151-c1-w45-3000-4000	15,772	4.6	0.011	1.9	2.8	1,742,995.47
fnl4461-51-c1-w60-3000-4000	719	3.6	0.000	0.1	0.1	716,001.04
fnl4461-61-c1-w60-3000-4000	641	3.2	0.000	0.0	0.1	816,843.99
fnl4461-71-c1-w60-3000-4000	1,801	3.8	0.035	0.1	0.2	821,157.56
fnl4461-81-c1-w60-3000-4000	4,037	4.1	0.060	0.4	0.7	920,873.42
fnl4461-91-c1-w60-3000-4000	5,571	4.5	0.000	0.5	0.7	1,124,714.21
fnl4461-101-c1-w60-3000-4000	5,513	3.9	0.035	0.5	0.9	1,131,643.96
fnl4461-111-c1-w60-3000-4000	10,151	4.1	0.015	1.1	1.7	1,229,215.76
fnl4461-121-c1-w60-3000-4000	15,089	4.5	7.524	1.8	2.7	1,329,079.16
fnl4461-131-c1-w60-3000-4000	16,981	4.1	0.005	2.2	3.0	1,434,432.55
fnl4461-141-c1-w60-3000-4000	18,942	4.4	0.003	2.3	3.2	1,535,963.23
fnl4461-151-c1-w60-3000-4000	14,224	4.2	0.028	1.9	4.3	1,741,589.55

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-51-c1-w75-3000-4000	570	3.2	0.004	0.0	0.1	713,779.32
fnl4461-61-c1-w75-3000-4000	1,402	3.9	0.008	0.1	0.2	915,796.20
fnl4461-71-c1-w75-3000-4000	1,958	3.7	0.027	0.2	0.4	818,857.51
fnl4461-81-c1-w75-3000-4000	3,080	3.9	0.000	0.3	0.4	1,022,750.57
fnl4461-91-c1-w75-3000-4000	6,090	4.3	0.015	0.5	1.0	1,022,617.96
fnl4461-101-c1-w75-3000-4000	6,167	3.9	0.021	0.7	1.7	1,125,300.28
fnl4461-111-c1-w75-3000-4000	18,327	4.9	0.075	2.0	6.2	1,027,317.68
fnl4461-121-c1-w75-3000-4000	24,534	4.7	0.005	3.4	4.6	1,126,356.33
fnl4461-131-c1-w75-3000-4000	18,154	4.4	7.417	2.5	10.2	1,332,735.15
fnl4461-141-c1-w75-3000-4000	33,020	4.6	0.041	4.6	11.9	1,335,668.38
fnl4461-151-c1-w75-3000-4000	21,788	4.2	0.056	2.9	7.6	1,537,608.85
fnl4461-51-c1-w90-3000-4000	509	3.4	0.010	0.0	0.1	613,238.30
fnl4461-61-c1-w90-3000-4000	2,321	4.1	12.205	0.3	0.5	815,156.81
fnl4461-71-c1-w90-3000-4000	2,235	3.8	0.110	0.3	0.5	718,793.16
fnl4461-81-c1-w90-3000-4000	9,219	4.6	0.000	1.0	1.4	818,231.24
fnl4461-91-c1-w90-3000-4000	5,278	4.4	0.000	0.6	0.8	1,022,458.87
fnl4461-101-c1-w90-3000-4000	6,507	4.2	0.091	0.7	2.2	1,024,906.67
fnl4461-111-c1-w90-3000-4000	10,391	4.2	0.043	1.3	3.6	1,126,455.37
fnl4461-121-c1-w90-3000-4000	21,929	4.8	0.020	3.4	6.5	1,229,926.65
fnl4461-131-c1-w90-3000-4000	21,365	4.5	0.028	2.8	5.2	1,232,351.07
fnl4461-141-c1-w90-3000-4000	40,669	4.8	0.028	7.2	13.7	1,434,522.89
fnl4461-151-c1-w90-3000-4000	70,649	5.0	0.168	12.4	40.3	1,238,464.94
fnl4461-51-c1-w120-3000-4000	1,605	3.9	0.003	0.2	0.3	611,216.96
fnl4461-61-c1-w120-3000-4000	3,694	4.3	0.000	0.4	0.5	614,602.57
fnl4461-71-c1-w120-3000-4000	9,517	4.7	0.000	1.2	1.5	715,113.13
fnl4461-81-c1-w120-3000-4000	29,426	5.5	0.025	5.3	8.2	818,144.04
fnl4461-91-c1-w120-3000-4000	8,391	4.5	10.811	0.8	2.3	920,187.74
fnl4461-101-c1-w120-3000-4000	44,809	5.6	0.047	8.7	13.6	922,090.81
fnl4461-111-c1-w120-3000-4000	22,265	4.6	0.013	3.0	4.3	927,472.33
fnl4461-121-c1-w120-3000-4000	29,039	4.7	0.126	4.5	14.5	1,129,866.96
fnl4461-131-c1-w120-3000-4000	36,145	4.6	0.123	5.8	64.5	1,132,011.40
fnl4461-141-c1-w120-3000-4000	70,419	5.1	0.228	13.7	51.6	1,236,128.94
fnl4461-151-c1-w120-3000-4000	69,997	4.8	0.076	12.9	41.2	1,336,367.94
nrw1379-51-c1-w45-3000-4000	352	3.0	0.000	0.1	0.2	818,663.69
nrw1379-61-c1-w45-3000-4000	661	3.6	0.000	0.1	0.1	1,022,337.14
nrw1379-71-c1-w45-3000-4000	1,656	4.0	0.000	0.2	0.2	1,022,994.36
nrw1379-81-c1-w45-3000-4000	2,235	4.0	0.000	0.2	0.3	1,122,818.39

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-91-c1-w45-3000-4000	1,434	3.7	0.000	0.2	0.2	1,226,159.73
nrw1379-101-c1-w45-3000-4000	3,179	3.7	0.000	0.3	0.4	1,231,474.12
nrw1379-111-c1-w45-3000-4000	2,939	3.7	0.000	0.3	0.4	1,635,360.31
nrw1379-121-c1-w45-3000-4000	3,825	3.7	0.041	0.4	1.0	1,533,965.88
nrw1379-131-c1-w45-3000-4000	8,743	4.1	0.000	1.0	1.3	1,737,978.90
nrw1379-141-c1-w45-3000-4000	10,255	4.4	0.003	1.1	1.5	1,946,832.43
nrw1379-151-c1-w45-3000-4000	10,809	4.0	0.001	1.2	1.8	1,745,182.79
nrw1379-51-c1-w60-3000-4000	545	3.4	0.000	0.0	0.1	715,450.19
nrw1379-61-c1-w60-3000-4000	636	3.2	0.000	0.1	0.1	1,021,865.56
nrw1379-71-c1-w60-3000-4000	2,105	4.1	0.004	0.2	0.3	1,020,686.48
nrw1379-81-c1-w60-3000-4000	1,511	3.6	0.009	0.1	0.2	1,329,711.62
nrw1379-91-c1-w60-3000-4000	1,821	3.8	0.012	0.2	0.3	1,426,254.48
nrw1379-101-c1-w60-3000-4000	4,151	4.1	0.000	0.4	0.6	1,327,802.02
nrw1379-111-c1-w60-3000-4000	2,828	3.8	0.003	0.3	0.4	1,434,304.21
nrw1379-121-c1-w60-3000-4000	4,458	3.7	6.125	0.5	0.8	1,635,073.26
nrw1379-131-c1-w60-3000-4000	7,982	3.9	0.004	0.9	1.4	1,538,350.09
nrw1379-141-c1-w60-3000-4000	10,878	4.3	0.009	1.3	2.6	1,740,189.25
nrw1379-151-c1-w60-3000-4000	15,082	4.3	0.036	2.1	4.2	1,844,076.33
nrw1379-51-c1-w75-3000-4000	932	4.0	0.005	0.1	0.2	716,630.85
nrw1379-61-c1-w75-3000-4000	1,090	3.9	0.000	0.1	0.2	1,021,288.00
nrw1379-71-c1-w75-3000-4000	2,217	4.1	0.005	0.3	0.4	1,120,583.31
nrw1379-81-c1-w75-3000-4000	1,545	3.8	0.000	0.2	0.2	1,325,830.74
nrw1379-91-c1-w75-3000-4000	5,289	4.5	0.000	0.7	0.8	1,226,942.55
nrw1379-101-c1-w75-3000-4000	3,645	3.7	0.007	0.5	0.7	1,429,728.22
nrw1379-111-c1-w75-3000-4000	5,067	4.2	0.000	0.6	0.8	1,534,615.51
nrw1379-121-c1-w75-3000-4000	8,197	4.3	0.052	1.0	2.6	1,534,424.84
nrw1379-131-c1-w75-3000-4000	17,602	4.6	0.180	2.6	4.0	1,439,946.99
nrw1379-141-c1-w75-3000-4000	17,091	4.5	0.005	2.2	3.2	1,843,107.49
nrw1379-151-c1-w75-3000-4000	18,024	4.5	0.028	2.5	4.7	1,741,507.82
nrw1379-51-c1-w90-3000-4000	797	3.6	0.000	0.1	0.1	815,530.19
nrw1379-61-c1-w90-3000-4000	1,407	4.0	0.000	0.1	0.2	922,134.13
nrw1379-71-c1-w90-3000-4000	2,411	4.2	0.000	0.2	0.3	819,453.29
nrw1379-81-c1-w90-3000-4000	2,413	3.9	0.001	0.3	0.4	1,125,181.20
nrw1379-91-c1-w90-3000-4000	6,001	4.3	0.000	0.6	0.8	1,127,910.97
nrw1379-101-c1-w90-3000-4000	11,267	4.5	0.014	1.5	2.2	1,127,125.18
nrw1379-111-c1-w90-3000-4000	6,090	3.9	0.036	0.7	1.3	1,230,490.09
nrw1379-121-c1-w90-3000-4000	8,286	4.1	0.034	1.0	2.4	1,332,111.42

Continued Table B.5 – *Three stacks vehicle of C1 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-131-c1-w90-3000-4000	27,109	4.9	6.426	4.1	7.8	1,535,509.98
nrw1379-141-c1-w90-3000-4000	23,999	4.9	0.042	3.0	7.6	1,538,208.72
nrw1379-151-c1-w90-3000-4000	33,827	4.8	0.044	5.0	10.2	1,539,895.47
nrw1379-51-c1-w120-3000-4000	1,520	4.2	0.000	0.2	0.3	712,946.42
nrw1379-61-c1-w120-3000-4000	2,144	4.3	0.007	0.2	0.3	820,018.66
nrw1379-71-c1-w120-3000-4000	8,010	5.1	0.005	1.2	1.5	1,022,295.21
nrw1379-81-c1-w120-3000-4000	4,165	4.2	0.006	0.5	0.7	1,022,933.37
nrw1379-91-c1-w120-3000-4000	16,502	4.9	0.000	2.4	3.0	1,025,130.16
nrw1379-101-c1-w120-3000-4000	12,357	4.6	0.054	1.8	2.9	1,127,009.55
nrw1379-111-c1-w120-3000-4000	14,415	4.7	8.049	1.7	6.4	1,227,448.77
nrw1379-121-c1-w120-3000-4000	17,445	4.6	0.025	2.6	4.1	1,228,777.04
nrw1379-131-c1-w120-3000-4000	25,556	4.7	6.951	4.9	7.4	1,433,036.60
nrw1379-141-c1-w120-3000-4000	73,101	5.4	0.025	17.6	23.8	1,439,145.78
nrw1379-151-c1-w120-3000-4000	59,740	5.0	0.059	10.3	20.7	1,436,828.65

Table B.6. Three Stacks Vehicle of C2 Class

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-51-c2-w15-500-1000	471	2.9	0.000	0.0	0.2	704,424.42
a280-61-c2-w15-500-1000	775	3.1	0.000	0.1	0.1	905,043.40
a280-71-c2-w15-500-1000	1,361	3.2	0.006	0.1	0.2	905,917.29
a280-81-c2-w15-500-1000	844	2.9	0.015	0.1	0.2	1,307,785.69
a280-91-c2-w15-500-1000	1,805	3.2	0.015	0.2	0.7	1,308,336.24
a280-101-c2-w15-500-1000	1,006	2.8	6.617	0.1	0.3	1,510,666.66
a280-111-c2-w15-500-1000	2,165	3.2	0.000	0.2	0.3	1,712,128.08
a280-121-c2-w15-500-1000	4,102	3.2	0.004	0.4	0.7	1,611,520.35
a280-131-c2-w15-500-1000	6,403	3.4	0.008	0.7	1.2	1,712,232.72
a280-141-c2-w15-500-1000	5,069	3.3	0.005	0.6	0.9	1,913,409.88
a280-151-c2-w15-500-1000	4,285	3.0	0.014	0.5	1.3	2,015,357.68
a280-51-c2-w15-1000-1200	360	2.9	0.003	0.0	0.1	804,353.97
a280-61-c2-w15-1000-1200	1,001	3.6	0.000	0.1	0.1	804,875.89
a280-71-c2-w15-1000-1200	2,097	4.2	0.013	0.2	0.3	906,909.90
a280-81-c2-w15-1000-1200	1,468	3.2	0.002	0.1	0.2	1,207,431.33
a280-91-c2-w15-1000-1200	2,134	3.6	0.004	0.2	0.3	1,208,754.64
a280-101-c2-w15-1000-1200	5,208	3.8	0.004	0.6	1.2	1,209,308.24
a280-111-c2-w15-1000-1200	7,583	4.1	0.011	0.7	2.1	1,210,475.12
a280-121-c2-w15-1000-1200	19,289	4.6	0.018	2.1	8.0	1,109,910.56

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-131-c2-w15-1000-1200	16,741	4.0	0.017	2.1	5.5	1,412,239.62
a280-141-c2-w15-1000-1200	11,600	4.0	0.004	1.3	1.9	1,713,062.11
a280-151-c2-w15-1000-1200	14,617	4.1	0.029	1.8	5.6	1,614,370.08
a280-51-c2-w15-1500-2000	4,387	4.5	0.000	0.4	0.5	503,814.14
a280-61-c2-w15-1500-2000	3,949	4.4	0.000	0.4	0.6	604,889.59
a280-71-c2-w15-1500-2000	13,659	4.7	0.010	1.5	2.6	805,232.26
a280-81-c2-w15-1500-2000	7,373	4.1	0.004	0.7	1.2	906,399.67
a280-91-c2-w15-1500-2000	26,282	4.9	0.015	3.0	5.6	808,043.99
a280-101-c2-w15-1500-2000	53,164	4.9	0.041	6.8	16.6	908,995.28
a280-111-c2-w15-1500-2000	50,083	4.7	0.021	7.2	13.1	1,110,593.61
a280-121-c2-w15-1500-2000	123,669	5.2	0.018	16.8	31.8	1,110,342.74
a280-131-c2-w15-1500-2000	376,833	5.9	0.045	60.2	181.3	1,111,472.17
a280-141-c2-w15-1500-2000	142,684	5.0	0.030	20.9	102.8	1,211,887.08
a280-151-c2-w15-1500-2000	331,949	5.4	0.053	52.9	446.1	1,213,060.25
a280-51-c2-w30-500-1000	648	3.1	0.000	0.1	0.1	604,267.38
a280-61-c2-w30-500-1000	2,181	3.8	0.002	0.2	0.3	804,537.82
a280-71-c2-w30-500-1000	2,270	3.5	0.010	0.3	0.4	905,704.57
a280-81-c2-w30-500-1000	4,321	3.7	0.004	0.6	1.0	1,107,020.96
a280-91-c2-w30-500-1000	2,198	3.2	0.011	0.2	0.4	1,208,021.85
a280-101-c2-w30-500-1000	3,437	3.3	0.004	0.4	0.7	1,309,392.35
a280-111-c2-w30-500-1000	8,656	3.9	0.018	1.3	3.0	1,310,275.38
a280-121-c2-w30-500-1000	11,536	3.8	6.573	1.5	4.6	1,510,585.06
a280-131-c2-w30-500-1000	15,304	3.7	0.008	2.2	4.2	1,511,080.09
a280-141-c2-w30-500-1000	14,146	3.8	0.001	1.9	2.7	1,612,884.12
a280-151-c2-w30-500-1000	19,557	3.9	0.016	2.7	6.5	1,612,740.72
a280-51-c2-w30-1000-1200	343	3.1	0.001	0.0	0.1	704,323.82
a280-61-c2-w30-1000-1200	1,621	3.9	14.199	0.1	0.2	704,338.20
a280-71-c2-w30-1000-1200	3,466	4.1	0.045	0.3	0.9	806,079.61
a280-81-c2-w30-1000-1200	2,859	3.6	0.007	0.4	0.5	1,006,877.62
a280-91-c2-w30-1000-1200	5,750	4.0	0.008	0.6	1.7	1,107,907.97
a280-101-c2-w30-1000-1200	9,237	4.0	0.004	1.0	1.6	1,109,226.37
a280-111-c2-w30-1000-1200	13,818	4.7	0.011	1.5	3.3	1,210,548.97
a280-121-c2-w30-1000-1200	18,987	4.3	0.008	2.9	5.8	1,410,916.90
a280-131-c2-w30-1000-1200	61,322	5.0	0.022	8.5	31.7	1,210,155.66
a280-141-c2-w30-1000-1200	46,250	4.6	0.017	6.3	14.6	1,411,698.78
a280-151-c2-w30-1000-1200	47,339	4.6	6.580	7.0	29.5	1,512,108.19
a280-51-c2-w30-1500-2000	4,079	4.4	0.005	0.5	0.7	503,573.43

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-61-c2-w30-1500-2000	22,696	5.3	0.009	3.4	5.3	504,708.32
a280-71-c2-w30-1500-2000	10,323	4.3	0.020	1.2	1.9	705,467.98
a280-81-c2-w30-1500-2000	20,745	5.1	0.019	2.6	6.8	706,655.09
a280-91-c2-w30-1500-2000	107,690	5.6	0.033	13.7	37.6	706,974.20
a280-101-c2-w30-1500-2000	97,000	5.2	0.021	14.6	32.1	908,736.04
a280-111-c2-w30-1500-2000	136,614	5.4	0.084	20.2	56.0	810,457.59
a280-121-c2-w30-1500-2000	284,748	5.5	0.019	51.6	88.1	1,109,764.05
a280-131-c2-w30-1500-2000	571,586	5.6	0.025	106.2	221.6	1,110,373.86
a280-141-c2-w30-1500-2000	880,558	5.9	0.021	171.7	722.8	1,210,914.54
a280-151-c2-w30-1500-2000	738,554	5.7	0.025	148.0	451.7	1,212,335.94
a280-51-c2-w45-500-1000	1,170	3.2	0.000	0.1	0.2	604,418.98
a280-61-c2-w45-500-1000	7,985	4.2	0.019	2.2	2.6	704,438.44
a280-71-c2-w45-500-1000	5,433	3.8	0.012	0.8	1.4	805,444.62
a280-81-c2-w45-500-1000	5,858	3.8	0.036	0.8	3.7	907,002.23
a280-91-c2-w45-500-1000	10,021	4.3	0.011	1.9	3.6	1,007,064.88
a280-101-c2-w45-500-1000	19,471	4.4	0.004	4.3	5.4	1,209,187.46
a280-111-c2-w45-500-1000	19,338	4.1	0.011	3.4	7.5	1,309,747.84
a280-121-c2-w45-500-1000	47,189	4.4	0.014	9.1	28.8	1,309,777.04
a280-131-c2-w45-500-1000	42,586	4.4	0.003	8.9	11.9	1,410,177.11
a280-141-c2-w45-500-1000	28,285	3.9	0.002	5.2	7.2	1,612,448.32
a280-151-c2-w45-500-1000	72,592	4.4	0.013	15.0	29.0	1,612,210.59
a280-51-c2-w45-1000-1200	2,150	3.9	0.002	0.4	0.6	703,879.79
a280-61-c2-w45-1000-1200	15,776	5.4	0.000	2.2	2.8	603,772.75
a280-71-c2-w45-1000-1200	10,262	4.7	0.004	1.6	2.3	705,150.28
a280-81-c2-w45-1000-1200	13,574	4.6	0.022	2.1	6.3	806,566.97
a280-91-c2-w45-1000-1200	38,870	5.6	0.012	9.1	14.9	907,833.24
a280-101-c2-w45-1000-1200	23,779	4.6	0.015	3.7	8.1	1,108,782.43
a280-111-c2-w45-1000-1200	53,828	5.0	9.000	9.5	45.1	1,109,652.28
a280-121-c2-w45-1000-1200	67,244	5.0	8.230	12.8	59.3	1,209,758.21
a280-131-c2-w45-1000-1200	176,265	5.4	8.220	36.1	240.9	1,209,973.93
a280-141-c2-w45-1000-1200	75,771	4.7	0.031	14.0	58.9	1,311,475.64
a280-151-c2-w45-1000-1200	192,001	5.4	0.056	40.1	422.2	1,312,939.24
a280-51-c2-w45-1500-2000	7,803	5.1	0.000	1.1	1.4	503,496.11
a280-61-c2-w45-1500-2000	24,557	5.2	0.004	3.0	4.1	604,088.49
a280-71-c2-w45-1500-2000	72,209	5.6	0.013	11.5	19.7	604,962.04
a280-81-c2-w45-1500-2000	114,885	5.8	0.038	17.4	42.4	607,006.02
a280-91-c2-w45-1500-2000	103,345	5.7	0.022	16.2	32.4	806,792.31

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
a280-101-c2-w45-1500-2000	130,468	5.4	0.033	25.7	45.3	808,250.41
a280-111-c2-w45-1500-2000	592,524	6.4	0.015	118.9	197.1	908,003.72
a280-121-c2-w45-1500-2000	1,678,731	7.1	0.012	463.2	770.3	1,109,669.35
a280-131-c2-w45-1500-2000	1,870,762	6.4	0.025	453.3	1,383.3	1,010,215.01
a280-141-c2-w45-1500-2000	1,349,312	5.9	0.018	355.5	618.7	1,211,235.19
a280-151-c2-w45-1500-2000	3,448,372	6.4	0.020	843.2	5,509.0	1,211,054.69
brd14051-51-c2-w45-3000-4000	116	2.5	0.000	0.0	0.0	1,429,425.92
brd14051-61-c2-w45-3000-4000	104	2.0	0.000	0.1	0.1	1,639,735.15
brd14051-71-c2-w45-3000-4000	357	3.1	0.023	0.1	0.1	1,437,430.55
brd14051-81-c2-w45-3000-4000	759	3.1	0.061	0.1	0.2	1,336,495.95
brd14051-91-c2-w45-3000-4000	422	2.7	0.000	0.1	0.1	1,852,599.71
brd14051-101-c2-w45-3000-4000	630	3.1	0.000	0.1	0.1	1,749,978.19
brd14051-111-c2-w45-3000-4000	1,530	3.7	0.000	0.2	0.3	2,052,468.84
brd14051-121-c2-w45-3000-4000	1,531	3.5	0.065	0.2	0.4	2,057,354.94
brd14051-131-c2-w45-3000-4000	930	3.0	0.000	0.1	0.2	2,677,705.88
brd14051-141-c2-w45-3000-4000	2,468	3.6	0.000	0.4	0.5	2,469,683.55
brd14051-151-c2-w45-3000-4000	6,684	4.3	3.747	1.0	2.4	2,668,892.80
brd14051-51-c2-w60-3000-4000	104	2.3	0.000	0.0	0.1	1,228,636.37
brd14051-61-c2-w60-3000-4000	187	2.6	0.000	0.0	0.1	1,230,492.03
brd14051-71-c2-w60-3000-4000	612	3.9	0.000	0.1	0.1	1,435,777.41
brd14051-81-c2-w60-3000-4000	400	2.9	0.000	0.0	0.1	1,639,742.68
brd14051-91-c2-w60-3000-4000	584	2.9	0.007	0.1	0.1	1,641,225.79
brd14051-101-c2-w60-3000-4000	678	3.4	0.000	0.1	0.1	1,847,898.23
brd14051-111-c2-w60-3000-4000	3,094	3.9	0.000	0.4	0.5	1,645,837.05
brd14051-121-c2-w60-3000-4000	1,313	3.1	0.010	0.2	0.3	2,466,877.97
brd14051-131-c2-w60-3000-4000	621	2.6	0.001	0.1	0.2	2,470,948.50
brd14051-141-c2-w60-3000-4000	4,377	3.8	0.002	0.5	0.7	2,263,935.06
brd14051-151-c2-w60-3000-4000	6,672	4.2	0.000	1.0	1.3	2,470,258.20
brd14051-51-c2-w75-3000-4000	198	2.8	0.000	0.0	0.1	1,025,843.14
brd14051-61-c2-w75-3000-4000	247	2.6	0.083	0.0	0.0	1,336,738.05
brd14051-71-c2-w75-3000-4000	391	2.7	0.039	0.0	0.1	1,435,375.18
brd14051-81-c2-w75-3000-4000	535	2.9	0.000	0.1	0.1	1,640,033.99
brd14051-91-c2-w75-3000-4000	907	3.1	0.057	0.1	0.1	1,540,175.99
brd14051-101-c2-w75-3000-4000	1,731	3.9	0.025	0.2	0.3	1,852,281.97
brd14051-111-c2-w75-3000-4000	1,701	3.7	0.004	0.2	0.3	1,852,875.25
brd14051-121-c2-w75-3000-4000	1,688	4.0	0.000	0.2	0.3	1,955,879.53
brd14051-131-c2-w75-3000-4000	1,225	3.0	0.000	0.2	0.2	2,167,640.35

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
brd14051-141-c2-w75-3000-4000	4,460	4.0	0.037	0.5	0.9	1,851,982.18
brd14051-151-c2-w75-3000-4000	9,106	4.3	0.000	1.3	1.6	2,056,823.85
brd14051-51-c2-w90-3000-4000	173	2.6	0.000	0.0	0.1	1,227,395.76
brd14051-61-c2-w90-3000-4000	395	3.2	0.000	0.0	0.1	1,131,544.26
brd14051-71-c2-w90-3000-4000	372	2.9	0.000	0.0	0.1	1,331,554.78
brd14051-81-c2-w90-3000-4000	515	2.9	0.000	0.1	0.1	1,438,542.78
brd14051-91-c2-w90-3000-4000	1,779	3.9	0.000	0.2	0.3	1,842,416.40
brd14051-101-c2-w90-3000-4000	1,568	3.5	0.000	0.2	0.3	1,753,149.00
brd14051-111-c2-w90-3000-4000	4,387	4.1	0.000	0.6	0.7	1,543,280.84
brd14051-121-c2-w90-3000-4000	1,453	3.2	0.031	0.2	0.5	2,057,762.26
brd14051-131-c2-w90-3000-4000	1,804	3.4	0.005	0.3	0.4	2,167,080.58
brd14051-141-c2-w90-3000-4000	6,359	4.0	0.000	1.0	1.2	2,256,816.70
brd14051-151-c2-w90-3000-4000	7,244	4.1	0.050	1.1	2.2	1,959,126.50
brd14051-51-c2-w120-3000-4000	297	2.8	0.000	0.0	0.1	1,124,620.39
brd14051-61-c2-w120-3000-4000	208	2.4	0.015	0.0	0.0	1,131,675.76
brd14051-71-c2-w120-3000-4000	390	2.8	0.030	0.0	0.1	1,234,087.40
brd14051-81-c2-w120-3000-4000	1,059	3.2	0.022	0.1	0.2	1,435,122.88
brd14051-91-c2-w120-3000-4000	672	3.0	0.009	0.1	0.1	1,542,815.26
brd14051-101-c2-w120-3000-4000	1,056	3.1	0.032	0.1	0.2	1,850,093.25
brd14051-111-c2-w120-3000-4000	3,525	3.7	0.016	0.4	0.6	1,540,458.68
brd14051-121-c2-w120-3000-4000	1,474	3.1	0.068	0.2	0.4	1,851,498.04
brd14051-131-c2-w120-3000-4000	1,487	3.1	0.035	0.2	0.4	1,959,299.94
brd14051-141-c2-w120-3000-4000	6,604	4.4	0.048	1.0	1.4	1,955,931.17
brd14051-151-c2-w120-3000-4000	15,782	4.9	0.118	4.0	6.6	1,955,626.18
d18512-51-c2-w45-3000-4000	103	2.1	0.000	0.0	0.1	1,334,829.84
d18512-61-c2-w45-3000-4000	136	2.4	0.001	0.0	0.0	1,434,275.55
d18512-71-c2-w45-3000-4000	338	3.0	0.013	0.0	0.1	1,435,207.06
d18512-81-c2-w45-3000-4000	401	2.8	0.000	0.0	0.1	1,335,806.81
d18512-91-c2-w45-3000-4000	450	2.8	0.101	0.1	0.1	1,745,020.66
d18512-101-c2-w45-3000-4000	1,239	3.4	0.000	0.1	0.2	1,747,912.75
d18512-111-c2-w45-3000-4000	676	2.9	0.028	0.1	0.2	2,057,507.25
d18512-121-c2-w45-3000-4000	707	3.0	0.005	0.1	0.2	2,467,386.41
d18512-131-c2-w45-3000-4000	2,027	3.8	0.013	0.3	0.4	2,364,494.52
d18512-141-c2-w45-3000-4000	1,864	3.6	0.093	0.3	0.4	2,470,198.64
d18512-151-c2-w45-3000-4000	1,641	3.2	0.008	0.2	0.3	2,365,162.54
d18512-51-c2-w60-3000-4000	108	2.4	0.000	0.0	0.0	1,234,236.81
d18512-61-c2-w60-3000-4000	255	2.7	0.084	0.0	0.1	1,129,637.41

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-71-c2-w60-3000-4000	331	2.8	0.000	0.0	0.1	1,332,572.92
d18512-81-c2-w60-3000-4000	351	2.6	0.000	0.0	0.1	1,443,166.89
d18512-91-c2-w60-3000-4000	615	2.9	0.000	0.1	0.1	1,746,582.50
d18512-101-c2-w60-3000-4000	1,171	3.6	0.023	0.2	0.2	1,950,489.34
d18512-111-c2-w60-3000-4000	1,019	3.4	0.012	0.1	0.2	2,161,004.00
d18512-121-c2-w60-3000-4000	1,292	3.3	0.000	0.2	0.2	2,369,462.86
d18512-131-c2-w60-3000-4000	3,496	4.0	0.000	0.5	0.6	2,163,245.51
d18512-141-c2-w60-3000-4000	5,655	4.0	0.046	0.8	1.2	2,060,225.89
d18512-151-c2-w60-3000-4000	3,388	3.9	0.013	0.5	0.7	2,571,094.43
d18512-51-c2-w75-3000-4000	112	2.2	0.000	0.0	0.0	1,129,919.41
d18512-61-c2-w75-3000-4000	292	2.7	0.000	0.0	0.0	1,131,832.72
d18512-71-c2-w75-3000-4000	333	2.9	0.000	0.0	0.1	1,435,030.78
d18512-81-c2-w75-3000-4000	395	2.8	0.000	0.0	0.1	1,741,331.87
d18512-91-c2-w75-3000-4000	1,544	3.6	0.002	0.2	0.3	1,537,494.05
d18512-101-c2-w75-3000-4000	1,348	3.4	0.044	0.2	0.3	1,643,606.04
d18512-111-c2-w75-3000-4000	1,519	3.7	0.005	0.2	0.3	1,854,577.01
d18512-121-c2-w75-3000-4000	1,081	3.5	0.000	0.2	0.2	2,162,914.08
d18512-131-c2-w75-3000-4000	909	2.9	4.687	0.1	0.2	2,156,812.74
d18512-141-c2-w75-3000-4000	5,202	3.9	4.444	0.7	1.1	2,257,054.90
d18512-151-c2-w75-3000-4000	3,331	3.6	0.066	0.5	0.7	2,165,967.69
d18512-51-c2-w90-3000-4000	147	2.4	0.000	0.0	0.0	1,026,879.76
d18512-61-c2-w90-3000-4000	310	2.9	0.088	0.0	0.1	1,230,401.43
d18512-71-c2-w90-3000-4000	459	2.9	0.062	0.0	0.1	1,231,027.53
d18512-81-c2-w90-3000-4000	505	3.0	0.062	0.1	0.1	1,438,248.08
d18512-91-c2-w90-3000-4000	920	3.6	0.000	0.1	0.2	1,845,072.45
d18512-101-c2-w90-3000-4000	1,813	3.8	0.000	0.2	0.3	1,643,509.41
d18512-111-c2-w90-3000-4000	860	3.0	0.000	0.1	0.2	1,951,322.46
d18512-121-c2-w90-3000-4000	1,323	3.2	0.075	0.2	0.3	2,059,559.91
d18512-131-c2-w90-3000-4000	1,986	3.5	4.902	0.3	0.8	2,056,202.66
d18512-141-c2-w90-3000-4000	5,181	4.3	0.007	0.9	1.3	1,956,422.78
d18512-151-c2-w90-3000-4000	6,009	4.2	0.017	1.1	1.4	2,467,967.46
d18512-51-c2-w120-3000-4000	104	2.1	0.058	0.0	0.0	1,130,584.03
d18512-61-c2-w120-3000-4000	211	2.6	0.000	0.0	0.0	1,131,741.95
d18512-71-c2-w120-3000-4000	571	3.0	0.000	0.1	0.1	1,229,391.78
d18512-81-c2-w120-3000-4000	1,303	3.5	0.004	0.1	0.2	1,233,493.43
d18512-91-c2-w120-3000-4000	787	3.3	0.033	0.1	0.2	1,541,741.88
d18512-101-c2-w120-3000-4000	2,445	4.1	0.000	0.4	0.5	1,545,264.93

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
d18512-111-c2-w120-3000-4000	1,349	3.1	0.028	0.2	0.4	1,751,009.00
d18512-121-c2-w120-3000-4000	1,281	3.1	0.012	0.2	0.3	1,961,978.35
d18512-131-c2-w120-3000-4000	6,551	4.3	0.035	1.7	3.2	1,750,253.74
d18512-141-c2-w120-3000-4000	6,218	4.1	0.079	1.0	1.8	1,956,833.39
d18512-151-c2-w120-3000-4000	11,546	4.3	0.001	2.2	2.7	2,161,196.06
fnl4461-51-c2-w45-3000-4000	193	2.7	0.000	0.0	0.1	714,979.49
fnl4461-61-c2-w45-3000-4000	623	3.4	0.000	0.1	0.1	1,219,256.18
fnl4461-71-c2-w45-3000-4000	1,433	3.4	0.061	0.1	0.2	918,301.38
fnl4461-81-c2-w45-3000-4000	4,455	4.2	0.039	0.4	0.6	921,220.74
fnl4461-91-c2-w45-3000-4000	6,251	3.9	0.008	0.6	0.9	1,021,381.75
fnl4461-101-c2-w45-3000-4000	2,930	3.3	0.017	0.3	0.5	1,327,639.68
fnl4461-111-c2-w45-3000-4000	3,271	3.6	0.029	0.3	0.7	1,329,326.21
fnl4461-121-c2-w45-3000-4000	5,403	3.7	0.070	0.6	1.9	1,434,892.04
fnl4461-131-c2-w45-3000-4000	6,200	3.7	0.035	0.6	2.1	1,638,724.11
fnl4461-141-c2-w45-3000-4000	8,327	3.7	0.009	1.0	1.7	1,740,635.71
fnl4461-151-c2-w45-3000-4000	12,152	3.9	0.014	1.4	3.5	1,638,896.58
fnl4461-51-c2-w60-3000-4000	402	3.1	0.000	0.0	0.1	715,290.92
fnl4461-61-c2-w60-3000-4000	850	3.3	0.007	0.1	0.1	816,052.97
fnl4461-71-c2-w60-3000-4000	3,076	4.1	0.146	0.3	0.7	820,385.20
fnl4461-81-c2-w60-3000-4000	2,442	3.3	0.070	0.2	0.3	922,908.87
fnl4461-91-c2-w60-3000-4000	6,202	4.2	0.003	0.6	0.8	1,021,321.17
fnl4461-101-c2-w60-3000-4000	7,816	4.1	0.015	0.8	1.2	1,126,184.33
fnl4461-111-c2-w60-3000-4000	4,607	3.8	0.052	0.5	1.7	1,330,114.45
fnl4461-121-c2-w60-3000-4000	6,198	3.8	0.027	0.6	1.9	1,330,780.36
fnl4461-131-c2-w60-3000-4000	6,799	3.7	0.034	0.7	1.6	1,539,452.79
fnl4461-141-c2-w60-3000-4000	10,563	4.1	0.051	1.1	3.5	1,536,759.69
fnl4461-151-c2-w60-3000-4000	18,045	4.1	0.082	2.1	6.2	1,540,506.98
fnl4461-51-c2-w75-3000-4000	645	3.5	0.000	0.0	0.1	714,576.91
fnl4461-61-c2-w75-3000-4000	947	3.1	0.019	0.1	0.1	815,999.00
fnl4461-71-c2-w75-3000-4000	1,369	3.4	0.009	0.1	0.3	918,376.14
fnl4461-81-c2-w75-3000-4000	2,743	3.8	0.005	0.3	0.4	919,530.42
fnl4461-91-c2-w75-3000-4000	4,075	3.9	0.059	0.5	1.2	1,023,900.79
fnl4461-101-c2-w75-3000-4000	3,324	3.7	0.037	0.4	0.9	1,225,903.50
fnl4461-111-c2-w75-3000-4000	5,706	3.8	0.129	0.6	2.9	1,130,999.99
fnl4461-121-c2-w75-3000-4000	8,448	3.8	0.063	0.9	2.6	1,432,616.69
fnl4461-131-c2-w75-3000-4000	14,512	4.3	0.049	1.6	6.3	1,434,903.88
fnl4461-141-c2-w75-3000-4000	8,291	3.7	0.051	0.9	3.2	1,637,689.44

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
fnl4461-151-c2-w75-3000-4000	24,986	4.4	0.029	3.0	8.4	1,540,360.19
fnl4461-51-c2-w90-3000-4000	548	3.5	0.000	0.1	0.1	714,327.75
fnl4461-61-c2-w90-3000-4000	1,336	4.0	0.163	0.1	0.2	817,382.42
fnl4461-71-c2-w90-3000-4000	2,517	3.8	0.023	0.3	0.4	815,992.95
fnl4461-81-c2-w90-3000-4000	1,572	3.6	0.054	0.1	0.2	921,585.08
fnl4461-91-c2-w90-3000-4000	6,573	4.3	9.606	0.7	2.0	1,022,957.38
fnl4461-101-c2-w90-3000-4000	5,332	3.7	8.039	0.6	2.0	1,225,577.75
fnl4461-111-c2-w90-3000-4000	15,131	4.2	0.001	1.8	2.4	1,227,613.81
fnl4461-121-c2-w90-3000-4000	9,563	3.9	0.065	1.1	3.8	1,332,946.61
fnl4461-131-c2-w90-3000-4000	22,620	4.4	0.183	2.8	8.2	1,233,130.49
fnl4461-141-c2-w90-3000-4000	30,011	4.7	0.070	3.7	10.8	1,435,430.83
fnl4461-151-c2-w90-3000-4000	39,590	4.4	0.063	5.2	23.8	1,538,586.78
fnl4461-51-c2-w120-3000-4000	1,077	3.4	0.044	0.1	0.3	611,149.55
fnl4461-61-c2-w120-3000-4000	1,961	3.7	0.063	0.2	0.4	714,722.89
fnl4461-71-c2-w120-3000-4000	10,661	5.1	0.005	1.6	2.1	715,189.64
fnl4461-81-c2-w120-3000-4000	9,616	4.6	0.039	1.3	2.4	819,861.58
fnl4461-91-c2-w120-3000-4000	17,886	4.6	0.019	2.4	3.6	919,380.62
fnl4461-101-c2-w120-3000-4000	12,729	4.2	0.052	1.4	4.5	1,125,947.35
fnl4461-111-c2-w120-3000-4000	24,683	4.6	8.720	3.6	16.2	1,125,604.41
fnl4461-121-c2-w120-3000-4000	22,209	4.5	7.941	3.5	14.1	1,229,653.79
fnl4461-131-c2-w120-3000-4000	41,363	4.7	0.081	5.8	14.5	1,132,683.64
fnl4461-141-c2-w120-3000-4000	73,513	5.0	0.072	15.6	30.6	1,336,041.89
fnl4461-151-c2-w120-3000-4000	68,288	4.8	0.117	11.7	46.2	1,337,997.98
nrw1379-51-c2-w45-3000-4000	425	3.2	0.000	0.1	0.2	815,723.63
nrw1379-61-c2-w45-3000-4000	588	3.2	0.000	0.0	0.1	918,778.59
nrw1379-71-c2-w45-3000-4000	1,253	3.7	0.000	0.1	0.2	1,121,131.08
nrw1379-81-c2-w45-3000-4000	1,227	3.5	0.061	0.1	0.2	1,231,182.73
nrw1379-91-c2-w45-3000-4000	1,453	3.3	0.000	0.2	0.2	1,330,074.59
nrw1379-101-c2-w45-3000-4000	3,096	3.6	0.039	0.3	0.5	1,329,716.37
nrw1379-111-c2-w45-3000-4000	3,442	3.8	0.037	0.3	0.7	1,534,214.38
nrw1379-121-c2-w45-3000-4000	3,568	3.5	0.069	0.3	0.9	1,539,555.93
nrw1379-131-c2-w45-3000-4000	2,858	3.4	0.007	0.3	0.4	1,842,083.56
nrw1379-141-c2-w45-3000-4000	5,239	3.6	0.012	0.6	0.9	1,945,084.54
nrw1379-151-c2-w45-3000-4000	3,360	3.6	0.025	0.4	0.8	2,051,154.32
nrw1379-51-c2-w60-3000-4000	503	3.6	0.000	0.0	0.1	714,314.05
nrw1379-61-c2-w60-3000-4000	924	3.7	0.136	0.1	0.2	818,782.39
nrw1379-71-c2-w60-3000-4000	1,285	3.9	0.029	0.1	0.3	1,122,155.38

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-81-c2-w60-3000-4000	1,568	3.3	0.000	0.2	0.2	1,226,778.19
nrw1379-91-c2-w60-3000-4000	1,795	3.3	0.018	0.2	0.3	1,228,500.47
nrw1379-101-c2-w60-3000-4000	2,502	3.6	0.000	0.2	0.3	1,429,735.52
nrw1379-111-c2-w60-3000-4000	2,967	3.5	0.010	0.3	0.5	1,634,836.10
nrw1379-121-c2-w60-3000-4000	4,837	3.8	0.028	0.5	1.4	1,539,881.33
nrw1379-131-c2-w60-3000-4000	10,874	4.3	0.090	1.3	3.8	1,639,294.46
nrw1379-141-c2-w60-3000-4000	10,439	4.0	0.010	1.3	2.4	1,947,046.01
nrw1379-151-c2-w60-3000-4000	15,122	4.1	5.304	1.9	4.2	1,844,156.77
nrw1379-51-c2-w75-3000-4000	892	4.0	0.000	0.1	0.2	918,881.93
nrw1379-61-c2-w75-3000-4000	572	3.1	0.006	0.0	0.1	1,019,512.92
nrw1379-71-c2-w75-3000-4000	1,541	3.8	0.000	0.1	0.2	1,020,804.06
nrw1379-81-c2-w75-3000-4000	1,775	3.3	0.043	0.2	0.4	1,125,447.56
nrw1379-91-c2-w75-3000-4000	3,686	4.1	0.000	0.4	0.5	1,229,755.72
nrw1379-101-c2-w75-3000-4000	4,588	4.1	0.000	0.5	0.7	1,225,428.40
nrw1379-111-c2-w75-3000-4000	3,296	3.6	0.000	0.3	0.5	1,435,236.21
nrw1379-121-c2-w75-3000-4000	5,300	3.6	0.009	0.5	0.9	1,530,436.88
nrw1379-131-c2-w75-3000-4000	6,368	3.8	5.908	0.7	1.9	1,637,411.15
nrw1379-141-c2-w75-3000-4000	12,014	3.9	0.047	1.5	4.1	1,741,382.40
nrw1379-151-c2-w75-3000-4000	9,740	4.2	0.030	1.2	3.4	1,743,505.71
nrw1379-51-c2-w90-3000-4000	582	3.4	0.010	0.0	0.1	815,953.28
nrw1379-61-c2-w90-3000-4000	1,191	3.7	0.000	0.1	0.2	1,222,460.71
nrw1379-71-c2-w90-3000-4000	3,634	4.2	0.000	0.4	0.5	1,021,147.48
nrw1379-81-c2-w90-3000-4000	5,557	4.5	0.000	0.6	0.8	1,023,997.64
nrw1379-91-c2-w90-3000-4000	4,466	4.2	0.013	0.5	0.7	1,323,626.23
nrw1379-101-c2-w90-3000-4000	3,007	3.6	0.003	0.4	0.5	1,427,968.74
nrw1379-111-c2-w90-3000-4000	5,025	3.9	0.011	0.5	0.8	1,329,356.28
nrw1379-121-c2-w90-3000-4000	11,961	4.6	0.000	1.8	2.2	1,334,100.22
nrw1379-131-c2-w90-3000-4000	16,779	4.5	0.017	2.3	3.8	1,334,712.82
nrw1379-141-c2-w90-3000-4000	21,821	4.5	6.026	2.9	9.1	1,641,069.58
nrw1379-151-c2-w90-3000-4000	8,379	4.0	0.029	1.0	2.7	1,638,621.79
nrw1379-51-c2-w120-3000-4000	2,014	4.4	0.000	0.3	0.3	615,545.78
nrw1379-61-c2-w120-3000-4000	1,425	3.9	0.023	0.1	0.2	818,667.33
nrw1379-71-c2-w120-3000-4000	4,303	4.6	0.019	0.5	0.9	1,023,418.02
nrw1379-81-c2-w120-3000-4000	3,999	4.1	0.006	0.5	0.6	924,129.12
nrw1379-91-c2-w120-3000-4000	5,505	4.0	0.000	0.8	1.0	1,122,968.98
nrw1379-101-c2-w120-3000-4000	5,794	3.9	0.048	0.6	1.8	1,230,463.26
nrw1379-111-c2-w120-3000-4000	12,604	4.7	0.186	1.6	7.0	1,233,305.80

Continued Table B.6 – *Three stacks vehicle of C2 class*

Instance	F_N	$Avg._{Req.}$	*Gap*	Sec_f	*Sec.*	Z^*
nrw1379-121-c2-w120-3000-4000	14,050	4.4	0.065	1.7	4.2	1,332,059.88
nrw1379-131-c2-w120-3000-4000	22,447	4.4	0.071	3.1	8.8	1,435,224.11
nrw1379-141-c2-w120-3000-4000	26,682	4.6	0.013	3.5	5.6	1,640,403.70
nrw1379-151-c2-w120-3000-4000	39,751	4.7	0.048	6.5	13.8	1,538,086.58

B.2. The PDPTWMS-S Computational Results

We applied our algorithm to the a280 instances of C2 class assuming the number of stacks is 4, 6 or 24 and fixing the vehicle capacity to 24 (24 stacks corresponds to the PDPTW solution). Tables B.7-B.9 show the computational results for these modified instances.

The columns of Tables B.7-B.9 represent the following: *Instance*: the instance name; F_N: the total number of non-dominated fragments; Sec_f: the CPU time in seconds used to generate non-dominated fragments, rounded to the nearest tenth; *Sec.*: the total CPU time in seconds used to solve the instance, including Sec_f time, rounded to the nearest tenth; Z^*: the optimal solution obtained within the time limit or left blank if no feasible solution found. If only a feasible solution was obtained within the time limit, then it is reported in bold in column Z^* as a best upper bound found; *Gap*: The percentage gap computed as $100(Z^* - Z_R)/Z^*$, where Z_R is the lower bound at the root node.

Table B.10 shows the percentage amount of saving in the total cost when using more stacks. For example, if the optimal costs for 3 and 24 stacks are Z^*_{3St} and Z^*_{24St} respectively, then $24St - vs. - 3St = ((Z^*_{3St} - Z^*_{24St})/Z^*_{24St}) \times 100$.

Table B.7. PDPTWMS-S for 4 Stacks Vehicle with Capacity 24

Instance(a280-C2)	F_N	*Gap*	Sec_f	*Sec.*	Z*
141-w15-500-1000	5,426	0.002	3.5	3.8	1,913,317.00
151-w15-500-1000	4,386	0.012	3.5	4.6	2,015,130.85
141-w15-1000-1200	11,464	0.018	6.7	9.1	1,712,984.69
151-w15-1000-1200	14,862	0.014	9.0	12.6	1,614,020.98
141-w15-1500-2000	136,702	8.289	158.9	263.5	1,211,594.58
151-w15-1500-2000	381,490	0.051	279.6	1,148.9	1,212,723.09
141-w30-500-1000	15,930	0.028	13.5	22.7	1,612,796.91
151-w30-500-1000	19,939	0.018	13.5	22.1	1,612,587.85
141-w30-1000-1200	42,676	0.021	36.9	54.1	1,411,836.82
151-w30-1000-1200	47,648	0.020	52.3	83.5	1,512,363.45
141-w30-1500-2000	846,802	0.022	917.9	1,161.6	1,210,906.74
151-w30-1500-2000	789,151	0.022	652.8	998.2	1,212,015.09
141-w45-500-1000	26,153	0.021	39.2	45.5	1,612,937.08
151-w45-500-1000	70,521	0.016	62.2	93.2	1,612,285.05
141-w45-1000-1200	95,451	7.593	146.7	246.2	1,311,031.31
151-w45-1000-1200	217,880	0.028	317.7	393.6	1,312,223.43
141-w45-1500-2000	1,392,558	0.024	3,135.8	4,041.3	1,211,259.52
151-w45-1500-2000	4,064,581		4,507.0		

Table B.8. PDPTWMS-S for 6 Stacks Vehicle with Capacity 24

Instance(a280-C2)	F_N	*Gap*	Sec_f	*Sec.*	Z*
141-w15-500-1000	5,634	5.219	3.6	4.5	1,913,172.11
151-w15-500-1000	4,757	0.030	3.6	4.7	1,915,843.87
141-w15-1000-1200	12,856	0.000	6.9	7.4	1,712,480.45
151-w15-1000-1200	15,588	0.022	10.0	15.9	1,613,947.12
141-w15-1500-2000	172,148	0.103	160.5	1,370.3	1,112,822.98
151-w15-1500-2000	442,218	0.052	347.4	1,485.6	1,212,484.09
141-w30-500-1000	16,690	6.160	15.0	24.8	1,612,356.58
151-w30-500-1000	21,223	0.026	15.7	26.3	1,612,597.69
141-w30-1000-1200	49,929	0.046	42.4	52.0	1,312,319.96
151-w30-1000-1200	52,690	6.602	62.6	190.9	1,512,111.16
141-w30-1500-2000	1,054,032	0.072	1,071.0	3,171.4	1,112,407.08
151-w30-1500-2000	1,030,131	8.107	797.5	2,501.4	1,211,586.62
141-w45-500-1000	32,909	0.012	37.7	47.4	1,612,149.74
151-w45-500-1000	78,436	0.015	69.9	89.9	1,612,147.86
141-w45-1000-1200	103,332	0.055	148.0	250.8	1,211,924.47
151-w45-1000-1200	225,985	0.020	307.6	375.5	1,311,855.52
141-w45-1500-2000	1,930,451	0.026	3,745.8	4,864.7	1,210,762.60
151-w45-1500-2000	4,949,498		5,007.8		

Table B.9. PDPTWMS-S for 24 Stacks Vehicle with Capacity 24

Instance(a280-C2)	F_N	*Gap*	Sec_f	*Sec.*	Z*
141-w15-500-1000	5,715	0.002	0.6	0.8	1,813,050.38
151-w15-500-1000	4,990	0.001	0.5	0.8	1,914,608.41
141-w15-1000-1200	13,310	0.000	1.4	1.9	1,712,432.73
151-w15-1000-1200	16,104	0.024	1.7	7.3	1,613,678.08
141-w15-1500-2000	199,549	0.051	23.5	352.6	1,111,767.29
151-w15-1500-2000	490,863	0.028	63.7	823.3	1,211,710.38
141-w30-500-1000	17,580	0.047	2.0	8.1	1,513,220.26
151-w30-500-1000	22,210	0.021	2.7	13.4	1,612,318.97
141-w30-1000-1200	58,309	0.038	6.7	43.6	1,311,682.70
151-w30-1000-1200	59,026	0.016	7.7	25.7	1,412,093.96
141-w30-1500-2000	1,254,108	0.037	183.6	869.8	1,111,290.47
151-w30-1500-2000	1,145,189	0.056	194.9	3,523.2	1,113,092.26
141-w45-500-1000	35,905	0.008	4.6	15.2	1,611,932.19
151-w45-500-1000	82,878	0.013	13.4	45.9	1,611,904.59
141-w45-1000-1200	114,192	0.031	19.1	85.3	1,211,229.81
151-w45-1000-1200	261,030	0.022	53.3	149.0	1,311,711.88
141-w45-1500-2000	2,540,821	0.020	467.9	1,462.4	1,210,451.08
151-w45-1500-2000	5,628,399	0.028	1,133.3	7,200.0	**1,210,755.16**

Table B.10. The Impact of Using More Stacks in the PDPTWMS-S (Vehicle Capacity 24)

Instance(a280-C2)	*24St-vs.-2St*	*24St-vs.-3St*	*24St-vs.-4St*	*24St-vs.-6St*
141-w15-500-1000	11.07	5.52	5.53	5.52
151-w15-500-1000	10.56	5.24	5.25	0.06
141-w15-1000-1200	5.88	0.02	0.03	0.00
151-w15-1000-1200	6.27	0.04	0.02	0.02
141-w15-1500-2000	18.05	9.00	8.98	0.09
151-w15-1500-2000	16.63	0.10	0.08	0.06
141-w30-500-1000	13.22	6.57	6.58	6.55
151-w30-500-1000	6.27	0.03	0.02	0.02
141-w30-1000-1200	15.31	7.61	7.64	0.05
151-w30-1000-1200	14.22	7.09	7.10	7.08
141-w30-1500-2000	9.11	8.97	8.96	0.10
151-w30-1500-2000	17.90	8.90	8.89	8.85
141-w45-500-1000	6.31	0.07	0.06	0.01
151-w45-500-1000	6.30	0.04	0.02	0.02
141-w45-1000-1200	8.35	8.25	8.24	0.06
151-w45-1000-1200	7.66	0.05	0.04	0.01
141-w45-1500-2000	8.39	0.08	0.07	0.03
151-w45-1500-2000	0.12			