Truck scheduling optimization in the logistics of crossdocking

Shakeri, Mojtaba

2013

Shakeri, M. (2013). Truck scheduling optimization in the logistics of crossdocking. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/54632>

<https://doi.org/10.32657/10356/54632>

TRUCK SCHEDULING OPTIMIZATION IN THE LOGISTICS OF CROSSDOCKING

MOJTABA SHAKERI

School of Computer Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of

Doctor of Philosophy

2013

Acknowledgments

I would like to thank my supervisor, Prof. Stephen John Turner, who has been a great source of inspiration and wisdom for me. His high standards of professional excellence have motivated me to make my work thorough, precise, and insightful at every turn. I am grateful to my co-supervisor, Dr. Eng Wah Lee, the Division Director of the Manufacturing System Division in Singapore Institute of Manufacturing Technology (SIMTech) for his trust and encouraging remarks. I should also thank my ex-supervisors, Dr. Malcolm Yoke Hean Low from D-SIMLAB Technologies Pte. Ltd. and Dr. Zhengping Li from the SIMTech Planning and Operations Management Group. Throughout his supervision, Dr. Low supported me with his patience and knowledge while allowing me the room to work in my own way. Likewise, Dr. Li directed me to define and work out the project from the logistics point of view. Finally, I want to express my gratitude to Pim Beers, the Senior AIMMS Specialist from Paragon Decision Technology Pte. Ltd., for his significant contributions in developing the crossdocking mathematical model under the AIMMS modeling environment.

Contents

Acknowledgments	i
List of Figures	v
List of Tables	vi
List of Procedures	ix
List of Notations	x
List of Abbreviations	xiii
Abstract	xv
 1 Introduction	 1
1.1 What Is Crossdocking?	1
1.2 Research Motivation and Objective	4
1.3 Thesis Organization and Contributions	7
 2 Literature Review	 9
2.1 Operational Strategies	9
2.2 Evaluation Requirements	17
2.3 Solution Techniques	19
2.4 Summary	24
 3 Formulation & Evaluation Methodology	 26
3.1 Crossdocking Model Development	27
3.1.1 Truck Scheduling Formulation	28
3.1.2 Problem Complexity	32
3.1.3 Complications in Standard Scheduling Notation	33
3.2 Integer Program	36

3.2.1	Virtual Pallets	43
3.2.2	Model Practicality and Limitation	43
3.3	Test Data Generation	44
3.4	Imposing Constraint on Crossdock Capacity	49
3.5	Summary	51
4	Solution Feasibility Construction	53
4.1	Truck Scheduling Feasibility	54
4.2	A Two-Phase Heuristic Algorithm	55
4.2.1	Sequencing Phase: DRS Heuristic	56
4.2.2	Assignment Phase: DF Heuristic	69
4.3	Experimental Results and Analysis	74
4.3.1	Heuristics vs. MIP Model: Efficiency	75
4.3.2	DRS vs. DR: Trailer-to-Door Ratio & Freight Mixed Level	77
4.3.3	Discussion on DRS Efficiency: Robustness & Solution Quality . .	86
4.4	Summary	91
5	Solution Quality Improvement	93
5.1	Optimization Framework	94
5.1.1	Search Space	94
5.1.2	Neighborhood Structure	95
5.2	Fractional Neighborhood Evaluation	97
5.2.1	Fractional Feasibility Verification	99
5.2.2	Fractional Solution Evaluation	100
5.3	Meta-Heuristics Algorithms	105
5.3.1	Tabu Search (TS) Algorithm	107
5.3.2	Variable Neighborhood Search (VNS) Algorithm	111
5.4	Solution Quality Metrics	116
5.4.1	Simplified MIP Model (S-MIP)	117
5.4.2	Multi-Start Local Search (M-LS)	121
5.5	Experimental Results and Analysis	124
5.5.1	Fractional vs. Full Neighborhood Evaluation	124

5.5.2	TS and VNS Evaluation	127
5.5.3	TS & VNS Solution Comparison with S-MIP & M-LS	138
5.6	Summary	145
6	Conclusion and Future Work	148
6.1	Research Accomplishments	148
6.2	Future Research Suggestions	151
	References	156
A	Detailed Presentation of Results	165
B	TS and VNS Parameter Tuning	180
C	Extended Variable Neighborhood Search	194
D	List of Publications	198

List of Figures

1.1	Crossdocking reduces supply chain inventory and transportation costs . .	2
1.2	A common crossdock layout	3
3.1	Structure and operation of the proposed crossdocking model	27
3.2	Interdependencies of six trailers exchanging products in a crossdock . . .	35
3.3	Timing details of trailer 5 in the crossdock	42
4.1	Complications in scheduling eight trailers in a four-door crossdock	55
4.2	DRS state transitions to sequence eight trailers in a four-door crossdock .	63
4.3	DRS performance in establishing solution feasibility	83
5.1	DRS+DF solution structure for the truck scheduling problem	95
5.2	Representation and evaluation of a solution and its neighbor	96
5.3	Solution s and the three examples of its perturbed solution s'	98
5.4	An example depicting the case required to displace the pivot backwards .	102
5.5	A general presentation of the fractional neighborhood evaluation	103
5.6	VNS neighborhood structures	116
5.7	Impact of initial solution vs. multiple runs on TS & VNS performance .	138
5.8	R-DRS, TS, VNS, R-TS, R-VNS efficiency in improving DRS solutions .	147

List of Tables

2.1	Overview on crossdocking operational strategies	13
2.2	Overview on truck scheduling evaluation and solution methodologies . . .	20
3.1	Test data characterization for the truck scheduling problem	46
3.2	Values of history weight table for uncorrelated and correlated approaches	48
3.3	Crossdock capacity utilization statistics	51
4.1	Parameters defined to describe the DRS algorithm	57
4.2	Interdependency matrix of eight trailers	60
4.3	Cluster formation and ranking of eight trailers	60
4.4	DRS procedure to sequence eight trailers in a four-door crossdock	64
4.5	Parameters defined to describe the DR algorithm	67
4.6	Parameters defined to describe the DF algorithm	70
4.7	The heuristics vs. the mathematical model for (small:1) instances	77
4.8	DRS vs. DR when FML is kept low and α varies	79
4.9	Percentage of feasibility for truck scheduling test instances	81
4.10	DRS vs. DR _b when $\alpha = 2$ and FML is high	84
4.11	DRS vs. DR _b when $\alpha = 4$ and FML is medium	85
4.12	DRS efficiency in the solution quality w.r.t. the initial cluster	90
5.1	Parameters defined to describe the FCHECK and SEVAL Procedures . . .	100
5.2	Interdependencies of eight trailers scheduled for crossdocking	101
5.3	Popular meta-heuristics and their standard control parameters	107
5.4	Parameters defined to describe the TS Procedure	109
5.5	Parameters defined to describe the VNS Procedure	112

5.6	Parameters defined to describe the POSTMIP Procedure	120
5.7	Parameters defined to describe the M-LS Procedure	122
5.8	Reduction in runtime when fractional neighborhood evaluation is applied	126
5.9	TS and VNS control parameters	128
5.10	Values assigned to TS and VNS control parameters after tuning	129
5.11	TS & VNS efficiency for small and medium instances	131
5.12	TS & VNS efficiency for large and huge instances	132
5.13	R-TS & R-VNS efficiency for small and medium instances	135
5.14	R-TS & R-VNS efficiency for large and huge instances	136
5.15	Solution comparison of TS with S-MIP[+Post] for small instances	140
5.16	TS & VNS solution comparison for small and medium instances	142
5.17	TS & VNS solution comparison for large and huge instances	143
5.18	TS & VNS runtime comparison for large and huge instances	145
A.1	DRS vs. DR for small instances when FML is kept low and α varies . . .	166
A.2	DRS vs. DR for medium instances when FML is kept low and α varies .	167
A.3	DRS vs. DR for large instances when FML is kept low and α varies . . .	168
A.4	DRS vs. DR for huge instances when FML is kept low and α varies . . .	169
A.5	Percentage of feasibility found by DRS and DR _b for small instances . . .	170
A.6	Percentage of feasibility found by DRS and DR _b for medium instances . .	170
A.7	Percentage of feasibility found by DRS and DR _b for large instances . . .	171
A.8	Percentage of feasibility found by DRS and DR _b for huge instances . . .	171
A.9	TS & VNS efficiency for large instances of $\alpha \geq 4$	172
A.10	TS & VNS efficiency for huge instances of $\alpha \geq 4$	173
A.11	R-TS & R-VNS efficiency for large instances of $\alpha \geq 4$	174
A.12	R-TS & R-VNS efficiency for huge instances of $\alpha \geq 4$	175
A.13	TS & VNS solution comparison for large instances of $\alpha \leq 2$ & high FML	176
A.14	TS & VNS solution comparison for large instances of $\alpha \geq 4$	177
A.15	TS & VNS solution comparison for huge instances of $\alpha \leq 2$ & high FML	178
A.16	TS & VNS solution comparison for huge instances of $\alpha \geq 4$	179
B.1	TS and VNS factor levels	182

B.2	Parameters used for the description of ANOVA	183
B.3	ANOVA template for a two-factor factorial design	183
B.4	ANOVA template for a three-factor factorial design	183
B.5	ANOVA for the factors of TS for small and medium instances	186
B.6	ANOVA for the factors of TS for large and huge instances	187
B.7	ANOVA for the significance of the TS factors in their middle values . . .	189
B.8	Final values considered for the parameters of the TS algorithm	190
B.9	ANOVA for the factors of VNS for small and medium instances	192
B.10	ANOVA for the factors of VNS for large and huge instances	193
C.1	E-VNS efficiency vs. TS & VNS for small and medium instances	196
C.2	E-VNS efficiency vs. TS & VNS for large and huge instances	197

List of Procedures

4.1	DEPENDENCYRANKINGSEARCH	65
4.2	DEPENDENCYRANKING ₁₂₃	68
4.3	DOORFITNESS	73
5.1	FCHECK(n, s, s', p, w)	100
5.2	SEVAL(n, s, s', p, s_{best})	104
5.3	TS(n, s_0, r, l, m)	110
5.4	SHAKE(n, s, k)	113
5.5	LS(n, s, r)	113
5.6	VNS($n, s_0, k_{\max}, r, m$)	113
5.7	POSTMIP(n', X, Y)	120
5.8	M-LS(n, e, m)	123

List of Notations

a_j	Assignment time of trailer j
C_j	Cluster corresponding to trailer j
c_j	Completion time of trailer j
$c_{\max}$	Schedule length (or makespan)
c_j^S	Contribution of trailer j when included in list L^S [based on its product exchange pattern]
c_j^A	Contribution of trailer j when included in list L^A [based on its product exchange pattern]
$d_{mm'}$	[rectilinear] distance from door m to door m'
e_j	Total number of exchanged pallets of trailer j
$f(s)$	Objective function [returns the makespan corresponding to solution s]
$f_{jj'}$	Amount of freight sent from trailer j to trailer j'
f_m	Fitness of door m [measured in time]
$g(j)$	Returns the number of available doors at the time of assigning trailer j
$h(j)$	Returns the order of trailers receiving pallets from trailer j
I	Intermediate storage [in number of pallets] reserved for each door
i, i'	Free indices representing any object
$i_{\max}$	Maximum number of iterations with no improvement in solution quality for TS or VNS algorithms
J	Set of trailers for crossdocking
J^K	Set of trailers whose assigned door is known
j, j'	Indices representing a trailer
$k_{\max}$	Maximum number of neighborhoods in VNS algorithm
L^A	List of trailers assigned yet have not released their assigned door
L^C	List of initial clusters sorted in the descending order of their rank
L^R	List of trailers that have released their assigned door
L^S	List of trailers for sequencing and assignment
L_j	Set of pallets loaded onto trailer j
l	Tabu list size
l_m	Physical location of door m
M	Set of crossdock doors
$M_{[j]}$	[candidate] assigned door of trailer j

m, m'	Indices representing a door
$N(s)$	Set of neighbors of solution s [based on adjacent pair-wise interchange structure]
$N'(s)$	A reduced sample of neighborhood $N(s)$ of solution s
$N_k(s)$	Set of neighbors of solution s based on neighborhood structure k ($k = 1$ to $k_{\max}$) in VNS algorithm
N_m^C	Set of neighbors of door m [based on a neighborhood defined by the average cluster size]
n	Total number of trailers for crossdocking
n'	Total number of crossdock doors
n^E	Number of elite solutions in solution pool O
$n_{jj'}$	Number of pallets sent from trailer j to trailer j'
$n_{\max}$	Maximum number of solutions examined to build solution pool O
O	Solution pool [contains solutions generated by restarting DRS]
$best(O)$	returns the best solution in solution pool O
$worst(O)$	returns the worst solution in solution pool O
P	Set of pallets
P_j	Set of predecessors of trailer j
p, p'	Indices representing a pallet
p^V	Virtual pallet
Q	A big number not less than the worst schedule length
Q_m	Queue of trailers to be assigned to door m
q_p	1, if pallet p is to be moved [by a forklift] before loaded onto its destination trailer, else 0
r	Number of neighbors in reduced neighborhood $N'(s)$ of solution s chosen randomly
r_j^E	External rank of cluster j
r_j^I	Internal rank of cluster j
r_j^S	Rank of trailer j in list L^S
r_j^U	Rank of trailer j which is still unsequenced
S	List of sequenced trailers
S_j	Set of successors of trailer j
s	Current solution [records the sequence of trailers for the door assignment]
s', s''	Solutions produced after perturbing solution s
$s_{[1..i]}$	Partial solution [contains a subsequence of trailers from position 1 to i]
s_0	Initial solution
s_{best}	Best solution
s_{l_best}	Local optimum in the neighborhood of solution s
s_{nt_best}	Best non-tabu solution recorded per iteration of TS algorithm
T	Tabu list
T^C	Trailer changeover time

t^L	Time taken to load a pallet
t^U	Time taken to unload a pallet
t_m	Availability time of door m
$t_{mm'}$	Time taken to move a pallet from door m to door m'
t_m^N	Total availability times of doors in neighborhood N_m^C of door m
U	Unsequenced trailers whose ranks have changed since the first iteration of DR algorithm
U_j	Set of pallets unloaded from trailer j
v	Pivot equal to position of the very first affected trailer in a perturbed solution
$v_{jmj'm'}$	1, if trailer j is assigned to door m and trailer j' is assigned to door m' , else 0
w	Window of affected trailers in a perturbed solution
$w_{pp'}$	1, if both pallets p and p' are to be moved [by a forklift] before loaded onto their destination trailers, else 0
X	Matrix storing values of decision variable x_{jm} generated by S-MIP model
x_{jm}	1, if trailer j is assigned to door m , else 0
Y	Matrix storing values of decision variable $y_{jj'}$ generated by S-MIP model
$y_{jj'}$	1, if for trailers j and j' assigned to the same door, j precedes j' , else 0
z_{pj}	1, if for trailer j assigned to the same door where pallet p is staged, the movement of p initiates after j 's assignment, else 0
α	The ratio of the number of trailers to the number of doors
β_p	Unloading position of pallet p
$\gamma_{pp'}$	1, if for pallets p and p' loaded onto the same trailer, p is loaded before p' , else 0
$\delta_{pp'}$	1, if for pallets p and p' staged onto the same staging area, p is moved before p' , else 0
λ_p	Time when moving of pallet p starts
μ_p	Time when moving of pallet p is completed
σ_p	Time when loading of pallet p is completed

List of Abbreviations

AI	Artificial Intelligence
ANOVA	Analysis of Variance
ACO	Ant Colony Optimization
B&B	Branch and Bound
BDP	Bounded Dynamic Programming
CDC	Crossdocking Distribution Center
DAG	Directed Acyclic Graph
DE	Differential Evolution
DF	Door Fitness
DG	Directed Graph
DP	Dynamic Programming
DR	Dependency Ranking
DRS	Dependency Ranking Search
EMA	Electro-Magnetism-like Algorithm
FMCG	Fast Moving Consumer Goods
FML	Freight Mixed Level
GA	Genetic Algorithms
GRASP	Greedy Randomized Adaptive Search Procedure
HS	Harmony Search
IT	Information Technology
JIT	Just in Time
L-VNS	Long Variable Neighborhood Search
LB	Lower Bound
LS	Local Search
LTL	Less than Truck Load
M-LS	Multi-start Local Search
MIP	Mixed Integer Programming
MOOP	Multi-Objective Optimization Problem
OR	Operations Research
PSO	Particle Swarm Optimization
QAP	Quadratic Assignment Problem

R-DRS	Re-starting Dependency Ranking Search
R-TS	Re-starting Tabu Search
R-VNS	Re-starting Variable Neighborhood Search
S-MIP	Simplified Mixed Integer Programming
SA	Simulated Annealing
SCC	Strongly Connected Components
SWO	Squeaky Wheel Optimization
TL	Full Truck Load
TS	Tabu Search
VNS	Variable Neighborhood Search

Abstract

Crossdocking is a warehousing strategy that moves products through flow consolidation centers or crossdocks without putting them into storage. The practice not only brings substantial reduction in the distribution cost but also shortens the product delivery time. This thesis addresses an important crossdocking planning operation which is the scheduling of trucks at the crossdocking facilities of the fast moving consumer goods (FMCG) industry and military logistics. The truck scheduling problem in this scenario decides on the sequence of a number of trucks that are to exchange some of their products at the dock doors of a crossdocking terminal. The decision is subject to the availability of the crossdock critical resources including dock doors and forklifts, whose allocation is considered non-preemptive. The non-preemptive assignment of trailers to dock doors makes it necessary to address the truck scheduling solution feasibility before its optimality as the problem might be entrapped in deadlock and no feasible solution is produced.

In the first place, an exact (optimal) mixed integer programming (MIP) model is developed to mathematically represent the crossdocking model and its associated truck scheduling problem. The objective function applied is to minimize the total crossdocking operation time (also referred to as makespan) starting when the first trailer is assigned to a door and ending when the last trailer releases its assigned door. Since there is no real data available to evaluate the formulated problem, synthetic data is generated based on the characteristics of a wide range of real-world crossdocking practices. The data set includes problem instances of different levels of difficulty representing small crossdocks hosting a few dozen trailers to huge ones hosting over 2000 trailers.

As the mathematical approach is impractical for solving real size problem instances, sub-optimal (heuristic) algorithms are developed alternatively. The first step addresses the feasibility of the truck scheduling problem by proposing a two-phase heuristic algorithm where in the first phase, a dependency ranking search (DRS) heuristic is deployed

to construct a feasible sequence of trucks for the assignment to dock doors and in the second phase, a door fitness (DF) heuristic is implemented to assign each sequenced truck to a proper dock door, subject to a limited number of forklifts, such that significant savings in the total crossdocking operation time are achieved. The efficiency of the proposed algorithm in establishing solution feasibility is evaluated against the exact mathematical model of the problem and a constructive dependency ranking (DR) heuristic developed for a similar truck scheduling problem. Experimental results show that the DRS algorithm is robust in avoiding deadlock and generates feasible solutions for the instances where the other two approaches cannot. It is also observed that re-starting DRS from a few initial points to generate a number of trailer sequence lists can noticeably improve the solution quality.

The last step addresses the optimality of the truck scheduling problem by developing two meta-heuristics to improve the initial heuristics solutions: tabu search (TS) and variable neighborhood search (VNS). The search space of the problem is formed by changing the sequence of trailers for the door assignment. As a neighborhood search method, each meta-heuristic is supplemented by a fractional evaluation of the neighborhood so that the runtime is more efficiently dedicated to solution optimization. The experimental results demonstrate that the fractional evaluation significantly reduces the solution assessment time. The experimentation continues by tuning the control parameters of the two meta-heuristics so that the trade-off between solution quality and runtime can be properly balanced. The TS and VNS algorithms are then invoked to improve the DRS+DF solutions. The analysis shows a consistent reduction in the makespan in reasonable running times for different characteristics of the experimental data. Further robustness and increase in the solution quality are achieved when the two meta-heuristics are initialized with a better initial solution obtained by the re-starting version of the DRS algorithm.

Chapter 1

Introduction

The increasing competition in the business sectors has accelerated the significant growth of the retail industry by using supply chain management concepts to streamline the flow of goods. One of the innovative methodologies to enhance the product flows in the supply chain is crossdocking [1]. Instead of being stored in the warehouse, products are shipped to their end destinations as soon as they are received. This thesis addresses an important crossdocking planning operation which is the scheduling of trucks – as the carriers of products – at crossdocking facilities.

1.1 What Is Crossdocking?

Crossdocking – almost as old as warehousing and relatively new as one moves into the 21st century – has a maritime and railway background [2]. From the maritime perspective, ships would discharge cargo “over the dock” onto smaller ships, barges and wagons for consolidated economic delivery. The practice became popular and prevalent with the development of the railway system, and especially when railways began to carry products for industrial and private use. The recent developments in the use of information technology (IT) to automate the flow of information in the supply chain and artificial intelligence (AI) to reduce uncertainties (e.g., demand forecasting) have since made the use of crossdocking the norm rather than the exception. According to Brockmann [3],

Figure 1.1: Crossdocking reduces supply chain inventory and transportation costs by (a) consolidating goods according to customer demands before immediate transshipment and (b) changing LTL transportation to TL transportation (adapted from [6])

the ever increasing pressure to reduce inventories and a trend toward smaller and fewer warehouses will transform many warehouse operations into crossdocking operations in the 21st century. Some examples of the companies which reported successful implementation of crossdocking in their distribution networks are Wal-Mart, Home Depot, Costco, FedEx Freight [4], Goodyear GB Ltd. [2], and Kodak Co. [5], among others.

Crossdocking is a warehousing strategy that moves products through flow consolidation centers or crossdocks without putting them into storage. The operation involves receiving product from a supplier or manufacturer for several end destinations and occasionally consolidating it with other suppliers products for common final delivery destinations, then shipped at the earliest opportunity. Crossdocking helps in reducing the supply chain inventory and transportation costs, thereby improving the financial flows and profitability of the organization [7]. Low inventory and transportation costs can be achieved through immediate transshipment of goods after being consolidated according to customer demands (see Figure 1.1(a)). As illustrated by Figure 1.1(b), crossdocking

Figure 1.2: A common crossdock layout

can reduce transportation costs by changing less-than-truckload (LTL) transportation to full truckload (TL) transportation from suppliers to a crossdocking distribution center (CDC) and from the CDC to customers.

Crossdocking is normally considered a two-stage product flow where the first stage contains truckloads of mostly similar items from suppliers, called the inbound, and the second stage contains truckloads of mostly different items to customers, called the outbound (see Figure 1.1). Products are unloaded from incoming trailers¹ and loaded onto outgoing trailers with little or no storage in between. To build up crossdocking, a big yard is required to accommodate incoming and outgoing trailers. The crossdock itself is [generally] a rectangular dock [8] with doors placed around its perimeter (see Figure 1.2). Whenever an incoming truck arrives at the yard of a crossdock, it is assigned to a dock door (or waits in a queue on the yard until it is assigned [9]), where inbound loads are unloaded and scanned to determine their intended destinations. The loads are then sorted, moved across the dock and loaded onto outgoing trucks (or staged in load positions waiting for their outbound trailers to be assigned to dock doors [10]). Depending on its size or shape, freight (typically palletized) is moved from inbound trailers to out-

¹In this thesis, the terms ‘trailers’ and ‘trucks’ convey the same concept and are used interchangeably.

bound trailers by different material handling equipment including forklifts, pallet jacks, and a conveyor belt system. Once an inbound (outbound) trailer has been completely unloaded (loaded), it is removed from the dock, replaced by another trailer, if any, and the course of action repeats.

1.2 Research Motivation and Objective

Despite its simple definition, crossdocking is a complex operation and involves planning, scheduling, and coordinating the associated docking activities. According to the survey conducted for this thesis, crossdocking operations planning, scheduling, and coordination are now the major focus of research by both academics and practitioners who study and work on crossdocking. The survey says that, of the papers published on crossdocking, the percentage of those addressing crossdocking planning and scheduling has increased from 22.2% to 40.2% since 2000.

When it is viewed as a “standalone node” in a supply chain network, the crossdock operations are normally confined to planning the schedule of trucks and managing the workload of material handling systems in the crossdock along with the coordination of uncertainties. The former is generally referred to as the truck scheduling problem and will be addressed in this thesis. The problem decides on the sequence of incoming and outgoing trucks at the dock doors of the terminal so that some performance criteria are met. On the basis of the truck schedule, each truck arriving at the yard is assigned to a specific dock door where shipments are processed. This elementary problem repeatedly arises during the daily crossdock operations and has a vital influence on a rapid transshipment process [11]. In other words, one of the primary means of reducing labor cost and congestion in a crossdocking terminal is truck scheduling as it determines material flow patterns within the terminal and the resulting load on each material handling system. According to Gue [12], by placing trailers at the correct doors, terminal managers

avoid having workers travel too far and reduce worker and forklift congestion, and hence, the total crossdocking operation time.

The variations in the crossdocking operational characteristics result in different modeling and evaluation methodologies for the schedule of trucks (see Chapter 2). This study focuses on the truck scheduling problem according to a crossdocking model which has practical applications in the fast moving consumer goods (FMCG) industry [13, 14] and military logistics [15]. In this crossdocking scenario, trucks are assigned to dock doors on a daily basis to exchange some of their products before being dispatched to their final customers. The objective is to address the problem from a practical point of view by proposing a realistic crossdocking model. Unlike a number of studies which model the crossdocking operations in an abstract way in the context of machine scheduling or assume an unlimited number of doors and forklifts², this research emulates the real world crossdocking by defining pallets as the unit of shipments exchanged between the trailers and imposing constraints on the number of doors and forklifts as the crossdock critical resources. Furthermore, as normally desired by terminal managers, the allocation of doors and forklifts will be considered non-preemptive throughout crossdocking. The non-preemptive door assignment incurs complexities in the feasibility of the studied truck scheduling problem. This is because the problem might be entrapped in deadlock and no feasible solution is produced (see Chapter 4 for more explanation). Accordingly, it is required to address the solution feasibility of the problem before its optimality.

An exact (optimal) mixed integer programming model (MIP) is first developed to mathematically represent the crossdocking model and its associated truck scheduling problem. The objective function applied is to minimize the total crossdocking operation time starting when the first trailer is assigned to a door and ending when the last trailer releases its assigned door. The objective function is also referred to as schedule length

²The related papers are reviewed in Chapter 2.

or *makespan*. Since there is no real data available to evaluate the formulated problem, synthetic data is generated based on the characteristics of a wide range of real-world crossdocking practices. The data set includes problem instances of different levels of difficulty representing small crossdocks hosting a few dozen trailers to huge ones hosting over 2000 trailers.

As the mathematical approach is impractical for solving real size problem instances, sub-optimal (heuristic) algorithms are developed alternatively. The first step addresses the feasibility of the truck scheduling problem. To manage the complexity of the studied truck scheduling problem, the proposed heuristic algorithm is broken into two phases. The first phase is dedicated to establishing solution feasibility when a limited number of doors are available. This is because, from a practical point of view, finding a feasible truck schedule with minimum number of doors can be very critical if opening an extra door is too costly. It comprises of a heuristic search to construct a feasible sequence of trucks for the assignment to dock doors. The detailed scheduling procedure of trucks is then carried out in the second phase according to the feasible sequence list generated in the first phase. The second phase includes a rule-based heuristic to assign each sequenced truck to a proper door so that significant savings in the truck schedule length are achieved. Such a schedule is subject to a limited number of forklifts. The proposed algorithmic approach aims at finding feasible solutions for problem instances of various types and difficulty levels and at the same time being readily implemented in an industrial setting.

The next step addresses the optimality of the problem by improving the solutions generated by the two-phase heuristic algorithm. The practicality of the studied truck scheduling problem enforces an easy-to-implement algorithmic design. Meanwhile, the proposed approach (or its variants) should have successful implementations in the related work. In this respect, two meta-heuristics including tabu search (TS) and variable neighborhood search (VNS) are chosen and developed for this problem. The search space

of the problem is formed by changing the sequence of trailers for the door assignment. Being a neighborhood search method, each meta-heuristic is supplemented by a fractional evaluation of the neighborhood so that the runtime is more efficiently dedicated to solution optimization. The objective is to improve the solutions generated by the proposed two-phase heuristic algorithm. This requires tuning the control parameters of the two meta-heuristics so that the trade-off between solution quality and runtime can be properly balanced. The efficiency of the proposed TS and VNS algorithms in improving the heuristics solutions is further augmented by initializing the algorithms with a better initial solution. (The better initial solution is obtained by the re-starting version of the two-phase heuristic algorithm.)

1.3 Thesis Organization and Contributions

The remainder of this thesis is organized as follows. Chapter 2 surveys some common variations of the crossdocking models developed in the published literature to formulate the truck scheduling problem. This includes the operational strategies assumed by the models together with the evaluation methodologies proposed for the problem. The crossdocking model proposed in this thesis is described in Chapter 3. The operational strategies considered in the model are discussed in detail from which an MIP model is developed to formulate the truck scheduling problem. The chapter also elaborates on the procedure of generating synthetic data in the absence of real data.

Chapter 4 focuses on the feasibility of the truck scheduling problem. First, the conditions leading to solution infeasibility are discussed. The findings are then used as the motivation for developing the two-phase heuristic algorithm. The robustness of the algorithm in avoiding deadlock and establishing solution feasibility for the various characteristics of input data will be assessed in this chapter. Improving the heuristics solutions of the truck scheduling problem is carried out in Chapter 5. Two meta-heuristics are

developed for this purpose: tabu search (TS) and variable neighborhood search (VNS). Being a neighborhood search method, each meta-heuristic is supplemented by a fractional evaluation of the neighborhood to more efficiently invest the runtime on solution optimization. Extensive experiments are conducted to evaluate both the efficiency of the fractional neighborhood evaluation in reducing the runtime and the two TS and VNS algorithms in improving the solutions of the two-phase heuristic algorithm. The performance of TS and VNS meta-heuristics will be re-assessed when they are fed in with better initial solutions. Finally, Chapter 6 highlights the concluding remarks and presents possible research directions as subjects of future work.

Chapter 2

Literature Review

Addressing a crossdocking decision problem requires a precise modeling of the entire operation. The modeling, however, is application dependent. According to a particular application, a particular crossdocking configuration with particular operational strategies need to be defined. The variations in the crossdocking operational characteristics result in different modeling and evaluation methodologies for the schedule of trucks. This chapter presents an extensive review of the crossdocking scenarios developed in the literature for the truck scheduling problem. The review outlines the crossdocking operational strategies assumed by the models together with the evaluation and solution methodologies proposed to address their associated truck scheduling problem.

2.1 Operational Strategies

One of the operational issues characterizing the truck scheduling problem is the assignment of inbound and outbound flows to dock doors. According to Bozer and Carlo [8], there are two types of door assignments: static and dynamic. With a static door assignment, the assignment of outbound destinations to outbound doors is fixed over a planning horizon (such as 3 to 6 months) while the inbound doors are assigned every night. The outbound doors are reassigned in monthly periods by deriving the destination distribution of the loads in inbound trailers. Such a fixed assignment seems especially suited

for steady commodity flows with a reliable distribution among inbound and outbound destinations [11]. Truck scheduling in this setting reduces to assigning outbound destinations to outbound doors over a predefined planning horizon and nightly scheduling inbound trucks based on a given outbound door assignment. Static door assignments are employed by the majority of papers addressing truck scheduling in less than truckload (LTL) terminals in which the operations are typical crossdocking operations [16]. There is also one variation of static door assignments practiced in “production crossdocks” where the arrival sequence of inbound trucks at receiving doors is known a priori. In this scenario, the number of outbound destinations is normally greater than the number of shipping doors so that the objective is to determine the best schedule of outbound trucks which should be present at the shipping doors of the crossdock at any given time to minimize crossdocking operational costs (see [17, 18] for more information).

On the other hand, with a dynamic door assignment, both the assignment of inbound and outbound trucks to inbound and outbound doors are determined every night. In other words, instead of working with an “average” or “biased” inbound trailer to determine the outbound door assignments, the individual daily flows from each inbound trailer to each outbound trailer are considered. The terms static and dynamic door assignments have been denoted by Boysen and Fliedner [11] as *mid-term* and *short-term* modes of planning, respectively. As they introduce a comprehensive classification of deterministic truck scheduling for the first time in the literature, their notations are adopted for the rest of this thesis. The list of papers investigating the mid- and short-term planning strategies is given in the first row of Table 2.1.

In the short-term planning, one strategy (rather than having two exclusive sets of dock doors for inbound and outbound assignments, respectively) is to allow the assignment of any inbound or outbound trailer to any dock door. The concept was first proposed by Peck [19] as a “floating dock” and its impact on having a more efficient material

flow was later evaluated by Stephan and Boysen [20]. It aims at improving productivity by abandoning technical restrictions for a separation of inbound and outbound trucks. Setting up a floating dock, however, requires instantaneous, system-wide information access and transfer to allow terminal managers to know exactly which shipments will be arriving on which trailers and the location of every shipment on the dock at anytime [12]. In Boysen and Fliedner's classification, the floating dock concept is referred to as a mixed assignment as opposed to the exclusive assignment. Although recent advances in information technology (IT) have made the implementation of a floating dock no longer a burden, the exclusive assignment (especially by designating one side of the dock to inbound and the other to outbound operations) is still the preferable choice applied in practice to ease product flows and supervision [11].

Nonetheless, the mixed assignment is inevitable in the crossdocking applications where trucks just exchange some of their products with each other. This is because no designated inbound and outbound trucks can be distinguished. Crossdocking in this scenario is interpreted by the following operations. A number of incoming trucks are docked in the crossdock; exchange their products according to customer demands; and then leave the crossdock as outgoing trucks. In this study, the above crossdocking strategy is referred to as a *mixed* crossdocking in contrast to the classical *exclusive* crossdocking strategy reviewed so far. The second row of Table 2.1 presents the list of papers addressing the exclusive and mixed crossdocking configurations, respectively.

The schedule of trucks in terms of their arrival or departure times in or from the crossdock yard is based on one of the following operational modes: *synchronous*, *just-in-time* (JIT), and a hybrid of both. The synchronous scheduling allows crossdocking terminal managers to plan in advance the arriving times of trucks for the next working day. So each trailer can be available in the crossdock at any time the terminal manager desires. In other words, the availability of trucks is synchronized and consistent with the

schedule. The time each trailer is assigned to a dock door is basically determined by an algorithmic approach with the objective of the earlier the better. In the JIT scheduling, on the other hand, each truck has its own constraint on when to arrive in or depart from the crossdock. So each trailer is available in the crossdock subject to an external arrival and departure time. Truck scheduling should be then carried out dynamically by forcing the crossdocking operations of a given trailer to be fulfilled within its specified time window. From the planning perspective, the synchronous scheduling is more flexible than the JIT scheduling. For example, it gives more freedom to establish a more efficient product assignment from inbound to outbound trucks as all trucks are accessible in the crossdock at desirable times. This is especially important in mixed crossdocking scenarios where trucks need to exchange some of their products based on customer demands. Constructing an efficient product exchange pattern can lead to significant savings in the crossdocking operation time and enhances resource utilization.

There is also one simplified mode of JIT scheduling which sets due dates only for outgoing trailers – as the final carriers of products to customers – while assuming incoming trailers are all available at the crossdock yard before the scheduling starts [21–23]. Finally, in the hybrid scenario, trucks arrive at the crossdock yard at different times (following a probability distribution, for example) but leave the crossdock similar to the synchronous scheduling, i.e., according to the time they complete their crossdocking operations. The list of papers applying the synchronous, JIT, and hybrid modes of scheduling is given in the third row of Table 2.1.

According to [24], scheduling problems can be understood in general as the problems of allocating resources over time to perform a set of tasks being parts of some processes. It might happen that a task waits for a resource already allocated to another task. Resource unavailability is generally represented as delays in the scheduling length. Addressing a scheduling problem in a resource-constrained environment is much more complicated

Table 2.1: Overview on crossdocking operational strategies for the truck scheduling problem

Operational Issue	Strategies	Related Work
planning	mid-term	[6, 8, 16–19, 25–35]
	short-term	[8, 9, 21–23, 36–47]
crossdocking	exclusive	[6, 8, 9, 16–18, 21–23] [25–35, 38–41, 43, 47]
	mixed	[36, 37, 44–46]
scheduling	synchronous	[6, 8, 9, 17, 18, 25–29] [31–41, 43, 47]
	just-in-time (JIT)	[16, 21–23, 32, 33, 44–46]
	hybrid	[30]
trailer-to-door ratio (α)	$\alpha = 1$	[8, 16, 26–29, 31, 44, 47]
	$\alpha \geq 1$	[9, 17, 18, 21–23, 25, 30] [36–43, 45, 46]
	not applicable	[6, 32–35]
intermediate storage	unlimited	[9, 22, 23, 36, 38–44]
	limited	[17, 18, 28, 31, 37, 45, 46]
	zero	[21]
	not applicable	[6, 8, 16, 25–27, 29, 30] [32–35, 47]
material handling systems [normally forklifts]	unlimited	[17, 18, 21, 26, 27, 29, 31] [36, 44–47]
	limited	[6, 8, 9, 16, 22, 23, 25, 28] [30, 32–35, 37–43]
transshipment time	fixed	[6, 9, 17, 18, 21–23, 32–35] [38–41, 43]
	function of door location	[8, 25–31, 36, 37, 44–47]
	function of material handling system	[28]
	function of resource [forklift] availability	[16, 25, 28, 30, 37]
	function of dock congestion	[28]

than solving the problem with unlimited resources as there must be strategies for the efficient management of resources to minimize delays. For the truck scheduling problem, the delays are normally caused by the unavailability of dock doors, intermediate storage, material handling equipment, and workers.

Door unavailability occurs when the number of parking trailers in the crossdock yard exceeds the number of dock doors. (The ratio of the number of trailers to the number of doors is characterized as *trailer-to-door* ratio and denoted by α .) A number of papers, however, ignore the above constraint and assume that the number of trucks does not go beyond the number of doors, i.e., $\alpha = 1$. This assumption might be justified in large

LTL crossdocking terminals where the queue length of waiting trailers seldom exceeds one [25]. In this scenario, truck scheduling is simplified to the door assignment problem. Moreover, no intermediate storage is required as no unloaded product needs to wait for its destination trailer to be docked in the crossdock.

In situations where $\alpha \geq 1$, truck scheduling must determine the sequence of the trucks assigned to the same door of the crossdocking terminal. A number of papers [9, 22, 23, 38–43] investigate truck scheduling in a crossdock with one inbound door and one outbound door, implying that α is always greater than one. They assume that the material handling equipment is limited to a conveyor belt system which moves products from the inbound door to the outbound door in a fixed travel time. The product assignments from incoming trucks to outgoing trucks are also determined during the schedule of trucks. Addressing truck scheduling in a crossdock with the minimum number of dock doors (two doors for the exclusive assignment and one door for the mixed assignment), though unrealistic, may be useful for bound computation and complexity issues (see [9, 11] for the related analysis). The fourth row of Table 2.1 lists the papers addressing the two settings of $\alpha = 1$ and $\alpha \geq 1$, respectively.

Sometimes, unloaded products cannot be moved and loaded directly onto their destination trailers but must be temporarily staged onto the intermediate storage inside the terminal; for example, when the number of outbound trailers is more than the number of outbound doors. The intermediate storage is generally in the form of staging areas reserved in front of each door [48]. Staging freight complicates the problem of truck scheduling as the location and capacity of available staging areas should be included in its formulation to emulate the true timing of docking activities. On the other hand, there are some crossdocking models which, depending on the application they support, do not allow freight staging. For example, the work by Boysen [21] addresses the zero-inventory crossdocks of the food industry that forbids any intermediate storage inside the terminal for perishable items.

Comparable complexity arises if the number of material handling systems to move freight in the crossdock is considered limited. The constraint is normally imposed on the number of forklifts. This is because forklifts are the most versatile freight handling equipment capable of moving any product in the crossdock, no matter its size or weight [12]. The fifth and sixth rows of Table 2.1 classify the crossdocking operational strategies employed by the related papers in terms of intermediate storage and material handling systems, respectively.

One of the complexities caused by accumulated freight is to aggravate forklift congestion, and hence, the *transshipment time*, by creating bottlenecks in front of doors with high levels of flows [28]. Boysen and Fliehdner [11] define the transshipment time as the time lag between the arrival of shipments inside the terminal after having unloaded them from their respective inbound truck until their availability at an outbound door. The transshipment time not only depends on congestion inside the terminal, but also depends on the type and availability of material handling systems (generally forklifts) used for moving shipments to their destination doors. The effect of dock congestion on the freight transshipment time, when a limited number of material handling systems operate in the crossdock, is investigated in detail by Bartholdi and Gue [28]. To reduce the complexity of truck scheduling, however, most papers consider the transshipment time to be fixed or at most the function of the location of dock doors (see the last row of Table 2.1). Such an approximation may be reasonable if a crossdocking terminal of only a few doors, or with unlimited intermediate storage and material handling systems is assumed.

For management purposes, resources in the crossdock are usually considered non-preemptive. If a trailer is assigned to a door, it does not free the door until it finishes its unloading/loading operations and is ready to leave the crossdock. The same applies to forklifts and workers. They cannot be interrupted for a new service while they are busy serving another activity. Doors are not preempted mostly because terminal managers

cannot handle trailers twice by parking them in the yard [28]. Two studies [39, 40] (both inspired by [49]), however, allow preemption for dock doors to preclude staging freight onto a temporary storage. A real world justification for such a crossdocking configuration is not apparent since crossdocks are big enough to temporarily accommodate freight [48]. As an alternative strategy, however, Alpan et. al [17, 18] explicitly model the temporary storage together with allowing the preemption of loading operations at shipping doors to establish a balance between inventory holding costs and truck replacement costs.

On the other hand, the non-preemptive door assignment incurs complexities in the solution feasibility of truck scheduling in mixed crossdocking scenarios where $\alpha > 1$. This is because the problem might be entrapped in deadlock and no feasible solution is produced (more discussion is provided in Chapter 4). The deadlock situation is observed in two independent works by Lim et. al [45, 46] and Shakeri et. al [36]. For the former, the authors suggest unfulfilled shipments to resolve the deadlock which due to the high cost of keeping freight in the crossdock (even for one additional day) are not recommended in practice. For the latter, solution infeasibility is the outcome. The infeasibility occurs because the proposed heuristic is not robust in detecting and avoiding deadlock while scheduling trailers. This implies that there may be a feasible schedule for a given problem instance while the algorithm fails to find it.

Finally, it is good to cite a number of papers which model the crossdocking operation in the context of machine scheduling [6, 32–35]. The idea is that by viewing crossdocking as two stages of downloading and unpacking inbound products and collecting and packing outbound products, it is possible to model the crossdocking problem (including truck scheduling) as a two-stage flow shop problem where the first stage consists of machines whose operations are to download and unpack the inbound products and the second stage contains those whose operations are to collect and pack the products with the same destination onto an outbound trailer [50]. Although such a modeling methodology

can help to exploit the vast machine scheduling literature to develop efficient solution approaches for the problem, it is not possible to envision truck scheduling precisely.

Consider the process of downloading and unpacking inbound products in the crossdock. It is required that inbound trailers be assigned to dock doors and a worker, probably with the help of material handling equipment, unloads and moves the products to their destination outbound doors. It is not clear, however, that the notion of machines refers to dock doors, or the team of workers and material handling systems, or both. The above inconsistency prevents the inclusion of the two operational issues representing the trailer-to-door ratio (α) and crossdock intermediate storage in these machine scheduling models (see the reflection in the fourth and fifth rows of Table 2.1). This means that crossdocking scheduling bears some peculiarities which cannot be directly formulated from a machine scheduling perspective. Alternatively, Boysen and Fliedner [11] introduce a classification of deterministic truck scheduling based on a tuple notation similar to machine scheduling, allowing both the existing and new variations of the truck scheduling problem to be formulated systematically by adopting the proposed notation. The interested reader is encouraged to read their paper to acquire a better insight into the possible crossdocking settings defined for the truck scheduling problem.

2.2 Evaluation Requirements

Truck scheduling evaluation requires the definition of an objective function. One of the common performance criteria for evaluating the truck scheduling problem is to minimize the total distance traversed for moving unloaded products to their destination doors. Shortest door-to-door distance, however, cannot represent traveling time accurately as other factors including delays due to resource unavailability and dock congestion should also be considered in its calculation (see [12]). The two metrics (i.e., travel distance and time) are normally applied to the crossdocking models where $\alpha = 1$. From a practical

viewpoint, especially for the cases where $\alpha > 1$, the traveling time from a receiving door to a shipping door might take less than five minutes, but the waiting time for one unit of freight in an incoming trailer to be unloaded and for the outgoing trailer to be loaded and ready to leave might exceed an hour [30]. This requires that the performance measure be a function of the entire crossdocking operation.

Depending on the scheduling mode of crossdocking, two more precise measures are defined to evaluate the truck scheduling problem. One is to minimize the total schedule length starting from the moment when the first product of the first scheduled truck is unloaded to the moment when the last product of the last scheduled truck is loaded. The objective is also referred to as *makespan*, similar to traditional machine scheduling problems. Minimization of the makespan is normally applied to the synchronous scheduling. In this strategy, as mentioned, all trailers can be available at the crossdock yard before the scheduling starts. It should be noted that in this mode of scheduling, minimization of the total completion time of all trailers has also been defined as the objective function. The other measure is to minimize earliness and tardiness penalties which applies to the JIT scheduling. Any deviation from predefined arrival and departure times incurs earliness and tardiness penalties, whichever applicable.

A few other performance criteria have been defined for the same problem as well. Boloori Arabani et. al [23] formulate their truck scheduling problem as a multi-objective optimization problem (MOOP) by defining two objective functions: minimization of the total crossdocking operation time and minimization of the total tardiness of outgoing trailers. For hybrid scheduling where incoming trailers arrive at the terminal at different times, the common objective is to minimize the average time a shipment spends in a terminal [16, 30, 51]. Another example is the work by Lim et. al [45, 46] that assigns hard time windows to trucks. In this scenario, unfulfilled shipments would be the penalty if a trailer fails to be serviced in its predefined time window. The strategy, as mentioned,

is not recommended in practice. Alternatively, soft time windows to incur earliness and tardiness penalties are normally used. Finally, if non-deterministic truck scheduling is to be addressed (when there are uncertainties in the truck arrival times due to traffic, accident, etc.) one performance criterion is to balance the idle times at the dock doors of the terminal to absorb the stochastic variability in the arrival and service schedules (see [51, 52] for more information). The first row of Table 2.2(a) presents the list of the relevant papers for each performance measure.

The second element required for the evaluation of a problem is experimental data. In case no real data is available to evaluate the solution approaches to a given problem, synthetic data is alternatively used. To demonstrate that synthetic test data can reliably represent the real data, some justification is required. In the conducted review, however, this is neglected by the majority of papers which use synthetic data to address their respective truck scheduling problem (see the second row of Table 2.2(a)). As an example, Boysen [21] addresses truck scheduling in the zero-inventory crossdocks of the food industry where, depending on the input, no feasible solution may exist for the problem. He resolves this issue by developing a data generation procedure that guarantees a feasible solution exists for each generated instance. However, no justification is given to confirm that the same attributes are applicable to real world case studies.

2.3 Solution Techniques

Irrespective of which crossdocking configuration is set up, the formulated truck scheduling problems are normally of the same nature and belong to the class of “combinatorial optimization problems”. The term is usually used for the problems in which the decision variables are discrete, i.e. where the solution is a set, or a sequence, or integers or other discrete objects [51]. The solution approaches to this sort of problems range from optimal *mathematical* models for benchmarking purposes to sub-optimal *heuristic* methods for

Table 2.2: Overview on truck scheduling evaluation and solution methodologies

(a) Requirements to evaluate the truck scheduling problem		
Evaluation Requirement	Methodologies	Related Work
performance measure	[weighted] travel distance	[8, 25–27, 29, 31, 44, 47]
	[weighted] travel time	[28, 30, 45, 46]
	schedule length (or makespan)	[6, 9, 23, 34–43]
	earliness & tardiness penalty	[21–23, 32, 33]
	others	[16–18, 21, 28, 30, 45, 46]
test data	real	[8, 16, 25, 28, 29]
	synthetic [justified]	[8, 16, 25, 28, 30, 37]
	synthetic [unjustified]	[6, 9, 17, 18, 21–23, 26, 27]
		[31–36, 38–47]
(b) Techniques to solve the truck scheduling problem		
Solution Technique	Related Work	
mathematical model	[6, 8, 9, 21, 22, 25, 26, 31, 32, 35, 37, 38, 44, 45]	
exact algorithms (B&B, DP)	[9, 17, 18, 21, 27, 34]	
constructive heuristics	[6, 9, 31, 34–36, 38]	
meta-heuristics	[8, 21–23, 28, 29, 32, 33, 39–47]	
hybrid algorithms	[8, 9, 25, 26, 32, 33, 40]	
simulation	[16, 25, 30]	

scalability and fast response time. The common solution techniques together with the complete list of the related papers are summarized in Table 2.2(b) and described in the next few paragraphs.

Optimal *exact* algorithms normally consist of dynamic programming (DP), cutting plane and branch-and-bound (B&B) techniques. The exact approaches are computationally restrictive and generally apply to problems of modest size. Furthermore, their search process is generally guided by applying appropriate lower and upper bounds. Bounded dynamic programming (BDP) has been used by [9] and [21] to solve the truck scheduling problem in a single inbound/single outbound door crossdock and zero-inventory crossdocks of the food industry, respectively. Both crossdocking scenarios have some operational restrictions on the size of data justifying the application of DP. The dynamic programming (DP) approach has also been employed by Alphan et. al [17, 18] to address the transshipment problem of goods in production crossdocks. However, the DP search space is constrained by some bounding strategies to make it time efficient for solving

larger size problems at the expense of not guaranteeing optimality. Likewise, Chen and Lee [34] have applied B&B for a single machine scheduling problem. Tsui and Chang [27] have also employed B&B in their crossdocking scenario but the algorithm's runtime increases dramatically with problem size.

On the other hand, sub-optimal heuristic algorithms are not only computationally very efficient to solve real-size problems but also capable of finding near-optimal solutions. This is perhaps the main reason why the majority of work on the truck scheduling solution methodology comprises heuristic designs. Sub-optimal algorithms are normally referred to as constructive heuristics, meta-heuristics and a hybrid approach exploiting both. Constructive heuristics are solution techniques which, based on some rules, build the solution from scratch. For the truck scheduling problem, two independent studies by [9] and [38] propose a heuristic algorithm where the rules are defined to prioritize scheduling those inbound and outbound trucks that can have the most direct transfer of products without the intervention of the temporary storage. Another example is the work by [6, 34, 35, 50] which addresses the crossdocking problem from a machine scheduling viewpoint. The constructive heuristics proposed are based on Johnson's algorithm introduced by [53] to optimally solve the classical two-machine flow shop problem in polynomial time.

The solution produced by a constructive heuristic is generally employed as the starting point for an iterative generation process to search for local improvements. The process is called a "meta-heuristic" as it guides and modifies other heuristics to reach better solutions [54]. In the conducted review, meta-heuristic algorithms and their hybrid variants are the most commonly used solution approach to address the truck scheduling problem. Examples include simulated annealing (SA) [8, 21, 28, 40–42], genetic algorithms (GA) [22, 29, 39, 41, 43, 47], tabu search (TS) [33, 41, 43, 44, 46], variable neighborhood search (VNS) [40, 41], electromagnetism-like algorithm (EMA) [39, 41], particle swarm optimization (PSO) [22, 40, 43], differential evolution (DE) [22, 43], and ant colony optimization (ACO) [43], among others.

The EMA, PSO, and DE algorithms are normally applied to continuous optimization problems. EMA is a population-based meta-heuristic which is based on the physical behavior of electric charges in an electromagnetic field. Each particle is a solution whose charge represents its fitness. PSO is also a population-based algorithm in which each particle representing a trailer has two attributes: velocity and position. For each generation, according to the local and global best position of particles, the velocity of each particle is updated and the value is used to specify the position of the particle (trailer) in the new population. Finally, DE belongs to the evolutionary algorithms and is quite similar to GA. To establish diversification via crossover, mutant vectors are generated for each individual. The vectors are mutated by adding them weighted random vector differentials.

Boloori Arabani et. al [43] are the sole authors that implement ACO for their studied truck scheduling problem. It is a population-based meta-heuristic inspired by the social behavior of ant colonies; a search path is detected by “laying the pheromone trail” as the real ants do. Each ant aims at constructing a sequence of trucks to form the population at each iteration. The best sequences at each iteration are determined and compared to find the best solution after the stopping criterion is met.

To evaluate their performance, Vahdani and Zandieh [41] develop and compare five meta-heuristics (GA, TS, SA, EMA, and VNS) for the truck scheduling problem in a single inbound/single outbound door crossdock. The results show that VNS outperforms the others in terms of solution quality. Boloori Arabani et. al [22] compare GA, PSO, and DE for the same truck scheduling problem and conclude that GA acquires the best performance (PSO is the best in terms of the runtime) while DE shows more robustness to problem size. Another study, also conducted by Boloori Arabani et. al [43], compares five meta-heuristics (GA, TS, PSO, ACO, and DE) for the same problem. Experimental results show that GA and DE slightly outperform the other three algorithms.

While running a meta-heuristic algorithm, there are some control parameters influencing the algorithm performance. Each control parameter, also referred to as a factor, takes some values and ranges from one to many different levels. To achieve a more robust algorithm, which does not produce functional variance due to the influence of the external environment, a parameter tuning technique can be applied for the adjustment and calibration of the values. One of the popular methods to study the effects of several factors and their interactions on a system's response is a full-factorial design [55]. The other applicable method for choosing the optimal or near-optimal parameter levels is Taguchi parameter design by which one can moderate the levels of parameters and also determine the pertinent levels of each factor having the most impact on the algorithm performance [22]. The Taguchi method has been used by [22, 39, 40, 42, 43] to set up a robust meta-heuristic for their respective truck scheduling problem.

The combination of meta-heuristics with each other or some other solution techniques has also been reported to be very effective (they are referred to as hybrid algorithms in this study). To solve their JIT two-phase parallel machine scheduling problem, Li et. al [32] develop a genetic algorithm in which a local search (LS) approach is applied to the best of each generation. The first LS is squeaky wheel optimization (SWO) in which the neighbors of a given solution are generated by assigning higher priorities to those jobs whose tardiness or earliness penalties are high. The second LS is a simplified MIP model whose constraints on machine assignment patterns have been removed, i.e., it is known which machine has been assigned a job. It is shown that GA+SWO is faster while GA+MIP is more efficient in terms of solution quality. For the same scheduling problem, Alvarez-Perez et. al [33] propose a combination of two meta-heuristics, Reactive GRASP (greedy randomized adaptive search procedure) and TS. For each iteration of Reactive GRASP, one feasible solution is produced. After a given number of iterations, an elite set of five diverse solutions is given as the initial points for TS for improvement.

Soltani and Sadjadi [40] develop a hybrid algorithm combining VNS with PSO. In the VNS algorithm, after generating the local optimum relative to a given neighborhood, PSO is applied to further improve the solution. The experimental results show that the proposed hybrid technique (VNS+PSO) outperforms the SA algorithm developed for the same problem in the aforementioned paper. Lastly, it is good to refer to [16, 25, 30] as a few studies which use detailed simulation models to evaluate the performance of their proposed strategies to accelerate freight transfer from inbound to outbound trucks under various crossdocking operational characteristics.

2.4 Summary

This chapter reviewed the related work on the truck scheduling problem from three perspectives. First, in terms of the operational strategies assumed in developing a crossdocking model required for the problem formulation, second, in terms of the requirements to evaluate the problem, and third, in terms of the solution techniques developed to solve the problem. The review affirms the diversity of the work and highlights the potential research gaps for future studies. The gaps mainly relate to addressing truck scheduling subject to constrained resources and operational uncertainties while evaluating different objective criteria. The resources are normally associated with dock doors, intermediate storage, and material handling equipment.

This thesis addresses the truck scheduling problem based on a particular crossdocking application with unique operational settings, making it different from the reviewed truck scheduling problems. The synchronous scheduling of trucks is studied in a short-term planning horizon of one day in a mixed crossdocking configuration. According to the introduction presented in Chapter 1, this scenario is generally practiced in the FMCG industry and military logistics where crossdocking has a key role in reducing the

turnaround time [13–15]. Much effort is dedicated to tackling the problem from a practical point of view¹. As presented in the next chapter, a resource-constrained crossdocking model is developed to formulate the truck scheduling problem. The constraints are imposed on the number of dock doors, forklifts and workers². The freight transshipment time is assumed to be a function of both the location of dock doors and availability of forklifts. Concerning the lack of real data required for the evaluation of the problem, the procedure of generating synthetic test data is introduced and its compatibility to real world applications will be justified.

¹This was mainly neglected by some of the reviewed papers.

²No constraint is imposed on the capacity of the crossdock intermediate storage.

Chapter 3

Formulation & Evaluation Methodology

This chapter introduces a resource-constrained crossdocking model to define and formulate the truck scheduling problem. A mixed integer programming (MIP) model is proposed to formulate the truck scheduling problem. The modeling methodology tends to be realistic by imposing constraints on the number of dock doors, forklifts and workers. Moreover, it considers pallets as the unit of shipments exchanged between trailers (the common case in real crossdocks). Concerning the lack of real test data, synthetic data are generated with a broad range of characteristics to represent the diversity exists in real world settings. These characteristics include the problem size in terms of the number of doors, the ratio of the number of trailers to the number of doors, freight distribution patterns between trailers, and correlation status of interdependent trailers. The aforementioned steps build a realistic reliable framework to propose and evaluate a number of algorithms for this combinatorial optimization problem. A realistic crossdocking model allows any accomplishments in terms of operational efficiency to be readily used in practice. Besides, the synthetic generated data provide a comparable assessment of the performance of the proposed solution techniques in the absence of real data.

Figure 3.1: Structure and operation of the proposed crossdocking model

3.1 Crossdocking Model Development

Figure 3.1 presents the proposed crossdocking model where a number of trailers are crossdocked in a multiple door crossdock. Crossdocking is done in a short-term planning horizon of one day in the mixed mode of service. That is, incoming trailers are assigned to dock doors on a daily basis to exchange some of their products before being dispatched to their customers. Products are staged onto staging areas until their destination trailers are docked in the crossdock. Without loss of generality, it is assumed that the products whose destination trailers have not yet been assigned to an available door are temporarily staged in front of the doors where they have been unloaded. The formation of staging areas is illustrated in Figure 3.1, based on Bartholdi and Gue's observation from real crossdocks [48]. As the figure shows, some space is reserved in front of each door to stage freight for that door.

In this study, the synchronous scheduling is employed to enable trucks to be available at the crossdocking terminal at any time the terminal manager desires. This strategy, as mentioned in Chapter 2, gives more freedom than the JIT scheduling to construct an efficient product exchange pattern between trailers¹. Truck scheduling starts when the first trailer is assigned to a door and ends when the last trailer leaves its assigned door. As soon as a trailer is assigned to a free door, a worker unloads the exchanged products onto the staging area provided in front of the door. The products are then moved by forklifts and discharged onto the respective staging areas of the doors to which their destination trailers are assigned. Similar to unloading, a worker is dedicated for loading the exchanged products onto a destination trailer. Simply put, what is done inside the crossdock is to unload, move, and load the exchanged products between the trailers.

3.1.1 Truck Scheduling Formulation

To formulate the truck scheduling problem addressed in this research, it is required to envision how products are exchanged among trailers in the crossdock. This is explained by applying the following operational characteristics to the crossdocking model:

- Products are packed in pallets where all the products in an exchanged pallet are destined to one destination trailer.
- All the pallets exchanged between trailers in the crossdock are similar in size and shape, leading to the following two statements:
 - Pallet unloading/loading time is constant and identical for all pallets.
 - Pallet moving time is only a function of the distance a pallet is moved. This also means that neither freight congestion nor interference among forklifts occurs to vary the forklift velocity.

¹Product assignments between trailers to establish a freight exchange pattern are done in advance by employing an algorithmic solution approach which is called product mixing/matching in Li et al. [56].

- The unloading operations of a trailer can be initiated as soon as it is assigned to a dock door.
- The loading operations of a trailer can start only after its unloading operations are completed.
- A trailer is ready to release its assigned door and leaves the crossdock when its loading operations are completed.
- Trailer changeover time is known in advance and is the same for all trailers.
- Pallets are unloaded or loaded by a worker one at a time. Also, only one worker can unload (load) pallets from (onto) a given trailer at any time.
- Pallets are unloaded in sequence according to the order they have been placed at the supplier side and loaded according to the order they arrive at the destination door. (The former is input to the problem while the latter is determined during the crossdocking operation.)
- A forklift moves only one pallet per trip. This means that for two pallets consecutively unloaded from a trailer and are to be loaded onto the same trailer, two forklift trips are needed to move them.
- A forklift moves a pallet (from its origin door) to its destination door only if the recipient trailer has been assigned to that door. This also implies that the location information of pallets is always available to forklifts for finding and moving them.
- The distance between any two doors is equal to the distance between their respective staging areas.
- Pallets are moved in rectilinear paths due to the aisles created by staged freight (see [25]).

In practice, the functionality of the aforementioned crossdocking operations is subject to the availability of the resources in the crossdock. The resources are considered dock doors, intermediate storage, forklifts and workers whose availability is confined by the following constraints:

- The number of trucks in the crossdock yard may exceed the number of dock doors, i.e., $\alpha \geq 1$, which may lead to queues of trucks waiting to be assigned to doors.
- The number of workers and forklifts to handle (i.e., unload, move, or load) pallets in the crossdock is limited.
- The intermediate storage in the crossdock is limited and represented by the staging areas distributed along the dock doors (see Figure 3.1).

It is deduced from the second constraint that the time duration to unload, move, or load pallets is subject to the workers and forklifts availability. If no worker is available, a delay should be incurred into the pallet unloading/loading time till one worker is available. Normally, these workers are the truck drivers themselves [57], allowing the delays to be negligible. In terms of forklifts, however, the amount of delays depends on the total number of forklifts operating in the crossdock and how they are scheduled for the service. If one assumes one forklift is dedicated to each door for moving its unloaded products, respectively, then the forklift scheduling problem is reduced to the problem of scheduling the movements of each single forklift based on the scheduling conditions of only those trailers which have been assigned to a door and been waiting to receive the unloaded pallets. Provided that the staging areas reserved for each door do not run over their assigned limit, a procedure can be embedded into the truck scheduling algorithm (see Chapter 4) to construct the moving order of the pallets unloaded in front of each door to their respective destination doors. Having known the orders, the delays can

be calculated successively given that neither freight congestion nor interference among forklifts occurs to vary the forklift velocity.

For example, for two pallets (unloaded at the same door) where the first one precedes the other in the moving order list, the latter should wait until the corresponding forklift moves the former to its destination door and then returns to the origin door. When the number of forklifts is fewer than the number of doors, an upper bound can be calculated on the delays by designating, for example, one forklift to two dock doors or more. In that case, it is required to integrate a more complex forklift scheduling procedure into the current problem. This is, however, beyond the scope of this research so, it is assumed in this thesis that the number of forklifts is equal to the number of doors.

Imposing the third constraint on the intermediate storage may turn the crossdock capacity into a bottleneck resource during the schedule of trucks. In the case of capacity overrun, it is required that freight unloading operations stop until enough space is provided after loading a number of staged pallets. On the other hand, if it is observed that the crossdock entire capacity does not overflow yet the designated staging areas overrun their assigned limit, a more sophisticated pallet moving procedure needs to be deployed. The procedure should not only efficiently schedule the movement of pallets by the forklifts but at the same time should avoid space overrun by moving the excess load to intermediate locations with available spaces. For both situations, the schedule length would definitely increase. The constraint, however, is not applied in this research, but rather an analysis is conducted in Section 3.4 to measure crossdock capacity utilization when a feasible schedule of trucks has already been constructed for crossdocking.

The allocation of the crossdock resources including dock doors, forklifts, and workers is considered non-preemptive. In terms of doors, this means that a trailer does not release its assigned door unless it completes its operations (i.e., unloading/loading). Regarding forklifts and workers, they cannot be interrupted for another service until they finish

moving and unloading/loading their assigned pallet, respectively. The non-preemptive door assignment makes it necessary to address the feasibility of truck scheduling before its optimality as the problem might be entrapped in deadlock and no feasible solution is produced (see Chapter 4).

The performance criterion could be defined either in terms of minimization of crossdocking total operation time or minimization of total completion time of all trailers. Regarding the related work, however, the former objective is more common for synchronous truck scheduling. Accordingly, the objective of the truck scheduling problem studied in this thesis is to find an optimal schedule of trucks for the assignment to dock doors so that the total crossdocking operation time starting when the first trailer is assigned to a door till the last trailer leaves its assigned door is minimized. Such a schedule is subject to a limited number of forklifts equal to the total number of doors. The objective function, i.e., schedule length, is also referred to as *makespan*.

3.1.2 Problem Complexity

According to Brown [58], the assignment of trailers to dock doors is related to the Quadratic Assignment Problem (QAP). In the QAP, there are a set of n facilities and a set of n locations. For each pair of locations, a distance is specified and for each pair of facilities a weight or flow is specified (e.g., the amount of supplies transported between the two facilities). The problem is to assign all facilities to different locations with the goal of minimizing the sum of the distances multiplied by the corresponding flows. The QAP is NP-hard to solve to optimality and to approximate within a constant approximation ratio (see [59–61]).

Likewise, the complexity of the truck scheduling problem studied in this thesis is NP-hard as one can transform the QAP instances to the instances of this problem. If the truck scheduling problem is simplified by assuming that the number of trailers is

equal to the number of doors, i.e., $\alpha = 1$, the resultant problem is the QAP as trailers are facilities and doors are locations. The flow defined for each pair of facilities is the amount of freight exchanged, if any, between the two corresponding trailers and the objective is to minimize the total flow exchange time.

3.1.3 Complications in Standard Scheduling Notation

It was stated in Chapter 2 that crossdocking scheduling bears some peculiarities which cannot be directly formulated from a machine scheduling perspective. Consequently, as noticed in the related work, the proposed modeling and algorithmic approaches may not be readily applicable practically. This section gives an insight into the subtle incompatibilities between the truck scheduling problem studied in this thesis and standard machine scheduling problems. The major issue is to specify the notion of jobs and machines in the problem. If jobs are tasks whose fulfillment requires machines as resources, then three types of jobs and two types of machines should be defined for this problem:

- j_{Ui} : represents the unloading operation of pallet i
- j_{Mi} : represents the moving operation of pallet i
- j_{Li} : represents the loading operation of pallet i
- m_{Ui} : represents resource i required for the unloading operation. (According to the problem definition, the resource is door i and its designated worker.)
- m_{Mi} : represents resource i required for the moving operation. (According to the problem definition, the resource is forklift i designated to door i .)
- m_{Li} : represents resource i required for the loading operation. (According to the problem definition, the resource is door i and its designated worker.)

Furthermore, the following precedence relations apply to the three types of jobs: $j_{Ui} \rightarrow j_{Mi} \rightarrow j_{Li}$. Pallet i must be unloaded before its movement starts, and must be moved before it is loaded. There are also some constraints on the assignment of jobs to machines. Those pallets unloaded (loaded) from (onto) a given trailer must be unloaded (loaded) by the same machine. All the pallets unloaded by machine m_{Ui} must be moved by machine m_{Mi} . Finally, it should be noted that each machine can process only one job at a time while each job is processed by only one machine.

The outcome of such a modeling results in a very detailed machine scheduling problem in which the number of jobs is three times the number of the total pallets exchanged between trailers. Consider a small instance of scheduling eight trailers. The average number of pallets the trailers exchange is about 60 leading to 180 jobs². For a large instance of scheduling 500 trailers, however, the average number of exchanged pallets increases to 6000 which is equal to 18000 jobs. This severely complicates the scheduling procedure which, due to the non-preemptive allocation of machines, a few feasible solutions may be available with respect to all the possible job permutations.

Alternatively, if the modeling methodology is simplified by mapping each trailer to one job, the resultant scheduling problem does not belong to the class of standard machine scheduling problems any more. The main difference is the relations between jobs, which is no longer of pure precedence. In this context, two jobs (trailers) are related if at least one [pallet] exchange exists from one of them to the other. Accordingly, it is possible that the two jobs are mutually related if both have exchanges with one another. The other difference is that job relations are of the assignment type and not completion. For two jobs where one is dependent on the other, the completion of the dependent job only requires the other job to be assigned to a free machine, i.e., its completion is not a prerequisite. This means that no precedence chain exists between jobs (in

²The data generation procedure will be introduced in Section 3.3.

Figure 3.2: (a) Product exchange matrix of six trailers (the row to each trailer indicates the number of products unloaded from that trailer while the column shows the number of products loaded onto the trailer), (b) the corresponding interdependency matrix and (c) directed graph (DG)

contrast to standard machine scheduling problems). Due to such discrepancies, the truck scheduling problem is not addressed in the context of machine scheduling in this thesis and the crossdocking terminology is applied instead. The following example introduces the terminology used for the problem along with the incompatibilities exist relative to standard machine scheduling:

Example. Consider six trailers which require to exchange some of their products before they leave the crossdock. The product exchange matrix is given by Figure 3.2(a). (Note that the matrix is the input to the problem.) In this figure, the row to each trailer indicates the number of products unloaded from that trailer while the column shows the number of products loaded onto the trailer. For example, according to the value of element e_{13} of the matrix, seven products are unloaded from trailer 1 and loaded onto trailer 3. As already mentioned, a trailer is related to another trailer if at least one product exchange exists from one to the other. Generally, two groups of trailers are related to a given trailer. The first group contains those trailers receiving products from the trailer, called the *successors* of the trailer, while the second contains those giving products, called the *predecessors* of that trailer. So the successors of trailer 3 in Figure 3.2(a) are trailers 2 and 4 whereas the predecessors are trailers 1 and 5. Trailer 1 has no predecessors (it is a pure donor trailer) and trailer 2 has no successors (it is a pure recipient trailer). The successors and predecessors of a trailer form the dependency relationships, or more concisely, the *interdependencies*, of that trailer.

To merely investigate the interdependencies of trailers, regardless of how many products are transferred for any exchange, an interdependency matrix or directed graph (DG) is more illustrative. As shown in Figure 3.2(b), for each pair of related trailers, a ‘1’ is filled in the corresponding entry of the interdependency matrix; otherwise, ‘0’ is assumed.

Likewise, the DG is formed by mapping trailers to nodes and their interdependencies to directed edges (see Figure 3.2(c)). According to the figure, unlike standard scheduling problems, the represented graph does not belong to the class of directed acyclic graphs (DAG). It is easy to notice that trailers 4, 5, and 6 are mutually dependent upon each other, i.e., each is both the successor and predecessor of the other. Furthermore, the precedence chains found in the graph do not convey the same meaning as that interpreted in normal scheduling. For example, although a precedence chain exists from trailer 1 to trailer 6, trailer 6 can complete its operations only if its predecessors, i.e., trailers 4 and 5, are assigned to an available door. Put differently, for a trailer which needs some products from another trailer, it is only required that the latter be assigned to a door so that the pallets are unloaded and moved for loading onto the former (even if the latter trailer itself is waiting for its required products to be unloaded from some other trailers). This means that the completion of a given trailer depends on the assignment of its predecessors and not their completion, making the concept of precedence chains, common in standard machine scheduling problems, irrelevant to this problem.

3.2 Integer Program

The truck scheduling problem studied in this thesis can be represented analytically in the context of an integer program. In the mixed integer programming (MIP) formulation, $|J|$ trailers are waiting for crossdocking in a crossdock of $|M|$ doors. The location of the crossdock doors is represented by a distance matrix. There are in total $|P|$ pallets exchanged between the trailers. For any two trailers which exchange some of their products with each other, a pallet unloaded from one of them is loaded onto the other. Accordingly, each trailer j has two sets of pallets. The first set contains those pallet IDs unloaded from the trailer³ (denoted by U_j) and the second contains those loaded onto the trailer (denoted by L_j). In other words, U_j and L_j contain pallet-truck mapping and pallet destination information. Note that the union of the pallets unloaded from or loaded onto the trailers is equal to the total pallets exchanged between the trailers.

The following notation is used to describe the MIP model:

³For a given n entities, entity IDs simply range from 1 to n .

Indices

m	door, $m \in M = \{1, \dots, M \}$
j	trailer, $j \in J = \{1, \dots, J \}$
p	pallet, $p \in P = \{1, \dots, P \}$

Input parameters

U_j	set of pallets unloaded from trailer j
L_j	set of pallets loaded onto trailer j
β_p	unloading position of pallet p
t^U	time taken to unload a pallet
$t_{mm'}^M$	time taken to move a pallet from door m to door m'
t^L	time taken to load a pallet
T^C	trailer changeover time
Q	a big number not less than the worst schedule length

Decision variables

$c_{\max}$	schedule length (or makespan)
a_j	assignment time of trailer j
c_j	completion time of trailer j
λ_p	time when moving of pallet p starts
μ_p	time when moving of pallet p is completed
σ_p	time when loading of pallet p is completed
$\delta_{pp'}$	1, if, for pallets p and p' staged onto the same staging area, p is moved before p' , else 0
$\gamma_{pp'}$	1, if, for pallets p and p' loaded onto the same trailer, p is loaded before p' , else 0
x_{jm}	1, if trailer j is assigned to door m , else 0
$y_{jj'}$	1, if, for trailers j and j' assigned to the same door, j precedes j' , else 0
q_p	1, if pallet p is to be moved [by a forklift] before loaded onto its destination trailer, else 0
$v_{jmj'm'}$	1, if trailer j is assigned to door m and trailer j' is assigned to door m' , else 0
$w_{pp'}$	1, if both pallets p and p' are to be moved [by a forklift] before loaded onto their destination trailers, else 0

With the above denotations and decision variables, the MIP formulation is as follows:

$$\begin{aligned}
 &\text{Minimize} && c_{\max} \\
 &\text{s.t.} && \\
 &&& \sum_{m \in M} x_{jm} = 1, \quad j \in J
 \end{aligned} \tag{3.1}$$

$$a_{j'} \geq c_j + T^C - Q \cdot (1 - y_{jj'}), \quad j, j' \in J, \quad j \neq j' \quad (3.2)$$

$$\lambda_p \geq a_j + t^U \cdot \beta_p, \quad p \in U_j, \quad j \in J \quad (3.3)$$

$$\lambda_p \geq a_j, \quad p \in L_j, \quad j \in J \quad (3.4)$$

$$\begin{aligned} \lambda_{p'} &\geq \lambda_p + 2 \cdot t_{mm'}^M \cdot v_{jmj'm'} - Q \cdot (1 - \delta_{pp'}), \\ p &\in U_j \cap L_{j'}, \quad p' \in P, \quad p \neq p', \quad j, j' \in J, \quad m, m' \in M \end{aligned} \quad (3.5)$$

$$\mu_p \geq \lambda_p + t_{mm'}^M \cdot v_{jmj'm'}, \quad p \in U_j \cap L_{j'}, \quad j, j' \in J, \quad m, m' \in M \quad (3.6)$$

$$\sigma_p \geq \mu_p + t^L, \quad p \in P \quad (3.7)$$

$$\sigma_p \geq a_j + \max_{p' \in U_j} t^U \cdot \beta_{p'} + t^L, \quad p \in L_j, \quad j \in J \quad (3.8)$$

$$\sigma_{p'} \geq \sigma_p + t^L - Q \cdot (1 - \gamma_{pp'}), \quad p, p' \in L_j, \quad p \neq p', \quad j \in J \quad (3.9)$$

$$c_j \geq \sigma_p, \quad p \in L_j, \quad j \in J \quad (3.10)$$

$$c_{\max} \geq c_j, \quad j \in J \quad (3.11)$$

Constraint (3.1) ensures that each trailer is assigned exactly to one door. Constraint (3.2) defines the timing dependencies between two different trailers that have been assigned to the same door. It states that the assignment time of a trailer to a door is after the loading completion time of the other (which precedes the former trailer in being assigned to the same door) plus the time it takes to leave the door, that is, the trailer changeover time. Having been assigned to a door, a trailer can start and complete unloading pallets consecutively according to the position they have been placed at the supplier side. Constraints (3.3)–(3.5) specify the conditions required to be met before an unloaded pallet can be moved. Constraints (3.3) and (3.4) state that the movement of a pallet cannot be started unless it is unloaded and its destination trailer is assigned to an available door. Constraint (3.5) schedules the movement of the unloaded pallets staged in front of a door according to the order which itself is decided during the scheduling. Note that Constraint (3.5) implicitly emulates the situation where exactly one forklift is designated to one door for moving its staged pallets. For two pallets where the first one precedes the other in the moving order list, the latter should wait until the corresponding forklift moves the former to its destination door and then returns to the origin door.

Constraint (3.6) determines when the moving operation of a given pallet is completed. Constraint (3.7) ensures that a pallet is loaded only after it is moved to its destination door. Constraint (3.8) indicates that a trailer can start its loading operations only after it completes its unloading operations. Constraint (3.9) states that the loading operations of the pallets onto a given trailer are completed consecutively according to the order they arrive at the destination door. Constraint (3.10) specifies that after loading its last pallet, a trailer is done and is ready to leave its assigned door. Constraint (3.11) defines the makespan which is equal to the maximum completion time of the trailers.

Finally, binary variables $y_{jj'}$, $\delta_{pp'}$, q_p , $w_{pp'}$, $\gamma_{pp'}$, and $v_{jmj'm'}$ are used as the control variables in the mathematical formulation and defined by the following constraints:

$$y_{jj'} + y_{j'j} = \sum_{m \in M} v_{jmj'm}, \quad j, j' \in J, \quad j \neq j' \quad (3.12)$$

$$\delta_{pp'} + \delta_{p'p} \leq \sum_{m \in M} v_{jmj'm} + Q \cdot (1 - w_{pp'}), \quad p \in U_j, \quad p' \in U_{j'}, \quad p \neq p', \quad j, j' \in J \quad (3.13)$$

$$\delta_{pp'} + \delta_{p'p} \geq \sum_{m \in M} v_{jmj'm} - Q \cdot (1 - w_{pp'}), \quad p \in U_j, \quad p' \in U_{j'}, \quad p \neq p', \quad j, j' \in J \quad (3.14)$$

$$\delta_{pp'} \leq q_p, \quad p, p' \in P, \quad p \neq p' \quad (3.15)$$

$$\delta_{pp'} \leq q_{p'}, \quad p, p' \in P, \quad p \neq p' \quad (3.16)$$

$$q_p \leq t_{mm'}^M + Q \cdot (1 - v_{jmj'm'}), \quad p \in U_j \cap L_{j'}, \quad j, j' \in J, \quad m, m' \in M \quad (3.17)$$

$$q_p \geq t_{mm'}^M / Q - Q \cdot (1 - v_{jmj'm'}), \quad p \in U_j \cap L_{j'}, \quad j, j' \in J, \quad m, m' \in M \quad (3.18)$$

$$w_{pp'} \leq q_p, \quad p, p' \in P \quad (3.19)$$

$$w_{pp'} \leq q_{p'}, \quad p, p' \in P \quad (3.20)$$

$$w_{pp'} \geq q_p + q_{p'} - 1, \quad p, p' \in P \quad (3.21)$$

$$\gamma_{pp'} \geq (\mu_{p'} - \mu_p) / Q, \quad p, p' \in L_j, \quad p \neq p', \quad j \in J \quad (3.22)$$

$$\gamma_{pp'} \leq 1 + (\mu_{p'} - \mu_p) / Q, \quad p, p' \in L_j, \quad p \neq p', \quad j \in J \quad (3.23)$$

$$v_{jmj'm'} \leq x_{jm}, \quad j, j' \in J, \quad m, m' \in M \quad (3.24)$$

$$v_{jmj'm'} \leq x_{j'm'}, \quad j, j' \in J, \quad m, m' \in M \quad (3.25)$$

$$v_{jmj'm'} \geq x_{jm} + x_{j'm'} - 1, \quad j, j' \in J, \quad m, m' \in M \quad (3.26)$$

Constraint (3.12) gives a freedom to the MIP solver to decide how to sequence the trailers assigned to the same door so that the optimal makespan is produced. A similar flexibility is provided by Constraints (3.13) and (3.14) to schedule the movement of pallets which have been staged in front of the same door. The scheduling, however, does not apply to those pallets whose destination trailers are also assigned to the door where they have been unloaded. Since in that case no movement [by using a forklift] is required and the pallets are already available for loading. This is realized by Constraints (3.15) and (3.16). Constraints (3.17) and (3.18) determine the value of q_p according to whether pallet p needs to be moved or not. (Note that the minimum distance between two doors is one, i.e., $t_{mm'}^M \geq 1$, for $m \neq m'$). Constraints (3.19)–(3.21) state that $w_{pp'} = q_p \cdot q_{p'}$. The value of the binary variable $\gamma_{pp'}$ specifying the sequence of pallets loaded onto a given trailer is identified by Constraints (3.22) and (3.23), according to the time they are available at the destination door. Finally, Constraints (3.24)–(3.26) express that $v_{jmj'm'} = x_{jm} \cdot x_{j'm'}$.

In this formulation, input parameter Q is a very big number not less than the worst schedule length. Its application in the model is two-fold. First, it represents the maximal moving time for Constraint (3.18) and the maximal completion time difference between two moving operations for Constraints (3.22) and (3.23). Second, it is Big-M for Constraints (3.2), (3.5), (3.9), (3.13), (3.14), (3.17), and (3.18). Choosing a proper value for Q is very critical when it is used as Big-M. In this case, if it is smaller than the upper bound of the associated decision variables, it may cut off valid solutions and if it is too large, the model may become numerically difficult to solve. In the implementation of the MIP model, however, parameter Q is applied to only Constraints (3.18), (3.22), and (3.23). Those constraints with Big-M are represented by *indicator constraints* to avoid unwanted side-effects and numerical instabilities incurred by Big-M formulations into a mathematical program [62]. Being introduced by the CPLEX solver version 10.0, an indicator constraint is a new way of controlling whether or not a constraint takes effect based

on the value of a binary variable. Using indicator constraints, such relationships between a constraint and a variable can be directly expressed in the constraint declaration.

It is thus required to ensure that the value of Q is bigger than the maximal moving time and the maximal completion time difference between two moving operations. The value for Q can be calculated by adding all pallets' unloading and loading times together with all the possibilities for their moving times plus total trailer changeover times. The corresponding formula is described as follows:

$$Q = (t^U + t^L + \sum_{m \in M} \sum_{m' \in M} t_{mm'}^M) \cdot |P| + T^C \cdot |J|$$

The timing relationships between trailers in the process of unloading, moving, and loading pallets, according to the constraints defined in the MIP model, is illustrated by the following example:

Example. Consider a trailer 5 which is to exchange some pallets with a number of trailers in the crossdock. Given that $U_5 = \{9, 10, 11, 12\}$ and $L_5 = \{1, 2, 5, 6, 14, 15\}$, let us assume that trailer 5 is assigned to door 3. At the time of assigning trailer 5, pallets 1 and 2 have already been staged in front of door 1 and pallets 5 and 6 are available in front of door 2. Also assume that a trailer 6 with $U_6 = \{13, 14, 15\}$ will be assigned to door 1 after the assignment of trailer 5. Figure 3.3 depicts the timing details of trailer 5 from the time it is assigned to door 3 at $t = 4$ till it releases the door at $t = 63$. As soon as trailer 5 is assigned, the pallets in L_5 , if they already have been unloaded onto the crossdock, can be moved and placed in front of door 3. Pallets 1 and 2 are moved by a single forklift from door 1 to door 3, and pallets 5 and 6 are moved by another forklift from door 2 to door 3. The remaining pallets 14 and 15, however, should be first unloaded in front of door 1 (which is the time when trailer 6 is assigned) and wait there till the corresponding forklift returns from door 3 – after moving pallet 2. It should be noted that the loading of pallets onto trailer 3 is carried out after the pallets in U_5 are unloaded. According to the figure, the loading order of the pallets in L_5 is $5 \rightarrow 1 \rightarrow 6 \rightarrow 2 \rightarrow 14 \rightarrow 15$. After the pallets have been loaded, trailer 5 is ready to leave and releases door 3.

It is important to note that the MIP model indeed solves three decision sub-problems embedded in the truck scheduling problem addressed in this study:

- (i) To which door are trailers assigned? (represented by decision variable x_{jm})

Figure 3.3: Timing details of trailer 5 in the crossdock when it is assigned to door 3 at $t = 4$ till it releases the door at $t = 63$

- (ii) How to sequence the queue of waiting trailers assigned to the same door? (represented by decision variable $y_{jj'}$)
- (iii) How to schedule the movement of pallets to their destination doors when one forklift is available at each origin door? (represented by decision variable $\delta_{pp'}$)

The above statement implies that by knowing the values of x_{jm} , $y_{jj'}$, and $\delta_{pp'}$, the values of the remaining decision variables are also known and the problem will be solved. Such a classification helps to direct the development of alternative solution techniques (specifically sub-optimal ones) for this problem where feasibility is as important as optimality in the process of solution construction. This will be investigated in the next two chapters.

3.2.1 Virtual Pallets

Given a product exchange pattern, it is possible that a trailer j is merely a donor (recipient) trailer. This results in having $L_j = \emptyset$ ($U_j = \emptyset$), violating the constraints defined to manage pallet unloading, moving, and loading operations (this is because the related index refers to a null object). To incorporate this special case into the MIP model without violating the affected constraints, virtual pallets are defined. For any trailer j , a virtual pallet p^V is assigned. Virtual pallet p^V represents a pallet unloaded and loaded onto the same trailer j , i.e., $p^V \in U_j$ and $p^V \in L_j$. Note that since it is virtual, the time taken to unload, move, and load a virtual pallet is considered to be zero. In other words, they are the carriers of zero products.

3.2.2 Model Practicality and Limitation

Based on the operational strategies introduced in Chapter 2 (see Table 2.1), this research studies the synchronous scheduling of trucks in a short-term planning horizon of one day in a mixed crossdocking configuration. The crossdocking operation is subject to constrained resources including dock doors and forklifts. The freight transshipment time is assumed to be a function of both the location of dock doors and availability of forklifts. Given the aforementioned characteristics, an MIP model has been developed to formulate the problem.

From a practical point of view, the proposed MIP model can be used to verify the feasibility of industrial solutions subject to the constraints defined. The reason is due to the modeling methodology which aims at being close to practice. Practicality is realized as pallets are considered the unit of shipments and realistic constraints are imposed on the number of doors, forklifts, and intermediate storage space. Furthermore, it is readily applicable to modify the model to support a few more operational strategies. Exclusive crossdocking can be realized in the model by defining two sets of inbound and outbound

doors, two sets of inbound and outbound trailers, and applying the same constraints to each, respectively. Note that due to the idea of virtual pallets, the constraints are still applicable to both inbound and outbound trailers. Mid-term planning is also attainable by fixing the assignment of outbound destinations to outbound doors in the MIP formulation.

On the negative side, however, the MIP model is limited to support synchronous crossdocking. In other words, the model is not applicable to scenarios where each truck has its own constraint on when to arrive in and depart from the crossdock. Moreover, the mathematical model cannot support a balanced utilization of forklifts in the crossdock as it is assumed that each door has its own designated forklift to move unloaded pallets to their destinations. Accordingly, for situations where the number of forklifts is fewer than the number of doors, the model is only applicable if it is specified at an early stage which forklift is designated to which door(s). Ultimately, what remains is the complexity of the model in solving the truck scheduling problem which will be discussed in the remaining chapters of the thesis.

3.3 Test Data Generation

When there is no real data available to evaluate the performance of a given algorithm to a problem, synthetic data is alternatively used. It is, however, required to present some justification to demonstrate that the generated synthetic data can reliably represent the real data. Although the majority of the papers reviewed in the previous chapter have not provided any justification for their proposed data generation methodology, this section aims to develop a realistically justified procedure to generate synthetic data for the truck scheduling problem studied in this thesis.

What is important in the data generation procedure is how the size of the crossdocking terminal (formally represented by the number of doors [48]), the queue length of

waiting trailers (i.e., the value of the trailer-to-door ratio), and most importantly, the product exchange patterns between trailers for crossdocking are characterized. A good characterization ensures that the performance of any solution algorithm to the problem is not biased towards the structure of the test data. This means that the algorithm must perform comparably if it is applied to real world case studies.

Unfortunately, in the review conducted in this study no survey paper on different crossdocking scenarios practiced in real life has been found. The industrial magazines which have been reviewed normally reported the successful implementation of crossdocking in the associated industrial enterprises and did not disclose details on the physical shape and layout of the terminal and freight flows information. Alternatively, the review on papers which used real data in their crossdocking scenarios reveals a large diversity on the values of the attributes outlined above to characterize the input data to crossdocking. Consequently, the proposed data generation procedure aims at generating a set of artificial data with a broad range of characteristics to represent the diversity exists in real world settings.

According to Gue [25], crossdocking terminals range in size from 6–8 doors to more than 200 and the queue of waiting trailers seldom exceeds 20. The latter is in contrast to what Miao et. al [46] have found in real world applications where crossdocks are usually strongly over-constrained, i.e., the number of trucks increases at a significantly faster rate than the number of doors available. To compensate for all this diversity, four data sizes from small to huge are defined with four different values of the trailer-to-door ratio for each size (see Table 3.1(a)). Likewise, the exchange composition of a trailer is given as a range by being varied between low, medium, and high mixed levels. A freight mixed level (FML) is quantified by the percentage of the trailers receiving the exchange volume of each single trailer. This also specifies the level of interdependencies defined between trailers. The three mixed levels are defined in Table 3.1(b). The amount of exchange

Table 3.1: Test data characterization

(a) Data sizes defined for the truck scheduling problem

Size	Number of Doors	Number of Trailers			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
small	8	8	16	32	64
medium	32	32	64	128	256
large	96	96	192	384	768
huge	256	256	512	1024	2048

(b) FMLs defined for the truck scheduling problem

FML	Percentage (%) of Trailers [that receive the exchange volume of each single trailer]
low	less than 10% of the total trailers
medium	10% to 25% of the total trailers
high	5% to 50% of the total trailers with 95% probability
	more than 50% of the total trailers with 5% probability

for the trailers ranges evenly from half⁴ to all of its initial carrying products. Given a maximum capacity of 60 pallets for each trailer, the initial number of products in each trailer is assumed to be uniformly distributed from 20 to 40 products.

Having specified the volume of products unloaded from a given trailer, the next step is to determine the number of trailers receiving this amount. A random number is uniformly generated in the interval corresponding to a particular freight mixed level introduced in the preceding paragraph. The exchange volume is then equally distributed among the recipient trailers. Note that in the extreme case, the number of recipient trailers is equal to the number of exchanged products, i.e., each trailer receives one product in a pallet.⁵

The final step is to identify the IDs of the recipient trailers for each trailer. This, in fact, forms the interdependencies defined between trailers, which as will be shown in the next chapter have a significant impact on the solution feasibility of the truck scheduling problem. Two approaches are proposed to build the interdependencies: *uncorrelated* and *correlated*. In the uncorrelated approach, the IDs are chosen uniformly from the entire

⁴It is not economical for a trailer to meet the crossdock for a lesser exchange volume [63].

⁵In the experiments conducted in Chapters 4 and 5, it is arbitrarily assumed that each pallet can accommodate at most four products destined to one destination trailer.

group of trailers while in the correlated one, the interdependency history of the trailer, if any, is involved in the procedure of specifying the IDs. By the interdependency history, one refers to the list of trailers having one of the following two features. They are either the predecessors of the trailer, i.e., have sent some products to the trailer, or, together with the trailer, are the successors of another trailer, i.e., have received products from the same donor trailer. The correlated approach assumes some weight for the trailers recorded in the history list so that they have a higher probability to be chosen as the recipient trailers (i.e., successors) of the given trailer. The following example better compares the two approaches:

Example. Consider the data generation procedure of eight trailers. In each iteration, the recipient trailer(s) of one trailer is(are) determined. Starting from trailer 1, it is presumed that the number of recipient trailers for this trailer is equal to four. For both uncorrelated and correlated approaches, the IDs of the four recipient trailers are chosen uniformly among the other seven trailers, i.e., $U[0, 7]$. This is because no history is available in the first iteration and all the trailers have an equal probability weight (see Table 3.2(a)). Now in case the IDs chosen are 3, 4, 6, and 8, respectively, then it is required that the history weight table be updated for the correlated approach. This is illustrated in Table 3.2(b). In the first iteration, trailers 1, 3, 4, 6, and 8 have formed an interdependency cluster so their corresponding history lists are updated for the second iteration. According to the table, the update policy increases the probability weight of the engaged trailers by one. As Table 3.2(b) shows, trailer 3 is correlated with trailers 1, 4, 6, and 8 due to having a higher probability of sending products to those trailers than the remaining ones. The same condition applies to trailers 4, 6, and 8, respectively. The above procedure continues until the IDs of the recipient trailers for trailer 8 are identified in the eighth iteration. (Note that the history update function is invoked for each iteration in the correlated approach.)

Increasing the probability weight by higher values (than that considered in the above example for each iteration) will lead to higher levels of correlation. Nevertheless, since no clear data on the exact level of correlation are available in practice, the conservative correlation level introduced in the given example will be used to generate the correlated test instances. So, according to the steps introduced for the data characterization, each category of the test data is represented in a 3-tuple (size: α ; FML; correlation status). In

Table 3.2: Values of the history weight table for uncorrelated and correlated approaches

(a) In the first iteration (similar for both)									
	1	2	3	4	5	6	7	8	
1	0	1	1	1	1	1	1	1	$= U[0, 7]$
2	1	0	1	1	1	1	1	1	$= U[0, 7]$
3	1	1	0	1	1	1	1	1	$= U[0, 7]$
4	1	1	1	0	1	1	1	1	$= U[0, 7]$
5	1	1	1	1	0	1	1	1	$= U[0, 7]$
6	1	1	1	1	1	0	1	1	$= U[0, 7]$
7	1	1	1	1	1	1	0	1	$= U[0, 7]$
8	1	1	1	1	1	1	1	0	$= U[0, 7]$

(b) In the second iteration for correlated approach									
	1	2	3	4	5	6	7	8	
1	–	–	–	–	–	–	–	–	–
2	1	0	1	1	1	1	1	1	$= U[0, 7]$
3	2	1	0	2	1	2	1	2	$= U[0, 11]$
4	2	1	2	0	1	2	1	2	$= U[0, 11]$
5	1	1	1	1	0	1	1	1	$= U[0, 7]$
6	2	1	2	2	1	0	1	2	$= U[0, 11]$
7	1	1	1	1	1	1	0	1	$= U[0, 7]$
8	2	1	2	2	1	2	1	0	$= U[0, 11]$

case one or more attributes are not reflected in the tuple, it means that the reduced tuple encompasses all the values of the missing attributes. For example, the tuple (small:1; low) represents both (small:1; low; uncorrelated) and (small:1; low; correlated) tuples.

Although different shapes for crossdocks have been proposed based on the number of doors they support (see [48]), this study adopts rectangular crossdocks for all the categories of the test data. In the crossdocking scenario studied in this thesis, no designated inbound and outbound doors are defined and all doors are treated similarly as potential candidates for the assignment of trailers. It is thus feasible to develop some door selection criteria that can be applied to other shapes and layouts for crossdocks. (These criteria are introduced in Section 4.2.2 of Chapter 4.) Given d the number of dock doors, the layout is formed by placing $d/8$ doors on the shorter side of the crossdock and $3d/8$ doors on the longer side. (Note that d is a multiple of 8.) The distance between doors

is specified by assuming that the distance between adjacent doors (and between each corner and the closest door) is 1 distance unit.

As mentioned in Section 3.1.1, pallet unloading/loading time is constant and identical for all pallets and it is arbitrarily assumed to be one second for all the experiments conducted in this thesis. Pallet moving time is considered to be equal to the distance a pallet is moved by forcing the forklift velocity to be one meter per second. Finally, the time taken to replace a trailer with another trailer for the assignment to a door, i.e., the trailer changeover time, which is assumed to be the same for all trailers, is arbitrarily set to 75 seconds. Note that these constants are parameters to the problem and not to the solution algorithm, and hence, do not bias the experimental results.

3.4 Imposing Constraint on Crossdock Capacity

This section investigates the situation where the crossdock capacity is assumed limited. The capacity is represented as limited staging areas distributed in front of crossdock doors. To analyze whether the crossdock capacity turns to a bottleneck in this crossdocking scenario, an experiment is conducted to monitor how the designated spaces for each door are utilized when a feasible schedule of trucks has already been constructed for crossdocking. It is assumed that the available space reserved in front of each door is the maximum number of pallets each trailer can carry, i.e., 60 pallets, where each pallet contains only one product. (In this case, pallets and products are interchangeable.) So the total crossdock available space [measured in number of pallets] is equal to 60 times the total number of doors. The assigned value is smaller than the one estimated in practice. According to Bartholdi and Gue [48], rectangular crossdocks are usually 60-120 feet wide (even wider), meaning that the reserved space is bigger than the entire capacity of standard 48-foot trailers typically used in distribution networks.

For each combination of the problem sizes, trailer-to-door ratios (α), freight mixed levels (FML), and uncorrelated and correlated interdependencies, i.e., $4 \times 4 \times 3 \times 2 = 96$ combinations, 10 random instances are generated and solved by a heuristic algorithm to construct a feasible truck schedule. The algorithm is a rule-based heuristic search which exhaustively searches the solution space to find a feasible schedule of trucks and will be introduced in Chapter 4. For each problem instance, the amount of space that a given trailer allocates, after being assigned to a door, is determined by measuring the number of its unloading and loading pallets plus the number of pallets unloaded from the previously assigned trailers (i.e., those which have been assigned to the same door before the given trailer) and yet have not been moved to their destination doors. The minimum, average, and maximum of the quantities are recorded afterwards. The total crossdock allocated space, i.e., the total number of pallets staged in front of all doors, is also updated every time the pallets of a given trailer are unloaded and the maximum value, in both absolute and relative [to the total crossdock available space] representations, is recorded. The two representations are denoted as abs. and rel. in Table 3.3, respectively.

The average experimental results for each problem size with a given trailer-to-door ratio (size: α) are summarized in Table 3.3. The results indicate that the entire crossdock capacity does not overflow. Even in the extreme instance of $\alpha = 8$, the utilization ratio is below 70% of the crossdock capacity. This means that the crossdock capacity in the studied crossdocking scenario does not turn out to be a bottleneck resource to temporarily stop the process of unloading pallets from trailers. Nonetheless, it is observed that on some occasions the allocated space in front of doors overruns its limit, which is set to 60 pallets. The above congestion needs to be resolved via balancing the number of staged pallets in front of each door. This issue will be considered one subject of future research and discussed in Chapter 6.

Table 3.3: Crossdock capacity utilization does not reach its upper limit

Data Category	Door Allocated Space			Crossdock Capacity		
				allocated		total
	min.	avg.	max.	abs.	rel.	available
(small:1)	35	50	66	180	0.38	480
(small:2)	31	55	88	183	0.38	
(small:4)	24	59	110	222	0.46	
(small:8)	22	77	163	332	0.69	
(medium:1)	25	49	81	674	0.35	1920
(medium:2)	25	56	107	776	0.40	
(medium:4)	22	60	127	821	0.43	
(medium:8)	21	77	191	1261	0.66	
(large:1)	22	49	98	1854	0.32	5760
(large:2)	23	57	127	2342	0.41	
(large:4)	20	60	146	2516	0.44	
(large:8)	20	78	199	3647	0.63	
(huge:1)	21	50	109	4737	0.31	15360
(huge:2)	20	57	141	6249	0.41	
(huge:4)	19	61	154	6679	0.43	
(huge:8)	19	77	219	9653	0.63	

3.5 Summary

Before any solution algorithm can be developed and evaluated for the truck scheduling problem, it is required that the problem be defined and formulated. Furthermore, the evaluation entails experimental data. This chapter addressed the two issues by first developing a crossdocking model and second a data generation methodology. The model supports a special type of crossdocking with unique operational characteristics compared to the previous work. It is not an abstract model since it considers pallets as the unit of shipments exchanged between the trailers (the common case in real crossdocks) and imposes constraints on the number of doors and forklifts as the crossdock critical resources. The synthetic data were generated with a broad range of characteristics to represent the diversity exists in real world settings. These characteristics include the problem size in terms of the number of doors, the ratio of the number of trailers to the number of doors,

freight distribution patterns between trailers, and correlation status of interdependent trailers. This has led to a data set containing problem instances of different levels of difficulty representing small crossdocks hosting a few dozen trailers to huge ones hosting over 2000 trailers with various freight exchange patterns.

Because the mathematical model developed to formulate truck scheduling cannot be regarded an efficient (or even practical) approach for solving the real size instances of this NP-hard problem, sub-optimal (heuristic) solution techniques should be developed as well so that fairly good solutions can be found in a reasonable amount of time, and scalability is maintained when the problem size increases to large scales. As there might be complexities in the solution feasibility of the truck scheduling problem (due to the non-preemptive door assignment), the problem feasibility is addressed before its optimality in the next chapter. Since the problem is operationally different from the truck scheduling problems reviewed in Chapter 2, the developed MIP model solved by the CPLEX solver will be considered the standard benchmarking tool to evaluate the efficiency of the proposed algorithms both in terms of solution feasibility and quality.

Chapter 4

Solution Feasibility Construction

The non-preemptive door assignment adopted in the truck scheduling problem studied in this thesis incurs complexities in its solution feasibility. The reason is that the problem may be entrapped in deadlock and no feasible solution is produced. This chapter therefore aims at developing an algorithmic approach that first, is capable of establishing solution feasibility for truck scheduling problem instances of various types and difficulty levels and second, can be readily implemented in an industrial setting. The proposed solution approach is a heuristic algorithm comprises of two phases. In the first phase, a heuristic search is deployed to construct a feasible sequence of trucks for the assignment to dock doors. In the second phase, a rule-based heuristic is implemented to assign each sequenced truck to a proper dock door so that significant savings in the truck schedule length are achieved. Such an assignment is subject to a limited number of forklifts equal to the total number of doors. Extensive experiments are conducted to evaluate the efficiency of the algorithm in terms of deadlock avoidance and solution quality. The evaluation is carried out against the solutions generated by the exact mathematical model of the problem developed in the previous chapter and a constructive heuristic proposed in [36] for a similar truck scheduling problem. Before introducing the algorithm, it is important to illustrate why establishing a feasible solution for the truck scheduling problem is a concern when the trailer-to-door assignment is not preempted.

4.1 Truck Scheduling Feasibility

The solution feasibility of the truck scheduling problem can be a complication when the number of trucks is more than the number of doors (i.e., $\alpha > 1$). In these situations, the problem might be entrapped in deadlock and no feasible solution is produced. This was first observed in the previous work of this research [36] where the authors proposed a dependency ranking (DR) constructive heuristic for a similar truck scheduling problem. The infeasibility occurs because the proposed DR heuristic is not robust in detecting and avoiding deadlock while constructing the sequence of trailers for the assignment to dock doors. (The DR heuristic will be re-used in this chapter for evaluation purposes.) More illustration is given by the following example:

Example. Consider the problem of scheduling eight trailers appointed to exchange some of their products in a four-door crossdock. The product exchange matrix is given by Figure 4.1(a). What decides on the solution feasibility of the truck scheduling problem is the interdependencies defined between the trailers and not the number of products they exchange. It is therefore more illustrative to investigate the feasibility using the corresponding directed graph (DG) depicted in Figure 4.1(b). One strategy to schedule the trailers is to decompose the interdependency DG into its strongly connected components (SCC), e.g., by Tarjan's algorithm [64], and schedule each SCC according to the order its vertices are reached, i.e., while its corresponding subtree is being formed. For the above example, the given interdependency DG is already strongly connected so a spanning tree would be generated. Figure 4.1(c) shows the constructed spanning tree from which the following trailer sequence is formed: $1 \rightarrow 5 \rightarrow 3 \rightarrow 2 \rightarrow 7 \rightarrow 8 \rightarrow 6 \rightarrow 4$.

Now if the trailers are assigned to the doors according to the above order, at the time of assigning trailer 7 no free door is available and the scheduling process is forced to stop. This is because all the four doors have been assigned trailers 1, 5, 3, and 2, respectively, and cannot be released. Trailer 1 cannot complete its operations (and thus cannot free its assigned door) unless trailers 6, 7, and 8 are assigned to a door and unload its required products. A similar situation applies to trailers 5, 3, and 2 which have been assigned to the remaining three doors. (The only trailer that could complete its operations and leaves the crossdock is trailer 4 which is scheduled after the assignment of trailer 3.) This situation is referred to as *deadlock*. It occurs because door preemption is not allowed throughout scheduling. The same condition applies if the DR heuristic is used to schedule the trucks (see [36]). Given the sequence $8 \rightarrow 1 \rightarrow 7 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$ generated by the DR algorithm, trailer 5 cannot be scheduled due to the deadlock caused after the assignment of trailers 1, 7, 3, and 4 to the four doors of the crossdock. On the other hand, if the trailers are scheduled according to the sequence $8 \rightarrow 7 \rightarrow 1 \rightarrow 2 \rightarrow 6 \rightarrow 5 \rightarrow 4 \rightarrow 3$, deadlock does not occur and a feasible solution will be produced.

Figure 4.1: (a) Product exchange matrix of eight trailers (b) the interdependency DG, and (c) the spanning tree formed by Tarjan's SCC algorithm

The preceding example helps to understand the importance of developing a solution technique capable of avoiding, or detecting and resolving deadlock¹. Suppose either of the suggested strategies (Tarjan or the DR algorithm) is implemented in an industrial setting. Both approaches oblige the terminal manager to allocate at least five doors to schedule the eight trailer while a feasible schedule does exist for only four doors. It may be, however, so costly, or even impossible from a practical viewpoint to open an extra door to crossdock the trailers. Accordingly, to manage the complexity of the studied truck scheduling problem, the proposed heuristic algorithm is broken into two phases such that the first phase is merely dedicated to establishing solution feasibility when a limited number of doors are available. The detailed truck scheduling procedure is then carried out in the second phase after a feasible schedule has been found in the first phase.

4.2 A Two-Phase Heuristic Algorithm

Due to the time complexity requirement, mathematical models like the one developed in the previous chapter are normally inefficient, or even impractical, for solving real

¹For some truck scheduling problem instances, deadlock occurrence is inevitable relative to a pre-specified number of doors. In that case, the truck scheduling problem is infeasible [for that number of doors].

size instances of an NP-hard combinatorial problem like the studied truck scheduling problem. The common approach to solve this sort of problems is to propose heuristic solution methods so that fairly good solutions can be found in a reasonable amount of time, and scalability is maintained when the problem size increases to large scales.

Having the same operational characteristics assumed in the development of the MIP model, a two-phase heuristic algorithm is proposed to solve the truck scheduling problem. In the first phase, which is called *truck sequencing*, a dependency ranking search (DRS) heuristic is developed to establish solution feasibility, subject to a given number of doors, by exhaustively searching the solution space to produce a feasible trailer sequence. The search path is dynamically guided by ranking trailers according to their interdependencies. In the second phase, which is called *door assignment*, trailers are scheduled, i.e., assigned to doors, according to the feasible sequence list generated in the first phase. This is done by developing a door fitness (DF) heuristic that deploys a fitness mechanism to choose and assign a proper door to a given trailer. The DF heuristic also schedules the movement of the forklifts operating in the crossdock. Note that all the timings concerning the schedule of trucks together with unloading, moving, and loading the exchanged pallets are determined in the second phase by the DF algorithm.

4.2.1 Sequencing Phase: DRS Heuristic

The primary criterion considered in developing the DRS heuristic is that establishing solution feasibility precedes optimality in addressing the problem. The proposed algorithm benefits from three main features to find a feasible solution for the truck scheduling problem. First, it incorporates the number of available doors into its sequencing procedure to ensure feasibility throughout the search. Second, the algorithm is based on a ranking methodology that attempts to avoid deadlock by maintaining door availability each time a trailer is sequenced. The strategy is to expedite the sequencing process of the trailers

which contribute products to more trailers (especially those that have been assigned and yet have not released their assigned door). Finally, the third feature is backtracking which enables the algorithm to recover from deadlock (whenever entrapped) by retrieving its previous state and adjust its search path.

DRS sequences one trailer per iteration. Trailers candidate for sequencing per iteration are stored in a list called L^S (see Table 4.1). When a trailer in L^S is sequenced and assigned to a door, it is moved to another list by the name of L^A . The trailer is still assigned to a door until its predecessor trailers (i.e., those giving products to the trailer), if any, are assigned to some doors. In that case, the trailer releases its assigned door and is moved from L^A to a list called L^R . L^S is updated each iteration to include a new list of trailers candidate for sequencing. This continues until all trailers are sequenced.

DRS localizes its sequencing process by starting from a subset of tightly interdependent trailers and gradually moving to other interdependent subsets until all trailers are sequenced. Localization is realized by defining clusters. Each trailer together with its successors and predecessors form a cluster. This implies that the number of clusters is equal to the number of trailers where each cluster ID is represented by the ID of the cor-

Table 4.1: Parameters defined to describe the DRS algorithm

Parameter	Description
S_j	set of successors of trailer j (i.e., trailers receiving products from trailer j)
P_j	set of predecessors of trailer j (i.e., trailers giving products to trailer j)
C_j	cluster corresponding to trailer j
r_j^I	internal rank of cluster j
r_j^E	external rank of cluster j
L^S	list of trailers for sequencing and assignment
L^A	list of trailers assigned yet have not released their assigned door
L^R	list of trailers that have released their assigned door
c_j^S	contribution of trailer j when included in list L^S
c_j^A	contribution of the trailer j when included in list L^A
r_j^S	rank of the trailer j in list L^S

responding trailer. Given the parameters defined in Table 4.1, cluster C_j corresponding to trailer j is formed as follows:

$$C_j = \{j\} \cup S_j \cup (P_j \setminus L^R), \quad j \notin L^R \quad (4.1)$$

Note that trailers with no predecessors are sequenced in advance and are not involved in the formation of clusters. Such trailers do not require products from any trailer but may provide products to some others. So in case they are assigned to a door, they can release their assigned door after unloading products required by other trailers, if any, and leave the crossdock.

The subset of tightly interdependent trailers to start sequencing with is one of the clusters referred to as the *initial* cluster. Two ranking criteria are introduced to choose the initial cluster. The first (and main) criterion is based on the internal contribution of the cluster members with one another and is referred to as the *internal* rank. The internal contribution is assessed by determining how many predecessors of each cluster member are inside the cluster itself. Internal rank r_j^I of cluster C_j is calculated according to the following formula:

$$r_j^I = \frac{\sum_{j' \in C_j} \frac{|P_{j'} \cap C_j|}{|P_{j'} \setminus L^R|}}{|C_j|} + \frac{\sum_{j' \in C_j} \left\lfloor \frac{|P_{j'} \cap C_j|}{|P_{j'} \setminus L^R|} \right\rfloor - 1}{|C_j| - 1} \quad (4.2)$$

For each member in the cluster, the percentage of its predecessors which are already in the cluster is calculated. The values are added and then scaled by dividing them by the size of the cluster – represented by the first term of Eq. (4.2). The second term of the equation considers an additional bonus if *all* the predecessors of any member (excluding the corresponding trailer – see '1') are in the cluster as well.

The second criterion is based on the external contribution of the cluster members to trailers outside the cluster and is referred to as the *external* rank. The external contribution is assessed by determining how many successors of each cluster member are

outside the cluster. External rank r_j^E of cluster C_j is calculated according to the following formula:

$$r_j^E = \sum_{j' \in C_j} |S_{j'} \setminus C_j| \quad (4.3)$$

For each member in the cluster, the number of its successors which are not in the cluster is recorded. The values are then added and the sum is considered as the external rank.

The initial cluster representing the most tightly interdependent subset of trailers is the one having the highest internal rank. In the case of ties, the cluster with the highest external rank is chosen. If there are clusters that have the same highest internal and external ranks, then, according to the number of pallets exchanged between their respective members, the one with the highest exchange volume is chosen. If none of the above is applicable, the cluster with the lowest ID is used as the initial cluster. The cluster formation and ranking procedure is illustrated by the following example:

Example. Consider eight trailers for scheduling in a four-door crossdock. As discussed earlier, the interdependencies defined between trailers determine the solution feasibility and not the number of products they exchange. So, to form and rank the clusters, the interdependencies are taken into account. In Chapter 3, the interdependency matrix was introduced as one way of representing the interdependencies. Table 4.2 shows the interdependency matrix for the eight trailers in this example. Given the input parameters presented by Table 4.3, Eqs (4.2) and (4.3) are invoked to find the best initial cluster. Table 4.3 also shows both the internal and external ranks of the eight clusters. Regarding the proposed selection strategy, cluster C_7 , shown in bold, has the highest internal rank and is chosen as the initial cluster. Note that trailer 8 will be sequenced earlier (since it has no predecessors) and is not involved in the formation of clusters. Also note that clusters 2 and 4 have identical members, leading to identical values for both their internal and external ranks, respectively.

When the initial cluster is identified, its members are included in list L^S . L^S holds a group of tightly interdependent trailers from which one trailer is sequenced in each iteration. Like clusters, each trailer in L^S is ranked and the one with the highest rank is chosen for sequencing. As indicated before, the methodology to rank a trailer in L^S is based on expediting the sequencing process of trailers which contribute more products

Table 4.2: Interdependency matrix of eight trailers

	1	2	3	4	5	6	7	8
1	0	0	0	1	0	1	1	0
2	1	0	1	0	1	0	1	0
3	0	1	0	1	0	1	0	0
4	0	1	0	0	1	0	1	0
5	1	1	0	0	0	1	0	0
6	1	0	1	0	1	0	0	0
7	1	0	1	1	0	1	0	0
8	0	0	1	1	0	0	0	0

Table 4.3: Cluster formation and ranking of the eight trailers of Table 4.2

j	S_j	P_j	C_j	r_j^I	r_j^E
1	{4,6,7}	{2,5,6,7}	{1,2,4,5,6,7}	1.25	3
2	{1,3,5,7}	{3,4,5}	{1,2,3,4,5,7}	1.25	4
3	{2,4,6}	{2,6,7,8}	{2,3,4,6,7}	0.7	6
4	{2,5,7}	{1,3,7,8}	{1,2,3,4,5,7}	1.25	4
5	{1,2,6}	{2,4,6}	{1,2,4,5,6}	0.65	5
6	{1,3,5}	{1,3,5,7}	{1,3,5,6,7}	0.62	5
7	{1,3,4,6}	{1,2,4}	{1,2,3,4,6,7}	1.26	3
8	{3,4}	{}	—	—	—

to more trailers and is built up from four elements. The first element represents the contribution of the trailer to the other trailers in L^S and those trailers that are still in L^A . The contribution is assessed by determining how many of these trailers are the successors of the given trailer. The contribution, however, is scaled relative to the trailer's current dependency level (specified by the number of its current unsequenced predecessors) to balance contribution vs. dependency. Contribution c_j^S of trailer j in list L^S is described analytically as follows:

$$c_j^S = \frac{|S_j \cap L^S| + |S_j \cap L^A|}{|P_j \setminus (L^A \cup L^R)|} \quad (4.4)$$

The contribution of these other trailers themselves is accounted as the second element of the proposed ranking mechanism. The trailers are either in L^S or L^A . For the former case, the contribution is calculated similarly according to Eq. (4.4). For the latter, the contribution of a trailer not only considers the trailers in L^S and L^A but also applies to

the remaining unsequenced trailers still waiting to be included in L^S . This is because the trailer has already unloaded the pallets required by all of its sequenced and unsequenced successor trailers. Similar to Eq. (4.4), the contribution is scaled relative to the trailer's current dependency level to balance contribution vs. dependency. The sum of the contributions is then scaled by dividing with the number of these trailers that are in L^S or L^A (see the second term of Eq. (4.6) below). The following formula computes contribution c_j^A of trailer j in list L^A :

$$c_j^A = \frac{|S_j \setminus L^R|}{|P_j \setminus (L^A \cup L^R)|} \quad (4.5)$$

The third element indicates how tightly the trailer is interdependent on the other trailers in L^S . This is assessed by determining how many trailers in L^S are both the successors and predecessors of the trailer. Like Eqs (4.4) and (4.5), the value is scaled by the number of the trailer's current unsequenced predecessors (see the third term of Eq. (4.6) below). Finally, the fourth element gives an additional bonus to the rank of the trailer if it serves as a predecessor of any trailer still in L^A . The bonus is represented as the inverse of the total number of the unsequenced predecessors of the trailer in L^A (see the fourth term of Eq. (4.6) below). The reason is to give higher ranks to those trailers in L^S which are more contributive to trailers in L^A . The following formula integrates the above four elements to calculate rank r_j^S of trailer j in list L^S :

$$\begin{aligned} r_j^S = c_j^S &+ \frac{\sum_{j' \in S_j \cap L^S} c_{j'}^S + \sum_{j' \in S_j \cap L^A} c_{j'}^A}{|S_j \cap L^S| + |S_j \cap L^A|} + \\ &\frac{|S_j \cap P_j \cap L^S|}{|P_j \setminus (L^A \cup L^R)|} + \sum_{j' \in S_j \cap L^A} \frac{1}{|P_{j'} \setminus (L^A \cup L^R)|} \end{aligned} \quad (4.6)$$

Two rules are defined to break ties, if any, when choosing the best trailer from L^S . The first one considers the *absolute* value of the trailer's contribution with respect to its current dependency level:

$$|S_j \cap L^S| + |S_j \cap L^A| - |P_j \setminus (L^A \cup L^R)| \quad (4.7)$$

The trailer having the largest value is then selected. The second rule is defined similarly except that instead of its contribution, the trailer's interdependency level is considered:

$$|S_j \cap P_j \cap L^S| - |P_j \setminus (L^A \cup L^R)| \quad (4.8)$$

In case neither works, the trailer with the lowest ID will be chosen.

Having sequenced and assigned the best trailer to a free door, the algorithm updates L^S by first deleting the trailer (now the trailer is in L^A) and second inserting its unsequenced predecessors, if any. It then monitors the allocation status of the doors. Two situations may occur. One is that the assignment of the trailer has maintained, or even improved, door availability. The other is that door availability is no longer maintained. In the latter case, DRS is entrapped in deadlock and, as explained later, requires backtracking to resolve it. In the former case, the algorithm searches for any trailer that can be released. A trailer is released if all of its initial predecessors have been sequenced and assigned to a door (irrespective of whether these trailers themselves have freed their assigned door or not). Depending on the status of a ready-to-release trailer, DRS does one of the following two actions. If the trailer is already assigned, the algorithm simply frees the assigned door, i.e., moves the trailer from L^A to list L^R . Otherwise, it will directly sequence the trailer by inserting it into the trailer sequence list (and also L^R). The above steps continue until no trailers can be released.

The algorithm then ranks the current trailers in L^S to find and sequence the next best trailer for the next iteration. The sequencing procedure continues until L^S becomes empty. If L^S is empty but there are some trailers which are still unsequenced, it means that there is at least one subset of trailers having no or loose interdependencies with those already-sequenced trailers. In that case, DRS resumes its sequencing procedure by ranking the clusters corresponding to the remaining trailers and reinitializing list L^S . The above courses of action repeat until all the trailers are sequenced.

It is possible that, after a trailer is sequenced and assigned to a free door, no door is left available and no assigned trailer can be released. In this situation, the sequencing process is not straightforward and encounters deadlock. In the introduction of DRS, it was mentioned that the algorithm is equipped with backtracking to resolve deadlock. By backtracking, the algorithm goes one level back and chooses the next best trailer from list L^S based on the ranking and selection criteria already described. If no more trailer is left untried, backtracking takes the sequencing status one further level back. In the case where backtracking has reached the first level and all the trailers of the initial cluster have been examined yet no feasible solution is available, DRS performs *re-starting* by choosing the next best cluster and resumes the sequencing process. This continues until a feasible solution is found or all the possible sequence patterns are tried by re-starting from the remaining clusters. In other words, DRS exhaustively searches the entire solution space to find a feasible trailer sequence.

Example. It is good to continue the truck scheduling example in Table 4.2. In the first part, the initial cluster, i.e., C_7 , was identified. Accordingly, its members are the initial trailers stored in L^S . The next step is to rank the trailers according to Eq. (4.6) to choose the first trailer to be sequenced in the first iteration. Table 4.4 presents the DRS sequencing procedure to sequence the eight trailers in a four-door crossdock. (As already mentioned, truck sequencing in DRS is done relative to the given number of doors.) Similar to the cluster formation and ranking, trailer 8 is not included in the sequencing process as it was sequenced in advance. According to the table, the sequence list is constructed after eight iterations and has the following order: $8 \rightarrow 4 \rightarrow 7 \rightarrow 2 \rightarrow 6 \rightarrow 1 \rightarrow 3 \rightarrow 5$. After the fourth iteration, the algorithm is entrapped in deadlock and cannot sequence any more trailer. The deadlock is resolved as DRS goes one level back (by backtracking) and chooses the next best trailer for sequencing. Figure 4.2 also depicts the state transitions of DRS until it finds the feasible truck sequence.

Figure 4.2: DRS state transitions to sequence the eight trailers of Table 4.2

Table 4.4: DRS procedure to sequence the eight trailers of Table 4.2 in a four-door crossdock

Level	State	L^S							L^A	L^R	d^*	Deadlock Occurrence
1	a	$\frac{j}{r_j^S}$	1	2	3	4	6	7	{}	{8}	4	no
		r_j^S	2.08	2.36	2.39	2.17	1.88	2.73				
2	a	$\frac{j}{r_j^S}$	1	2	3	4	6			{8}	3	no
		r_j^S	2.67	2.94	3.39	2.5	2.58		{7}			
3	a	$\frac{j}{r_j^S}$	1	2	4	6			{3,7}	{8}	2	no
		r_j^S	3.11	3.61	3.75	3.25						
4	a	$\frac{j}{r_j^S}$	1	2	6					{8}	1	no
		r_j^S	4.83	5.5	3.25				{3,4,7}			
5	a	$\frac{j}{r_j^S}$	1	5	6				{2,3,4,7}	{8}	0	yes, backtrack to level 4
		r_j^S	—	—	—							
4	b	$\frac{j}{r_j^S}$	1	2	6					{8}	1	no
		r_j^S	4.83	5.5	3.25				{3,4,7}			
5	b	$\frac{j}{r_j^S}$	2	5	6				{1,3,7}	{4,8}	1	no
		r_j^S	8.38	5.39	5.89							
6	a	$\frac{j}{r_j^S}$	5	6						{4,7,8}	1	no
		r_j^S	7.67	7.33					{1,2,3}			
7	a	$\frac{j}{r_j^S}$	n/a						{}	{1,2,3,4,5,6,7,8}	4	no, sequencing is done
		r_j^S	—									

*represents the number of available doors

It was mentioned that DRS exhaustively searches the solution space until it either finds a feasible solution or visits all the possible sequence patterns by re-starting from the remaining clusters. This means that if no solution is found, one can guarantee that deadlock occurrence is inevitable with respect to the given number of doors (see Theorem 4.1 for the proof). Procedure 4.1 describes the DRS heuristic in pseudo codes:

Procedure 4.1 DEPENDENCYRANKINGSEARCH

1. Sequence any trailer j whose $P_j \setminus L^R = \emptyset$;
 2. Form and rank C_j for each unsequenced trailer j ;
 3. **for all** j where C_j has not been tried yet **do**
 4. **if** no cluster is found **then**
 5. Stop; {no feasible solution is found}
 6. **else**
 7. Choose the cluster with the highest rank;
 8. **end if**
 9. **end for**
 10. Initialize L^S with the trailers of the best cluster;
 11. **while** $L^S \neq \emptyset$ **do**
 12. Rank the trailers in L^S ;
 13. **for all** $j \in L^S$ where trailer j has not been tried yet **do**
 14. **if** no trailer is found **and** sequencing is in Level 1 **then**
 15. **goto** Line 3; {re-start from the next best cluster}
 16. **else if** no trailer is found **then**
 17. Retrieve the previous level; {backtrack occurs}
 18. **goto** Line 13;
 19. **else**
 20. Sequence the trailer with the highest rank;
 21. Remove the trailer from L^S and add it to L^A ;
 22. Add its unsequenced predecessors to L^S ;
 23. Check for door availability;
 24. **if** door availability is not maintained **then**
 25. Retrieve the previous level; {backtrack occurs}
 26. **goto** Line 13;
 27. **else**
 28. Search for any trailer that can be released;
 29. Update L^S , L^A , and L^R accordingly;
 30. **end if**
 31. **end if**
 32. **end for**
 33. **end while**
 34. **if** any unsequenced trailer is left **then**
 35. **goto** Line 2;
 36. **else** {a feasible solution is found}
 37. **return** the sequence list;
 38. **end if**
-

Theorem 4.1 DRS guarantees infeasibility if it finds no solution.

Proof. Suppose there exists one sequence of trailers which is known to be feasible. It will be shown that the given sequence can be modified to another sequence list visited by DRS whereas the feasibility is still maintained. For any trailer in the list, one of the following perturbations is applied:

- 1a. If the trailer has no predecessor, shift it to the extreme left of the sequence list. (Note that the feasibility is maintained since the trailer is not dependent on the other trailers.)
- 1b. If the trailer has no successor, shift it to the extreme right of the sequence list. (Note that the feasibility is maintained since no trailer is dependent on the trailer.)
- 1c. If the trailer is placed after all of its predecessors in the sequence list or is after at least one of its successors, leave the trailer as it is.
- 1d. If at least one predecessor of the trailer is placed after it in the sequence list and all of its successors are after the trailer in the list, depending on the positions of the last predecessor and first successor of the trailer in the list, one of the following conditions applies:
 - 2a. If its last predecessor is placed before its first successor, shift and place the trailer immediately after its last predecessor. (Note that the shift maintains the feasibility since no trailer is dependent on the trailer in the shifting interval.)
 - 2b. If its first successor is placed before its last predecessor, shift and place the trailer immediately before its nearest successor. (Note that the shift maintains the feasibility since no trailer is dependent on the trailer in the shifting interval.)

The question is whether the trailer can be placed after its successor and the feasibility still holds? The current order requires that the trailer be assigned first then followed by the assignment of its successor. The trailer assignment, however, does not help to release any assigned door as no assigned trailer is its successor. The trailer itself cannot free its assigned door either since at least one of its predecessors has not been assigned yet. All of this means that another door must be available to assign the successor. As a matter of fact, two doors must be available for the respective assignment of the trailer and its successor. This implies that the trailer can be placed after its successor in the sequence list without invalidating the feasibility.

Each regular trailer (trailers that have both successors and predecessors) in the resultant sequence list acquires one of the following two attributes. They are either placed after all of their predecessors or after at least one of their successors. Both situations have been considered in the DRS sequencing procedure. For the former, after the assignment of a trailer, it is always checked whether any trailer can be released, i.e., all of its predecessors have been sequenced before the trailer. Likewise, for the latter, when a trailer is sequenced and assigned to a door, its unsequenced predecessors (which the trailer is their successor) are inserted into the candidate list and considered to be sequenced afterwards. So the modified sequence list must have been visited by DRS and hence, the feasibility would have been achieved.

Table 4.5: Parameters defined to describe the DR algorithm

Parameter	Description
S_j	set of successors of trailer j (i.e., trailers receiving products from trailer j)
P_j	set of predecessors of trailer j (i.e., trailers giving products to trailer j)
S	list of sequenced trailers
r_j^U	rank of unsequenced trailer j
U	unsequenced trailers whose ranks have changed since the first iteration

The efficiency of the DRS algorithm in establishing solution feasibility is evaluated against the DR heuristic. In the DR sequencing procedure proposed in [36], each trailer is ranked according to the ratio of the number of its successors to the number of its predecessors. Given the parameters defined in Table 4.5, rank r_j^U of an unsequenced trailer j is analytically described as follows:

$$r_j^U = \frac{|S_j|}{|P_j \setminus S|} \quad (4.9)$$

The trailer with the highest rank along with its predecessors are sequenced afterwards. The sequenced trailers are removed from the lists of unsequenced trailers and their unsequenced predecessors, respectively. The remaining unsequenced trailers are ranked again so that the next trailer with the highest rank together with its [unsequenced] predecessors are sequenced. The above steps repeat until all trailers are sequenced.

As the DR algorithm does not verify the feasibility of the generated sequence list against a certain number of doors, solution feasibility is not ensured. This section thus proposes two other variations of the algorithm on the quest for reducing deadlock probability. In the first variation, rather than repeatedly sequencing a trailer and its predecessors, after sequencing one trailer, the algorithm searches for any unsequenced trailer whose rank is infinity, i.e., the number of its unsequenced predecessors is zero. If any trailer is found, it is sequenced immediately. The second variation, while including the previous amendment, localizes the sequencing procedure. After the best trailer and its

predecessors are sequenced in the first iteration, the selection mechanism finds the next best trailers – for the remaining iterations – among those unsequenced trailers whose ranks have changed until then. Procedure 4.2 describes the three versions of DR denoted as DR_1 , DR_2 , and DR_3 in pseudo codes:

Procedure 4.2 DEPENDENCYRANKING₁₂₃

Procedure 4.2.1 DR_1

1. Rank all trailers;
2. **while** any unsequenced trailer is left **do**
3. Sequence trailer j with the highest rank;
4. **while** $P_j \setminus S \neq \emptyset$ **do**
5. Sequence one unsequenced predecessor of trailer j ;
6. **end while**
7. Repeat ranking the remaining unsequenced trailers;
8. **end while**
9. **return** S ;

Procedure 4.2.2 DR_2

1. Rank all trailers;
2. **while** any unsequenced trailer is left **do**
3. Sequence trailer j with the highest rank;
4. Sequence any trailer j' where $P_{j'} \setminus S = \emptyset$;
5. **while** $P_j \setminus S \neq \emptyset$ **do**
6. Sequence one unsequenced predecessor of trailer j ;
7. Sequence any trailer j' where $P_{j'} \setminus S = \emptyset$;
8. **end while**
9. Repeat ranking the remaining unsequenced trailers;
10. **end while**
11. **return** S ;

Procedure 4.2.3 DR_3

1. $U \leftarrow \emptyset$; $\{U$ has no member initially}
 2. Rank all trailers;
 3. **while** any unsequenced trailer is left **do**
 4. **if** $U \neq \emptyset$ **then**
 5. Sequence trailer $j \in U$ with the highest rank;
 6. **else**
 7. Sequence trailer j with the highest rank;
 8. **end if**
 9. Sequence any trailer j' where $P_{j'} \setminus S = \emptyset$;
 10. **while** $P_j \setminus S \neq \emptyset$ **do**
 11. Sequence one unsequenced predecessor of trailer j ;
 12. Sequence any trailer j' where $P_{j'} \setminus S = \emptyset$;
 13. **end while**
 14. Repeat ranking the remaining unsequenced trailers;
 15. Update U ;
 16. **end while**
 17. **return** S ;
-

The three versions of the DR heuristic form a so-called DR_b algorithm whose solution output is the minimum of the three solutions and runtime is the sum of the three running times. It should be noted that irrespective of which algorithm is deployed (i.e. DRS or any variation of DR), a trailer sequence list will be generated as the output. The sequence list is then given to the DF assignment algorithm to produce the final solution which is the makespan.

4.2.2 Assignment Phase: DF Heuristic

The DF heuristic follows a greedy approach to choose a proper door for each trailer picked from the sequence list. (Note that the trailer sequence list is generated in the first phase.) The selection mechanism is developed by defining a fitness for each door. The door attaining the best fitness is then chosen and assigned to the trailer.

There are a number of criteria to define the fitness. The most apparent criterion to characterize the fitness of a door is its availability time. A door having the earlier availability time is a better candidate for the assignment. To realize whether the interdependencies of the given trailer should also be considered by the selection mechanism or not, they are divided into two categories. The first category includes both the successors and predecessors of the trailer whose position in the sequence list is before the trailer and the second category includes the successors and predecessors of the trailer whose position in the sequence list is after the trailer. The assignment information of the trailers in the first category, including the assignment time and assigned door, is known at the time of assigning the given trailer itself whereas for the trailers in the second category, no exact information is available and estimation should be applied. In the preliminary experiments, however, it was observed that the estimated information does not help to improve the solution quality but rather severely worsens the final results. This is mainly due to the high complexity of the problem making the estimation procedure complicated, and

Table 4.6: Parameters defined to describe the DF algorithm

Parameter	Description
M	set of crossdock doors
t_m	availability time of the door m
$d_{mm'}$	[rectilinear] distance from door m to door m'
l_m	physical location of door m
S_j	set of successors of trailer j
P_j	set of predecessors of trailer j
J^K	set of trailers whose assigned door is known
$M_{[j]}$	[candidate] assigned door of trailer j
$n_{jj'}$	number of pallets sent from trailer j to trailer j'
e_j	total number of exchanged pallets of trailer j
N_m^C	set of neighbors of door m formed by the average cluster size
t_m^N	total availability times of the doors in neighborhood N_m^C of door m
f_m	fitness of door m [measured in time]

hence, inaccurate. On the contrary, including the exact assignment information (of the trailers in the second category) in the fitness function did help to improve the makespan.

For the cases where no exact assignment information is available about any of the successors or predecessors of the trailer (e.g., when the first trailer in the sequence list is to be assigned), the physical location of the doors in the crossdock will be included in the fitness function. The physical location of a door is assessed by its average [rectilinear] distance to all other doors on the dock. According to Bartholdi and Gue [48], this is the simplest and most conservative way to measure the convenience of a door. Given the parameters defined in Table 4.6, physical location l_m of door m is calculated as follows:

$$l_m = \frac{\sum_{m' \in M \setminus \{m\}} d_{mm'}}{|M| - 1} \quad (4.10)$$

To calculate the fitness of a door relative to a given trailer, the availability time of the door is added to the cumulative travel times of the pallets exchanged between the trailer and those of its successors and predecessors whose assigned door is known at the time of assigning the trailer itself. Fitness $f_{M_{[j]}}$ of door $M_{[j]}$, candidate for the assignment to trailer j , is computed according to the following formula:

$$f_{M_{[j]}} = t_{M_{[j]}} + \sum_{j' \in S_j \cap J^K} n_{jj'} \cdot d_{M_{[j]}M_{[j']}} + \sum_{j' \in P_j \cap J^K} n_{j'j} \cdot d_{M_{[j']}M_{[j]}} \quad (4.11)$$

In case no exact assignment information is available, the value representing the physical location of the door is multiplied by the total exchanged pallets of the trailer. The resultant value (which is measured in time because distance and time are interchangeable) is added to the availability time of the door to form its fitness. The following formula analytically describes how fitness $f_{M[j]}$ of door $M[j]$, candidate for the assignment to trailer j , is calculated in that case:

$$f_{M[j]} = t_{M[j]} + e_j \cdot l_{M[j]} \quad (4.12)$$

The door acquiring the smallest value scores the best fitness and is chosen for the assignment to the trailer. Regardless of which formula is invoked to compute the fitness, the following measure is used to break ties when choosing the best door:

$$t_m^N = \sum_{m' \in N_m^C} t_{m'} \quad (4.13)$$

While calculating the fitness of door m , the availability times of the doors in its neighborhood N_m^C are also recorded². The size of the neighborhood is problem dependent and is equal to the average size of the clusters defined between the trailers (see Section 4.2.1 for the definition of clusters). For example, if the average cluster size is seven, then the door's neighborhood consists of its seven closest doors. The values are then added (denoted as t_m^N for door m) and the door scoring the smallest is the one chosen for the assignment. In case the tie is still persistent, the door with the lowest ID will be chosen.

Note that Eq. (4.11) is used only if the assigned door of at least one successor or predecessor of the trailer is known; otherwise, Eq. (4.12) is invoked for computing the fitness. Also note that at the time a trailer is assigned, the fitness construction includes only those doors whose availability time is known. The availability time of a door can be measured only if the release time of its latest assigned trailer is known. The release time

²In case any door $m' \in N_m^C$ is not free and still assigned to a trailer, the assignment time of the trailer is considered its availability time.

of a trailer depends on the location of the doors to which the predecessors of the trailer are assigned. Consequently, if no exact information is available about the assigned door of at least one of the predecessors, it is not possible to calculate the release time of the trailer, and hence, the availability time of its assigned door.

It is also important to explain why forklift availability times are not included in the computation of the fitness function. The main reason is that the fitness of a door, for a given trailer, does not depend on the availability time of the forklift dedicated to the door but rather depends on the availability times of a number of forklifts that move the pallets loaded onto the trailer from a number of origin doors. A direct inclusion of these values in the fitness function would bias the fitness from its main element which is the door availability time. On the other hand, an indirect inclusion demands a more complicated and exhaustive analysis beyond the scope of this research.

Having assigned the trailer to the door, one needs to schedule the movement of both the pallets unloaded from the trailer and those loaded onto the trailer. For the latter, the scheduling is straightforward. Given the order it is unloaded, each pallet is moved by the forklift dedicated to its origin door. For the former, however, the situation is different as all the pallets³ have to be moved by one particular forklift. Since the pallets might have different destinations, a scheduling policy is required to order their movements. The suggested strategy is to prioritize the movement of the pallets according to the completion status of their recipient trailers. The completion status of a trailer is assessed by the latest position that one of its predecessors holds in the sequence list. The recipient trailers are then sorted in ascending order of their completion status. This means that a trailer which acquires a smaller value receives its required pallets earlier than another with a larger value. The smallest possible value belongs to one (or more) recipient trailer(s) whose last predecessor is the currently assigned trailer. Accordingly, as soon as the trailer receives

³This includes only those pallets which are ready to be moved. That is, their recipient trailers have already been assigned to a door.

its required pallets, it can complete its final loading operations and releases its assigned door. It is expected that the proposed scheduling policy shortens the completion time of the trailers, and hence, reduces the makespan. A similar expectation is also desired for the algorithm's fitness function.

Having defined the fitness of each door and the forklift scheduling policy, it is time to describe the DF assignment procedure. As the feasibility of the trailer sequence list is verified by the DRS algorithm, the assignment procedure is done in a straightforward way. For a given trailer picked from the sequence list, the algorithm finds a door with the best fitness and assigns the trailer. It then schedules the movement of the pallets both unloaded from and loaded onto the trailer and updates the availability times of the involved forklifts accordingly. The above procedure continues until all the trailers are assigned and complete their crossdocking operations. On the other hand, if the feasibility of the sequence list has not been verified by the sequencing algorithm (like DR), it is possible that DF encounters deadlock and fails to produce the final makespan. Procedure 4.3 describes the DF heuristic in pseudo codes:

Procedure 4.3 DOORFITNESS

1. **while** sequence list $\neq \emptyset$ **do**
 2. Pick the current trailer from the sequence list;
 3. **for all** m where door m is available **do**
 4. **if** no available door is found **then**
 5. Stop; {deadlock occurs}
 6. **else**
 7. Calculate the fitness of the available doors;
 8. Assign the trailer to the door with the best fitness;
 9. Schedule the forklifts that move the exchanged pallets;
 10. Update timings of the docking activities involved;
 11. **end if**
 12. **end for**
 13. **end while**
 14. **return** the makespan;
-

4.3 Experimental Results and Analysis

The proposed heuristics are coded in C++ and run on a CoreTM2 Quad 2.66GHz computer with 3GB RAM. To set up the experimental data, ten random instances are generated for each combination of the problem sizes, trailer-to-door ratios (α), freight mixed levels (FML), and uncorrelated and correlated interdependencies (this equals to $4 \times 4 \times 3 \times 2 \times 10 = 960$ total instances).

At the first attempt, the mathematical model developed in the previous chapter is used to evaluate the performance of the heuristics in finding high-quality solutions in reasonable running times. Because no constraint is imposed on the amount of space reserved in front of each dock door, the model has been simplified by eliminating its capacity variables and constraints (see Chapter 3). The MIP model is implemented in the AIMMS modeling platform [62] and solved by the ILOG CPLEX 12.1. The computer platform considered for running the MIP model is a 64-bit machine with two Intel Xeon Quad Core E5335 2.0GHz processors and 4GB RAM. The mathematical model is also used to verify the feasibility of the solutions generated by the heuristics. Solution feasibility verification requires that a given solution [produced by the heuristics] not violate any constraints imposed on the crossdocking model. In the verification process, the decision variables of the MIP model (which has been implemented in AIMMS) are taken as input parameters. The verification is then carried out by fixing the variables to the result of the heuristics.

At the second attempt, the efficiency of the DRS algorithm in establishing solution feasibility is evaluated against the DR_b sequencing heuristic. The evaluation measure adopted in the experiments is *relative difference* (*r. d.*). Given two algorithms A and B for the problem, the relative difference of the results of B from those of A is calculated according to the following formula:

$$r. d. (\%) = \frac{C_{\max}(B) - C_{\max}(A)}{C_{\max}(A)} \times 100 \quad (4.14)$$

As the problem objective is to minimize the makespan, an improvement in the solution quality (which is obtained by a new algorithm, here B) is represented by a negative value of the relative difference. Also note that when the comparison of two (or more) solution techniques is based on average results, the mean function is applied to only those instances where all the compared solution techniques could produce a feasible solution. That is, if one algorithm finds a feasible solution for a given problem instance while the other fails, the value is not included in the mean function for the former algorithm. Finally, the following terminology is adopted to present the experimental results in tables:

- **Bt:** Number of backtracks invoked during the execution of DRS
- **Rs:** Number of re-starts invoked during the execution of DRS
- **ST(s):** Time [in seconds] taken for DRS to establish solution feasibility after the last re-starting, if any
- **TT(s):** Algorithm's total runtime [in seconds]

4.3.1 Heuristics vs. MIP Model: Efficiency

The performance of the heuristics algorithms against the MIP model is evaluated for small size instances. Both solution quality and runtime are considered for the comparison. This is because crossdocking is a real-time logistics practice and operational time is as important as operational quality. Given a time limit of two hours, the CPLEX solver is able to find a feasible (but not optimal) solution only for 41 (out of 60) instances belonging to the (small:1) category. This is in fact the simplest form of the experimental data! The truck scheduling problem is to be solved in an eight-door crossdock where the number of trucks is equal to the number of doors. The sequencing procedure of trucks is straightforward as exactly one door is available for each truck.

The best results obtained by the proposed heuristic techniques (including DRS and the three variations of DR, combined with DF) are compared with those of the CPLEX solver. Table 4.7 presents the results for the two approaches. To make a fair comparison, the running times of the four heuristic combinations are added and represented as the total runtime. According to the table, heuristics exhibit much better performance in terms of both solution quality and runtime than the mathematical model for all the combinations of the freight mixed level and correlation status – excluding two instances in the (small:1;low) category. The advantage is more apparent for the higher freight mixed levels where the interdependencies of trucks are more disseminated. Note that the feasibility of the heuristics results has been verified by the mathematical model.

It can be concluded from the above observation that the current formulation is not efficient for solving this detailed scheduling problem. The MIP model requires the solver to decide on the timing values and handling orders of every single pallet in the crossdock before producing the final solution, which is the overall schedule length. Although this flexibility in time gives total freedom to find the optimal solution, at the same time, it makes it very complicated for the solver to systematically branch and cut the search space as many intricate domain indices are defined in the constraints of the model. The current formulation is not even efficient for verification purposes. The CPLEX solver could not verify the feasibility of the heuristics results for the test cases in which the number of trucks exceeds 64. Huge memory requirements and a substantial amount of time to evaluate some of the domain indices of the constraints are the main reasons for the lack of scalability. As the further work, Chapter 5 scrutinizes the applicability of a simplified, yet approximate, MIP formulation of the problem to enable benchmarking and verifying the sub-optimal solutions for larger size problem instances.

Table 4.7: The heuristics vs. the mathematical model for (small:1) instances

	CPLEX _{2-hour}	Heuristics		<i>r. d.</i> (%)	CPLEX _{2-hour}	Heuristics		<i>r. d.</i> (%)
	$C_{\max}$	$C_{\max}$	TT(s)		$C_{\max}$	$C_{\max}$	TT(s)	
	(small:1;low;uncorrelated)				(small:1;low;correlated)			
1	62	32	0.36	-48.39	170	35	0.28	-79.41
2	47	53	0.16	12.77	119	36	0.19	-69.75
3	54	46	0.17	-14.81	41	41	0.15	0
4	40	38	0.22	-5	36	30	0.11	-16.67
5	46	40	0.25	-13.04	n/a	41	0.16	–
6	40	38	0.19	-5	70	30	0.15	-57.14
7	70	46	0.14	-34.29	n/a	36	0.12	–
8	112	32	0.28	-71.43	188	41	0.15	-78.19
9	74	28	0.15	-62.16	35	46	0.15	31.43
10	123	54	0.19	-56.1	157	70	0.15	-55.41
avg.	66.8	40.7	0.21	-39.07	102	41.1	0.17	-59.68
	(small:1;medium;uncorrelated)				(small:1;medium;correlated)			
1	68	45	0.33	-33.82	110	54	0.19	-50.91
2	n/a	47	0.16	–	99	50	0.19	-49.49
3	n/a	49	0.15	–	117	40	0.16	-65.81
4	70	36	0.14	-48.57	114	42	0.19	-63.16
5	82	45	0.25	-45.12	n/a	38	0.22	–
6	144	58	0.26	-59.72	122	37	0.14	-69.67
7	n/a	58	0.16	–	105	40	0.12	-61.9
8	59	45	0.22	-23.73	111	36	0.19	-67.57
9	n/a	52	0.26	–	114	58	0.17	-49.12
10	70	43	0.26	-38.57	n/a	50	0.2	–
avg.	82.2	45.3	0.24	-44.83	111.5	44.6	0.17	-59.98
	(small:1;high;uncorrelated)				(small:1;high;correlated)			
1	54	52	0.36	-3.7	n/a	66	0.2	–
2	n/a	46	0.15	–	101	52	0.17	-48.51
3	94	41	0.22	-56.38	124	46	0.15	-62.9
4	93	40	0.19	-56.99	n/a	55	0.14	–
5	n/a	54	0.17	–	n/a	64	0.12	–
6	122	45	0.14	-63.11	147	58	0.2	-60.54
7	124	48	0.22	-61.29	91	46	0.22	-49.45
8	n/a	61	0.17	–	n/a	54	0.11	–
9	n/a	49	0.16	–	n/a	53	0.16	–
10	n/a	53	0.11	–	n/a	40	0.22	–
avg.	97.4	45.2	0.22	-53.59	115.7	50.5	0.19	-56.37

4.3.2 DRS vs. DR Algorithms

DRS is compared with DR_b by first varying the value of α and keeping FML low. This allows the investigation of the complexity of the truck scheduling problem solely relative to the trailer-to-door ratio. The comparison will then continue for more difficult problem

instances of the higher mixed levels. Since the comparison of the two algorithms in terms of solution quality is based on the value of the makespan, it is required that DR_b (in addition to DRS) be integrated with the DF heuristic. In that case, the average cluster size (which is used in the fitness function of DF) is set to the average number of successors and predecessors of the trailers.

4.3.2.1 Trailer-to-Door Ratio (α)

Table 4.8 shows the average results for each combination of the problem sizes, trailer-to-door ratios, and uncorrelated and correlated interdependencies relative to the low freight mixed level (see Appendix A for the full coverage of the results). The table indicates that for the majority of instances representing large crossdocks with short queues of waiting trailers (the latter is specified by the value of α), DR_b exhibits a better performance in solution quality than DRS – see the positive values of *r. d. (%)*. The advantage is mainly due to the sequencing strategy of DR_b which is based on the repeated sequencing of trailers and their predecessors. It is thus more probable that the trailers are assigned to adjacent doors in the assignment phase, leading to more savings in pallet moving times and hence, the overall makespan (this is more significant for large crossdocks). Moreover, one should not forget that DR_b benefits from some sort of diversification (by combining the three versions of the DR algorithm) which may help to improve solution quality.

On the contrary, as α increases and the problem becomes more complex, DRS performs better in terms of solution quality than DR_b – see the negative values of *r. d. (%)*. The reason should be sought in the DRS sequencing scheme. DRS aims at constructing a chain of sequenced trailers in which each trailer is highly interdependent on both its direct and indirect neighbors. Thus, irrespective of the number of total trailers, a neighborhood of tightly interdependent trailers can be found in any part of the sequence list that have been assigned to a number of adjacent doors in the crossdock. The accumulated savings gradually outweigh those of DR_b as the value of α increases.

Table 4.8: DRS vs. DR_b when FML is kept low and α varies

Data Category	DR_b+DF		DRS+DF				<i>r. d. (%)</i>
	$C_{\max}$	TT(s)	$C_{\max}$	TT(s)	Bt	Rs	
(small:1;low;uncorrelated)	53.1	0.08	42.8	0.13	0	0	-19.4
(small:1;low;correlated)	52	0.07	47.4	0.09	0	0	-8.85
(small:2;low;uncorrelated)	110.5	0.1	111.7	0.13	0	0	1.09
(small:2;low;correlated)	110.7	0.1	110.1	0.13	0	0	-0.54
(small:4;low;uncorrelated)	269.6	0.18	262.3	0.21	0	0	-2.71
(small:4;low;correlated)	285.1	0.18	278.9	0.21	0	0	-2.17
(small:8;low;uncorrelated)	856.9	0.26	790.4	0.29	0.1	0	-7.75
(small:8;low;correlated)	838.2	0.25	797.1	0.31	0.2	0	-4.91
(medium:1;low;uncorrelated)	135.8	0.26	165.4	0.16	0	0	21.8
(medium:1;low;correlated)	126.9	0.24	139.4	0.15	0	0	9.85
(medium:2;low;uncorrelated)	257.9	0.43	260.3	0.3	0	0	0.93
(medium:2;low;correlated)	255.1	0.46	274.5	0.32	0	0	7.6
(medium:4;low;uncorrelated)	555.9	0.87	546.9	0.68	0	0	-1.62
(medium:4;low;correlated)	550.7	0.83	537.8	0.69	0	0	-2.34
(medium:8;low;uncorrelated)	1481.9	1.73	1313	1.98	0	0	-11.4
(medium:8;low;correlated)	1700.6	1.7	1371.9	1.98	0	0	-19.33
(large:1;low;uncorrelated)	404.3	1.25	498.6	0.67	0	0	23.32
(large:1;low;correlated)	452.8	1.25	475.8	0.68	0	0	5.08
(large:2;low;uncorrelated)	699.9	2.46	731.9	1.53	0	0	4.57
(large:2;low;correlated)	732.9	2.46	738.7	1.67	0	0	0.79
(large:4;low;uncorrelated)	1361.4	4.85	1442.6	4.56	0	0	5.96
(large:4;low;correlated)	1386.9	4.85	1381.7	4.55	0	0	-0.37
(large:8;low;uncorrelated)	3635.9	9.67	3042.1	14.39	0	0	-16.33
(large:8;low;correlated)	3396	9.73	3133.1	14.94	0	0	-7.74
(huge:1;low;uncorrelated)	1240.7	7.09	1669	3.73	0	0	34.52
(huge:1;low;correlated)	1270.8	7.08	1388.6	3.85	0	0	9.27
(huge:2;low;uncorrelated)	2008.1	14.23	2299.6	9.74	0	0	14.52
(huge:2;low;correlated)	2103.7	14.06	2291.6	10.46	0	0	8.93
(huge:4;low;uncorrelated)	3621.4	28.56	3877.9	29.54	0	0	7.08
(huge:4;low;correlated)	3593.2	28.36	3829.6	30.43	0	0	6.58
(huge:8;low;uncorrelated)	8749.8	57.29	7972.7	103.6	0	0	-8.88
(huge:8;low;correlated)	8713.6	57.64	7790.3	103.5	0	0	-10.6

It is observed that both heuristics maintain scalability in runtime when the problem size increases to huge scales. For the extreme case of scheduling over 2000 trailers in a 256-door crossdock, the total runtime of the heuristics does not exceed two minutes. It should also be noted that DR_b fails to generate a feasible trailer sequence, relative to the given number of dock doors, for one instance in the (small:8;low;uncorrelated) data category and two instances in the (small:8;low;correlated) category – see Appendix A.

4.3.2.2 Freight Mixed Level (FML)

As mentioned in Section 4.1, the truck scheduling problem studied in this thesis might not have a feasible solution for the cases where $\alpha > 1$. The experimentation conducted in the previous section, however, shows that establishing solution feasibility is not an issue when the freight mixed level is kept low, even for $\alpha = 8$. Accordingly, the investigation on the problem feasibility should be carried out for more difficult instances of the higher mixed levels. The infeasibility of a problem with respect to a given data set can be proven either analytically or experimentally by exploring the entire solution space. The latter approach applies here as one can make use of the DRS algorithm, which is inherently an exhaustive search technique⁴, to decide on the problem feasibility. Since the solution space of the truck scheduling problem is exponentially increased by linearly increasing the number of trailers and doors⁵, DRS is unable to search the entire solution space in a reasonable amount of time to confirm infeasibility. Alternatively, a *miniature* test case of scheduling 4, 8, 16, and 32 trucks in a four-door crossdock is generated using the same methodology introduced in Chapter 3 to enable DRS to explore the whole solution space within moderate running times.

For each combination of the trailer-to-door ratios, freight mixed levels, and uncorrelated and correlated interdependencies, i.e., $4 \times 3 \times 2 = 24$ combinations, 200 random instances are generated and the number of feasible ones are recorded. Table 4.9(a) shows the percentage of the feasible solutions for each combination. The results indicate that the product of the large trailer-to-door ratios and the higher freight mixed levels lead to infeasibility. This is because in such conditions dock doors are critical resources which are normally exhausted by some assigned trailers that due to high levels of interdependencies cannot complete their operations and release their assigned door.

⁴DRS exhaustively explores the search space until it finds a feasible solution.

⁵Note that the problem is NP-hard.

Table 4.9: Percentage (in %) of feasibility for

(a) Miniature instances (guaranteed by DRS)

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	93.5
	correlated	100	100	100	98.5
medium	uncorrelated	100	100	10	0
	correlated	100	100	13	0
high	uncorrelated	100	92	0	0
	correlated	100	98	0	0

(b) Small to huge instances (found by DR_b & DRS)

FML	Trailer-to-Door Ratio	Correlation Status	DR _b	DRS
low	$\alpha = 8$	uncorrelated	97.5	100
		correlated	95	100
medium	$\alpha = 4$	uncorrelated	0	5
		correlated	0	15
high	$\alpha = 2$	uncorrelated	42.5	90
		correlated	57.5	100

According to the table, the infeasibility region is formed by the combination of $\alpha = 8$ and both the medium and high mixed levels, and $\alpha = 4$ and the high mixed level (represented in bold **0**s in the same table). Note that feasibility is poor when $\alpha = 4$ and FML is medium (only over 10 percent of the instances have feasible solutions) and is not guaranteed when $\alpha = 2$ and FML is high, and $\alpha = 8$ and FML is low. It can be claimed that a similar allocation of feasible and infeasible regions, with comparable percentage, is applicable to larger size instances as first, the scaling is proportional and second, all data are generated according to an identical generation procedure. The above generalization is strongly supported by the results obtained by the DRS algorithm.

To demonstrate how robust DRS is in avoiding deadlock and reaching feasibility with respect to larger size instances, the percentage of feasible solutions found by the DRS algorithm in a time limit of two hours is presented in Table 4.9(b). The table includes the aggregate results from small to huge size problems for the six combinations of the trailer-to-door ratios, freight mixed levels, and uncorrelated and correlated interdependencies

where 100% feasibility is not guaranteed. The comparable performance of the DRS algorithm in establishing solution feasibility for various characteristics of input data is clearly affirmed by comparing the results of Tables 4.9(a) and 4.9(b). Table 4.9(b) also shows that DRS is more efficient than DR_b in generating feasible solutions. The DR_b algorithm severely lacks robustness in avoiding deadlock and finding feasible solutions with respect to input data. This is because DR_b is a single-pass heuristic and cannot capture the heterogeneity of input data to establish solution feasibility. There are totally 47 instances where DR_b is not able to find a solution while DRS is. (For more information, see Appendix A for the percentage of feasible solutions found by both DRS and DR_b algorithms relative to each individual problem size.)

Lastly, to better illustrate the robustness of DRS, in establishing solution feasibility relative to DR_b , Figure 4.3 presents the aggregate results (i.e., percentage of feasible solutions) from small to huge size problems for the three combinations of the trailer-to-door ratios and freight mixed levels, where 100% feasibility is not guaranteed, and compares them with those of miniature instances. The numbers strongly support the claim that the DRS algorithm performs comparably in finding feasible solutions relative to the different combinations of the trailer-to-door ratios and freight mixed levels.

Tables 4.10 and 4.11 show the results for the individual problem instances from small to huge sizes when $\alpha = 2$ and FML is high, and when $\alpha = 4$ and FML is medium, respectively. The results demonstrate that the combination of the ranking methodology and backtracking mechanism embedded in DRS works efficiently. DRS can find a feasible solution for the 76 (out of 80) instances listed in Table 4.10 whereas DR_b did it for only 40 instances. Likewise in Table 4.11, while DR_b is unable to find a feasible solution for any instance, DRS can find for eight instances within the two hour time limit. Although DRS is designed to exhaustively search the solution space until it finds a feasible trailer sequence, it does it efficiently. Consider all the 47 data instances where DR_b is not able

Figure 4.3: DRS performs comparably in establishing solution feasibility for various characteristics of input data

to produce a feasible solution while DRS is (see Tables 4.10, 4.11, and Appendix A), for almost 66% of them, i.e., 31 out of 47, the number of backtracks does not exceed 60.

It can be seen from Tables 4.9, 4.10, and 4.11 that the percentage of feasible solutions is slightly higher for correlated problem instances. This is because correlation makes the interdependencies of the trailers more clustered so it is more probable to find a feasible solution based on a systematic scheduling of trailers in each cluster – as it is employed in the heuristics. The above observation implies that for those real world case studies where the level of correlation in trailer interdependencies is higher than that considered in this work, the proposed DRS algorithm performs efficiently in finding feasible solutions. In summary, truck scheduling solution feasibility becomes a concern when both the level of trailer interdependencies and the ratio of the number of trailers to the number of doors increase. Finally, in terms of solution quality, it is observed that DRS performs considerably worse than DR_b for some instances, e.g., see sample 6 in the (large:2;high;correlated) category in Table 4.10 where the relative difference is over 100%. The further work on improving the quality of the solutions generated by the DRS+DF algorithms will be presented briefly in Section 4.3.3.2 and in detail in Chapter 5.

Table 4.10: DRS vs. DR_b when $\alpha = 2$ and FML is high

DR _b +DF		DRS+DF				$r. d.$		DR _b +DF		DRS+DF				$r. d.$	
C_{max}	TT(s)	C_{max}	ST(s)	TT(s)	Bt	Rs	(%)	C_{max}	TT(s)	C_{max}	ST(s)	TT(s)	Bt	Rs	(%)
(small:2;high:uncorrelated)															
1	n/a	0.11	265	60.4	40761	0	–	n/a	0.09	305	–	0.19	6	0	–
2	n/a	0.05	n/a	1935	1153285	15	–	n/a	0.06	281	–	0.28	52	0	–
3	n/a	0.09	257	27.1	16352	0	–	n/a	0.05	353	–	0.2	4	0	–
4	n/a	0.05	n/a	1936	1200698	15	–	n/a	0.08	322	–	54.9	32701	0	–
5	n/a	0.05	305	0.17	5	0	–	291	0.16	307	–	1.84	1068	0	5.5
6	n/a	0.05	375	54.2	32708	0	–	254	0.16	335	0.02	121	71997	1	31.89
7	n/a	0.05	n/a	1936	1185367	15	–	n/a	0.08	264	8.52	1339	831752	11	–
8	n/a	0.05	305	0.23	48	0	–	n/a	0.08	332	–	0.25	4	0	–
9	n/a	0.05	n/a	1936	1107601	15	–	n/a	0.06	251	–	0.17	0	0	–
10	n/a	0.08	367	0.22	10	0	–	n/a	0.08	295	–	0.17	0	0	–
avg.	–	–	–	–	–	–	–	272.5	0.09	321	0.02	61.23	37074	0.5	17.8
(medium:2;high;uncorrelated)															
1	n/a	0.47	1429	8.64	1097	0	–	n/a	0.42	1210	–	0.98	19	0	–
2	1052	0.67	971	1.61	144	0	-7.7	n/a	0.41	1191	–	1.11	28	0	–
3	n/a	0.39	1283	547.3	78227	3	–	n/a	0.36	881	–	0.72	1	0	–
4	n/a	0.41	1074	0.98	6	0	–	813	0.72	1313	–	0.81	5	0	61.5
5	n/a	0.37	1062	0.73	1	0	–	792	0.87	991	–	1	1	0	25.13
6	n/a	0.42	1308	1.16	18	0	–	n/a	0.44	1080	–	1.14	34	0	–
7	n/a	0.37	907	0.84	5	0	–	n/a	0.36	1033	–	0.81	4	0	–
8	n/a	0.36	1125	0.84	24	0	–	n/a	0.36	1436	–	39.98	5849	0	–
9	n/a	0.39	1136	1.06	30	0	–	846	0.75	1354	–	0.97	27	0	60.05
10	n/a	0.36	760	0.75	0	0	–	n/a	0.42	1243	–	1.01	26	0	–
avg.	1052	0.67	971	1.61	144	0	-7.7	817	0.78	1219.3	–	0.93	11	0	49.24
(large:2;high;uncorrelated)															
1	2520	3.39	2704	5.76	0	0	7.3	3917	3.22	2179	–	5.81	0	0	-44.37
2	n/a	2.08	3426	6.22	26	0	–	2294	3.23	1916	–	5.83	0	0	-16.48
3	n/a	2.08	2019	5.66	0	0	–	3920	3.3	3765	–	5.83	3	0	-3.95
4	3212	3.37	2171	5.89	0	0	-32.41	3103	3.25	2264	–	5.73	0	0	-27.04
5	3572	2.51	3941	5.98	15	0	10.33	2673	3.25	2052	–	5.53	0	0	-23.23
6	2705	3.109	2348	5.44	0	0	-13.2	2309	3.31	4699	–	6.11	18	0	103.5
7	n/a	2.2	2690	5.81	3	0	–	4095	2.59	2657	–	5.66	0	0	-35.12
8	2465	3.09	2870	5.59	0	0	16.43	3322	3.45	1889	–	5.64	0	0	-43.14
9	2746	3.3	2381	5.87	0	0	-13.29	n/a	2.05	4432	–	6.48	52	0	–
10	n/a	2.12	4365	6.81	37	0	–	n/a	2.16	1761	–	6.11	0	0	–
avg.	2870	3.13	2736	5.76	2.5	0	-4.67	3204	3.2	2678	–	5.77	2.6	0	-16.43
(huge:2;high;uncorrelated)															
1	7887	15	11614	56.2	13	0	47.25	7809	16.2	5988	–	51.8	0	0	-23.32
2	11665	14.4	9532	54.1	3	0	-18.28	6797	16.6	5230	–	52.3	0	0	-23.05
3	5818	16.9	5145	51.9	0	0	-11.57	9273	16.5	10749	–	54.2	10	0	15.92
4	7127	16.2	4742	52.6	0	0	-33.46	7036	16.2	6801	–	53.9	0	0	-3.34
5	6740	16.5	6369	52	0	0	-5.5	7205	16.5	5476	–	53.4	0	0	-24
6	7341	16.1	5626	53.7	0	0	-23.36	6476	16.5	5298	–	53.4	0	0	-18.19
7	8538	15.8	8018	53	1	0	-6.09	10081	15.8	5246	–	51.6	0	0	-47.96
8	6342	16.4	6300	52.4	0	0	-0.66	8542	16.1	6687	–	53.6	0	0	-21.72
9	8206	15.7	5968	51.4	0	0	-27.27	7414	15.6	4971	–	53.8	0	0	-32.95
10	7199	16.2	7522	52.9	0	0	4.49	12053	15.4	6012	–	51.8	0	0	-50.12
avg.	7686.3	15.9	7084	53	1.7	0	-7.84	8269	16.1	6246	–	53	1	0	-24.5

Table 4.11: DRS vs. DR_b when $\alpha = 4$ and FML is medium

DR _b +DF		DRS+DF				
$C_{\max}$	TT(s)	$C_{\max}$	ST(s)	TT(s)	Bt	Rs
(small:4;medium;uncorrelated)						
—	—	—	—	—	—	—
(medium:4;medium;uncorrelated)						
2	n/a	0.42	3414	2.36	4708	461173
—	—	—	—	—	—	—
(large:4;medium;uncorrelated)						
5	n/a	2.34	17385	64.11	969.4	29429
—	—	—	—	—	—	—
—	—	—	—	—	—	—
(huge:4;medium;uncorrelated)						
—	—	—	—	—	—	—

DR _b +DF		DRS+DF				
$C_{\max}$	TT(s)	$C_{\max}$	ST(s)	TT(s)	Bt	Rs
(small:4;medium;correlated)						
4	n/a	0.08	792	0.09	121.3	43988
(medium:4;medium;correlated)						
2	n/a	0.41	3630	2.28	364.9	36962
3	n/a	0.41	2810	2.05	544.8	54859
(large:4;medium;correlated)						
3	n/a	2.39	16362	54.55	2465	76879
4	n/a	2.31	18341	55.56	359	11097
9	n/a	2.37	17322	32	6355	197296
(huge:4;medium;correlated)						
—	—	—	—	—	—	—

4.3.3 Discussion on DRS Efficiency

The experimental results indicate that the DRS ranking methodology on its own cannot maintain a deadlock-free search path at all times and requires a backtracking mechanism to adjust it. In other words, the occasional failure of the methodology in avoiding deadlock is compensated by backtracking. The frequency of the failures, however, varies from one occasion to another, implying that the ranking mechanism is dependent on the input data. (This is in fact a complication generic to constructive heuristics which, due to their single-pass behavior, cannot capture the heterogeneity of their input data instances thoroughly to guarantee feasibility or optimality.) The problem with recurring backtracks, as a result of frequent failures, is the huge overhead incurred on the algorithm runtime, impeding its capability to systematically find a feasible solution within a specified time limit, and hence, impairing its robustness. The following section thus introduces the mechanism deployed in the experiments to maintain a robust DRS and verifies its efficiency.

4.3.3.1 Robustness

It was observed in the conducted preliminary experiments that the choice of the initial cluster significantly affects the DRS performance to avoid deadlock and to build a feasible sequence of trailers. A poor initial cluster causes the algorithm to construct the sequence list from an improper collection of trailers, leading to recurring entrapment in deadlock. Consequently, a large number of backtracks are required to fix the constructed sequence list. In the worst case, no feasible sequence list can be found with respect to the given initial cluster and the algorithm must initiate a *re-start* to resume the sequencing procedure from a new initial cluster.

To loosen the negative impact of a poor initial cluster, and hence, reduce the total runtime, the length of time where DRS is searching the solution space, after every selection of the initial cluster, is limited to a time when the number of backtracks is less than

a certain limit. (The limit is varied from 1000 to 50000 backtracks in the experiments.) Whenever the solution time exceeds the assigned limit, the algorithm stops searching and initiates re-starting to choose the next best initial cluster according to the ranking and selection criteria defined earlier. The sequencing process is then resumed and the solution time is monitored again. This continues until a feasible sequence list is constructed, or the two hour time limit expires, or no more cluster is left to be selected as the initial cluster.

The proposed amendment is very effective. Regarding those problem instances where at least one re-start is required till a feasible sequence list is constructed, the solution time after the last re-starting, denoted by $ST(s)$, is considerably smaller than the given time limit equivalent to the assigned maximum number of backtracks – see Tables 4.10 and 4.11. (Note that $ST(s)$ is defined in case at least one re-start is invoked throughout the solution construction.) It is also worth noting that there are four samples in the (small;2;high;uncorrelated) category in Table 4.10 where DRS is unable to find a feasible solution after examining each of the total 16 clusters in the given time limit, respectively.

Although DRS proved to be robust in avoiding deadlock and generating feasible solutions, the imposed timing restrictions no longer ensure that the algorithm guarantees infeasibility in the case it does not find a feasible solution. For example, the average number of clusters examined by DRS within the assigned time limits is 24 and 12 for large and huge size instances, respectively. This is considerably fewer than the potential number of initial clusters (which could be as many as 768 for large problems and 2048 for huge ones). As noticed in Section 4.3.2.2, finding a feasible solution is a bottleneck when $\alpha = 4$ and FML is medium. In this data category, the number of initial clusters is 384 and 1024 for large and huge size instances, respectively, implying that the DRS algorithm is able to check only 6.2% and 1.2% of their total number of clusters. This means that the robustness of the algorithm is heavily dependent on the ranking methodology developed in Section 4.2.1 for choosing the initial cluster.

To evaluate the efficiency of the proposed cluster ranking methodology in choosing a good initial cluster, a purely random selection method was applied to uniformly choose the initial cluster among all the candidate ones for both large and huge size instances where $\alpha = 4$ and FML is medium. It is interesting to mention that at the first round, the random method even failed to produce a feasible solution for the three instances in the (large:4;medium;uncorrelated) and (large:4;medium;correlated) categories in the two hour time limit – this already confirms that the cluster ranking methodology is fairly efficient. Nevertheless, after the next nine rounds⁶, the random approach was able to find a feasible solution within the two hour time limit for two more instances (where the rank-based method could not), one in the (large:4;medium,uncorrelated) data category and the other in the (huge:4;medium,correlated) category. The DRS current performance in establishing solution feasibility can be further improved by an *intelligent* randomization of the cluster selection step. The concept is introduced as the future work in Chapter 6.

4.3.3.2 Solution Quality

Not only does the initial cluster influence the robustness of the DRS algorithm in establishing solution feasibility, it also affects the quality of the final solution produced. An experimentation is conducted to evaluate how much improvement in the solution quality would be obtained if the initial cluster changes. The experimental data consist of those random instances where the relative difference of the DRS solution results from those of DR_b exceeds 20% (totally 102 instances). The DRS+DF heuristics are then run ten times per instance where, each time, one cluster among the best ten clusters is selected as the initial cluster. (The first ten clusters acquiring the highest ranks form the best ten clusters.) Of the ten solutions generated, the smallest one, i.e., the best one, is recorded afterwards. The amount of improvement (i.e., minimization) for each random instance is scaled relative to the original value according to Eq. (4.14).

⁶The random selection method was run for ten times.

The experimental results indicate that the amount of improvement is absolutely significant for most of the instances. Over -20% relative difference in the solution quality has been obtained for nearly 70% (71 out of 102) of the test instances. The highest reduction is reported as 57.67% and the lowest is 0%. (The latter belongs to the (small:1) category where the number of initial clusters is at most eight.) The average reduction is equal to 26.46% (the total average is 29.46%) and the average deviation is 9.37%. The experimental results for 12 instances where the reduction exceeds 40% are shown in Table 4.12. The instances are sorted in descending order of their improvement. The table also presents some useful statistics on the results obtained from the whole 102 instances. By looking at the number of backtracks (Bt) required to adjust the search path for both the original and improved versions of DRS, it is inferred that choosing the best initial cluster has enabled the sequencing procedure to construct the feasible solutions systematically based on the ranking and selection criteria proposed. As the second last row of the table shows, the total number of backtracks for the entire 102 instances reduces from 73146 to only 5 times in the case of constructing the sequence list from the best initial cluster. Note that the number of re-starts (Rs) in the Improved DRS[+DF] column of Table 4.12 can be used to identify which cluster has led to the best solution. For example, if the number of re-starts is zero it means that the initial cluster is the first best cluster. If the number is 9 it means that the initial cluster is the tenth best cluster and so on.

As a comparison, the DRS+DF solution results, including the original and the improved ones, are evaluated against those of the DR_b +DF heuristics. The last two columns of Table 4.12 indicate that the improved DRS outperforms DR_b on average. (See the positive and negative values of *r. d. (%)* for both the original and improved versions of DRS.) All of this implies that if DRS is augmented to a re-starting heuristic in which the solution construction is carried out iteratively from a number of clusters as the initial points, it is expected that much improvement would be attained in terms of deadlock avoidance (represented by the number of backtracks), solution quality, and runtime.

Table 4.12: DRS efficiency in the solution quality w.r.t. the initial cluster

Data Category	#	DR _b [+DF]		Original DRS[+DF]				Improved DRS[+DF]						DRS Improvement		
		$C_{\max}$		$C_{\max}$	Bt	Rs		$C_{\max}$	ST(s)	Bt	Rs			$r.d.(%)$	$r.d.(%)$ r.t. DR _b	
															Org.	Imp.
(large:2;high:correlated)	6	2309		4699	18	0		1989	4.45	0	7			-57.67	103.5	-13.86
(medium:1;medium:correlated)	9	182		296	0	0		143	0.09	0	2			-51.69	62.64	-21.43
(huge:1;low:uncorrelated)	9	1410		2154	0	0		1117	1.22	0	4			-48.14	52.77	-20.78
(medium:1;low:uncorrelated)	3	115		203	0	0		108	0.03	0	2			-46.8	76.52	-6.09
(medium:2;high:correlated)	4	813		1313	5	0		731	0.55	0	4			-44.33	61.5	-10.09
(small:1;medium;uncorrelated)	6	58		70	0	0		39	0.01	0	4			-44.29	20.69	-32.76
(large:1;high:correlated)	9	736		1356	0	0		757	1.12	0	2			-44.17	84.24	2.85
(huge:1;low;uncorrelated)	2	1067		1991	0	0		1117	1.41	0	2			-43.9	86.6	4.69
(small:1;low;correlated)	5	41		73	0	0		41	0	0	6			-43.84	78.05	0
(large:4;low;uncorrelated)	10	1464		2018	0	0		1167	2.3	0	2			-42.17	37.84	-20.29
(large:1;low;uncorrelated)	7	346		622	0	0		361	0.22	0	7			-41.96	79.77	4.33
(huge:2;low;correlated)	10	2124		3296	0	0		1942	4.69	0	8			-41.08	55.18	-8.57
sum (for the total 102 instances)		93450		126714	73146	—		89381	—	5	—			-29.46	35.59	-4.35
average (for the total 102 instances)		916.2		1242.3	—	—		876.3	2.89	—	—			-29.46	35.59	-4.35

4.4 Summary

This chapter developed an algorithmic approach to establish solution feasibility for truck scheduling problem instances of various types and difficulty levels that could be readily implemented in an industrial setting. The proposed solution approach was a heuristic algorithm comprised of two phases. In the first phase, a heuristic search was deployed to construct a feasible sequence of trucks for the assignment to dock doors. In the second phase, a rule-based heuristic was implemented to assign each sequenced truck to a proper dock door so that significant savings in the truck schedule length could be achieved. Such an assignment is subject to a limited number of forklifts equal to the total number of doors.

Extensive experiments were conducted to evaluate the efficiency of the algorithm in terms of deadlock avoidance and solution quality. The evaluation was carried out against the solutions generated by the mixed integer programming (MIP) model of the problem and the three variations of a dependency ranking (DR_b) constructive heuristic developed by Shakeri et. al [36] for a similar truck scheduling problem. The findings, when compared by the percentage of feasible solutions guaranteed by DRS for a very small subset of instances and also the solutions generated by the other two approaches, demonstrated that DRS performs comparably in establishing solution feasibility for various characteristics of input data. This includes the problem size, trailer-to-door ratio, freight mixed level, and correlation status. Furthermore, DRS+DF produced better solutions than DR_b +DF for more complicated problems in which the ratio of the number of trucks to the number of doors is high.

It was also observed that the DRS heuristic can be systematically augmented to a re-starting algorithm to generate a pool of feasible solutions with diverse attributes. The overall solution quality can be further improved if a neighborhood search method such as local search (LS) is deployed to optimize each individual solution in the pool. The truck

scheduling solution optimality via improving the DRS+DF solutions will be addressed in detail in the next chapter.

Chapter 5

Solution Quality Improvement

Efficient truck scheduling is a prerequisite to attain fast product flows in the crossdock, as the hub of a supply chain network. This chapter thus focuses on improving the solutions generated by the DRS+DF algorithm proposed in the previous chapter. Strongly supported by the related work on truck scheduling – see Chapter 2 – meta-heuristics (and their hybrid variants) have been a superior alternative to exact (optimal) algorithms in terms of solution quality improvement. They proved to be very efficient for finding high-quality solutions to an ever-growing number of complex, ill-defined real world problems, in particular, combinatorial ones (see [65] and [66]). Solution quality improvement is carried out by developing two meta-heuristics: tabu search (TS) and variable neighborhood search (VNS).

Being a neighborhood search method, each algorithm is supplemented by a fractional evaluation of the neighborhood so that the runtime is more efficiently dedicated to solution optimization. Extensive experiments are conducted to evaluate the efficiency of the proposed algorithms in terms of solution quality and runtime. The experimentation starts with tuning the control parameters of both meta-heuristics with the aim of balancing the trade-off between solution quality and runtime. The two algorithms are then deployed to improve the solutions generated by the DRS+DF heuristics. The impact of enhancing the initial solution by re-starting DRS on the performance of the algorithms

is also evaluated. As the development methodology enforces an optimization framework to be set up for the problem, the first section discusses how a solution can be represented and perturbed on the quest for better solutions.

5.1 Optimization Framework

In their book, Glover and Kochenberger [65] introduce meta-heuristics as solution methods that orchestrate an interaction between local improvement procedures and higher level strategies to create a process which is capable of escaping local optima and performing a robust search of a solution space. This implies that for developing a meta-heuristic algorithm, not only is a *search space* required to be defined for the problem, but also one needs to configure a *neighborhood structure* to realize the local moves.

5.1.1 Search Space

The search space to a problem is defined and formed by the attributes of its solutions. Each attribute addresses one or more decision variables of the problem. Given its formulation described in Section 3.2 of Chapter 3, a solution to the truck scheduling problem entails the following three attributes:

- (i) the sequence of trailers for the door assignment
- (ii) the assigned doors of trailers
- (iii) the moving order of pallets to their destination trailers

In the two-phase heuristic algorithm (i.e., DRS+DF) proposed in Chapter 4, the first attribute is determined in the first phase (by the DRS algorithm) while the last two are specified in the second phase (by the DF algorithm) – see Figure 5.1. The inclusion of all the three attributes leads to an extremely complex search space. Alternatively, this optimization framework restricts the search space to represent only the first attribute

Figure 5.1: The solution structure of the two-phase DRS+DF heuristic algorithm to determine the three decision variables of the truck scheduling problem

which determines the feasibility of the truck scheduling problem as well. This means that the search space is traversed from one solution to another by altering the sequence of the trailers for the door assignment. The quality of each visited solution is evaluated by the DF heuristic. Figure 5.2(a) illustrates the truck scheduling optimization framework by depicting how a solution is represented and evaluated in the search space. According to the figure, a solution s is represented by a sequence list of n trailers. The quality of the solution is assessed by a fitness function that determines the assigned doors of the trailers (according to their position in the sequence list) and schedules the movement of unloaded pallets to their destination trailers. The evaluation measure is the makespan, as the objective function $f(s)$ of solution s . (Note that the fitness function is the DF heuristic.)

As seen in Figure 5.2, a simple notation is adopted in this chapter to refer to any element(s) of an array or list in the related figures, equations and pseudo codes. Given an array or list X with k elements, $X_{[i]}$ ($1 \leq i \leq k$) refers to the element located at the position i of X whereas $X_{[i..i']}$, where $1 \leq i < i' \leq k$, refers to the elements stored from the position i to position i' .

5.1.2 Neighborhood Structure

To generate a neighbor of a given solution s , “adjacent pair wise interchange” neighborhood structure – the neighborhood is denoted as $N(s)$ – is applied. In this method,

Figure 5.2: (a) How a solution s is represented and evaluated in the search space (b) a new solution s' in neighborhood $N(s)$ is produced by interchanging two adjacent trailers b and c located at positions i and $i + 1$ of solution s

the positions of two consecutive trailers in the current solution are exchanged with each other. See Figure 5.2(b) where two trailers b and c located at positions i and $i + 1$ of solution s , respectively, are interchanged to form solution s' . As the resultant solution must be evaluated by the DF heuristic, the change in the assignment order of the two trailers may lead to a better or worse makespan. So the neighborhood of a solution s includes all the solutions that have been produced by interchanging the positions of two adjacent trailers in solution s . This neighborhood structure is adopted in accordance with the nature of the problem. It is a rule of thumb that efficient scheduling requires that interdependent trailers be sequenced in nearby positions from each other¹. Though very restrictive, adjacent pair wise interchange neighborhood structure encompasses the following benefits. First, it ensures that no trailer is unreasonably moved to a position far from its interdependent trailers. Second, the neighborhood generation complexity is of $O(n)$ for a solution of n trailers, implying that the neighborhood search is scalable with respect to the problem size. Last and most importantly is the capability to exploit the attributes of a solution for the evaluation of its neighbors. This eliminates the need

¹As discussed in Chapter 4, the DRS sequencing strategy is to produce a chain of interdependent trailers for the assignment to dock doors.

to carry out the assessment from scratch, and thus, can lead to significant reduction in the runtime. The concept is introduced in the next section.

5.2 Fractional Neighborhood Evaluation

In the design of any algorithm, computational effort should be taken into consideration as importantly as solution quality. This section tackles the truck scheduling optimization from the computational perspective by introducing a new methodology in the implementation of the truck scheduling problem that assesses each neighbor of a given solution using the information the solution already holds. The objective is to reduce the runtime by invoking the DF heuristic to evaluate only those portions of a neighboring solution that cannot be retrieved from the current solution.

Figure 5.3(a) shows a sample solution s representing the sequence list of eight trailers. Neighboring solution s' is produced by interchanging trailers 8 and 4 located at the fifth and sixth positions of solution s , respectively. Since the door assignment is performed according to the order the trailers are positioned, for both solutions s and s' , the first four trailers are assigned exactly the same, meaning that all the crossdocking timings are the same before the fifth trailer (trailer 8 for solution s and trailer 4 for solution s') is assigned. In other words, the assignment procedure changes from the fifth trailer and the change may extend to the remaining trailers so that a different makespan is produced for each of the two solutions. This turning point is referred to as *pivot* (and denoted as v) in this study. The other term is *window* defined as the number of trailers explicitly affected by a perturbation in the current solution to produce a new solution (it is denoted as w). The window starts from the very first position in the solution sequence list whose designated trailer has changed (which is the pivot) and ends at the very last position whose assigned trailer has also changed. For the example of Figure 5.3(a), the pivot is equal to five and the window is two.

5.3(a): s' is formed by an adjacent pair wise interchange of trailers 8 and 4 in solution s (note that $v = 5$ and $w = 2$)

5.3(b): s' is formed by an arbitrary re-ordering of the trailers located from position 2 to position 5 of solution s (note that $v = 2$ and $w = 4$)

5.3(c): s' is formed by a pair wise interchange of trailers 3 and 7 in solution s (note that $v = 1$ and $w = 8$)

Figure 5.3: Solution s and the three examples of its perturbed solution s'

Depending on the perturbation made, different values for v and w may be obtained. See Figure 5.3(b) where solution s' is produced after arbitrarily re-ordering the four trailers located from the second to the fifth position of s . In this figure, $v = 2$ and the window covers the four affected trailers, i.e., $w = 4$. According to the definition of the window of affected trailers, if two trailers located at different positions of the solution sequence list are exchanged with each other, the trailers positioned in between are also affected. Consider the example of Figure 5.3(c) where the first trailer in solution s is interchanged with the last trailer. In this case, the window will cover the entire trailers in the list and the pivot is equal to one.

The preceding paragraphs help in describing the proposed fractional evaluation strategy more clearly. Given a solution and its neighbor, all the information regarding the assigned doors and crossdocking timings of the trailers positioned before the pivot in the neighboring solution can be retrieved from the current solution. For those trailers placed after [and including] the pivot, however, the required information must be obtained by

invoking the DF algorithm. As perturbing the order of trailers for the assignment might distort feasibility, the fractional evaluation proceeds in two stages. First, the feasibility is checked and then, in the case the solution is feasible, its quality will be evaluated.

5.2.1 Fractional Feasibility Verification

For the fractional verification of the feasibility of solution s' , produced after perturbing solution s , it is required to record the number of available doors each time a trailer – picked from the sequence list of solution s – is assigned. The number of available doors in solution s' remains the same as the number in s when the trailers positioned before the pivot are assigned one after the other. From the pivot, however, the number of available doors must be re-calculated before the assignment of each trailer. If in the meantime the number reaches zero, it shows that solution s' is not feasible. On the other hand, if, after passing the window of affected trailers, the number of available doors before assigning the next trailer is equal or greater than the one in solution s , it is concluded that s' is feasible and the verification procedure terminates; otherwise, the procedure continues till the last trailer is assigned to an available door. In that case, solution s' is also feasible. Given $g(j)$, the function to return the number of available doors at the time of assigning trailer j – see Table 5.1 – the feasibility of solution s' can be verified before assigning the last trailer if the following logical relation holds:

$$\forall i_{v \leq i < (v+w)} \cdot (g(s'_{[i]}) > 0) \wedge \left[\exists i_{(v+w) \leq i < n} \forall i'_{(v+w) \leq i' < i} \cdot (g(s'_{[i]}) \geq g(s_{[i]})) \wedge (g(s'_{[i']}) > 0) \right] \vdash$$

$$feasibility(s') = \text{true}$$

Procedure 5.1 describes the fractional verification of the feasibility of solution s' in pseudo codes based on the parameters defined in Table 5.1. According to the code, it is required to restore the availability status of the doors after the trailers positioned before the pivot have all been assigned. The assigned trailer should then be recorded for each unavailable door. Note that solution s' is a neighbor of s when $w = 2$, i.e., $s' \in N(s)$, or else it could be any solution produced after arbitrarily perturbing solution s .

Procedure 5.1 FCHECK(n, s, s', v, w)

1. Restore door availability after the trailers positioned before pivot have been assigned;
 2. **for** $i = v$ **to** n **do**
 3. **if** $g(s'_{[i]})=0$ **then**
 4. **return false**; {feasibility of s' is not verified}
 5. **else if** $i \geq v + w$ **and** $g(s'_{[i]}) \geq g(s_{[i]})$ **then**
 6. **return true**; {feasibility of s' is verified}
 7. **else**
 8. Assign trailer $s'_{[i]}$ to a free door;
 9. Update door availability if any assigned trailer can release its door;
 10. **end if**
 11. **end for**
 12. **return true**; {feasibility of s' is verified}
-

5.2.2 Fractional Solution Evaluation

The fractional evaluation of a feasible solution s' , produced after perturbing solution s , involves restoration of the timings from solution s for the trailers positioned before the pivot and then invoking the DF algorithm to schedule the remaining trailers (i.e., those positioned after and including the pivot). The crossdocking state just before the assignment of the trailer placed at the pivot can be readily retrieved using the information stored in solution s . The information includes the interdependencies and timings of those trailers located before the pivot, together with their associated pallets. It is used to restore

Table 5.1: Parameters defined to describe the FCHECK and SEVAL Procedures

Parameter	Description
n	total number of trailers
s	current solution [records the sequence of trailers for the door assignment]
s'	solution produced after perturbing solution s
v	pivot equal to position of the very first affected trailer in a perturbed solution
w	window of affected trailers in a perturbed solution
$s_{[1..i]}$	partial solution [contains a subsequence of trailers from position 1 to i]
s_{best}	best solution
$g(j)$	returns the number of available doors at the time of assigning trailer j
$h(j)$	returns the order of trailers receiving pallets from trailer j
$f(s)$	objective function [returns the makespan corresponding to solution s]

the allocation status of the doors and availability times of both the doors and forklifts at the time where the trailer positioned at the pivot is to be assigned.

Unfortunately, the restoration procedure is not straightforward as the change in the scheduling order of the trailers may affect the crossdocking process for the trailers scheduled before the pivot. Consider the problem of optimizing the schedule of eight trailers in a crossdocking terminal. The interdependencies of the trailers, i.e., the list of their successors and predecessors, are given by Table 5.2. A local search (LS) algorithm (based on the adjacent pair wise interchange neighborhood) is used to optimize the schedule. Suppose the algorithm is currently at solution s and is evaluating its neighbor $s' \in N(s)$ (see Figure 5.4). According to the figure, the pivot is seven implying that the first six trailers should be scheduled exactly similarly to solution s . However, the implication is not true when trailer 1 at the sixth position is to be assigned.

According to Table 5.2, 1 has some pallets destined to trailers 5 and 6, respectively, which based on the proposed crossdocking model must be moved by a single forklift. The forklift scheduling function embedded in the DF algorithm decides on the movement of pallets according to the completion status of their recipient trailers assessed by the latest position that one of the predecessors of the trailers holds in the solution sequence list. A trailer having a smaller value receives its required pallets earlier than another with

Table 5.2: Interdependencies of eight trailers scheduled for crossdocking

Trailer	Successors	Predecessors
1	{2, 4, 5, 6}	{2, 3, 4}
2	{1, 4, 5}	{1, 3, 4, 5, 8}
3	{1, 2, 5}	{5, 6, 7, 8}
4	{1, 2, 6}	{1, 2, 8}
5	{2, 3, 7}	{1, 2, 3}
6	{3, 7, 8}	{1, 4, 7, 8}
7	{3, 6, 8}	{5, 6, 8}
8	{2, 3, 4, 6, 7}	{6, 7}

Figure 5.4: The moving order of the pallets unloaded from trailer 1 onto trailers 5 and 6 is different for solutions s and s' . This requires that the pivot be displaced backwards from position 7 to position 6 for the fractional evaluation of solution s'

a larger value (see Section 4.2.2 of Chapter 4). Accordingly, trailer 5 should receive its pallets earlier than trailer 6 for solution s . The last predecessor of trailer 5, which is trailer 2, is located at the seventh position while the last predecessor of trailer 6, which is trailer 4, is at the eighth position. For solution s' , the moving order is reversed as the positions of trailers 2 and 4 are interchanged. The change can lead to different timings which, like dominoes falling, may affect the next trailers. It is thus required to carry out a quick assessment on the moving order of the pallets unloaded from the trailers placed before the pivot to determine the very first trailer in the solution sequence list, if any, whose unloaded pallets would have a different moving schedule (relative to the previous solution). If one is found, the original pivot is displaced backwards to the position of the trailer.

The idea is better illustrated in Figure 5.5. A solution s' is generated by perturbing a given solution s with the total n trailers. As the fractional evaluation procedure depends on the value of the pivot and not the window of affected trailers, without loss of generality, it is assumed that $s' \in N(s)$ and is formed by interchanging two adjacent trailers c and d located at positions v and $v+1$ of solution s . The quick assessment function is performed from the first trailer in solution s' and continues until it finds a trailer b in a position $v' < v$ with a pallet moving schedule different from that in solution s . (It is possible that no such a trailer is found and all the pallets unloaded from the trailers positioned before the pivot in solution s' follow a moving schedule similar to the one in solution s .) Given

Figure 5.5: The assessment function detects that the moving order of pallets unloaded from a trailer b located at a position v' of solution s' is different from that in solution s

the assessment function is denoted by $h(j)$ for trailer j – see Table 5.1 – it is required to displace the pivot backwards to position v' if the following logical relations hold:

$$\begin{aligned} \exists v' \cdot h(s'_{[v']}) \neq h(s_{[v]}) \vdash f(s'_{[1..v']}) \neq f(s_{[1..v]}) \\ \forall i \cdot h(s'_{[i]}) = h(s_{[i]}) \vdash f(s'_{[1..v'-1]}) = f(s_{[1..v'-1]}) \end{aligned}$$

$1 < v' < v$ $1 \leq i \leq v'-1$

Since for the example in Figure 5.4, $h(s'_{[6]}) = (6, 5)$ is different from $h(s_{[6]}) = (5, 6)$, the pivot should be repositioned from the seventh to the sixth position.

Having updated the pivot, if necessary, the evaluation procedure uses the information stored in solution s to retrieve the crossdocking state while the trailers located before the pivot are being assigned. The schedule of the trailers positioned after [and including] the pivot is then carried out by invoking the DF algorithm. For situations where the initial pivot has changed, it is possible to reduce the assessment time by retrieving the assigned door of those trailers located between the new and old values of the pivot directly from solution s . The requirement is that the availability times of the available doors have not changed relative to those in solution s . In the case of any difference, the evaluation procedure must use DF to assign the remaining trailers (till the last trailer) to the best available door, respectively. Procedure SEVAL is described in pseudo codes as follows:

Procedure 5.2 sEVAL(n, s, s', v, s_{best})

```
1.  $v' \leftarrow v$ ;
2. for  $i = 1$  to  $v' - 1$  do
3.   if  $h(s'_{[i]}) \neq h(s_{[i]})$  then
4.      $v' \leftarrow i$ ; {pivot is displaced backwards}
5.   end if
6. end for
7. for  $i = 1$  to  $v' - 1$  do
8.   Retrieve the assigned door and crossdocking timings from  $s_{[i]}$  when  $s'_{[i]}$  is assigned;
9.   if  $f(s'_{[1..i]}) \geq f(s_{best})$  then
10.    return  $f(s_{best})$ ; { $s'$  cannot be better than the current best solution}
11.   end if
12. end for
13. for  $i = v'$  to  $n$  do
14.   if  $i < v$  and door availability times have not changed then
15.     Retrieve the assigned door from  $s_{[i]}$  and re-calculate crossdocking timings when  $s'_{[i]}$  is assigned;
16.   else
17.     Invoke DF algorithm to assign  $s'_{[i]}$  to the best available door;
18.   end if
19.   if  $f(s'_{[1..i]}) \geq f(s_{best})$  then
20.    return  $f(s_{best})$ ; { $s'$  cannot be better than the current best solution}
21.   end if
22. end for
23.  $s_{best} \leftarrow s'$  {best solution has improved}
24. return  $f(s_{best})$ ;
```

The ultimate goal is to embed the sEVAL procedure in an optimization algorithm so that new solutions are evaluated on the quest for a better solution. Further improvement in the procedure runtime is achieved by immediately terminating the evaluation process if it is detected in the meantime that the solution being evaluated cannot be better than the best solution found so far. This is accomplished by evaluating the partial sequence list $s_{[1..i]}$ updated each time a new trailer is scheduled per iteration (see Procedure 5.2). The objective function (makespan) of the partial schedule is then compared with that of the current best schedule to decide whether to continue the evaluation or stop the procedure. It is important to note that for situations in which it is required to fully evaluate a solution (e.g., to make a non-improving move as it is done in a tabu search-based algorithm – see Section 5.3.1), the s_{best} entry in the sEVAL procedure is set to a null value.

No lower bound is imposed on the completion times of the remaining unscheduled trailers whenever a trailer is scheduled. As further explained in Section 5.4, obtaining tight lower bounds requires estimating which door would be assigned to the trailers to incorporate pallet travel distances. Unfortunately, correct estimations cannot be guaranteed or are even achievable because of the high complexity of the truck scheduling problem. In the case that pallet travel distances are excluded, the calculated lower bounds would be so poor that they do not help to reduce the runtime but rather incur considerable overhead, counteracting the current savings.

In the end, it should be noted that in the real implementation of both `FCHECK` and `SEVAL` procedures, solution s shares its data with solution s' . Put differently, no individual database is considered for the solutions, eliminating the laborious process of copying data per evaluation. The complication is that both procedures may change the initial data required for the evaluation of the next solution produced after perturbing solution s , if any². For this purpose, any changes are stored in a temporary database and will later be used to restore the initial information for the next evaluation. In the case the search makes a move to a new solution, the corresponding information will be updated accordingly.

5.3 Meta-Heuristics Algorithms

Meta-heuristics, being truly diverse in nature, are mainly similar in performing operations to escape local optima. There are a number of criteria to choose and apply a particular meta-heuristic to a problem. One criterion is the ease of application. This can be in part understood by reviewing whether it has been employed widely for similar problems or not. The algorithm complexity is also an important criterion. If two algorithms produce

²For example, if a steepest descent algorithm is to be employed, one must evaluate the objective function for every element of neighborhood $N(s)$ of a given solution s before making the next best move.

similar results but one is significantly simpler than the other, then the simpler of the two is a superior algorithm. The complexity is generally assessed by the number of control parameters used in the algorithm [67]. Each major type of meta-heuristics has a number of parameters that must be set before the algorithm execution. As introduced in Chapter 2, control parameters (or factors) are the configurable components of a meta-heuristic algorithm that can be changed to alter its performance.

Meta-heuristics using many control parameters are more complex than procedures with few parameters. The effort needed to tune or understand these parameters is far greater as their number increases. Aside from this is the complexity of the studied truck scheduling problem which, due to its practicality, enforces a simple easy-to-implement algorithmic design. All of this suggests the employment of algorithms with a low degree of complexity for simpler explanation, analysis, and implementation.

In their paper, Silberholz and Golen [67] outline the basic control parameters required for six major meta-heuristics including ant colony optimization (ACO), genetic algorithm (GA), harmony search (HS), simulated annealing (SA), tabu search (TS), and variable neighborhood search (VNS) (see Table 5.3). Among them, variable neighborhood search with no control parameter and tabu search with only one parameter (which is the tabu list length) are considered the two meta-heuristic algorithms implemented for the truck scheduling problem addressed in this thesis. Both meta-heuristics have successful implementations in the related work. According to the review conducted in Chapter 2, both TS and VNS algorithms have been applied to other variations of the truck scheduling problem and the results recorded in terms of both solution quality and runtime have been among the best ones (see [33, 40, 41, 44, 46]). Finally, it is important to note that for both TS and VNS meta-heuristics, this study seeks an approach that is straightforward to implement, as a basis for developing comparisons. Advanced TS and VNS variants which are not applied here but may be readily used to supplement the current work, are a subject of future research.

Table 5.3: Popular meta-heuristics and their standard control parameters [67]

Name	Control Parameters
ant colony optimization (ACO)	pheromone evaporation parameter pheromone weighting parameter
genetic algorithm (GA)	crossover probability mutation probability population size
harmony search (HS)	distance bandwidth memory size pitch adjustment rate rate of choosing from memory
simulated annealing (SA)	annealing rate initial temperature
tabu search (TS)	tabu list length
variable neighborhood search (VNS)	none

5.3.1 Tabu Search (TS) Algorithm

Originally proposed by Glover in 1986 [68], tabu search is an extension of classical local search (LS) methods. According to Gendreau and Potvin [69], the basic principle of TS is to pursue LS whenever it encounters a local optimum by allowing non-improving moves. Cycling back to previously visited solutions is prevented by the use of memories, called tabu lists, that record the recent history of the search. Similar to LS, it follows that the two basic elements of any TS algorithm are the definition of its search space and its neighborhood structure.

For the TS algorithm proposed in this thesis, the search space defined in Section 5.1.1 and the neighborhood structure defined in Section 5.1.2 are adopted, respectively. This means that a local move is realized by interchanging the positions of two consecutive trailers (i.e., adjacent pair wise interchange) in the solution sequence list. Given an ini-

tial solution, normally produced by a constructive heuristic³, the TS algorithm starts exploring the search space by evaluating its neighbors. To make a move, a pre-specified number of neighbors are generated randomly and evaluated by the DF algorithm. In the *regular* TS, one must evaluate the objective function for every element of the neighborhood $N(s)$ of the current solution s . For the truck scheduling problem addressed in this study, however, the full evaluation is extremely expensive from the computational standpoint (more data is provided in Section 5.5.1). As a consequence, this research deploys *probabilistic* TS by considering only a random sample $N'(s)$ of $N(s)$ to significantly reduce the computational burden. (The number of random neighbors is determined by a control parameter.) While the added randomness can act as an anti-cycling mechanism, one may miss good solutions. This is because the best solutions returned by probabilistic TS will not be local optima with respect to the neighborhood structure being used (see [69]). As a subject of future work, Chapter 6 discusses the strategies to overcome this limitation.

The current solution will then be replaced with either a better solution, if any, or the best solution among the non-tabu moves evaluated for that iteration. The two trailers whose positions have been exchanged with each other in the current move are put in the tabu list. So any move which is made by exchanging these two trailers is considered tabu. The trailer pair stays in the tabu list for a pre-defined number of iterations called the tabu tenure. The length of the tabu tenure is determined randomly for each pair within the interval equal to the tabu list size. The termination criterion applied is to stop the TS algorithm after a pre-specified number of iterations without an improvement in the objective function value. (The maximum number of non-improving iterations is specified by a control parameter.)

³The ultimate objective of implementing both TS and VNS meta-heuristics is to improve the solutions generated by the DRS+DF heuristics and will be addressed in Section 5.5.2.

Table 5.4: Parameters defined to describe the TS Procedure

Parameter	Description
n	total number of trailers
s_0	initial solution
s	current solution
s'	solution produced after perturbing solution s
v	pivot equal to position of the very first affected trailer in a perturbed solution
$N'(s)$	a reduced sample of neighborhood $N(s)$ of solution s
r	number of neighbors in $N'(s)$ chosen randomly
T	tabu list
l	tabu list size
$i_{\max}$	maximum number of iterations with no improvement in solution quality
s_{nt_best}	best non-tabu solution recorded per TS iteration
s_{best}	best solution
$f(s)$	objective function [returns the makespan corresponding to solution s]

Aspiration criteria are employed to ensure that moves which are exceptionally promising are not ignored because of the tabu status of some component elements. The aspiration criterion applied in the implementation of the proposed TS algorithm is to allow a move, even if it is tabu, if it results in a solution with an objective function value better than that of the current best-known solution. Procedure 5.3 describes the TS algorithm in pseudo codes based on the parameters defined in Table 5.4. According to the code, the procedure evaluates tabu moves to check whether they lead to solutions better than the overall best solution or not. In this case, the procedure `SEVAL` is invoked by having the best solution (i.e., s_{best}) so that it can terminate the evaluation in early stages, to reduce the runtime, if it detects the current solution cannot be better than the overall best solution. On the other hand, for a non-tabu move, the evaluation cannot terminate unless it is detected that the solution is not better than the best solution produced from the previous non-tabu moves (denoted by s_{nt_best}) at the same iteration. In the end, note that due to the application of adjacent pair wise interchange neighborhood structure, the window of affected trailers (which is used as the input parameter to the procedure `FHECK`) for each perturbation is equal to two.

Procedure 5.3 TS($n, s_0, r, l, i_{\max}$)

```
1.  $s \leftarrow s_0$ ;
2. Initialize  $T$  by invalidating all its entries;
3.  $i \leftarrow 0$ ;
4.  $s_{best} \leftarrow s_0$ ; {solution  $s_0$  is initially considered the best solution}
5. repeat
6.    $s_{nt\_best} \leftarrow \text{NULL}$ ;
7.    $flag \leftarrow \text{false}$ ;
8.   for all  $1 \leq k \leq l$  where  $T_{[k]}$  is valid do
9.     if tabu tenure of  $T_{[k]}$  is zero then
10.      Invalidate  $T_{[k]}$ ; { $T_{[k]}$  refers to the element at position  $k$  of  $T$ }
11.    else
12.      Decrease tabu tenure of  $T_{[k]}$  by one;
13.    end if
14.  end for
15.  Build  $N'(s)$  by randomly choosing  $r$  neighbors;
16.  for  $k = 1$  to  $r$  do
17.    Initialize  $s'$  with one neighbor in  $N'(s)$ ;
18.    Initialize  $v$  according to the difference between  $s$  and  $s'$ ;
19.    if FCHECK( $n, s, s', v, 2$ ) = true then
20.      if move to  $s'$  is tabu then
21.        SEVAL( $n, s, s', v, s_{best}$ );
22.      else
23.        SEVAL( $n, s, s', v, s_{nt\_best}$ );
24.      end if
25.      if  $f(s') < f(s_{best})$  then
26.         $s_{best} \leftarrow s'$ ; {best solution is updated whether or not the move to  $s'$  is tabu}
27.         $flag \leftarrow \text{true}$ ;
28.      else if move to  $s'$  is non-tabu and  $f(s') < f(s_{nt\_best})$  then
29.         $s_{nt\_best} \leftarrow s'$ ; {record the best solution for non-tabu moves}
30.      end if
31.    end if
32.  end for
33.  if  $flag = \text{true}$  then
34.     $s \leftarrow s_{best}$ ; {move to the best solution}
35.    if move to  $s_{best}$  is non-tabu then
36.      Add the exchanged trailer pair in  $s_{best}$  to  $T$ ; {tabu tenure is randomly chosen in  $[1, l]$ }
37.    else
38.      Update the tabu tenure of the exchanged trailer pair in  $s_{best}$  randomly in  $[1, l]$ ;
39.    end if
40.     $i \leftarrow 0$ ;
41.  else
42.     $s \leftarrow s_{nt\_best}$ ; {move to the best solution among non-tabu moves}
43.    Add the exchanged trailer pair in  $s_{nt\_best}$  to  $T$ ; {tabu tenure is randomly chosen in  $[1, l]$ }
44.     $i \leftarrow i + 1$ ; {no improvement}
45.  end if
46. until  $i = i_{\max}$ 
47. return  $f(s_{best})$ ;
```

Most tabu designs normally include strategies such as the use of a long-term memory (e.g., see [70, 71]) or strategic oscillation by passing through infeasible solutions (also see [70]) to diversify the search into other areas of the solution space. Ensuring proper search diversification, however, is not an easy task and, as Gendreau and Potvin [69] stress, is possibly the most critical issue in the design of TS heuristics. So this study abstains from applying the above strategies to the current implementation of the TS algorithm. Rather, the stochastic evaluation of the neighborhood is exploited as a diversification mechanism by running the algorithm for a few times successively per problem instance.

5.3.2 Variable Neighborhood Search (VNS) Algorithm

Variable neighborhood search (VNS) is a meta-heuristic, proposed by Mladenovic and Hansen [72], based on the idea of a systematic change of neighborhood both in a descent phase to find a local optimum and in a perturbation phase to get out of the corresponding valley. In other words, VNS proceeds by a descent method to a local minimum, then explores, systematically or at random, increasingly distant neighborhoods of this solution. Each time, one or several points within the current neighborhood are used as an initial solution for a local descent. One jumps from the current solution to a new one if and only if a better solution has been found. It is important to note that unlike TS, variable neighborhood search does not specify forbidden moves.

To begin with, a finite set of neighborhood structures denoted by $N_k(k = 1, \dots, k_{\max})$ needs to be determined. Similar to TS, the solution produced by a constructive heuristic is considered the initial solution. VNS starts searching by generating a solution at random from neighborhood $N_k(k = 1)$ of the initial solution (the procedure is referred to as *shaking*). The shaking is followed by a local search routine on the quest for better solutions. A move is made to a new solution if it is better than the current solution. Otherwise, the algorithm jumps to $N_k(k \leftarrow k + 1)$, to invoke the shaking and local

Table 5.5: Parameters defined to describe the VNS Procedure

Parameter	Description
n	total number of trailers
s_0	initial solution
s	current solution
s', s''	solutions produced after perturbing solution s
v	pivot equal to position of the very first affected trailer in a perturbed solution
w	window of affected trailers in a perturbed solution
$k_{\max}$	maximum number of neighborhoods
$N_k(s)$	neighbors of solution s based on neighborhood structure k ($k = 1$ to $k_{\max}$)
$N'(s)$	a reduced sample of neighborhood $N(s)$ used in the LS procedure
r	number of neighbors in $N'(s)$ chosen randomly
$i_{\max}$	maximum number of iterations with no improvement in solution quality
$s_{l,best}$	best local solution found by the LS procedure per iteration
$f(s)$	objective function [returns the makespan corresponding to solution s]

search procedures from there. If the current solution improves, the search resumes from the first neighborhood, $N_k(k = 1)$, or else VNS will move to the next neighborhood, $N_k(k \leftarrow k + 1)$, if any, to repeat the shaking and local search procedures from that neighborhood. The algorithm will stop if after a pre-specified number of iterations, no improvement is made by exploring each neighborhood N_k per iteration. (The maximum number of non-improving iterations is specified by a control parameter.)

Unlike the basic VNS which employs a deterministic local search by evaluating every element of a neighborhood [72], the proposed algorithm adopts *reduced* local search by randomly choosing a pre-specified number of neighbors to evaluate per iteration (by the DF algorithm). This is because similar to the TS algorithm, the full evaluation is too costly from the computational perspective. (The number of random neighbors evaluated per iteration by the local search procedure is determined by a control parameter.) On the negative side, however, reduced LS may lead to loss of good solutions throughout the search because the best solutions returned will not be local optima. As a subject of future work, Chapter 6 discusses the strategies to overcome this limitation. Note that the neighborhood structure applied in LS is adjacent pair wise interchange. Given the parameters defined in Table 5.5, Procedure VNS is described in pseudo codes as follows:

Procedure 5.4 SHAKE(n, s, k)

1. **repeat**
 2. Generate a random solution $s' \in N_k(s)$;
 3. Initialize v and w according to the difference between s and s' ;
 4. **until** FCHECK(n, s, s', v, w) = **true**
 5. **return** s' ;
-

Procedure 5.5 LS(n, s, r)

1. $s_{l_best} \leftarrow s$;
 2. **repeat**
 3. $s \leftarrow s_{l_best}$;
 4. Build $N'(s)$ by randomly choosing r neighbors;
 5. **for** $i = 1$ **to** r **do**
 6. Initialize s' with one neighbor in $N'(s)$;
 7. Initialize v according to the difference between s and s' ;
 8. **if** FCHECK($n, s, s', v, 2$) = **true** **then**
 9. SEVAL(n, s, s', v, s_{l_best});
 10. **if** $f(s') < f(s_{l_best})$ **then**
 11. $s_{l_best} \leftarrow s'$;
 12. **end if**
 13. **end if**
 14. **end for**
 15. **until** $f(s) \leq f(s_{l_best})$
 16. $s_{l_best} \leftarrow s$;
 17. **return** s_{l_best} ;
-

Procedure 5.6 VNS($n, s_0, k_{\max}, r, i_{\max}$)

1. $s \leftarrow s_0$;
 2. $i \leftarrow 0$;
 3. **repeat**
 4. $k \leftarrow 1$;
 5. **repeat**
 6. $s' \leftarrow \text{SHAKE}(n, s, k)$;
 7. Initialize v according to the difference between s and s' ;
 8. SEVAL(n, s, s', v, NULL);
 9. $s'' \leftarrow \text{LS}(n, s', r)$;
 10. **if** $f(s'') < f(s)$ **then**
 11. $s \leftarrow s''$; {make a move}
 12. $k \leftarrow 1$; {first neighborhood}
 13. $i \leftarrow 0$;
 14. **else**
 15. $k \leftarrow k + 1$; {next neighborhood}
 16. **end if**
 17. **until** $k = k_{\max}$
 18. $i \leftarrow i + 1$;
 19. **until** $i = i_{\max}$
 20. **return** $f(s)$;
-

Note that the SHAKE procedure ensures that a feasible solution is randomly chosen from the current neighborhood. It is also required to fully evaluate the random solution before invoking the local search routine. For this purpose, the entry to the best solution (i.e., s_{best}) in the sEVAL procedure has been set to a null value in the code.

What remain not addressed in this section are what N_k should be used, how many of them, and how they should be ordered in the search. Depending on the studied problem, this could be a routine or challenging procedure. For example, in the p-median problem addressed by Hansen and Mladenovic [73], a solution and its neighbor in neighborhood N_k simply differ in k locations. On the other hand, to address their truck scheduling problem, Vahdani and Zandieh [41] develop nine different neighborhood structures intuitively, mainly based on random exchanges of inbound and outbound trucks in their individual list, respectively. The experimental results demonstrate that the proposed solution approach is remarkably efficient for scheduling problem instances up to 26 trucks. For large size problems, e.g., with more than 100 trucks, however, purely random exchanges of trucks without considering their freight distributions may not be efficient to find high quality solutions.

The neighborhood structures defined for the VNS algorithm proposed in this thesis are partly in accordance with those implemented by Vahdani and Zandieh in [41] due to similarities exist between the two problems. The main difference is that the suggested structures have been adapted to the nature of the problem. The neighborhoods are defined within clusters of interdependent trailers such that random exchanges of trailers are carried out in the cluster and not in the whole trailer list. In the original implementation of the proposed VNS algorithm, eight neighborhood structures were defined (i.e., $k_{max} = 8$). In the preliminary experimentations conducted, however, it was noticed that solution quality improved in normally three or four of the neighborhoods combined with the fact that exploring all the eight neighborhoods incurs considerable computational

overhead. This is consistent with what was observed by Hansen et. al [74] that the best value for parameter $k_{\max}$ is often two or three. Regarding their order in the search, neighborhoods are usually ranked in such a way that solutions are explored increasingly far from the current solution. In this way, intensification of the search around the current solution is followed naturally by diversification [73]. Given the above explanation, the proposed VNS implementation is complemented by defining three neighborhood structures ($k_{\max} = 3$) according to the following order:

- (i) **Adjacent pair wise interchange (N_1)**. In this structure, a trailer is chosen randomly and its position in the solution sequence list is exchanged with its subsequent trailer. This neighborhood structure is illustrated in Figure 5.6(a).
- (ii) **Shift between two random points within a cluster (N_2)**. Given two points p_1 and p_2 , randomly chosen within a cluster of interdependent trailers in the solution sequence list, a forward shift is applied if $p_2 > p_1$ or else a backward shift is applied. Figure 5.6(b) depicts the two situations where the forward and backward shifts are applied, respectively. It is good to note that for both shifts in the figure the pivot and window of affected trailers are equal to four.
- (iii) **Inverting and moving a subsequence of trailers within a cluster (N_3)**. Again two points p_1 and p_2 are randomly chosen within a cluster of interdependent trailers to form a trailer subsequence. The subsequence is first inverted and then moved forward if $p_2 > p_1$ or backward otherwise (see Figure 5.6(c)). Note that for the forward move in the figure, the pivot is three while for the backward, it is equal to two. Nonetheless, the window of affected trailers for both moves is six, which is twice the trailer subsequence size. It is also good to note that the values of p_1 and p_2 may be adjusted to prevent the move from crossing beyond the borders of the solution sequence list.

Figure 5.6: VNS neighborhood structures

5.4 Solution Quality Metrics

It is generally accepted that the best metric to use in solution quality comparison is the deviation of the solutions returned by [sub-optimal] algorithms from optimality. This, however, requires knowledge of the optimal solution for every problem instance that is tested. Unfortunately, the truck scheduling problem investigated in this thesis is so complex that optimal solutions even for small size problems are not attainable in reasonable running times. Two alternative metrics are commonly used when optimal solutions are not available for all (or potentially any) problem instances. One metric quantifies deviation from best-known solutions for a problem and the other measures deviation between

the algorithms being compared. The problem with the two comparison methods is that neither gives a reader any sense of the actual error of solutions [from the optimal solution]. Therefore, it is required to accompany such relative measures with additional information, including relaxation results for some of the problem instances or deviation from tight lower bounds to make them more meaningful.

Deriving tight lower bounds on the objective function value of the studied truck scheduling problem requires bounding door and forklift unavailability times, and pallet moving distances, that cannot be realized unless the location and sequence of trailers at the crossdock doors are taken into consideration. This indeed implies an estimation-based procedure to be carried out rather than bound computation, leading to unreliable conclusions on the solution quality of the compared algorithms. Alternatively, relaxation results for some of the problem instances obtained by a simplified version of the MIP model (S-MIP) – the original model was developed in Chapter 3 – and the solutions produced by a multi-start local search (M-LS) algorithm are considered the supplementary information that help to give a sense of the global optimality of the compared solution techniques. The two approaches are introduced in the next two sections.

5.4.1 Simplified MIP Model (S-MIP)

It was observed in Section 4.3.1 of Chapter 4 that the proposed MIP model was impractical to find optimal (and even feasible) solutions for this detailed truck scheduling problem. The model requires the solver to decide on the timing values and handling orders of every single pallet in the crossdock before producing the makespan. Alternatively, this section introduces a simplified version of the MIP model (S-MIP) where the exchanges of freight between trailers are not carried out in the form of pallets by using a limited number of forklifts but rather done collectively in a time equal to the distance they are moved (no unloading and loading times are thus considered). In that case, the

solver only needs to decide on the assignment order of trailers to the respective doors. The simplified model is applicable to verify the feasibility of the solutions generated by the proposed sub-optimal algorithms, as done by the original MIP model for the solutions of the DRS+DF heuristics in the previous chapter.

The following notation is used to describe the simplified MIP (S-MIP) model:

Indices	
m	door, $m \in M = \{1, \dots, M \}$
j	trailer, $j \in J = \{1, \dots, J \}$
Input parameters	
$f_{jj'}$	amount of freight sent from trailer j to trailer j'
$d_{mm'}$	[rectilinear] distance between doors m and m'
P_j	set of the predecessors of trailer j
T^C	trailer changeover time
Decision variables	
$c_{\max}$	schedule length (or makespan)
a_j	assignment time of trailer j
c_j	completion time of trailer j
x_{jm}	1, if trailer j is assigned to door m , else 0
$y_{jj'}$	1, if, for trailers j and j' assigned to the same door, j precedes j' , else 0
$v_{jmj'm'}$	1, if trailer j is assigned to door m and trailer j' is assigned to door m' , else 0

With the above denotations and decision variables, the S-MIP formulation can be described as follows:

$$\begin{array}{ll} \text{Minimize} & C_{\max} \\ \text{s.t.} & \end{array}$$

$$\sum_{m \in M} x_{jm} = 1, \quad j \in J \quad (5.1)$$

$$a_{j'} \geq c_j + T^C - Q \cdot (1 - y_{jj'}), \quad j, j' \in J, \quad j \neq j' \quad (5.2)$$

$$c_j \geq a_j, \quad j \in J \quad (5.3)$$

$$c_{j'} \geq a_j + f_{jj'} \cdot d_{mm'} \cdot v_{jmj'm'}, \quad j \in P_{j'}, \quad j' \in J, \quad m, m' \in M \quad (5.4)$$

$$c_{\max} \geq c_j, \quad j \in J \quad (5.5)$$

$$y_{jj'} + y_{j'j} = \sum_{m \in M} v_{jmj'm'}, \quad j, j' \in J, \quad j \neq j' \quad (5.6)$$

$$v_{jmj'm'} \leq x_{jm}, \quad j, j' \in J, \quad m, m' \in M \quad (5.7)$$

$$v_{jmj'm'} \leq x_{j'm'}, \quad j, j' \in J, \quad m, m' \in M \quad (5.8)$$

$$v_{jmj'm'} \geq x_{jm} + x_{j'm'} - 1, \quad j, j' \in J, \quad m, m' \in M \quad (5.9)$$

Constraint (5.1) ensures that each trailer is assigned exactly to one door. Constraint (5.2) defines the timing dependencies between two different trailers that have been assigned to the same door. It states that the assignment time of a trailer to a door is after the completion time of the other (which precedes the former trailer in being assigned to the same door) plus the time it takes to leave the door, i.e., the trailer changeover time. Constraints (5.3) and (5.4) define the completion time of a trailer. A trailer completes its operations after being assigned and receives its required freight from its predecessors. (Note that the latter operation would happen after the predecessor trailers themselves have been assigned.) Constraint (5.5) bounds the makespan which is equal to the maximum completion time of the trailers. Constraint (5.6) gives a freedom to the MIP solver to decide how to sequence the trailers assigned to the same door so that the optimal makespan is produced. Finally, Constraints (5.7)–(5.9) express that $v_{jmj'm'} = x_{jm} \cdot x_{j'm'}$.

The produced makespan, however, is not accurate requiring that a post-processing heuristic be developed to construct the true schedule length according to the operational characteristics of the crossdocking model introduced in Chapter 3. That is, the heuristic should decide on the moving order of pallets to their respective destination trailers.

The solution generated after solving the S-MIP model entails the door assigned to a trailer (specified by decision variable x_{jm}) and the order of trailers assigned to the same door (specified by decision variable $y_{jj'}$). This information is given to Procedure POSTMIP to build the true truck schedule. Initially, the assignment queue of trailers is constructed for each door, respectively, using the data provided by the S-MIP model. The procedure then repeats the following steps until all the trailers are scheduled. First, the trailers at the top of their corresponding queues are assigned to their respective doors.

Table 5.6: Parameters defined to describe the POSTMIP Procedure

Parameter	Description
n'	total number of crossdock doors
X	matrix storing values of decision variable x_{jm} generated by the S-MIP model
Y	matrix storing values of decision variable $y_{jj'}$ generated by the S-MIP model
Q_m	queue of trailers to be assigned to door m

Next, it is required to schedule the movement of unloaded pallets to their individual recipient trailers provided that they are already assigned to a door. Given the parameters defined in Table 5.6, Procedure POSTMIP is described in pseudo codes as follows:

Procedure 5.7 POSTMIP(n' , X , Y)

```

1. for  $m = 1$  to  $n'$  do
2.   Construct  $Q_m$  according to values of  $X$  and  $Y$ ;
3. end for
4. repeat
5.   for  $m = 1$  to  $n'$  do
6.     if door  $m$  is free and  $Q_m \neq \emptyset$  then
7.       Assign trailer  $Q_{m[1]}$  to door  $m$ ; {door  $m$  is no longer available}
8.       Schedule pallets moved from  $Q_{m[1]}$  to its assigned successors based on their completion
       status;
9.       Assign forklift  $m$ , designated to door  $m$ , to move the pallets;
10.      Update timings of the docking activities involved;
11.    end if
12.  end for
13.  for  $m = 1$  to  $n'$  do
14.    if door  $m$  is assigned and its assigned trailer is ready to be released then
15.      Remove the trailer from  $Q_m$ ; {door  $m$  is now available}
16.    end if
17.  end for
18. until all trailers are scheduled
19. return the makespan;

```

The scheduling policy is almost similar to that proposed in the DF algorithm developed in Section 4.2.2 of Chapter 4. That is, the order of movements is prioritized according to the completion status of the recipient trailers, assessed by the latest position that one of their predecessors holds in its corresponding assignment queue. Note that all the pallets unloaded from a given trailer are moved by one particular forklift. Having updated the timings of the docking activities involved, the procedure searches

for any trailer whose operations have been completed and can release its assigned door. The trailer, if any, is then removed from its corresponding assignment queue to allow its succeeding trailer in the queue to be assigned to the door. After all trailers are scheduled, the procedure terminates by returning the makespan.

5.4.2 Multi-Start Local Search (M-LS)

In the introduction of Section 5.4, it was stated that one common metric to evaluate the solution quality of a sub-optimal algorithm to a given problem is to measure its deviation from the best-known solutions of the problem. Given the fact that in the solution space of a combinatorial problem, there are typically a few local optima of which one is a global optimum, this section develops a multi-start local search (M-LS) algorithm to produce a number of local optima from a number of initial solutions.

It is well known that search methods based on local optimization that aspire to find a global optimum usually require some type of diversification to overcome local optimality. One way to achieve diversification is to re-start the search from a new solution once a region has been explored. In this respect, multi-start strategies are employed to guide the construction of new solutions in a long term horizon of the search process. According to Mart et. al [75], multi-start methods are two-phase procedures where in the first phase, the solution is generated (also called the construction phase) and in the second, the solution is improved normally by a local search algorithm. Then, each global iteration produces a solution (here a local optimum) where the best overall is the algorithm's output [and could be a global optimum].

It was shown in Chapter 4 that the DRS heuristic could be re-started from different initial clusters to generate a number of feasible trailer sequence lists of various qualities. (Note that the quality of each sequence list is evaluated by the DF algorithm.) This realizes the possibility of devising a multi-start procedure where the solutions generated

Table 5.7: Parameters defined to describe the M-LS Procedure

Parameter	Description
n	total number of trailers
L^C	list of initial clusters sorted in the descending order of their rank
O	solution pool [contains solutions generated by restarting DRS]
n^E	number of elite solutions in solution pool O
$n_{\max}$	maximum number of solutions examined to build solution pool O
$best(O)$	returns the best solution in solution pool O
$worst(O)$	returns the worst solution in solution pool O
s	current solution
$s_{l.best}$	local optimum in the neighborhood of solution s
s_{best}	best solution
$f(s)$	objective function [returns the makespan corresponding to solution s]

by re-starting DRS+DF are used by a local search algorithm as the initial solutions for further improvement. Given the parameters defined in Table 5.7, the construction phase proceeds by running the DRS+DF heuristics iteratively from a number initial clusters stored in the descending order of their rank in L^C (see Section 4.2.1 of Chapter 4 for the information on the cluster ranking methodology) to produce maximum $n_{\max}$ solutions from which n^E elite solutions will ultimately form a solution pool O .

After the first n^E solutions are generated and stored in solution pool O , the pool will be refined based on the following rules. When a solution s with $f(s) < f(worst(O))$ is generated, it is a candidate to be added to the pool replacing the worst solution if solution s is different enough from all of the solutions in the pool. To measure how different solution s is from a solution s' in the pool, the number of trailers whose positions in the two solution sequence lists s and s' are farther than the average cluster size are counted. If the number is smaller than 50% of the total number of trailers, solution s is discarded unless an aspiration criteria is satisfied, that is $f(s) < f(best(O))$. The n^E elite solutions will then be given to a local search algorithm to find their corresponding local optimum (according to the adjacent pair wise interchange neighborhood structure). Procedure 5.8 describes the M-LS algorithm in pseudo codes:

Procedure 5.8 M-LS($n, n^E, n_{\max}$)

```
1. Rank all the clusters and store them in descending order of their rank in  $L^C$ ;
2.  $O \leftarrow \emptyset$ ; {solution pool is initially empty}
3.  $timer \leftarrow 0$ ; {program timer}
4.  $i \leftarrow 0$ ;
5. repeat
6.   Initialize  $s$  by invoking DRS+DF from initial cluster  $L_{[1]}^C$ ;
7.   if  $s \neq \text{NULL}$  {a feasible solution has been produced} then
8.      $i \leftarrow i + 1$ ;
9.      $timer \leftarrow 0$ ; {reset the timer}
10.    if  $i \leq n^E$  then
11.      Insert solution  $s$  to pool  $O$ ;
12.      else if  $f(s) < f(best(O))$  {aspiration criterion is applied} then
13.        Replace the worst solution with solution  $s$ ;
14.      else if  $f(s) < f(worst(O))$  and  $s$  is different enough from all the solutions in  $O$  then
15.        Replace the worst solution with solution  $s$ ;
16.      end if
17.    end if
18.    Remove cluster  $L_{[1]}^C$  from  $L^C$ ;
19.  until  $i = n_{\max}$  or  $timer > 7200$  {stop if  $n_{\max}$  solutions are examined or the timer exceeds 2 hour}
20.   $n^E \leftarrow \min(n^E, i)$ ; {finalize the number of elite solutions in  $O$ }
21.   $s_{best} \leftarrow best(O)$ ;
22.  for  $i = 1$  to  $n^E$  do
23.    Initialize  $s$  with one solution in  $O$ ;
24.     $s_{l\_best} \leftarrow \text{LS}(n, s, n - 1)$ ;
25.    if  $f(s_{l\_best}) < f(s_{best})$  then
26.       $s_{best} \leftarrow s_{l\_best}$ ; {record the current best solution};
27.    end if
28.  end for
29. return  $f(s_{best})$ ;
```

Note that the construction phase terminates if the DRS algorithm cannot find another feasible trailer sequence list in a two-hour time since it has found one. It is thus possible that the total number of solutions in pool O is fewer than the pre-specified number n^E . Each elite solution in pool O is improved by the LS algorithm described by Procedure 5.5. To implement a deterministic local search, rather than a reduced form like that deployed in the VNS algorithm, the entry r in the LS procedure is set to the total number of neighbors to ensure that every element of the neighborhood of the current solution will be evaluated. Given the adjacent pair wise interchange neighborhood structure, the total number of neighbors defined for a solution s is equal to $n - 1$, where n is the number of

trailers. The algorithm will then return the best solution found after improving all the elite solutions.

5.5 Experimental Results and Analysis

All the proposed algorithms are coded in C++ and run on a Core™2 Quad 2.66GHz computer with 3GB RAM. Since the objective is to improve the quality of the solutions generated by the DRS+DF heuristics in Chapter 4, the experimental data consist of those instances where DRS could establish solution feasibility within the specified time limits. (The total number of feasible instances is 644 relative to the 960 instances generated initially.) Similar to the experiments conducted in the previous chapter, *relative difference* (*r. d.*) is adopted as the evaluation metric – see Eq. (4.14) – to compare the performance of the algorithms both in terms of solution quality and runtime. So if, for example, an algorithm *B* is compared with an algorithm *A*, a negative value of the relative difference implies an improvement in the solution quality or runtime of the algorithm *B* with respect to the algorithm *A*.

The other measure used in the experiments is *feasibility ratio* (*f. r.*) to indicate the percentage of times a search algorithm stays in feasibility regions of solution space and does not cross boundaries to enter infeasibility areas. This is determined by calculating the ratio of the number of feasible solutions evaluated by the algorithm relative to the total number of potential solutions visited for evaluation throughout the search.

5.5.1 Fractional vs. Full Neighborhood Evaluation

This section evaluates the efficiency of the proposed fractional neighborhood evaluation in reducing the runtime of neighborhood search methods. A deterministic local search algorithm is applied for this purpose (see the code in Procedures 5.5 and 5.8). The fractional evaluation is compared against the *full* neighborhood evaluation. Note that in the

latter, each neighbor of a given solution is assessed from scratch (by using the DF heuristic) whereas in the former, the assessment also uses the information the solution already holds wherever applicable. All the 644 problem instances whose solution feasibility has been established by the DRS heuristic are chosen for the experiments. The LS algorithm commences the search from the solution generated by the DRS+DF heuristics for each problem instance.

Table 5.8 presents the average experimental results for all the combinations of the problem sizes, trailer-to-door ratios (α), and freight mixed levels (FML). The second column of the table shows the search time when the full neighborhood evaluation has been applied while the third column shows the search time for the fractional evaluation. As seen in the table, the amount of reduction is very significant for large and huge size instances. (The reduction in the runtime is represented by negative values of the relative difference.) As an example, consider the extreme instance of scheduling 2048 trailers in a 256-door crossdock, i.e., the (huge:8) data category. The average time taken for LS to find a local optimum reduces from nearly 54 hours in the full evaluation to less than 16 hours in the fractional one.

On the other hand, the full neighborhood evaluation takes a comparable (or even a shorter) runtime relative to the fractional evaluation for small size instances. The reason is that the overhead incurred to carry out the restorations and updates associated with the fractional evaluation may offset the resultant savings given the fact that the full evaluation can be done fairly fast for small problems. It is also observed that the efficiency of the fractional evaluation slightly deteriorates for the problem instances of the higher freight mixed levels. This is typically due to the backward displacements of the pivot which occur more frequently for the higher levels of trailer interdependencies. Nevertheless, the deterioration in runtime is negligible compared to the overall savings obtained for large and huge size problems.

Table 5.8: Reduction in LS runtime when fractional neighborhood evaluation is applied

Data Category	LS Runtime(s)		<i>r. d.</i> (%)	LS Evaluations		<i>f. r.</i> (%)
	full	fractional		feasible	total	
(small:1;low)	0.11	0.06	-43.92	11	11	100
(small:1;medium)	0.16	0.12	-21.52	15	15	100
(small:1;high)	0.2	0.18	-5.59	17	17	100
(small:2;low)	0.65	0.52	-19.94	35	35	100
(small:2;medium)	1.05	0.98	-6.26	52	52	100
(small:2;high)	1.05	1.18	12.56	43	54	79.63
(small:4;low)	3.71	2.99	-19.46	99	99	100
(small:4;medium)	3.78	3.53	-6.59	89	124	71.77
(small:8;low)	21.12	17.06	-19.24	283	290	97.59
(medium:1;low)	7.05	3.25	-53.99	113	113	100
(medium:1;medium)	10.29	5.57	-45.82	156	156	100
(medium:1;high)	15.7	10.38	-33.85	158	158	100
(medium:2;low)	26.3	12.66	-51.86	214	214	100
(medium:2;medium)	29.21	16.41	-43.82	239	239	100
(medium:2;high)	81.89	52.05	-36.43	480	523	91.78
(medium:4;low)	126.9	64.59	-49.1	514	514	100
(medium:4;medium)	323.3	179.6	-44.45	1455	1693	85.94
(medium:8;low)	587.2	297.8	-49.29	1173	1173	100
(large:1;low)	181.3	65.1	-64.09	470	470	100
(large:1;medium)	267.5	99.49	-62.8	674	674	100
(large:1;high)	597.8	273	-54.34	1088	1088	100
(large:2;low)	664.5	233.6	-64.85	879	879	100
(large:2;medium)	716.8	283.8	-60.41	1041	1041	100
(large:2;high)	1448.9	579.2	-60.02	1702	1729	98.44
(large:4;low)	2587.5	988.5	-61.79	1685	1685	100
(large:4;medium)	11203.8	3848.9	-65.65	9907	11299	87.68
(large:8;low)	12661.2	4583.8	-63.8	4065	4065	100
(huge:1;low)	4440.6	1391.9	-68.66	1925	1925	100
(huge:1;medium)	4852.4	1708.8	-64.78	2065	2065	100
(huge:1;high)	13688.7	5464.1	-60.08	4399	4399	100
(huge:2;low)	12642.6	4031.3	-68.11	2734	2734	100
(huge:2;medium)	13878.5	4510.1	-67.5	3475	3475	100
(huge:2;high)	22314.8	6668.4	-70.12	4897	4906	99.82
(huge:4;low)	47355.9	14997	-68.33	5064	5064	100
(huge:8;low)	195378.75	55965.34	-71.35	10030	10030	100

The data recorded in the history files demonstrate that the strategy to update the partial makespan each time a trailer is scheduled (see Procedure 5.2) proved to be efficient for further reducing the runtime. For the majority of instances, there were solutions that could be detected not to be better than the overall best solution while only half of

the trailers had been scheduled. In other words, the partial makespan calculated after scheduling the first half of the trailers was worse than the best makespan.

It is important to note that even though the strategies proposed in the fractional neighborhood evaluation have been shown to be very efficient in reducing the search time, the application of a deterministic local search procedure through evaluating every element of the neighborhood is considerably time consuming for huge size problem instances. This can be a good reason to justify why a reduced random selection of the neighborhood has been adopted in the development of both TS and VNS meta-heuristics in this chapter.

Finally, the last three columns of Table 5.8 present statistics on the feasibility behavior of the neighboring solutions. The findings show that for the instances in which finding feasible solutions (using the DRS algorithm) is a bottleneck, typically those where $\alpha = 4$ and FML is medium, see Section 4.3.2.2 of Chapter 4, the feasibility percentage in the neighborhood is sufficiently high. It may thus be more time efficient to carry out the optimization of such instances in the neighborhood of only one feasible solution rather than spending time on establishing solution feasibility in different regions of the search space to ensure search diversification.

5.5.2 TS and VNS Evaluation

The proposed TS and VNS algorithms have a number of control parameters (or factors) that must be tuned before the performance of each is compared with one another. The objective of parameter tuning is mainly to balance the trade-off between the solution quality and runtime of both TS and VNS meta-heuristics. For example, by increasing the maximum number of non-improving iterations allowed for the TS algorithm, one can expect to find better solutions at the expense of a longer runtime. The question, however, is to what extent the parameter should be increased such that the expected improvement in solution quality is comparable to the extra computation time spent. Parameter tuning

answers this question. Appendix B contains the detailed analysis conducted to tune the control parameters of both algorithms. Given the control parameters defined for TS and VNS in Table 5.9, Table 5.10 summarizes the values assigned to them after being tuned based on the procedure carried out in Appendix B.

According to the values set up for their control parameters, the efficiency of both TS and VNS algorithms is evaluated in terms of improving the solutions generated by the DRS+DF heuristics. In the first step, the initial solution fed into the two meta-heuristics is the first feasible solution found by DRS (and evaluated by DF). In the second step, the best solution produced after re-starting DRS from a number of initial clusters is considered the initial solution. In the latter case, the TS and VNS meta-heuristics are denoted by R-TS and R-VNS to avoid confusion. (Note that in the two evaluation steps, both algorithms are invoked to improve all the 644 problem instances whose feasibility has been verified by the DRS heuristic.)

5.5.2.1 TS vs. VNS

According to the values of the control parameters tuned for both TS and VNS algorithms (see Appendix B), VNS demands a much longer time than TS to terminate its search process. A fair comparison of the two algorithms in terms of solution quality, however, should proceed with comparable running times. For this purpose, the time taken by the

Table 5.9: TS and VNS control parameters

(a) TS control parameters	
Parameter	Description
l	tabu list size
r	number of neighbors chosen randomly per iteration
$i_{\max}$	maximum number of non-improving iterations allowed
(b) VNS control parameters	
Parameter	Description
r	number of neighbors chosen randomly per iteration of LS procedure
$i_{\max}$	maximum number of non-improving iterations allowed

Table 5.10: Values assigned to TS and VNS control parameters after tuning

(a) Values assigned to TS control parameters

Problem Size	l	r	$i_{\max}$
small	10 if $\alpha \leq 4$ v^* if $\alpha = 8$	10	100
medium	10 if $\alpha = 1$ v^* if $\alpha \geq 2$	10	100
large	v^*	10 if $\alpha \leq 2$ 20 if $\alpha \geq 4$	100 if $\alpha \leq 2$ 50 if $\alpha \geq 4$
huge	v^*	10 if $\alpha \leq 2$ 20 if $\alpha \geq 4$	50

*indicates that the size of the tabu list varies between 0, 10, and 20

(b) Values assigned to VNS control parameters

Problem Size	r	$i_{\max}$
small	50	20
medium	50	20
large	50	20
huge	50	20

TS algorithm to optimize a given solution is fed into VNS, as an input parameter, to allow the latter to return the solution that has been improved within that time limit.

The stochastic components embedded in the implementation of both TS and VNS meta-heuristics require that the algorithms be run for a few times per problem instance to establish robustness in solution quality. (The number of times each algorithm is run per instance is set to 5 in the experiments to achieve a balance between solution quality robustness and runtime.) Since the solutions produced each run may be different in values, the corresponding running times may be different as well. Accordingly, to accomplish a synchronization between both algorithms, the average time taken by TS to optimize a solution per run is calculated and fed into VNS. The VNS algorithm should then return the improved solution within that particular time, which is the same for all the five runs. Note that for a specified number of runs, the runtime recorded for each algorithm is the total time taken to improve a solution for all the runs.

The average experimental results on the efficiency of TS and VNS in improving the DRS+DF solutions are shown in Tables 5.11 and 5.12. Table 5.11 presents the results for

all the combinations of the problem sizes, trailer-to-door ratios, freight mixed levels, and uncorrelated and correlated interdependencies for small and medium size instances while Table 5.12 shows the results for large and huge size instances. (Appendix A presents the detailed statistics for a subset of instances in the large and huge data categories.)

It can be observed from the tables that both TS and VNS algorithms perform almost equally well in improving the solutions of the DRS+DF heuristics relative to different characteristics of the test data. (Note that VNS optimizes a given solution within the time taken by TS to optimize the same solution.) Nevertheless, there are a few data categories where one algorithm slightly outperforms the other. It is noticed in particular that VNS is moderately better than TS in improving the majority of more difficult larger size instances of the higher trailer-to-door ratio and freight mixed level. As the search space of these instances is dramatically large, efficient diversification strategies are required to diversify the search into other areas of the solution space. It seems that the shaking mechanism embedded in the VNS algorithm works sufficiently well compared to TS where diversification is merely realized by the random evaluation of the neighborhood and running the algorithm for a few times repeatedly per problem instance. The slight differences in the solution quality for the remaining data categories are mainly due to the stochastic behavior of both algorithms, and thus, can be ignored.

The amount of improvement in solution quality is more noticeable for the simpler instances of the smaller trailer-to-door ratio and low freight mixed level. (The value is more significant for large and huge size instances – the reduction is less than 35% on average.) This indicates that both algorithms are successful in escaping local optima and exploring more promising regions of the search space for these data categories. Finally, note that the percentage of feasible solutions visited by the algorithms is adequately high even for the instances in which finding feasible solutions (using DRS) is a bottleneck, typically those where $\alpha = 4$ and FML is medium (see Chapter 4).

Table 5.11: Efficiency of TS and VNS in improving DRS+DF solutions for small and medium size instances

Data Category	DRS+DF		TS		VNS	
	$C_{\max}$	time(s)	$C_{\max}$	time(s)	$C_{\max}$	time(s)
(small:1;low;uncorrelated)	42.8	0.015	32	26.14	32	23.84
(small:1;medium;uncorrelated)	54.4	0.012	35.3	37.46	35.8	35.3
(small:1;high;uncorrelated)	55.7	0.009	38	53.62	38.1	51.37
(small:1;low;correlated)	47.4	0.016	31.5	25.26	31.5	22.01
(small:1;medium;correlated)	49.8	0.013	33.1	35.06	32.1	32.87
(small:1;high;correlated)	54.4	0.008	38.9	49.87	38.7	49.14
(small:2;low;uncorrelated)	111.7	0.023	87.7	104.48	87.9	104.77
(small:2;medium;uncorrelated)	160.4	0.03	115.7	165.88	117	166.22
(small:2;high;uncorrelated)	312.3	23.67	238.3	201.14	239.2	201.6
(small:2;low;correlated)	110.1	0.022	86.6	104.61	85.1	104.87
(small:2;medium;correlated)	148.6	0.028	116.5	143.51	118.7	143.86
(small:2;high;correlated)	304.5	151.72	231.3	199.11	230.8	199.54
(small:4;low;uncorrelated)	262.3	0.055	219.3	241	221.9	242.56
(small:4;low;correlated)	278.9	0.056	221.3	246.32	222.6	246.87
(small:4;medium;correlated)	792	121.06	666	439	639	439.8
(small:8;low;uncorrelated)	800	0.147	696.7	613.71	688.6	614.9
(small:8;low;correlated)	887.5	0.55	757.3	607.66	762.7	608.85
(medium:1;low;uncorrelated)	165.4	0.08	90.4	253.61	87.5	254.48
(medium:1;medium;uncorrelated)	172.1	0.114	110.7	430.28	112.2	431.22
(medium:1;high;uncorrelated)	323.8	0.191	206.9	1026.9	210.3	1028.5
(medium:1;low;correlated)	139.4	0.084	81.4	272.96	82.6	273.83
(medium:1;medium;correlated)	210.7	0.117	117.5	374.73	117.6	375.77
(medium:1;high;correlated)	303.3	0.191	220.4	858.72	220.3	860.32
(medium:2;low;uncorrelated)	260.3	0.197	175.6	544.11	174.8	545.73
(medium:2;medium;uncorrelated)	397.9	0.32	291.4	738.37	292.8	740.05
(medium:2;high;uncorrelated)	1105.5	56.15	700.2	2335.7	743.5	2338.2
(medium:2;low;correlated)	274.5	0.198	174.2	519.43	175.8	521.22
(medium:2;medium;correlated)	374.7	0.323	285.2	781.7	284.8	783.56
(medium:2;high;correlated)	1173.2	4.68	745.9	2015.9	787.8	2018.6
(medium:4;low;uncorrelated)	546.9	0.545	409.7	1356.1	411	1359.4
(medium:4;medium;uncorrelated)	341.4	4708.6	265.8	1680.1	258.5	1683.7
(medium:4;low;correlated)	537.8	0.55	395.3	1323.5	391.8	1326.7
(medium:4;medium;correlated)	3220	454.9	2436	2038.6	2405	2042
(medium:8;low;uncorrelated)	1313	1.644	1089.6	2980.1	1074.8	2986.1
(medium:8;low;correlated)	1371.9	1.651	1156.5	2725.6	1152.7	2732
				-15.7	100	-15.98
				-24.28	100	-24.28
				-27.15	100	-27.15
				-25.31	100	-25.31
				-18.14	100	-18.14
				-15.98	100	-15.98

Table 5.12: Efficiency of TS and VNS in improving DRS+DF solutions for large and huge size instances

Data Category	DRS+DF		TS				VNS			
	C_{max}	time(s)	C_{max}	time(s)	$r. d. (%)$	$f. r. (%)$	C_{max}	time(s)	$r. d. (%)$	$f. r. (%)$
(large:1;low;uncorrelated)	498.6	0.552	274.6	1698.7	-44.93	100	273.7	1703.3	-45.11	100
(large:1;medium;uncorrelated)	576.9	0.872	385.1	2263.3	-33.25	100	388.5	2268.6	-32.66	100
(large:1;high;uncorrelated)	1077.1	1.474	709.5	7224.2	-34.13	100	693.2	7232.1	-35.64	100
(large:1;low;correlated)	475.8	0.651	288.6	1700.3	-39.34	100	279.1	1705.4	-41.34	100
(large:1;medium;correlated)	584.5	0.88	383.8	2910.8	-34.34	100	377.5	2916.5	-35.41	100
(large:1;high;correlated)	1056.9	1.466	692	6241.3	-34.52	100	668.4	6248.5	-36.76	100
(large:2;low;uncorrelated)	731.9	1.389	497.8	3138.6	-31.98	100	490.3	3148.2	-33.01	100
(large:2;medium;uncorrelated)	1102.4	2.798	799.4	3686.1	-27.48	100	784.7	3695.6	-28.82	100
(large:2;high;uncorrelated)	2891.5	5.37	2133.1	7814.1	-26.23	98.37	2065.7	7825.2	-28.56	98.44
(large:2;low;correlated)	738.7	1.469	473.7	3377.7	-35.87	100	482.3	3387.3	-34.71	100
(large:2;medium;correlated)	1096.2	2.767	783.6	3648.1	-28.52	100	769.9	3657	-29.77	100
(large:2;high;correlated)	2761.4	5.279	2043.4	8130	-26	98.91	2028.8	8141.2	-26.53	98.87
(large:4;low;uncorrelated)	1442.6	4.136	1004.8	6607.1	-30.35	100	989.9	6626.8	-31.38	100
(large:4;medium;uncorrelated)	17385	971.98	12907	9857.2	-25.76	89.27	11737	9866.9	-32.49	89.92
(large:4;low;correlated)	1381.7	4.165	997.2	5757	-27.83	100	992.8	5776.7	-28.15	100
(large:4;medium;correlated)	17341.7	3068.5	12904.7	9077.9	-25.59	88.93	13030.3	9089.8	-24.86	88.57
(large:8;low;uncorrelated)	3042.1	13.68	2465.7	13318.6	-18.95	100	2465.8	13354.6	-18.94	100
(large:8;low;correlated)	3133.1	13.65	2492.7	12761.4	-20.44	100	2479.2	12796.9	-20.87	100
(huge:1;low;uncorrelated)	1669	3.566	844.6	5615	-49.39	100	861.8	5644.3	-48.36	100
(huge:1;medium;uncorrelated)	1817.6	6.597	1170.7	8921.9	-35.59	100	1166.8	8948.8	-35.8	100
(huge:1;high;uncorrelated)	3087.1	11.73	2230.8	24682.1	-27.74	100	2140.2	24720.5	-30.67	100
(huge:1;low;correlated)	1388.6	3.514	823.8	6022.7	-40.67	100	836.6	6051.1	-39.75	100
(huge:1;medium;correlated)	1828.8	6.565	1197.9	9486.6	-34.5	100	1194.4	9516.7	-34.69	100
(huge:1;high;correlated)	3214.5	11.78	2073.8	24725.9	-35.49	100	2053	24769	-36.13	100
(huge:2;low;uncorrelated)	2299.6	9.331	1493.9	11199.3	-35.04	100	1498.7	11251.4	-34.83	100
(huge:2;medium;uncorrelated)	3078	25.45	2244.7	10588.8	-27.07	100	2229.2	10636.8	-27.58	100
(huge:2;high;uncorrelated)	7083.6	51.63	5506.6	21460.7	-22.26	99.75	5473.9	21510	-22.72	99.79
(huge:2;low;correlated)	2291.6	9.484	1485.4	10536.1	-35.18	100	1479.7	10589.2	-35.43	100
(huge:2;medium;correlated)	2993.7	25.14	2203.9	11743.4	-26.38	100	2207.2	11788.9	-26.27	100
(huge:2;high;correlated)	6245.8	51.69	5184.3	19449	-16.99	99.88	5189.8	19501.6	-16.91	99.89
(huge:4;low;uncorrelated)	3877.9	28.59	2715.1	36064	-29.98	100	2679.5	36179.3	-30.9	100
(huge:4;low;correlated)	3829.6	28.46	2663.5	39066.9	-30.45	100	2669.9	39173	-30.28	100
(huge:8;low;uncorrelated)	7972.7	98.09	6189.2	63015.8	-22.37	100	6198.9	63219.6	-22.25	100
(huge:8;low;correlated)	7790.3	97.72	6109.9	68026.4	-21.57	100	6190.1	68235.8	-20.54	100

Since the efficiency of the VNS algorithm in improving the DRS+DF solutions was biased by an external time limit, it is good to assess its performance by allowing the algorithm to continue its search till its own termination criterion is met. This is done in Appendix C. The experimental results indicate that any marginal improvement of the solution quality is achieved by an exponential increase in the runtime. For example, the -3% relative difference in the solution quality of the (huge:8;low;correlated) category has been obtained after allowing the algorithm to continue running for additional five days! Given the fact that the original VNS algorithm could return an improved solution in this data category in nearly 19 hours, no convincing justification can be provided for the implementation of this extended version of VNS in an industrial setting.

It is thus required to combine some other approaches with the basic components of the proposed TS and VNS algorithms so that an efficient exploration of the search space is achieved in reasonable running times. The technique investigated in this chapter is to combine the re-starting version of the DRS algorithm (denoted as R-DRS) with the TS and VNS meta-heuristics. The idea is that the best solution produced by R-DRS is fed into each algorithm as the initial solution. The detailed analysis is conducted in the following section.

5.5.2.2 R-TS & R-VNS vs. TS & VNS

The efficiency of TS and VNS in improving the solutions generated by the DRS+DF heuristics is re-evaluated in the case that the initial solution is the best solution recorded after re-starting DRS. (The two meta-heuristics are denoted as R-TS and R-VNS, respectively.) The procedure of re-starting DRS from a number of initial clusters has been described in Section 5.4.2 where the multi-start local search (M-LS) algorithm was introduced (see Procedure 5.8, Lines 5–19). The only difference is that rather than returning a pool of elite solutions, the best solution in the pool is given as the output and considered the initial solution for both R-TS and R-VNS. The maximum number of solutions

required to be constructed for having a reasonably high-quality solution in the re-starting procedure is set to 10 in the experiments. Nevertheless, the procedure terminates if the DRS algorithm fails to find another feasible trailer sequence list in the two-hour time since it has found one. This implies that the number of generated solutions may be fewer than the pre-specified 10 solutions.

Tables 5.13 and 5.14 show the average experimental results for the proposed configuration of the R-TS and R-VNS algorithms. The former table presents the results for all the combinations of the problem sizes, trailer-to-door ratios, freight mixed levels, and uncorrelated and correlated interdependencies for small and medium size instances while the latter shows the results for large and huge size instances. (Appendix A presents the detailed statistics for a subset of instances in the large and huge data categories.) Note that the second column of both tables lists the makespan for the first feasible solution found and evaluated by the DRS+DF heuristics whereas the third column shows the best makespan recorded by the re-starting version of DRS (i.e., R-DRS). The forth and fifth columns indicate the time taken and amount of improvement in the solution quality obtained by the R-DRS+DF heuristics, respectively. Also note that the number of runs per problem instance for both R-TS and R-VNS algorithms is reduced from 5 to 4 to compensate for the additional time taken by the re-starting procedure.

Similar to the first phase of the experimentation, both R-TS and R-VNS algorithms perform almost equally well in improving the solutions of the heuristics relative to different characteristics of the test data. (Again note that R-VNS optimizes a given solution within the time taken by R-TS to optimize the same solution.) By comparing the two tables with Tables 5.11 and 5.12, one can notice a consistent robustness and increase in the solution quality of the two algorithms when they are initialized with a better solution. The robustness is observed by comparing the variations in the value of relative difference in the data categories (in particular, large and huge ones), which have been reduced in Tables 5.13 and 5.14.

Table 5.13: R-TS and R-VNS efficiency in improving DRS+DF solutions after re-starting for small and medium size instances

Data Category	DRS+DF			R-DRS+DF			R-TS			R-VNS		
	$C_{\max}$	$C_{\max}$	time(s)	$r. d. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$	$C_{\max}$	time(s)
(small:1;low;uncorrelated)	42.8	40.3	0.076	-5.84	32	19.47	-25.23	32	18.27	-25.23	32	18.27
(small:1;medium;uncorrelated)	54.4	44.7	0.094	-17.83	35.3	26.74	-35.11	36.1	26.06	-33.64	36.1	26.06
(small:1;high;uncorrelated)	55.7	48.7	0.1	-12.57	37.8	46.34	-32.14	38.4	42.64	-31.06	38.4	42.64
(small:1;low;correlated)	47.4	37.5	0.077	-20.89	31.5	19.96	-33.54	31.6	17.66	-33.33	31.6	17.66
(small:1;medium;correlated)	49.8	40.4	0.09	-18.87	32.9	25.34	-33.94	32.7	24.6	-34.34	32.7	24.6
(small:1;high;correlated)	54.4	45.3	0.103	-16.73	38.9	38.97	-28.49	38.6	38.42	-29.04	38.6	38.42
(small:2;low;uncorrelated)	111.7	102.2	0.211	-8.5	85.9	85.31	-23.1	85.9	85.52	-23.1	85.9	85.52
(small:2;medium;uncorrelated)	160.4	132.5	0.272	-17.39	112.8	109.74	-29.68	112.3	110.01	-29.99	112.3	110.01
(small:2;high;uncorrelated)	312.3	290.7	352.17	-6.94	248.8	158.16	-20.33	247.8	158.5	-20.65	247.8	158.5
(small:2;low;correlated)	110.1	98.4	0.217	-10.63	87.3	80.74	-20.71	85.8	80.95	-22.07	85.8	80.95
(small:2;medium;correlated)	148.6	140.7	0.292	-5.32	116.7	111.38	-21.47	117.5	111.65	-20.93	117.5	111.65
(small:2;high;correlated)	304.5	278.4	522.11	-8.57	220.7	154.2	-27.52	222.3	154.35	-26.99	222.3	154.35
(small:4;low;uncorrelated)	262.3	243.4	0.544	-7.2	216.8	177.84	-17.35	219.2	178.33	-16.43	219.2	178.33
(small:4;low;correlated)	278.9	239.2	0.551	-14.23	209.5	201.23	-24.88	213.3	201.7	-23.52	213.3	201.7
(small:4;medium;correlated)	792	747	1025	-5.68	609	521.7	-23.11	639	522.14	-19.32	639	522.14
(small:8;low;uncorrelated)	800	731.6	56.9	-8.55	636.3	431.6	-20.46	642.2	432.5	-19.725	642.2	432.5
(small:8;low;correlated)	887.5	779.3	1.847	-12.19	679.4	435.61	-23.45	679	436.6	-23.49	679	436.6
(medium:1;low;uncorrelated)	165.4	118.5	0.728	-28.35	86.8	202.59	-47.52	85.7	203.25	-48.19	85.7	203.25
(medium:1;medium;uncorrelated)	172.1	145.9	1.083	-15.22	112.4	280.23	-34.69	114.5	280.98	-33.47	114.5	280.98
(medium:1;high;uncorrelated)	323.8	263.4	1.792	-18.65	206.1	676.95	-36.35	207.3	678.33	-35.98	207.3	678.33
(medium:1;low;correlated)	139.4	119.9	0.741	-13.99	84.9	194.99	-39.1	83.8	195.67	-39.88	83.8	195.67
(medium:1;medium;correlated)	210.7	151.5	1.096	-28.1	113.6	288.22	-46.08	114.8	289.03	-45.51	114.8	289.03
(medium:1;high;correlated)	303.3	262.1	1.781	-13.58	219.2	755.02	-27.73	218.3	756.37	-28.02	218.3	756.37
(medium:2;low;uncorrelated)	260.3	224.5	1.861	-13.75	178.5	454.66	-31.42	176.8	455.9	-32.08	176.8	455.9
(medium:2;medium;uncorrelated)	397.9	346.7	3.139	-12.87	293.8	561.38	-26.16	293	562.81	-26.36	293	562.81
(medium:2;high;uncorrelated)	1105.5	931.9	596.41	-15.7	641.1	1671	-42.01	697.6	1673.1	-36.9	697.6	1673.1
(medium:2;low;correlated)	274.5	213.6	1.906	-22.19	173	451.11	-36.98	172.1	452.42	-37.3	172.1	452.42
(medium:2;medium;correlated)	374.7	338.2	3.178	-9.74	287.9	658.36	-23.16	290.5	659.71	-22.47	290.5	659.71
(medium:2;high;correlated)	1173.2	901.4	178.17	-23.17	683	1597	-41.78	716.1	1599.1	-38.96	716.1	1599.1
(medium:4;low;uncorrelated)	546.9	480.1	5.403	-12.21	408.2	1164.8	-25.36	406.5	1167.4	-25.67	406.5	1167.4
(medium:4;medium;uncorrelated)	3414	3414	11947.9	0	2647	1550.6	-22.47	2647	1552.6	-22.47	2647	1552.6
(medium:4;low;correlated)	537.8	466.6	5.378	-13.24	399.6	1002	-25.7	394.4	1004.7	-26.66	394.4	1004.7
(medium:4;medium;correlated)	3220	2995	17831.8	-6.99	2350.5	1676.6	-27	2458.5	1678.9	-23.65	2458.5	1678.9
(medium:8;low;uncorrelated)	1313	1189	16.07	-9.44	1018.8	2248.5	-22.41	1029.3	2254.1	-21.61	1029.3	2254.1
(medium:8;low;correlated)	1371.9	1230.9	16.33	-10.28	1073.9	2239.1	-21.72	1073.6	2244.3	-21.74	1073.6	2244.3

Table 5.14: R-TS and R-VNS efficiency in improving DRS+DF solutions after re-starting for large and huge size instances

Data Category	DRS+DF		R-DRS+DF			R-TS			R-VNS		
	C_{max}	C_{max}	time(s)	$r. d. (%)$	C_{max}	time(s)	$r. d. (%)$	C_{max}	time(s)	$r. d. (%)$	
(large:1;low:uncorrelated)	498.6	383.4	4.914	-23.1	280.9	1146.7	-43.66	276.4	1150.4	-44.56	
(large:1;medium:uncorrelated)	576.9	502.1	8.284	-12.97	383.3	2270.5	-33.56	375.4	2275	-34.93	
(large:1;high:uncorrelated)	1077.1	857.4	14.12	-20.4	661.6	6164.1	-38.58	674.2	6170.1	-37.41	
(large:1;low:correlated)	475.8	402	5.11	-15.51	277.8	1291.9	-41.61	273.1	1295.7	-42.6	
(large:1;medium:correlated)	584.5	481.4	8.276	-17.64	381.2	1759.1	-34.78	377.8	1763.3	-35.36	
(large:1;high:correlated)	1056.9	843.6	13.74	-20.18	661.5	5099.3	-37.41	656	5105.3	-37.93	
(large:2;low:uncorrelated)	731.9	616.2	13.27	-15.81	493.3	2559.8	-32.6	470.4	2568.1	-35.73	
(large:2;medium:uncorrelated)	1102.4	937.9	27.63	-14.92	789.5	2534.3	-28.38	775.1	2541.6	-29.69	
(large:2;high:uncorrelated)	2891.5	2061.9	51.85	-28.69	1726.9	5361.3	-40.28	1729.7	5369.6	-40.18	
(large:2;low:correlated)	738.7	649.8	13.87	-12.03	491.4	2515	-33.48	481.1	2523	-34.87	
(large:2;medium:correlated)	1096.2	941.1	27.58	-14.15	773.2	2690.8	-29.46	777.1	2697.7	-29.11	
(large:2;high:correlated)	2761.4	1993	50.38	-27.83	1595.3	5995.1	-42.23	1604	6003.9	-41.91	
(large:4;low:uncorrelated)	1442.6	1209.3	40.25	-16.17	1014.9	4780.8	-29.65	995.6	4795.2	-30.99	
(large:4;medium:uncorrelated)	17385	17385	8218.4	0	12502	6732.9	-28.09	12688	6744.2	-27.02	
(large:4;low:correlated)	1381.7	1197	40.34	-13.37	978.1	5229.8	-29.21	981.3	5245.7	-28.98	
(large:4;medium:correlated)	17341.7	16970	17693.7	-2.14	12920	8177.5	-25.5	12950	8188.5	-25.32	
(large:8;low:uncorrelated)	3042.1	2818.9	133.58	-7.34	2489.3	10646.2	-18.17	2492.2	10672.8	-18.08	
(large:8;low:correlated)	3133.1	2777.2	132.66	-11.36	2447.3	10593.8	-21.89	2437.4	10621.6	-22.2	
(huge:1;low:uncorrelated)	1669	1222.1	31.8	-26.78	845.9	4864	-49.32	849.2	4887.8	-49.12	
(huge:1;medium:uncorrelated)	1817.6	1523.1	62.25	-16.2	1187.3	7043.8	-34.68	1191.1	7067	-34.47	
(huge:1;high:uncorrelated)	3087.1	2618.1	113.33	-15.19	2084.5	17323.3	-32.48	2068.8	17355.1	-32.99	
(huge:1;low:correlated)	1388.6	1159.2	31.81	-16.52	843	4182.3	-39.29	839.9	4204	-39.51	
(huge:1;medium:correlated)	1828.8	1460.8	62.79	-20.12	1184.3	7545.6	-35.24	1194.3	7568.6	-34.69	
(huge:1;high:correlated)	3214.5	2478.7	113.59	-22.89	2071.2	19206.9	-35.57	2031.6	19239.6	-36.8	
(huge:2;low:uncorrelated)	2299.6	1931.3	88.44	-16.02	1540.5	8262	-33.01	1479.5	8309.1	-35.66	
(huge:2;medium:uncorrelated)	3078	2522.4	252.32	-18.05	2221.5	8503.8	-27.83	2221.7	8538.9	-27.82	
(huge:2;high:uncorrelated)	7083.6	5807.3	514.9	-18.02	4991.3	13784.2	-29.54	5008	13823.8	-29.3	
(huge:2;low:correlated)	2291.6	1959	89.67	-14.51	1456.7	9716.6	-36.43	1509	9761.1	-34.15	
(huge:2;medium:correlated)	2993.7	2568	249.82	-14.22	2185.6	9263.5	-26.99	2190.1	9303.1	-26.84	
(huge:2;high:correlated)	6245.8	5268.6	514.46	-15.65	4670.8	12058.5	-25.22	4668.9	12102.7	-25.25	
(huge:4;low:uncorrelated)	3877.9	3196.4	275.88	-17.57	2662.6	28693.4	-31.34	2682.7	28778.9	-30.82	
(huge:4;low:correlated)	3829.6	3268.8	276.82	-14.64	2692.2	30737.4	-29.7	2707	30822.5	-29.31	
(huge:8;low:uncorrelated)	7972.7	7088.5	1105.2	-11.09	6182.2	53864.1	-22.46	6213.3	54037.5	-22.07	
(huge:8;low:correlated)	7790.3	6906.9	1096.1	-11.34	6045.2	54674	-22.4	6164.9	54839.7	-20.86	

The impact of the initial solution on the performance of the TS and VNS algorithms is very significant for the problem instances of the high mixed level (especially those where $\alpha = 2$). For example, the average reduction in the makespan for the samples in the (large;2;high;correlated) category is more than 16%. There is a comparable reduction for medium and huge size instances with similar attributes. On the contrary, the effect of the initial solution is relatively insignificant for the simpler instances of $\alpha \leq 2$, and the low and medium freight mixed levels. It is generally accepted that a good initial solution can help a search method to start searching directly from a promising region of the solution space and reach remarkably high quality solutions (in a reasonable amount of time) which might be computationally very expensive to be obtained if the algorithm is initialized with a poor initial solution. The above statement specifically applies to optimizing difficult instances of the high mixed level. For simpler instances, the two meta-heuristics prove to be successful in guiding their search path towards more favorable areas of the solution space irrespective of the value of the initial solution.

In general, the combination of R-DRS with the two meta-heuristics is efficient in both obtaining better solutions and saving runtime which would be greatly spent for successively running the algorithms per problem instance to achieve high quality solutions.⁴ The above claim can be justified in Figure 5.7. As shown in the given charts, the improvement obtained in solution quality as the number of runs increases becomes less for both situations where TS and VNS are fed in with the first feasible solution generated by the DRS+DF heuristics and with the best solution obtained after restarting DRS from a number initial clusters (Note that the algorithms in the latter situation are denoted as R-TS and R-VNS). Based on these charts, it is not worth spending long time on letting the algorithms run beyond 3 to 4 runs per problem instance to obtain marginal improvement in solution quality.

⁴It is noticed that the re-starting procedure is not time consuming at all for almost all the problem instances except for those where establishing solution feasibility is a bottleneck, i.e., the instances where $\alpha = 4$ and FML is medium (see Chapter 4).

Figure 5.7: Impact of initial solution vs. multiple runs on the performance of the TS and VNS meta-heuristics in achieving high quality solutions

Finally, it is good to analyze the functionality of the R-TS and R-VNS algorithms for the complex instances of $\alpha = 8$. The impact of the initial solution is influential on the quality of the final solution produced for small and medium size instances. Such an advantage, however, is not observed for large and huge ones. In other words, a better initial solution could not help the meta-heuristics to more efficiently exploit the large search space of these instances according to the specified termination criteria. It is thus necessary to devise further advanced strategies such that efficiency is maintained when the problem size and complexity increase. This is considered a subject of future research and will be discussed in Chapter 6.

5.5.3 TS and VNS Solution Quality Assessment

In Section 5.4, two metrics were introduced to assess the quality of solutions generated by a sub-optimal algorithm. The first metric measures deviation from the relaxation results of a simplified version of the MIP model (S-MIP) – the original model was developed

in Chapter 3 – and the second compares the solutions with those of a multi-start local search (M-LS) algorithm. The comparison is conducted in the next two sections.

5.5.3.1 Comparison with S-MIP

As discussed in Section 5.4.1, the solution generated by the S-MIP model entails the door assigned to a trailer and the order of trailers assigned to the same door. This information should then be given to a post-processing heuristic (see Procedure 5.7) to build the true truck schedule by deciding on the moving order of pallets to their respective destination trailers. (The hybrid approach is denoted by S-MIP[+Post].)

Similar to the original MIP model, S-MIP is implemented in the AIMMS modeling platform and solved by the ILOG CPLEX 12.1 on a 64-bit machine with two Intel Xeon Quad Core E5335 2.0GHz processors and 4GB RAM. The objective is to run the S-MIP model for all the small instances of different characteristics. This, however, can be intractable for the instances where $\alpha > 1$. The CPLEX solver was unable to find the optimal solution for such instances and could not terminate the program. Alternatively, an external termination criterion is imposed on the solver such that if the incumbent solution cannot be further improved within a specified timing interval, the solver stops and returns the incumbent as the final solution; otherwise, the runtime is extended for another interval. The imposed timing interval is equivalent to a pre-specified number of iterations taken by the solver to solve a given problem instance. (The number is set to 10^7 in the experiments to give CPLEX enough time to search the solution space.)

Tables 5.15(a) and 5.15(b) present the S-MIP+[Post] results for all the instances in the (small:1) data category, and two samples in the (small:2;low;uncorrelated) and (small:4;low;uncorrelated) categories, respectively. For all the (small:1) instances, the CPLEX solver is able to establish solution optimality. For the two larger samples, however, the solver is forced to stop and return a not-necessarily optimal solution according to the termination criterion already introduced.

Table 5.15: Solution comparison of TS with S-MIP[+Post] for small instances

(a) Comparison with the (small:1) data category

	S-MIP [+Post]	TS			S-MIP [+Post]	TS			S-MIP [+Post]	TS	
	$C_{\max}$	$C_{\max}$	$r. d.(\%)$		$C_{\max}$	$C_{\max}$	$r. d.(\%)$		$C_{\max}$	$C_{\max}$	$r. d.(\%)$
	(small:1;low;uncorrelated)				(small:1;medium;uncorrelated)				(small:1;high;uncorrelated)		
1	46	28	-39.13		58	39	-32.76		46	38	-17.39
2	53	38	-28.3		78	33	-57.69		78	38	-51.28
3	54	28	-48.15		70	36	-48.57		42	35	-16.67
4	84	36	-57.14		53	34	-35.85		53	35	-33.96
5	78	28	-64.1		39	30	-23.08		54	38	-29.63
6	93	30	-67.74		70	34	-51.43		54	36	-33.33
7	119	36	-69.75		62	40	-35.48		43	36	-16.28
8	78	28	-64.1		70	36	-48.57		54	43	-20.37
9	40	28	-30		78	35	-55.13		52	41	-21.15
10	59	40	-32.2		88	36	-59.09		68	40	-41.18
avg.	70.4	32	-54.54		66.6	35.3	-47		54.4	38	-30.15
	(small:1;low;correlated)				(small:1;medium;correlated)				(small:1;high;correlated)		
1	47	28	-40.42		78	32	-58.97		68	53	-22.06
2	69	29	-57.97		107	29	-72.9		52	40	-23.08
3	106	41	-61.32		44	34	-22.73		58	41	-29.31
4	77	30	-61.04		62	37	-40.32		58	38	-34.48
5	70	36	-48.57		29	29	0		46	40	-13.04
6	90	30	-66.67		39	36	-7.69		46	36	-21.74
7	62	28	-54.84		86	30	-65.12		43	34	-20.93
8	107	32	-70.09		46	32	-30.43		48	36	-25
9	41	29	-29.27		53	39	-26.41		46	37	-19.56
10	89	32	-64.04		70	33	-52.86		46	34	-26.09
avg.	75.8	31.5	-58.44		61.4	33.1	-46.09		51.1	38.9	-23.87

(b) Comparison with two samples in the (small:2) and (small:4) data categories

	S-MIP[+Post]		TS				S-MIP[+Post]		TS		
	$C_{\max}$	time(s)	$C_{\max}$	time(s)	$r. d.(\%)$		$C_{\max}$	time(s)	$C_{\max}$	time(s)	$r. d.(\%)$
	(small:2;low;uncorrelated)						(small:4;low;uncorrelated)				
1	151	45107.8	89	97.75	-41.06		444	473516.3	226	277.84	-49.1

The final solutions produced by the post-processing heuristic are compared with the solutions improved by the basic proposed TS algorithm. As shown in the tables, the solutions are remarkably inferior to those of the TS algorithm, even for the (small:1) instances. The main reason for such a deficiency is that the S-MIP model decides on the assignment of trailers to doors with no knowledge of the detailed handling operations of pallets exchanged. So the resultant configuration, which is verified to be optimal by the solver, does not necessarily remain optimal when the true truck schedule is constructed by the post-processing heuristic. On the other hand, restricting the comparison on the solutions directly generated by the S-MIP model, i.e., simplifying the problem by ignoring pallets, is not accurate as the complexity of the truck scheduling problem in terms of

solution optimality mainly lies in its detailed modeling structure. Apart from that is the complication of the simplified model which is still not scalable to the problem size. See Table 5.15(b) where the solution time increases to 12.5 hours and nearly 5.5 days for the two instances in the (small:2;low;uncorrelated) and (small:4;low;uncorrelated) categories, respectively. It is concluded from the above experiment that any improvement in the mathematical model should be applied to the original formulation in Chapter 3 to maintain accuracy. An important property of the given formulation is the large number of variables it contains. One of the techniques to deal with such formulations with a huge number of variables is column generation which provides a decomposition of the problem into master and sub-problems [76]. The technique is considered a subject of future work and will be discussed in Chapter 6.

5.5.3.2 Comparison with M-LS

The last strategy that remains for solution quality assessment of the TS and VNS meta-heuristics is to measure deviation from the solutions generated by the M-LS algorithm proposed in Section 5.4.2. The M-LS algorithm is applied to all the 644 feasible instances. Unlike the TS and VNS meta-heuristics, no timing restrictions are imposed on M-LS and the algorithm continues searching till it finds a local optimum for each of the elite solutions in the solution pool. The deviation is then measured relative to the best local optimum that is found. Note that the number of elite solutions is 5 while the maximum number of solutions required to be examined in the pool to maintain diversity is 10.

Tables 5.16 and 5.17 compare the average solutions generated by the TS, R-TS, VNS, and R-VNS algorithms with those of M-LS. The former table presents the results for all the combinations of the problem sizes, trailer-to-door ratios, freight mixed levels, and uncorrelated and correlated interdependencies for small and medium problem instances while the latter shows the results for large and huge size instances. (Appendix A presents the detailed statistics for a subset of instances in the large and huge data categories.)

Table 5.16: Solution comparison of TS and VNS with M-LS for small and medium size instances

Data Category	M-LS		TS		R-TS		VNS		R-VNS	
	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$
(small:1:low;uncorrelated)	36.2	32	-11.6	32	-11.6	32	-11.6	32	-11.6	
(small:1:medium;uncorrelated)	38.5	35.3	-8.31	35.3	-8.31	35.8	-7.01	36.1	-6.23	
(small:1:high;uncorrelated)	45.5	38	-16.48	37.8	-16.92	38.1	-16.26	38.4	-15.6	
(small:1:low;correlated)	33.5	31.5	-5.97	31.5	-5.97	31.5	-5.97	31.6	-5.67	
(small:1:medium;correlated)	36.1	33.1	-8.31	32.9	-8.86	32.1	-11.08	32.7	-9.42	
(small:1:high;correlated)	42.9	38.9	-9.32	38.9	-9.32	38.7	-9.79	38.6	-10.02	
(small:2:low;uncorrelated)	95.6	87.7	-8.26	85.9	-10.15	87.9	-8.05	85.9	-10.15	
(small:2:medium;uncorrelated)	121.1	115.7	-4.46	112.8	-6.85	117	-3.39	112.3	-7.27	
(small:2:high;uncorrelated)	263.7	238.3	-9.61	248.8	-5.63	239.2	-9.29	247.8	-6	
(small:2:low;correlated)	90.4	86.6	-4.2	87.3	-3.43	85.1	-5.86	85.8	-5.09	
(small:2:medium;correlated)	128.2	116.5	-9.13	116.7	-8.97	118.7	-7.41	117.5	-8.35	
(small:2:high;correlated)	248.2	231.3	-6.81	220.7	-11.08	230.8	-7.01	222.3	-10.43	
(small:4:low;uncorrelated)	226.4	219.3	-3.14	216.8	-4.24	221.9	-1.99	219.2	-3.18	
(small:4:low;correlated)	221.9	221.3	-0.27	209.5	-5.59	222.6	0.31	213.3	-3.88	
(small:4:medium;correlated)	747	666	-10.84	609	-18.47	639	-14.46	639	-14.46	
(small:8:low;uncorrelated)	657.9	696.7	5.9	636.3	-3.28	688.6	4.67	642.2	-2.39	
(small:8:low;correlated)	700.4	757.3	8.12	679.4	-3	762.7	8.89	679	-3.05	
(medium:1:low;uncorrelated)	94.1	90.4	-3.93	86.8	-7.76	87.5	-7.01	85.7	-8.93	
(medium:1:medium;uncorrelated)	127.6	110.7	-13.24	112.4	-11.91	112.2	-12.07	114.5	-10.27	
(medium:1:high;uncorrelated)	232.8	206.9	-11.12	206.1	-11.47	210.3	-9.66	207.3	-10.95	
(medium:1:low;correlated)	92.9	81.4	-12.38	84.9	-8.61	82.6	-11.09	83.8	-9.79	
(medium:1:medium;correlated)	129.5	117.5	-9.27	113.6	-12.28	117.6	-9.19	114.8	-11.35	
(medium:1:high;correlated)	244.8	220.4	-9.97	219.2	-10.46	220.3	-10.01	218.3	-10.82	
(medium:2:low;uncorrelated)	186.7	175.6	-5.94	178.5	-4.39	174.8	-6.37	176.8	-5.3	
(medium:2:medium;uncorrelated)	305.4	291.4	-4.58	293.8	-3.8	292.8	-4.13	293	-4.06	
(medium:2:high;uncorrelated)	779.5	700.2	-10.17	641.1	-17.75	743.5	-4.62	697.6	-10.51	
(medium:2:low;correlated)	181.9	174.2	-4.22	173	-4.89	175.8	-3.35	172.1	-5.39	
(medium:2:medium;correlated)	294.7	285.2	-3.22	287.9	-2.31	284.8	-3.36	290.5	-1.42	
(medium:2:high;correlated)	781.5	745.9	-4.55	683	-12.6	787.8	0.81	716.1	-8.37	
(medium:4:low;uncorrelated)	410.5	409.7	-0.19	408.2	-0.56	411	0.12	406.5	-0.97	
(medium:4:medium;uncorrelated)	2713	2658	-2.03	2647	-2.43	2585	-4.72	2647	-2.43	
(medium:4:low;correlated)	405.3	395.3	-2.47	399.6	-1.41	391.8	-3.33	394.4	-2.69	
(medium:4:medium;correlated)	2580.5	2436	-5.6	2350.5	-8.91	2405	-6.8	2458.5	-4.73	
(medium:8:low;uncorrelated)	1051.3	1089.6	3.64	1018.8	-3.09	1074.8	2.23	1029.3	-2.09	
(medium:8:low;correlated)	1082.9	1156.5	6.8	1073.9	-0.83	1152.7	6.45	1073.6	-0.86	

Table 5.17: Solution comparison of TS and VNS with M-LS for large and huge size instances

Data Category	M-LS		TS		R-TS		VNS		R-VNS	
	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$
(large:1;low;uncorrelated)	290.7	-5.54	274.6	-3.37	280.9	-3.37	273.7	-5.85	276.4	-4.92
(large:1;medium;uncorrelated)	400.1	-3.75	385.1	-4.2	383.3	-4.2	388.5	-2.9	375.4	-6.17
(large:1;high;uncorrelated)	730.1	-2.82	709.5	-9.38	661.6	-9.38	693.2	-5.05	674.2	-7.66
(large:1;low;correlated)	295	-2.17	288.6	-5.83	277.8	-5.83	279.1	-5.39	273.1	-7.42
(large:1;medium;correlated)	396.2	-3.13	383.8	-3.79	381.2	-3.79	377.5	-4.72	377.8	-4.64
(large:1;high;correlated)	722.4	-4.21	692	-8.43	661.5	-8.43	668.4	-7.47	656	-9.19
(large:2;low;uncorrelated)	504.2	-1.27	497.8	-2.16	493.3	-2.16	490.3	-2.76	470.4	-6.7
(large:2;medium;uncorrelated)	793.2	0.78	799.4	-0.47	789.5	-0.47	784.7	-1.07	775.1	-2.28
(large:2;high;uncorrelated)	1802.3	18.35	2133.1	-4.18	1726.9	-4.18	2065.7	14.61	1729.7	-4.03
(large:2;low;correlated)	491.4	-3.6	473.7	0	491.4	0	482.3	-1.85	481.1	-2.1
(large:2;medium;correlated)	797.6	-1.75	783.6	-3.06	773.2	-3.06	769.9	-3.47	777.1	-2.57
(large:2;high;correlated)	1665.3	22.7	2043.4	-4.2	1595.3	-4.2	2028.8	21.83	1604	-3.68
(large:4;low;uncorrelated)	1006.7	-0.19	1004.8	0.81	1014.9	0.81	989.9	-1.67	995.6	-1.1
(large:4;medium;uncorrelated)	12699	1.64	12907	-1.55	12502	-1.55	11737	-7.57	12688	-0.09
(large:4;low;correlated)	979.9	1.76	997.2	-0.18	978.1	-0.18	992.8	1.32	981.3	0.14
(large:4;medium;correlated)	13871.7	-6.97	12904.7	-6.86	12920	-6.86	13030.3	-6.06	12950	-6.64
(large:8;low;uncorrelated)	2461.6	0.17	2465.7	1.12	2489.3	1.12	2465.8	0.17	2492.2	1.24
(large:8;low;correlated)	2421.2	2.95	2492.7	1.08	2447.3	1.08	2479.2	2.39	2437.4	0.67
(huge:1;low;uncorrelated)	831.2	1.61	844.6	1.77	845.9	1.77	861.8	3.68	849.2	2.16
(huge:1;medium;uncorrelated)	1163.6	0.61	1170.7	2.04	1187.3	2.04	1166.8	0.27	1191.1	2.36
(huge:1;high;uncorrelated)	2148.1	3.85	2230.8	-2.96	2084.5	-2.96	2140.2	-0.37	2068.8	-3.69
(huge:1;low;correlated)	833.2	-1.13	823.8	1.18	843	1.18	836.6	0.41	839.9	0.8
(huge:1;medium;correlated)	1206.7	-0.73	1197.9	-1.86	1184.3	-1.86	1194.4	-1.02	1194.3	-1.03
(huge:1;high;correlated)	2106.3	-1.54	2073.8	-1.67	2071.2	-1.67	2053	-2.53	2031.6	-3.55
(huge:2;low;uncorrelated)	1479.2	0.99	1493.9	4.14	1540.5	4.14	1498.7	1.32	1479.5	0.02
(huge:2;medium;uncorrelated)	2208.8	1.62	2244.7	0.57	2221.5	0.57	2229.2	0.92	2221.7	0.58
(huge:2;high;uncorrelated)	4946.4	11.32	5506.6	0.91	4991.3	0.91	5473.9	10.66	5008	1.24
(huge:2;low;correlated)	1462.3	1.58	1485.4	-0.38	1456.7	-0.38	1479.7	1.19	1509	3.19
(huge:2;medium;correlated)	2184	0.91	2203.9	0.07	2185.6	0.07	2207.2	1.06	2190.1	0.28
(huge:2;high;correlated)	4619.5	12.23	5184.3	1.11	4670.8	1.11	5189.8	12.34	4668.9	1.07
(huge:4;low;uncorrelated)	2645	2.65	2715.1	0.66	2662.6	0.66	2679.5	1.3	2682.7	1.42
(huge:4;low;correlated)	2614.9	1.86	2663.5	2.96	2692.2	2.96	2669.9	2.1	2707	3.52
(huge:8;low;uncorrelated)	5995.2	3.24	6189.2	3.12	6182.2	3.12	6198.9	3.4	6213.3	3.64
(huge:8;low;correlated)	5919.2	3.22	6109.9	2.13	6045.2	2.13	6190.1	4.58	6164.9	4.15

It can be observed that both TS and VNS algorithms and their re-starting variants have been successful in escaping local optima and finding high quality solutions for most of the small, medium, large and some of the huge size instances in reasonable running times (the latter have been shown in Tables 5.11–5.14, and 5.18). In particular, the R-TS and R-VNS variants effectively compensate for the high positive relative difference of the TS and VNS solutions from those of M-LS for large and huge size instances of $\alpha = 2$ and the high mixed level. A similar situation is noticed for the small and medium size instances of $\alpha = 8$. As seen in Tables 5.16 and 5.17, the variations in the value of relative difference have been reduced for the R-TS and R-VNS algorithms.

On the contrary, for the majority of the large size instances of $\alpha = 8$ and most of the huge size instances, the four meta-heuristics slightly fall behind the M-LS algorithm. The justification for such declining performance is based on the results of Table 5.18. The table presents the average runtime of M-LS, TS, R-TS, VNS, and R-VNS for the (large:8) category and all the combinations of the huge size instances. As shown in the table, M-LS spends a substantially longer time than the other four algorithms for these data categories. For example, the runtime of M-LS is nearly twice that of the R-TS and R-VNS algorithms for the (large:8) instances. As another example, consider the (huge:8) data categories. The solution time taken by M-LS is on average 60 hours longer than that taken by these four meta-heuristics. As the improvement in the solution quality is marginal compared to the time spent⁵, the following conclusions can be derived from the performance of the TS and VNS meta-heuristics:

- The trade-off between solution quality and runtime has been properly balanced.

This implies that the control parameters of the two algorithms, particularly TS, have been correctly tuned.

⁵A similar situation is also observed in the extended version of VNS (E-VNS), see Appendix C.

Table 5.18: Runtime comparison of TS and VNS with M-LS for large and huge size instances

Data Category	M-LS	TS	R-TS	VNS	R-VNS
(large:8;low;uncorrelated)	21646.8	13318.6	10646.2	13354.6	10672.8
(large:8;low;correlated)	20614.1	12761.4	10593.8	12796.9	10621.6
(huge:1;low;uncorrelated)	6617.4	5615	4864	5644.3	4887.8
(huge:1;medium;uncorrelated)	9012.5	8921.9	7043.8	8948.8	7067
(huge:1;high;uncorrelated)	25713.8	24682.1	17323.3	24720.5	17355.1
(huge:1;low;correlated)	5817.3	6022.7	4182.3	6051.1	4204
(huge:1;medium;correlated)	9890.7	9486.6	7545.6	9516.7	7568.6
(huge:1;high;correlated)	25226.6	24725.9	19206.9	24769	19239.6
(huge:2;low;uncorrelated)	20297.3	11199.3	82612	11251.4	8309.1
(huge:2;medium;uncorrelated)	20329.5	10588.8	8503.8	10636.8	8538.9
(huge:2;high;uncorrelated)	31747.5	21460.7	13784.2	21510	13823.8
(huge:2;low;correlated)	20194	10536.1	9716.6	10589.2	9761.1
(huge:2;medium;correlated)	21425.8	11743.4	9263.5	11788.9	9303.1
(huge:2;high;correlated)	29462.5	19449	12058.5	19501.6	12102.7
(huge:4;low;uncorrelated)	69611.1	36064	28693.4	36179.3	28778.9
(huge:4;low;correlated)	73063.6	39066.9	30737.4	39173	30822.5
(huge:8;low;uncorrelated)	292411.3	63015.8	53864.1	63219.6	54037.5
(huge:8;low;correlated)	270696.8	68026.4	54674	68235.8	54839.7

- Additional efficiency in terms of solution quality without incurring huge computational overhead requires the implementation of further advanced techniques. One strategy that can be studied as a subject of future research is the decomposition of the problem such that the optimization procedure is directed to solve the decomposed (smaller sized) problems. The idea will be discussed in Chapter 6.

5.6 Summary

This chapter addressed the optimality of the truck scheduling problem by developing two neighborhood search meta-heuristics to improve the solutions of the DRS+DF heuristics: tabu search (TS) and variable neighborhood search (VNS). The search space of the problem was formed by changing the sequence of trailers for the door assignment. The neighborhood structure implemented was adjacent pair wise interchange which exchanges

the positions of two consecutive trailers in the solution sequence list. Being a neighborhood search method, each meta-heuristic was supplemented by a fractional evaluation of the neighborhood so that runtime is more efficiently dedicated to solution optimization. The proposed strategy was to evaluate each neighbor of a given solution using the information the solution already holds, wherever applicable, rather than performing the evaluation from scratch.

Extensive experiments were conducted to evaluate both the efficiency of the fractional neighborhood evaluation in reducing runtime and the two TS and VNS algorithms in improving the solutions of the DRS+DF heuristics. The experimental results demonstrated that the fractional evaluation achieved 60% to 70% reduction in the runtime of a local search (LS) algorithm after optimizing large and huge size problem instances. The experimentation continued by tuning the control parameters of the two meta-heuristics so that the trade-off between solution quality and runtime could be properly balanced. The TS and VNS algorithms were invoked to improve the DRS+DF solutions with two different values of the initial solution. In the first case, the first feasible solution found by DRS was considered the initial solution and in the second, the best solution obtained by the re-starting version of DRS (R-DRS) was assumed the initial solution. (In the latter case, the two meta-heuristics were denoted by R-TS and R-VNS, respectively.)

In either of the two cases, the TS and VNS algorithms performed equally well and were able to significantly improve the heuristics solutions in reasonable running times. It was also noticed that initializing the two meta-heuristics with a better solution led to a consistent robustness and increase in solution quality. The robustness is observed by comparing the variations in the value of relative difference of the improved solutions from the DRS+DF heuristics solutions, which have been reduced for the R-TS and R-VNS algorithms. In conclusion, the combination of R-DRS with TS and VNS proves to be efficient in terms of solution quality and runtime provided that establishing solution

feasibility does not lead to a bottleneck. Figure 5.8 compares the average performance of R-DRS, TS, VNS, and their re-starting variants (R-TS and R-VNS) in improving the solutions of the DRS+DF heuristics.

The solutions improved by the TS and VNS algorithms (and their hybrid variants R-TS and R-VNS) were compared with those of a multi-start local search (M-LS) algorithm. The comparison showed that the proposed meta-heuristics have been successful in escaping local optima and finding high quality solutions for most of the small, medium, large and some of the huge size problem instances in reasonable running times. The better performance of M-LS compared to TS and VNS in terms of solution quality (in ranges of 2% to 4% relative difference) for some large and huge size instances was mainly due to the substantial amount of time taken by M-LS to find the local optima. This clearly indicates that the trade-off between solution quality and runtime has been properly balanced by both TS and VNS meta-heuristics. The above capability for the VNS algorithm, however, has been primarily accomplished by using the solution time of the TS algorithm, which is fed into VNS as an input parameter. This implies that the current implementation of the TS meta-heuristic is a better candidate to be applied in an industrial setting.

Figure 5.8: Relative difference of the solutions of R-DRS, TS, VNS, R-TS, and R-VNS from the solutions of the DRS+DF heuristics

Chapter 6

Conclusion and Future Work

This thesis addressed the truck scheduling problem in a crossdocking terminal whose resources including doors, forklifts, and workers were assumed limited and non-preemptive. The crossdocking model, while having practical instances in the broad scope of the FMCG industry and military logistics, was purposefully specialized to visualize truck scheduling in the context of an integer program. Since the non-preemptive door assignment incurs complexities in the feasibility of truck scheduling, as the problem might be entrapped in deadlock and no feasible solution is produced, establishing solution feasibility was investigated before optimality. The final effort was then to improve the feasible solutions found for truck scheduling problem instances of various types and difficulty levels. The aim of this chapter is to present a summary of the accomplished work and introduce a few ideas based on the limitations of the current study as subjects of future research.

6.1 Research Accomplishments

Industrial practicality has been the key element in conducting this research both from the modeling and algorithmic perspectives. Nevertheless, direct exploitation of the findings and accomplishments in a real crossdocking scenario requires careful consultation with an industrial partner. For example, if in a real crossdock, the number of forklifts is fewer than the number of doors, the modeling needs to be slightly modified so that a forklift

may be designated to two dock doors or more. In that case, the solution algorithms should be modified accordingly.

Considering the related work on crossdocking and more specifically truck scheduling, the accomplishments are summarized as follows:

- Truck scheduling was formulated in a crossdocking model with unique and realistic operational settings. The proposed model considers pallets as the unit of shipments exchanged between the trailers (the common case in real crossdocks) and imposes constraints on the number of doors and forklifts as the crossdock critical resources. The allocation of doors and forklifts is also assumed non-preemptive throughout crossdocking, which is normally desired by terminal managers.
- The experimental data, though synthetically generated, entail a broad range of characteristics to represent the diversity exists in real world settings. These characteristics include the problem size in terms of the number of doors, the ratio of the number of trailers to the number of doors, freight distribution patterns between trailers, and correlation status of interdependent trailers. This has led to a data set containing problem instances of different levels of difficulty representing small crossdocks hosting a few dozen trailers to huge ones hosting over 2000 trailers with various freight exchange patterns.
- The algorithmic approach to address the truck scheduling feasibility is capable of generating feasible solutions for problem instances of various types and difficulty levels which at the same time can be readily implemented in an industrial setting. The proposed approach was a two-phase heuristic algorithm where, in the first phase, a dependency ranking search (DRS) heuristic was implemented to construct a feasible sequence of trucks for the assignment to dock doors and, in the second, a door fitness (DF) heuristic was used to assign each sequenced truck to a

proper dock door, subject to a limited number of forklifts equal to the total number of doors, such that significant savings in the makespan could be achieved. Of the total 960 test instances, DRS was able to establish solution feasibility for 644 instances within a two hour time limit. The findings, when compared by the percentage of feasible solutions guaranteed by DRS for a very small subset of instances, demonstrated that DRS performs comparably in establishing solution feasibility for various characteristics of input data.

- The optimality of the truck scheduling problem was addressed by developing two meta-heuristics: tabu search (TS) and variable neighborhood search (VNS). The objective was to improve the solutions generated by the DRS+DF heuristics. Significant work was carried out to enhance runtime efficiency of the proposed algorithms to justify their implementation in an industrial setting. First, as a neighborhood search method, each meta-heuristic was supplemented by a fractional evaluation of the neighborhood in order that the runtime was more efficiently dedicated to the solution optimization. The proposed methodology was to evaluate each neighbor of a given solution using the information the solution already holds, wherever applicable, rather than performing the evaluation from scratch. Second was tuning the control parameters of the two meta-heuristics so that the trade-off between solution quality and runtime could be properly balanced. All of this led to a consistent reduction in the makespan in reasonable running times from 20% to 28% on average for different characteristics of the experimental data. The average time taken by the TS algorithm to improve a solution in the extreme instance of scheduling 2048 trailers in a 256-door crossdock is nearly four hours (with standard deviation of 1.5 hour) per run. Further robustness and increase in the solution quality were achieved when the two meta-heuristics were initialized with a better initial solution obtained by the re-starting version of the DRS algorithm (the amount of reduction

in the makespan ranges from 23% to 29% on average). The robustness is observed by comparing the variations in the value of relative difference of the improved solutions from the DRS+DF heuristics solutions, which have been reduced for the latter case.

The aforementioned research accomplishments have led to a number publications that are listed in Appendix D.

6.2 Future Research Suggestions

Seven subjects of future research are suggested based on the limitations of the current study both in the modeling and algorithmic designs. Of the seven topics, two deal with the modeling methodology of crossdocking and the other five relate to the truck scheduling algorithmic design:

- The truck scheduling problem should be re-investigated by imposing a limit on the amount of space reserved in front of each door of the crossdock. In the analysis conducted in Section 3.4 of Chapter 3, it was observed that the allocated space in front of dock doors might overrun the assigned limit. It is thus necessary to avoid space overrun while truck scheduling is in progress. One applicable strategy is to simultaneously monitor the number of products staged in front of each dock door. In case overrun occurs, a procedure is invoked to move the excess load to the appropriate locations in which sufficient space is available. A post-processing procedure can also be used to balance the number of staged products in front of each door by utilizing forklift idle times. The utilization may even help to reduce the makespan by pre-moving pallets to their destination doors.
- The MIP model formulated in Chapter 3 consists of too many variables, making it impractical for the CPLEX solver to establish solution optimality even for small

size problem instances. Originally invented by Ford and Fulkerson [77] in 1958, column generation is a technique to handle such mathematical models with a huge number of variables. Successful implementations of column generation for solving a wide range of industrial problems to optimality or to near optimality have been reported since then (e.g., see [77–82]). The idea is to begin with a small, manageable part of the problem (which involves a few of the entire variables), solving that part, analyzing that partial solution to discover the next part of the problem (by adding one or more variables to the model), and then re-solving the enlarged model. Column generation repeats that process until it achieves a satisfactory solution to the whole problem. In this respect, the truck scheduling solution procedure by the CPLEX solver can start with a small set of trailer-to-door assignment and pallet moving patterns (represented by decision variables x_{jm} and $\delta_{pp'}$ in the MIP formulation, respectively). It is checked afterwards whether the solution can be improved by adding a new pattern. This continues till no improvement can be obtained.

- The analysis described in Section 4.3.3.1 of Chapter 4 reveals that the DRS efficiency in finding feasible solutions is heavily dependent on the choice of the initial cluster of trailers. To further improve the robustness of the DRS algorithm in establishing solution feasibility, the current ranking heuristic rules can be supplemented by a few more ranking functions of different characteristics. An intelligent randomization function is then designed to choose the rules for ranking the clusters based on the status of the search. In other words, the selection of each ranking function is subject to a probability whose value is dynamically updated by the history of the search. This indicates that the choice of the initial cluster is adaptive (and not deterministic) to the input data. The important issue is how to define and adjust

the ranking criteria of the functions such that they continually complement each other throughout the search.

- In the design of both TS and VNS algorithms, the sampling strategy to randomly evaluate a subset of neighbors of a given solution may lead to loss of good solutions throughout the search. An effective technique to overcome this limitation is to add an intensification step without sampling on cases where the algorithm is exploring different regions of the search space. The idea is applicable to both probabilistic TS and the reduced LS of VNS algorithm.
- The experimental results obtained in Chapter 5 indicate that additional improvement in the solution quality of the proposed TS and VNS meta-heuristics cannot be achieved in reasonable running times for large and huge size instances unless further advanced techniques are combined with the current basic implementation. One proposed strategy is to decompose the problem into a few smaller sized sub-problems and direct the optimization procedure to solve the decomposed problems. It is expected that the decomposition technique can be successfully applied to complex large and huge size instances where $\alpha = 8$ and FML is low. This is because the number of interdependent trailers is generally low for such instances and, according to the statistics recorded in the history data files, they are normally placed at nearby positions in the solution sequence list. It is thus possible to restrict the search exploration by evaluating a few clusters of interdependent trailers, respectively, rather than evaluating the entire trailers. On the contrary, the decomposition strategy may not be straightforward for the instances of the high mixed level. It was noticed in some (huge:2;high) instances that the interdependent trailers had been separated from each other by as many as 300 positions while the total number of trailers was 512.

- There are indeed three interrelated sub-problems embedded in the truck scheduling problem addressed in this thesis (see Section 3.2 of Chapter 3). The optimization framework proposed in Chapter 5, however, restricts the search space on the first sub-problem of truck scheduling which decides on the sequence of trailers for the door assignment. Further improvement in the solution quality of the problem may be achieved if the search space is extended to incorporate the other two sub-problems (i.e., to also change the assigned doors of the trailers and moving orders of pallets to their destination trailers). In that case, both TS and VNS meta-heuristics are enhanced to three-phase optimization algorithms where in each phase, exploring the solution space is based on the characteristics of the corresponding sub-problem. The best solution obtained in each phase is then fed into the next phase as the initial solution. This cycle continues until the overall best solution no longer improves. It is nonetheless required that the practicality of the proposed optimization model be carefully examined especially for complex large and huge size problem instances.
- The strategies proposed in Section 5.4 of Chapter 5 to assess the quality of the solutions generated by the TS and VNS algorithms could not tell the reader how far they are from optimality as all the comparisons have been relative to one another. On the other hand, finding optimal solutions are almost impractical for the existing problem instances. An alternative approach is to use “geometrically constructed solutions”, for which optimal or near-optimal solutions are apparent, as benchmark solutions [67]. These specially designed problem instances can be applied to provide a good estimate of the error of each meta-heuristic tested.

In spite of the fact that both the solution feasibility and optimality of the studied truck scheduling problem are potential areas of further research, the problem itself can

be considered a practical case study in the domain of operations research (OR) to compare and evaluate the performance of different classes of meta-heuristics, in particular, evolutionary algorithms. It is readily possible to develop an evolution-based approach for the studied truck scheduling problem based on the optimization framework introduced in Chapter 5 and compare it with the TS and VNS meta-heuristics. The solution representation proposed in Section 5.1 can be used to represent an individual in an evolutionary algorithm. The initial population can be formed by restarting the DRS algorithm from different initial clusters according to the strategy introduced in Section 5.4.2. Variation operators including mutation and recombination to produce a new generation of individuals per algorithm iteration are defined on the sequence list of trailers. And finally, the fractional neighborhood evaluation mechanism developed in Section 5.2 can be used as the fitness function for evaluating each individual. In that case, the fitness evaluation of new individuals that have been created from old individuals can be carried out by using the information the old ones already hold. The comparison is then made by evaluating solution quality improvement vs. solution runtime.

References

- [1] Apte, U. and Viswanathan, S., “Strategic and technological innovations in supply chain management,” *International Journal of Manufacturing Technology & Management* **4**(3-4), 264–282 (2002).
- [2] Kinnear, E., “Is there any magic in cross-docking?,” *Supply Chain Management: An International Journal* **2**(2), 49–52 (1997).
- [3] Brockmann, T., “21 warehousing trends in the 21st century,” *IIE Solutions* **31**(7), 36–40 (1999).
- [4] Gue, K. R., “Cross-docking: Just-In-Time for distribution,” teaching notes, Graduate School of Business and Public Policy, Naval Postgraduate School, Monterey, CA 93943 (2001).
- [5] Katz, J., “Meeting at the crossdock,” *Industry Week* **255**(5), 50 (2006).
- [6] Song, K. and Chen, F., “Scheduling cross docking logistics optimization problem with multiple inbound vehicles and one outbound vehicle,” in [*IEEE Int. Conf. Automation and Logistics, vols 1-6*], 3089–3094 (2007).
- [7] Apte, U. M. and Viswanathan, S., “Effective cross docking for improving distribution efficiencies,” *International Journal of Logistics: Research and Applications* **3**(3), 291–302 (2000).
- [8] Bozer, Y. A. and Carlo, H. J., “Optimizing inbound and outbound door assignments in less-than-truckload crossdocks,” *IIE Transactions* **40**(11), 1007–1018 (2008).
- [9] Boysen, N., Fliedner, M., and Scholl, A., “Scheduling inbound and outbound trucks at cross docking terminals,” *OR Spectrum* **32**(1), 135–161 (2010).

REFERENCES

- [10] Cohen, Y. and Keren, B., “A simple heuristic for assigning doors to trailers in cross-docks,” in [*Int. Conf. Industrial Logistics (ICIL 2008): Logistics in a Flat World: Strategy, Management and Operations*], 1–14 (2008).
- [11] Boysen, N. and Fliedner, M., “Cross dock scheduling: classification, literature review and research agenda,” *Omega-International Journal of Management Science* **38**(6), 413–422 (2010).
- [12] Gue, K. R., *Freight Terminal Layout and Operations*, PhD thesis, Georgia Institute of Technology, Atlanta, Georgia (1995).
- [13] Li, Z., Low, M. Y. H., Lim, Y. G., and Ma, B., “Optimal decision-making on product ranking for crossdocking/warehousing operations,” in [*6th IEEE Int. Conf. Industrial Informatics, vols 1-3*], 836–841 (2008).
- [14] Li, Z., Low, M. Y. H., Sim, C. H., and Lim, Y. G., “Optimal product allocation for crossdocking and warehousing operations in FMCG supply chain,” in [*IEEE Int. Conf. Service Operations and Logistics, and Informatics, vols 1-2*], 2963–2968 (2008).
- [15] Li, Z., Sim, C. H., He, W., and Chen, C. C., “A solution for cross-docking operations planning, scheduling and coordination,” in [*IEEE Int. Conf. Service Operations and Logistics, and Informatics, vols 1-2*], 2957–2962 (2008).
- [16] Deshpande, P. J., Yalcin, A., Zayas-Castro, J., and Herrera, L. E., “Simulating less-than-truckload terminal operations,” *Benchmarking* **14**(1), 92–101 (2007).
- [17] Alpan, G., Larbi, R., and Penz, B., “A bounded dynamic programming approach to schedule operations in a cross docking platform,” *Computers & Industrial Engineering* **60**(3), 385–396 (2011).
- [18] Alpan, G., Ladier, A.-L., Larbi, R., and Penz, B., “Heuristic solutions for transshipment problems in a multiple door cross docking warehouse,” *Computers & Industrial Engineering* **61**(2), 402–408 (2011).

REFERENCES

- [19] Peck, K. E., *Operational Analysis of Freight Terminals Handling Less Than Container Load Shipments*, PhD thesis, University of Illinois at Urbana-Champaign, Urbana, Illinois (1983).
- [20] Stephan, K. and Boysen, N., “Vis-a-vis vs. mixed dock door assignment: A comparison of different cross dock layouts,” *Operations Management Research* **4**(3-4), 150–163 (2011).
- [21] Boysen, N., “Truck scheduling at zero-inventory cross docking terminals,” *Computers & Operations Research* **37**(1), 32–41 (2010).
- [22] Boloori Arabani, A. R., Fatemi Ghomi, S. M. T., and Zandieh, M., “A multi-criteria cross-docking scheduling with just-in-time approach,” *International Journal of Advanced Manufacturing Technology* **49**(5-8), 741–756 (2010).
- [23] Boloori Arabani, A. R., Zandieh, M., and Fatemi Ghomi, S. M. T., “Multi-objective genetic-based algorithms for a cross-docking scheduling problem,” *Applied Soft Computing* **11**(8), 4954–4970 (2011).
- [24] Blazewicz, J., Ecker, K., Pesch, E., Schmidt, G., and Weglarz, J., [*Handbook on Scheduling: From Theory to Applications*], Springer, Berlin (2007).
- [25] Gue, K. R., “The effects of trailer scheduling on the layout of freight terminals,” *Transportation Science* **33**(4), 419–428 (1999).
- [26] Tsui, L. Y. and Chang, C.-H., “A microcomputer based decision support tool for assigning dock doors in freight yards,” *Computers & Industrial Engineering* **19**(1-4), 309–312 (1990).
- [27] Tsui, L. Y. and Chang, C.-H., “An optimal solution to a dock door assignment problem,” *Computers & Industrial Engineering* **23**(1-4), 283–286 (1992).
- [28] Bartholdi, J. and Gue, K., “Reducing labor costs in an LTL crossdocking terminal,” *Operations Research* **48**(6), 823–832 (2000).

REFERENCES

- [29] Bermudez, R. and Cole, M., “A genetic algorithm approach to door assignments in break-bulk terminals,” Technical Report MBTC-1102, Mack Blackwell Transportation Center, University of Arkansas, Fayetteville, Arkansas (2001).
- [30] Wang, J.-F. and Regan, A., “Real-time trailer scheduling for crossdock operations,” *Transportation Journal* **47**(2), 5–20 (2008).
- [31] Cohen, Y. and Keren, B., “Trailer to door assignment in a synchronous cross-dock operation,” *International Journal of Logistics Systems & Management* **5**(5), 574–590 (2009).
- [32] Li, Y., Lim, A., and Rodrigues, B., “Crossdocking - JIT scheduling with time windows,” *Journal of the Operational Research Society* **55**(12), 1342–1351 (2004).
- [33] Alvarez-Perez, G. A., Gonzalez-Velarde, J. L., and Fowler, J. W., “Crossdocking-Just in Time scheduling: an alternative solution approach,” *Journal of the Operational Research Society* **60**(4), 554–564 (2009).
- [34] Chen, F. and Lee, C.-Y., “Minimizing the makespan in a two-machine cross-docking flow shop problem,” *European Journal of Operational Research* **193**(1), 59–72 (2009).
- [35] Chen, F. and Song, K., “Minimizing makespan in two-stage hybrid cross docking scheduling problem,” *Computers & Operations Research* **36**(6), 2066–2073 (2009).
- [36] Shakeri, M., Low, M. Y. H., Li, Z., and Lee, E. W., “Two efficient constructive heuristics for scheduling trucks at crossdocking terminals,” in [*IEEE Int. Conf. Service Operations and Logistics, and Informatics*], 177–182 (2010).
- [37] Shakeri, M., Low, M. Y. H., and Lee, E. W., “Development of a novel resource-constrained crossdocking model for the truck scheduling problem,” in [*IEEE Int. Conf. Emerging Technologies and Factory Automation*], 1–9 (2010).
- [38] Yu, W. and Egbelu, P. J., “Scheduling of inbound and outbound trucks in cross docking systems with temporary storage,” *European Journal of Operational Research* **184**(1), 377–396 (2008).

REFERENCES

- [39] Vahdani, B., Soltani, R., and Zandieh, M., “Scheduling the truck holdover recurrent dock cross-dock problem using robust meta-heuristics,” *International Journal of Advanced Manufacturing Technology* **46**(5-8), 769–783 (2010).
- [40] Soltani, R. and Sadjadi, S. J., “Scheduling trucks in cross-docking systems: A robust meta-heuristics approach,” *Transportation Research Part E-Logistics and Transportation Review* **46**(5), 650–666 (2010).
- [41] Vahdani, B. and Zandieh, M., “Scheduling trucks in cross-docking systems: robust meta-heuristics,” *Computers & Industrial Engineering* **58**(1), 12–24 (2010).
- [42] Boloori Arabani, A. R., Ramtin, F., and Rafienejad, S. N., “Applying simulated annealing algorithm for cross-docking scheduling,” in [*WCECS 2009: World Congress on Engineering and Computer Science, Vols I and II*], 1248–1252 (2009).
- [43] Boloori Arabani, A. R., Fatemi Ghomi, S. M. T., and Zandieh, M., “Meta-heuristics implementation for scheduling of trucks in a cross-docking system with temporary storage,” *Expert Systems with Applications* **38**(3), 1964–1979 (2011).
- [44] Lim, A., Ma, H., and Miao, Z., “Truck dock assignment problem with time windows and capacity constraint in transshipment network through crossdocks,” in [*Int. Conf. Computational Science and Its Applications (ICCSA 2006), Pt. 3, Vol. 3982*], 688–697 (2006).
- [45] Lim, A., Ma, H., and Miao, Z., “Truck dock assignment problem with operational time constraint within crossdocks,” in [*19th Int. Conf. Industrial, Engineering and Other Applications of Applied Intelligent Systems, Vol. 4031*], 262–271 (2006).
- [46] Miao, Z., Lim, A., and Ma, H., “Truck dock assignment problem with operational time constraint within crossdocks,” *European Journal of Operational Research* **192**(1), 105–115 (2009).
- [47] Ley, S. and Elfayoumy, S., “Cross dock scheduling using genetic algorithms,” in [*Int. Symp. Computational Intelligence in Robotics and Automation*], 540–544 (2007).

REFERENCES

- [48] Bartholdi, J. and Gue, K., “The best shape for a crossdock,” *Transportation Science* **38**(2), 235–244 (2004).
- [49] Yu, W., *Operational Strategies for Cross Docking Systems*, PhD thesis, Iowa State University, Ames, Iowa (2002).
- [50] Chen, R., Fan, B., and Tang, G., “Scheduling Problems in Cross Docking,” in [*3rd Int. Conf. Combinatorial Optimization and Applications, Vol. 5573*], 421–429 (2009).
- [51] Acar, M., *Robust dock assignments at less-than-truckload terminals*, Master’s thesis, University of South Florida (2004).
- [52] Acar, K., Yalcin, A., and Yankov, D., “Robust door assignment in less-than-truckload terminals,” *Computers & Industrial Engineering* (2012). <http://dx.doi.org/10.1016/j.cie.2012.04.008>.
- [53] Johnson, S. M., “Optimal two- and three-stage production schedules with setup times included,” *Naval Research Logistics Quarterly* **1**(1), 61–68 (1954).
- [54] Burke, E. K. and Kendall, G., eds., [*Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*], Springer Science+Business Media, LLC (2005).
- [55] Montgomery, D. C., [*Design and Analysis of Experiments*], John Wiley & Sons, Inc., 7th ed. (2009).
- [56] Li, Z., Low, M. Y. H., and Ghai, J. S., “Optimal product mixing and matching for crossdocking operations,” internal technical report, Singapore Institute of Manufacturing Technology (SIMTech), Singapore 638075 (2008).
- [57] Yu, F.-T., *Door Allocation Problem at Intermediate Consolidation Terminals of Less-Than-Truckload Motor Carriers*, PhD thesis, University of Michigan, Ann Arbor, MI 48109 (2004).

REFERENCES

- [58] Brown, A. M., *Improving the efficiency of hub operations in a less-than-truckload distribution network*, Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia (2003).
- [59] Cela, E., [*The Quadratic Assignment Problem; Theory and Algorithms*], Kluwer Academic Publishers (1998).
- [60] Sahni, S. and Gonzalez, T., "P-complete approximation problems," *Journal of the ACM* **23**(3), 555–565 (1976).
- [61] Garey, M. R. and Johnson, D. S., [*Computers and Intractability: A Guide to the Theory of NP-Completeness*], W. H. Freeman & Co., New York, NY, USA (1979).
- [62] Roelofs, M. and Bisschop, J., "The AIMMS Language Interface." <http://www.aimms.com/downloads/manuals/language-reference> (Apr. 2010).
- [63] Napolitano, M., [*Making the Move to Cross Docking: A Practical Guide to Planning, Designing and Implementing a Cross Dock Operation*], Warehousing Education and Research Council (WERC) (2000).
- [64] Tarjan, R. E., "Depth- first search and linear graph algorithms," *SIAM Journal on Computing* **1**(2), 146–160 (1972).
- [65] Glover, F. and Kochenberger, G. A., eds., [*Handbook of Metaheuristics*], International Series in Operations Research & Management Science, Kluwer Academic Publishers, first ed. (2003).
- [66] Gendreau, M. and Potvin, J.-Y., eds., [*Handbook of Metaheuristics*], International Series in Operations Research & Management Science, Springer Science+Business Media, LLC, second ed. (2010).
- [67] Silberholz, J. and Golden, B., [*Handbook of Metaheuristics*], ch. Comparison of Metaheuristics, 625–640, International Series in Operations Research & Management Science, Springer Science+Business Media, LLC, second ed. (2010).
- [68] Glover, F., "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research* **13**(5), 533–549 (1986).

REFERENCES

- [69] Gendreau, M. and Potvin, J.-Y., [*Handbook of Metaheuristics*], ch. Tabu Search, 41–59, International Series in Operations Research & Management Science, Springer Science+Business Media, LLC, second ed. (2010).
- [70] Rolland, E., Schilling, D., and Current, J., “An efficient tabu search procedure for the p-Median Problem,” *European Journal of Operational Research* **96**(2), 329–342 (1997).
- [71] Ghosh, D., “Neighborhood search heuristics for the uncapacitated facility location problem,” *European Journal of Operational Research* **150**(1), 150–162 (2003).
- [72] Mladenovic, N. and Hansen, P., “Variable neighborhood search,” *Computers & Operations Research* **24**(11), 1097–1100 (1997).
- [73] Hansen, P. and Mladenovic, N., “Variable neighborhood search for the p-median,” *Location Science* **5**(4), 207–226 (1997).
- [74] Hansen, P., Mladenovic, N., Brimberg, J., and Moreno Perez, J. A., [*Handbook of Metaheuristics*], ch. Variable Neighborhood Search, 61–86, International Series in Operations Research & Management Science, Springer Science+Business Media, LLC, second ed. (2010).
- [75] Mart, R., Marcos Moreno-Vega, J., and Duarte, A., [*Handbook of Metaheuristics*], ch. Advanced Multi-Start Methods, 265–281, International Series in Operations Research & Management Science, Springer Science+Business Media, LLC, second ed. (2010).
- [76] Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W. P., and Vance, P. H., “Branch-and-price: Column generation for solving huge integer programs,” *Operations Research* **46**(3), 316–329 (1998).
- [77] Ford Jr., L. R. and Fulkerson, D. R., “A suggested computation for maximal multi-commodity network flows,” *Management Science* **5**(1), 97–101 (1958).
- [78] Leventhal, T., Nemhauser, G., and Trotter Jr., L., “Column generation algorithm for optimal traffic assignment,” *Transportation Science* **7**(2), 168–176 (1973).

REFERENCES

- [79] Lavoie, S., Minoux, M., and Odier, E., “A new approach for crew pairing problems by column generation with an application to air transportation,” *European Journal of Operational Research* **35**(1), 45–58 (1988).
- [80] Vanderbeck, F., “Computational study of a column generation algorithm for bin packing and cutting stock problems,” *Mathematical Programming, Series B* **86**(3), 565–594 (1999).
- [81] Choi, E. and Tcha, D. W., “A column generation approach to the heterogeneous fleet vehicle routing problem,” *Computers & Operations Research* **34**(7), 2080–2095 (2007).
- [82] Riera-Ledesma, J. and Salazar-Gonzalez, J., “A column generation approach for a school bus routing problem with resource constraints,” *Computers & Operations Research* **40**(2), 566–583 (2013).

Appendix A

Detailed Presentation of Results

Table A.1: DRS vs. DR_b for small size instances when FML is kept low and α varies

	DR _b +DF		DRS+DF				<i>r. d. (%)</i>
	<i>C</i> _{max}	TT(s)	<i>C</i> _{max}	TT(s)	Bt	Rs	
(small:1;low;uncorrelated)							
1	62	0.19	32	0.17	0	0	-48.39
2	53	0.08	53	0.08	0	0	0
3	46	0.06	46	0.11	0	0	0
4	70	0.06	38	0.16	0	0	-45.71
5	40	0.06	54	0.19	0	0	35
6	70	0.09	38	0.09	0	0	-45.71
7	70	0.06	46	0.08	0	0	-34.29
8	38	0.06	32	0.22	0	0	-15.79
9	28	0.11	35	0.05	0	0	25
10	54	0.06	54	0.12	0	0	0
avg.	53.1	0.08	42.8	0.13	0	0	-19.4
(small:2;low;uncorrelated)							
1	114	0.16	94	0.19	0	0	-17.54
2	101	0.08	104	0.11	0	0	2.97
3	113	0.09	114	0.12	0	0	0.88
4	116	0.08	114	0.12	0	0	-1.72
5	120	0.09	121	0.08	0	0	0.83
6	115	0.09	104	0.12	0	0	-9.56
7	104	0.12	117	0.11	0	0	12.5
8	115	0.08	122	0.12	0	0	6.09
9	105	0.11	120	0.14	0	0	14.29
10	102	0.08	107	0.17	0	0	4.9
avg.	110.5	0.1	111.7	0.13	0	0	1.09
(small:4;low;uncorrelated)							
1	239	0.2	259	0.3	0	0	8.37
2	241	0.19	231	0.22	0	0	-4.15
3	260	0.19	216	0.22	0	0	-16.92
4	248	0.19	266	0.17	0	0	7.26
5	272	0.16	263	0.16	0	0	-3.31
6	296	0.19	265	0.17	0	0	-10.47
7	301	0.17	276	0.17	0	0	-8.31
8	306	0.16	352	0.17	0	0	15.03
9	273	0.17	238	0.31	0	0	-12.82
10	260	0.16	257	0.23	0	0	-1.15
avg.	269.6	0.18	262.3	0.21	0	0	-2.71
(small:8;low;uncorrelated)							
1	1288	0.28	1141	0.42	1	0	-11.41
2	n/a	0.14	886	0.34	0	0	-
3	1091	0.22	871	0.28	0	0	-20.16
4	814	0.3	887	0.28	0	0	8.97
5	752	0.26	697	0.26	0	0	-7.31
6	500	0.28	520	0.28	0	0	4
7	584	0.26	750	0.26	0	0	28.42
8	959	0.3	773	0.25	0	0	-19.39
9	738	0.25	739	0.25	0	0	0.13
10	986	0.2	736	0.33	0	0	-25.35
avg.	856.9	0.26	790.4	0.29	0.1	0	-7.75
(small:1;low;correlated)							
1	62	0.14	35	0.14	0	0	-2.78
2	53	0.06	46	0.12	0	0	27.78
3	46	0.06	54	0.09	0	0	31.71
4	70	0.05	30	0.06	0	0	-57.14
5	40	0.06	73	0.09	0	0	78.05
6	70	0.06	30	0.09	0	0	-56.52
7	70	0.06	36	0.06	0	0	-48.57
8	38	0.06	54	0.09	0	0	31.71
9	28	0.05	46	0.11	0	0	0
10	54	0.09	70	0.06	0	0	0
avg.	53.1	0.07	47.4	0.09	0	0	-8.85
(small:2;low;correlated)							
1	114	0.11	122	0.14	0	0	3.39
2	101	0.08	108	0.11	0	0	-11.47
3	113	0.08	111	0.12	0	0	15.62
4	116	0.09	114	0.11	0	0	10.68
5	120	0.09	106	0.09	0	0	-7.02
6	115	0.09	105	0.16	0	0	7.14
7	104	0.14	115	0.14	0	0	-5.74
8	115	0.09	100	0.17	0	0	-1.96
9	105	0.09	102	0.11	0	0	0
10	102	0.12	118	0.14	0	0	-9.23
avg.	110.7	0.1	110.1	0.13	0	0	-0.54
(small:4;low;correlated)							
1	239	0.25	296	0.37	0	0	9.63
2	241	0.17	329	0.22	0	0	35.39
3	260	0.16	262	0.25	0	0	-2.6
4	248	0.17	264	0.17	0	0	-7.37
5	272	0.14	282	0.14	0	0	-1.4
6	296	0.17	263	0.22	0	0	-18.07
7	301	0.17	268	0.25	0	0	-11.84
8	306	0.16	244	0.19	0	0	-2.4
9	273	0.17	244	0.12	0	0	-10.95
10	260	0.2	337	0.14	0	0	-3.44
avg.	285.1	0.18	278.9	0.21	0	0	-2.17
(small:8;low;correlated)							
1	1288	0.33	696	0.39	0	0	-16.55
2	n/a	0.22	1229	0.28	2	0	25.15
3	1091	0.26	711	0.33	0	0	-28.11
4	814	0.23	570	0.34	0	0	-25.29
5	752	0.11	1078	0.31	0	0	-
6	500	0.3	799	0.33	0	0	-7.74
7	584	0.19	788	0.2	0	0	-19.01
8	959	0.11	1420	4.08	1002	0	-
9	738	0.23	642	0.36	0	0	0.94
10	986	0.28	942	0.28	0	0	42.08
avg.	856.9	0.25	797.1	0.31	0.2	0	-4.91

Table A.2: DRS vs. DR_b for medium size instances when FML is kept low and α varies

	DR _b +DF				DRS+DF				<i>r. d. (%)</i>	DR _b +DF				DRS+DF				<i>r. d. (%)</i>		
	<i>C</i> _{max}	TT(s)		<i>C</i> _{max}	Bt	Rs	<i>C</i> _{max}	TT(s)		<i>C</i> _{max}	Bt	Rs	<i>C</i> _{max}	TT(s)		<i>C</i> _{max}	Bt		Rs	
		(medium:1;low;uncorrelated)	(medium:2;low;uncorrelated)					(medium:1;low;correlated)						(medium:2;low;correlated)						
1	157	0.31	141	0.22	0	0	-10.19	123	0.31	133	0.2	0	0	8.13						
2	134	0.26	197	0.17	0	0	47.01	152	0.22	112	0.14	0	0	-26.32						
3	115	0.25	203	0.17	0	0	76.52	153	0.23	148	0.17	0	0	-3.27						
4	112	0.25	132	0.12	0	0	17.86	139	0.25	154	0.12	0	0	10.79						
5	123	0.25	158	0.19	0	0	28.45	135	0.25	127	0.2	0	0	-5.93						
6	152	0.23	165	0.14	0	0	8.55	89	0.2	127	0.11	0	0	42.7						
7	138	0.26	145	0.2	0	0	5.07	125	0.25	131	0.11	0	0	4.8						
8	139	0.22	188	0.16	0	0	35.25	131	0.22	164	0.09	0	0	25.19						
9	147	0.27	185	0.16	0	0	25.85	128	0.28	200	0.22	0	0	56.25						
10	141	0.25	140	0.11	0	0	-0.71	94	0.2	98	0.12	0	0	4.25						
avg.	135.8	0.26	165.4	0.16	0	0	21.8	126.9	0.24	139.4	0.15	0	0	9.85						
(medium:2;low;uncorrelated)																				
1	274	0.48	267	0.39	0	0	-2.55	315	0.48	252	0.39	0	0	-20						
2	259	0.44	229	0.3	0	0	-11.58	245	0.44	247	0.34	0	0	0.82						
3	241	0.45	283	0.33	0	0	17.43	250	0.44	240	0.28	0	0	-4						
4	268	0.41	241	0.26	0	0	-10.07	232	0.48	272	0.33	0	0	17.24						
5	261	0.42	27	0.26	0	0	3.06	235	0.47	272	0.36	0	0	15.74						
6	275	0.42	317	0.31	0	0	15.27	304	0.44	273	0.23	0	0	-10.2						
7	261	0.41	268	0.23	0	0	2.68	240	0.45	317	0.33	0	0	32.08						
8	254	0.47	211	0.39	0	0	-16.93	283	0.45	335	0.3	0	0	18.37						
9	216	0.44	266	0.26	0	0	23.15	213	0.45	272	0.31	0	0	27.7						
10	270	0.42	252	0.25	0	0	-6.67	234	0.47	265	0.33	0	0	13.25						
avg.	257.9	0.43	260.3	0.3	0	0	0.93	255.1	0.46	274.5	0.32	0	0	7.6						
(medium:4;low;uncorrelated)																				
1	537	0.91	611	0.78	0	0	13.78	622	0.84	555	0.76	0	0	-10.77						
2	548	0.91	47	0.7	0	0	-13.32	553	0.8	586	0.61	0	0	5.97						
3	561	0.87	517	0.67	0	0	-7.84	598	0.84	605	0.67	0	0	1.17						
4	612	0.84	532	0.59	0	0	-13.07	592	0.83	457	0.58	0	0	-22.8						
5	581	0.8	496	0.73	0	0	-14.63	453	0.81	495	0.78	0	0	9.27						
6	610	0.84	527	0.62	0	0	-13.61	537	0.8	468	0.64	0	0	-12.85						
7	502	0.91	617	0.69	0	0	22.91	447	0.91	539	0.66	0	0	20.58						
8	530	0.94	539	0.64	0	0	1.7	521	0.83	648	0.73	0	0	24.38						
9	524	0.86	566	0.69	0	0	8.01	647	0.84	520	0.73	0	0	-19.63						
10	554	0.86	589	0.7	0	0	6.32	537	0.81	505	0.72	0	0	-5.96						
avg.	555.9	0.87	546.9	0.68	0	0	-1.62	550.7	0.83	537.8	0.69	0	0	-2.34						
(medium:8;low;uncorrelated)																				
1	1428	1.83	1420	2.55	0	0	-0.56	1447	1.75	1185	1.89	0	0	-18.11						
2	1307	1.81	1481	1.78	0	0	13.31	1489	1.56	1324	1.83	0	0	-11.08						
3	1495	1.75	1607	1.83	0	0	7.49	1856	1.76	1414	1.89	0	0	-23.81						
4	1740	1.59	1143	1.77	0	0	-34.31	2088	1.72	1583	1.95	0	0	-24.19						
5	1429	1.76	1203	1.92	0	0	-15.81	1982	1.7	1340	2.2	0	0	-32.39						
6	1638	1.76	1155	2.44	0	0	-29.49	1541	1.69	1631	2.59	0	0	5.84						
7	1302	1.75	1097	1.97	0	0	-15.74	1665	1.73	1241	1.87	0	0	-25.46						
8	1388	1.67	1307	1.81	0	0	-5.84	1959	1.8	1473	1.91	0	0	-24.81						
9	1586	1.67	1435	2.03	0	0	-9.52	1480	1.7	1245	1.84	0	0	-15.88						
10	1506	1.75	1282	1.73	0	0	-14.87	1499	1.62	1283	1.83	0	0	-14.41						
avg.	1481.9	1.73	1313	1.98	0	0	-11.4	1700.6	1.7	1371.9	1.98	0	0	-19.33						

Table A.3: DRS vs. DR_b for large size instances when FML is kept low and α varies

	DR _b +DF		DRS+DF				<i>r. d. (%)</i>
	<i>C</i> _{max}	TT(s)	DR _b +DF		<i>r. d. (%)</i>		
			<i>C</i> _{max}	TT(s)		Bt	
(large:1;low;uncorrelated)							
1	354	1.33	449	0.72	0	0	26.84
2	430	1.33	596	0.69	0	0	38.6
3	463	1.28	506	0.73	0	0	9.29
4	394	1.22	341	0.67	0	0	-13.45
5	313	1.17	607	0.66	0	0	93.93
6	409	1.19	370	0.58	0	0	-9.53
7	346	1.36	622	0.73	0	0	79.77
8	398	1.2	425	0.64	0	0	6.78
9	486	1.28	479	0.69	0	0	-1.44
10	450	1.17	591	0.62	0	0	31.33
avg.	404.3	1.25	498.6	0.67	0	0	23.32
(large:2;low;uncorrelated)							
1	675	2.47	730	1.5	0	0	8.15
2	798	2.48	652	1.61	0	0	-18.3
3	657	2.37	692	1.47	0	0	5.33
4	789	2.45	865	1.5	0	0	9.63
5	662	2.56	828	1.51	0	0	25.07
6	706	2.51	643	1.51	0	0	-8.92
7	710	2.5	640	1.48	0	0	-9.86
8	679	2.44	865	1.51	0	0	27.39
9	730	2.41	659	1.69	0	0	-9.73
10	593	2.42	745	1.5	0	0	25.63
avg.	699.9	2.46	731.9	1.53	0	0	4.57
(large:4;low;uncorrelated)							
1	1502	4.94	1494	5.12	0	0	-0.53
2	1205	4.86	1185	4.25	0	0	-1.66
3	1327	4.76	1389	4.42	0	0	4.67
4	1445	4.75	1332	5.01	0	0	-7.82
5	1341	5.17	1142	4.47	0	0	-14.84
6	1293	4.92	1410	4.91	0	0	9.05
7	1447	4.91	1636	4.56	0	0	13.06
8	1306	4.97	1405	4.5	0	0	7.58
9	1284	4.48	1415	4.23	0	0	10.2
10	1464	4.73	2018	4.08	0	0	37.84
avg.	1361.4	4.85	1442.6	4.56	0	0	5.96
(large:8;low;uncorrelated)							
1	3246	10.01	2782	13.92	0	0	-14.29
2	3516	9.22	3495	14.25	0	0	-0.6
3	4094	9.55	3572	14.17	0	0	-12.75
4	3477	10	3004	14.01	0	0	-13.6
5	3374	9.92	2522	14.33	0	0	-25.25
6	4385	9.69	3479	14.73	0	0	-20.66
7	3898	9.59	2991	14.16	0	0	-23.27
8	3671	9.66	2893	16.56	0	0	-21.19
9	3107	9.62	2724	13.5	0	0	-12.33
10	3591	9.45	2959	14.31	0	0	-17.6
avg.	3635.9	9.67	3042.1	14.39	0	0	-16.33
(large:2;low;correlated)							
1	730	2.5	632	1.84	0	0	-13.42
2	757	2.47	617	1.72	0	0	-18.49
3	778	2.56	715	1.62	0	0	-8.1
4	690	2.45	749	1.48	0	0	8.55
5	752	2.45	898	1.66	0	0	19.41
6	783	2.42	763	1.69	0	0	-2.55
7	631	2.51	774	1.7	0	0	22.66
8	784	2.45	657	1.61	0	0	-16.2
9	757	2.36	780	1.78	0	0	3.04
10	667	2.41	802	1.62	0	0	20.24
avg.	732.9	2.46	738.7	1.67	0	0	0.79
(large:4;low;correlated)							
1	1234	4.66	1379	4.59	0	0	11.75
2	1365	4.84	1620	4.5	0	0	18.68
3	1094	5.12	1403	4.23	0	0	28.24
4	1536	5.12	1358	5.11	0	0	-11.59
5	1481	4.83	1386	5.16	0	0	-6.41
6	1485	4.84	1498	4.25	0	0	0.87
7	1579	4.92	1312	4.23	0	0	-16.91
8	1283	4.86	1357	4.37	0	0	5.77
9	1508	4.66	1414	4.64	0	0	-6.23
10	1304	4.67	1090	4.45	0	0	-16.41
avg.	1386.9	4.85	1381.7	4.55	0	0	-0.37
(large:8;low;correlated)							
1	3836	9.31	3645	14.41	0	0	-4.98
2	2908	9.69	2886	14.09	0	0	-0.76
3	3359	10.26	2673	15.97	0	0	-20.42
4	3765	9.44	3587	14.37	0	0	-4.73
5	3149	10.22	3376	13.98	0	0	7.21
6	3547	9.34	3022	14.31	0	0	-14.8
7	2853	10.19	2940	14.3	0	0	3.05
8	3349	9.52	2875	14.39	0	0	-14.15
9	3542	9.58	3326	16.75	0	0	-6.1
10	3652	9.78	3001	16.84	0	0	-17.83
avg.	3336	9.73	3133.1	14.94	0	0	-7.74

Table A.4: DRS vs. DR_b for huge size instances when FML is kept low and α varies

	DR _b +DF				DRS+DF				<i>r. d. (%)</i>	DR _b +DF				<i>r. d. (%)</i>
	<i>C</i> _{max}	TT(s)	<i>C</i> _{max}	TT(s)	Bt	Rs	<i>C</i> _{max}	TT(s)		<i>C</i> _{max}	TT(s)	Bt	Rs	
(huge:1;low;uncorrelated)														
1	1076	7.3	1646	3.89	0	0	52.97	1231	7.01	1739	3.76	0	0	41.27
2	1067	7.01	1991	3.76	0	0	86.6	1241	7.01	1522	3.59	0	0	22.64
3	1392	7.42	1218	3.75	0	0	-12.5	1340	7.09	1203	3.64	0	0	-10.22
4	1353	7.09	1732	3.83	0	0	28.01	1507	7.03	1516	3.67	0	0	0.6
5	1205	6.98	1429	3.78	0	0	18.59	1145	6.94	1420	4.22	0	0	24.02
6	1310	6.94	1124	3.81	0	0	-14.2	1388	7.25	1421	4.36	0	0	2.38
7	1146	7.16	1713	3.64	0	0	49.48	1413	7.22	1294	3.92	0	0	-8.42
8	1097	6.95	2074	3.7	0	0	89.06	1044	6.98	1150	3.7	0	0	10.15
9	1410	7	2154	3.58	0	0	52.77	1255	7.14	1151	3.72	0	0	-8.29
10	1351	7.08	1609	3.53	0	0	19.1	1144	7.08	1470	3.92	0	0	28.5
avg.	1240.7	7.09	1669	3.73	0	0	34.52	1270.8	7.08	1388.6	3.85	0	0	9.27
(huge:2;low;uncorrelated)														
1	2155	14.61	2375	11	0	0	10.21	2387	14.36	2420	9.92	0	0	1.38
2	1959	14.5	2437	9.45	0	0	24.4	2229	13.75	2330	10.03	0	0	4.53
3	2145	14.23	2700	9.36	0	0	25.87	2298	14.19	2230	9.75	0	0	-2.96
4	2080	14.26	1990	9.73	0	0	-4.33	1930	14.53	2210	10.75	0	0	14.51
5	1839	14.22	2123	9.83	0	0	15.44	1898	13.98	2068	9.86	0	0	8.96
6	2074	14.14	2068	9.937	0	0	-0.29	1841	14.01	2216	10	0	0	20.37
7	2107	13.98	2210	9.25	0	0	4.89	2124	13.8	1774	10.75	0	0	-16.48
8	1778	13.97	2749	9.62	0	0	54.61	2201	13.91	2287	12.28	0	0	3.91
9	2076	14.16	2564	9.91	0	0	23.51	2005	14.22	2085	10.53	0	0	3.99
10	1868	14.25	1780	9.31	0	0	-4.71	2124	13.84	3296	10.78	0	0	55.18
avg.	2008.1	14.23	2299.6	9.74	0	0	14.52	2103.7	14.06	2291.6	10.46	0	0	8.93
(huge:4;low;uncorrelated)														
1	3366	28.48	3820	28	0	0	13.49	3606	27.81	3932	33.51	0	0	9.04
2	3842	27.75	3819	29.62	0	0	-0.6	3470	28.53	3466	30.87	0	0	-0.11
3	3908	29.14	3876	29.78	0	0	-0.82	3385	28.8	3940	28.26	0	0	16.4
4	3587	28.64	3358	30.01	0	0	-6.38	3941	28.19	3888	27.98	0	0	-1.34
5	3664	27.81	4106	29.22	0	0	12.06	3387	28.48	4270	33.97	0	0	26.07
6	3350	28.86	4031	34.22	0	0	20.33	3782	28.48	3456	29.45	0	0	-8.62
7	4075	28.89	3990	28.59	0	0	-2.09	3878	28.25	4027	28.39	0	0	3.84
8	3671	28.61	3522	28.09	0	0	-4.06	3560	28.20	3726	30.09	0	0	4.66
9	3324	28.48	4219	30.14	0	0	26.92	3545	28.58	4376	33.09	0	0	23.44
10	3427	28.98	4038	27.7	0	0	17.83	3378	28.31	3215	28.62	0	0	-4.82
avg.	3621.4	28.56	3877.9	29.54	0	0	7.08	3593.2	28.36	3829.6	30.43	0	0	6.58
(huge:8;low;uncorrelated)														
1	8081	57.94	7527	95.87	0	0	-6.86	8756	58.31	7194	99.06	0	0	-17.84
2	9160	57.58	8518	98.22	0	0	-7.01	7964	57.34	7941	100.4	0	0	-0.29
3	8223	56.61	8109	120.6	0	0	-1.39	8103	58.06	8543	98.78	0	0	5.43
4	10009	56.5	7846	101	0	0	-21.61	9261	57.7	7836	100.7	0	0	-15.39
5	8005	58.08	8386	100.1	0	0	4.76	8940	56.59	7682	117.5	0	0	-14.07
6	8988	57.28	7697	98.67	0	0	-14.36	9260	58.69	7305	96.7	0	0	-21.11
7	9594	55.08	9130	102.6	0	0	-4.84	8143	58.84	7989	98.11	0	0	-1.89
8	8487	58.17	7608	101.5	0	0	-10.36	8393	58.25	7816	120.7	0	0	-6.87
9	9079	56.34	7527	101.5	0	0	-17.09	8529	56.81	7425	100.4	0	0	-12.94
10	7872	59.36	7379	116.2	0	0	-6.26	9787	55.75	8172	102.4	0	0	-16.5
avg.	8749.8	57.29	7972.7	103.6	0	0	-8.88	8713.6	57.64	7790.3	103.5	0	0	-10.6

Table A.5: Percentage (in %) of feasibility found by DRS and DR_b for small size instances

(a) Percentage of feasible solutions found by DRS

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	100
	correlated	100	100	100	100
medium	uncorrelated	100	100	0	0
	correlated	100	100	10	0
high	uncorrelated	100	60	0	0
	correlated	100	100	0	0

(b) Percentage of feasible solutions found by DR_b

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	90
	correlated	100	100	100	80
medium	uncorrelated	100	100	0	0
	correlated	100	100	0	0
high	uncorrelated	100	0	0	0
	correlated	100	20	0	0

Table A.6: Percentage (in %) of feasibility found by DRS and DR_b for medium size instances

(a) Percentage of feasible solutions found by DRS

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	100
	correlated	100	100	100	100
medium	uncorrelated	100	100	10	0
	correlated	100	100	20	0
high	uncorrelated	100	100	0	0
	correlated	100	100	0	0

(b) Percentage of feasible solutions found by DR_b

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	100
	correlated	100	100	100	100
medium	uncorrelated	100	100	0	0
	correlated	100	100	0	0
high	uncorrelated	100	10	0	0
	correlated	100	30	0	0

Table A.7: Percentage (in %) of feasibility found by DRS and DR_b for large size instances

(a) Percentage of feasible solutions found by DRS

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	100
	correlated	100	100	100	100
medium	uncorrelated	100	100	10	0
	correlated	100	100	30	0
high	uncorrelated	100	100	0	0
	correlated	100	100	0	0

(b) Percentage of feasible solutions found by DR_b

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	100
	correlated	100	100	100	100
medium	uncorrelated	100	100	0	0
	correlated	100	100	0	0
high	uncorrelated	100	60	0	0
	correlated	100	80	0	0

Table A.8: Percentage (in %) of feasibility found by DRS and DR_b for huge size instances

(a) Percentage of feasible solutions found by DRS

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	100
	correlated	100	100	100	100
medium	uncorrelated	100	100	0	0
	correlated	100	100	0	0
high	uncorrelated	100	100	0	0
	correlated	100	100	0	0

(b) Percentage of feasible solutions found by DR_b

FML	Correlation Status	Trailer-to-Door Ratio			
		$\alpha = 1$	$\alpha = 2$	$\alpha = 4$	$\alpha = 8$
low	uncorrelated	100	100	100	100
	correlated	100	100	100	100
medium	uncorrelated	100	100	0	0
	correlated	100	100	0	0
high	uncorrelated	100	100	0	0
	correlated	100	100	0	0

Table A.9: TS & VNS efficiency in improving DRS+DF solutions for large size instances of $\alpha \geq 4$

	DRS+DF		TS				VNS			
	$C_{\max}$	time(s)	$C_{\max}$	time(s)	$r. d. (\%)$	$f. r. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$	$f. r. (\%)$
(large:4;low;uncorrelated)										
1	1494	4.203	1011	6807.7	-32.33	100	1062	6827.7	-28.92	100
2	1185	4.094	931	6054.9	-21.43	100	971	6073.7	-18.06	100
3	1389	4.219	988	8501.5	-28.87	100	995	8523.8	-28.37	100
4	1332	4.156	995	6538	-25.3	100	950	6557.9	-28.68	100
5	1142	4.203	1028	7493.8	-9.98	100	927	7514.3	-18.83	100
6	1410	4.016	1013	6519.5	-28.16	100	974	6533.9	-30.92	100
7	1636	4.297	1084	5324.9	-33.74	100	1077	5344.4	-34.17	100
8	1405	4.266	1055	4733	-24.91	100	1012	4753.5	-27.97	100
9	1415	4.032	978	6627.9	-30.88	100	951	6650.1	-32.79	100
10	2018	3.875	965	7469.7	-52.18	100	980	7488.6	-51.44	100
avg	1442.6	4.136	1004.8	6607.1	-30.35	100	989.9	6626.8	-31.38	100
(large:4;medium;uncorrelated)										
5	17385	971.98	12907	9857.2	-25.76	89.27	11737	9866.9	-32.49	89.92
(large:4;low;correlated)										
1	1379	4.328	914	4406.7	-33.72	100	943	4427.4	-31.62	100
2	1620	4.219	1005	5137.9	-37.96	100	1008	5159.1	-37.78	100
3	1403	3.953	931	7100.1	-33.64	100	896	7119.4	-36.14	100
4	1358	4.156	1100	6268.6	-19	100	1034	6288.8	-23.86	100
5	1386	4.313	1040	5322.4	-24.96	100	1004	5342.5	-27.56	100
6	1498	4	1030	5523.7	-31.24	100	1059	5542.8	-29.31	100
7	1312	4.093	1044	6852.3	-20.43	100	1058	6872.8	-19.36	100
8	1357	4.093	969	5594.9	-28.59	100	1018	5615.4	-24.98	100
9	1414	4.313	989	5424.1	-30.06	100	981	5442.1	-30.62	100
10	1090	4.187	950	5939.5	-12.84	100	927	5956.3	-14.95	100
avg.	1381.7	4.165	997.2	5757	-27.83	100	992.8	5776.7	-28.15	100
(large:4;medium;correlated)										
3	16362	2471.5	11834	8545.2	-27.67	90.67	12094	8557.7	-26.08	90.03
4	18341	358.69	14005	9737.4	-23.64	87.69	14256	9749.4	-22.27	87.13
9	17322	6375.2	12875	8951	-25.67	88.66	12741	8962.4	-26.45	88.79
avg.	17341.7	3068.5	12904.7	9077.9	-25.59	88.93	13030.3	9089.8	-24.86	88.57
(large:8;low;uncorrelated)										
1	2782	13.59	2255	11077	-18.94	100	2211	11117.5	-20.52	100
2	3495	13.56	2559	10928.4	-26.78	100	2591	10966.3	-25.86	100
3	3572	13.485	2805	13295.9	-21.47	100	2759	13322.5	-22.76	100
4	3004	13.42	2288	15818.8	-23.83	100	2260	15855.7	-24.77	100
5	2522	14.28	2127	13847.2	-15.66	100	2132	13886.8	-15.46	100
6	3479	14.64	2799	12018.4	-19.54	100	2817	12050	-19.03	100
7	2991	13.69	2504	12037.3	-16.28	100	2539	12076.3	-15.11	100
8	2893	13.67	2426	20775.8	-16.14	100	2456	20811.9	-15.1	100
9	2724	12.89	2191	13571.6	-19.57	100	2279	13611.3	-16.34	100
10	2959	13.58	2703	9815.2	-8.65	100	2614	9847.9	-11.66	100
avg.	3042.1	13.68	2465.7	13318.6	-18.95	100	2465.8	13354.6	-18.94	100
(large:8;low;correlated)										
1	3645	14.03	2587	13300.7	-29.03	100	2552	13329.8	-29.99	100
2	2886	13.61	2531	13734.5	-12.3	100	2499	13768.5	-13.41	100
3	2673	13.05	2232	11481.3	-16.5	100	2312	11521.4	-13.5	100
4	3587	13.84	2630	16126.2	-26.68	100	2592	16165.5	-27.74	100
5	3376	13.34	2382	13483.3	-29.44	100	2317	13525.5	-31.37	100
6	3022	13.78	2402	8369.7	-20.52	100	2498	8406.17	-17.34	100
7	2940	13.98	2610	13528.1	-11.22	100	2505	13562.3	-14.79	100
8	2875	14.16	2422	13117	-15.76	100	2498	13148.7	-13.11	100
9	3326	13.27	2458	11154.5	-26.1	100	2357	11188.6	-29.13	100
10	3001	13.47	2673	13319.1	-10.93	100	2662	13352.6	-11.3	100
avg.	3133.1	13.65	2492.7	12761.4	-20.44	100	2479.2	12796.9	-20.87	100

Table A.10: TS & VNS efficiency in improving DRS+DF solutions for huge size instances of $\alpha \geq 4$

	DRS+DF		TS				VNS			
	$C_{\max}$	time(s)	$C_{\max}$	time(s)	$r. d. (\%)$	$f. r. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$	$f. r. (\%)$
(huge:4;low;uncorrelated)										
1	3820	27.7	2606	31321.7	-31.78	100	2739	31443.2	-28.3	100
2	3819	28.92	2733	31116.1	-28.44	100	2700	31220.6	-29.3	100
3	3876	29.05	2631	47110	-32.12	100	2640	47214.1	-31.89	100
4	3358	29.83	2737	37201.3	-18.49	100	2589	37320.9	-22.9	100
5	4106	28.34	2880	34300.5	-29.86	100	2860	34426.5	-30.34	100
6	4031	28.92	2649	34967	-34.28	100	2619	35062.9	-35.03	100
7	3990	27.81	2775	27843.8	-30.45	100	2723	27975.1	-31.75	100
8	3522	27.48	2705	45922.4	-23.2	100	2530	46034.1	-28.17	100
9	4219	30.86	2667	30247.9	-36.79	100	2742	30371.8	-35.01	100
10	4038	26.97	2768	40609.6	-31.45	100	2653	40724.1	-34.3	100
avg.	3877.9	28.59	2715.1	36064	-29.98	100	2679.5	36179.33	-30.9	100
(huge:4;low;correlated)										
1	3932	28.98	2614	43631.5	-33.52	100	2718	43759.7	-30.87	100
2	3466	30	2541	46117.8	-26.69	100	2592	46227.4	-25.22	100
3	3940	27.41	2681	31080.5	-31.95	100	2619	31173.6	-33.53	100
4	3888	27.37	2699	36147.4	-30.58	100	2717	36232.1	-30.12	100
5	4270	28.5	2698	35321.6	-36.81	100	2625	35439.8	-38.52	100
6	3456	28.81	2719	38442.2	-21.32	100	2793	38544.7	-19.18	100
7	4027	28.12	2651	33136.4	-34.17	100	2667	33249.1	-33.77	100
8	3726	29.28	2776	38201.8	-25.5	100	2786	38309.4	-25.23	100
9	4376	27.86	2590	43387.2	-40.81	100	2561	43492.7	-41.48	100
10	3215	28.25	2666	45202.4	-17.08	100	2621	45301.9	-18.48	100
avg.	3829.6	28.46	2663.5	39066.9	-30.45	100	2669.9	39173	-30.28	100
(huge:8;low;uncorrelated)										
1	7527	93.34	6063	60584.6	-19.45	100	6057	60780	-19.53	100
2	8518	97.41	6106	67948.8	-28.32	100	6357	68140.9	-25.37	100
3	8109	99.58	6053	63362.5	-25.35	100	6179	63558.7	-23.8	100
4	7846	99.12	6535	68370.2	-16.71	100	6303	68540.6	-19.67	100
5	8386	98.53	6058	58643.6	-27.76	100	5991	58858.4	-28.56	100
6	7697	99.06	6399	67056.7	-16.86	100	6251	67258.7	-18.79	100
7	9130	101.55	6907	51291.2	-24.35	100	6874	51501.6	-24.71	100
8	7608	98.87	5993	64871.2	-21.23	100	6102	65079.5	-19.79	100
9	7527	97.87	5987	60233.9	-20.46	100	6076	60428.2	-19.28	100
10	7379	95.52	5791	67795.7	-21.52	100	5799	68049.9	-21.41	100
avg.	7972.7	98.09	6189.2	63015.8	-22.37	100	6198.9	63219.6	-22.25	100
(huge:8;low;correlated)										
1	7194	95.41	5742	76185.7	-20.18	100	6094	76412.2	-15.29	100
2	7941	98.06	6241	54344.6	-21.41	100	6271	54556.8	-21.03	100
3	8543	97.37	6282	56108.5	-26.47	100	6247	56327.8	-26.88	100
4	7836	98.58	6010	68406.8	-23.3	100	6077	68585.7	-22.45	100
5	7682	96.16	6036	61032.4	-21.43	100	6132	61230.1	-20.18	100
6	7305	98.06	5811	62900.4	-20.45	100	5884	63115.1	-19.45	100
7	7989	95.01	6028	80091.8	-24.55	100	6068	80301.8	-24.05	100
8	7816	99	6203	73424.5	-20.64	100	6305	73665.6	-19.33	100
9	7425	99.03	5881	93761	-20.79	100	5833	93957.5	-21.44	100
10	8172	100.48	6865	54008.1	-15.99	100	6990	54205.3	-14.46	100
avg.	7790.3	97.72	6109.9	68026.4	-21.57	100	6190.1	68235.8	-20.54	100

Table A.11: R-TS & R-VNS efficiency in improving DRS+DF solutions after re-starting for large size instances of $\alpha \geq 4$

	DRS+DF	R-DRS+DF			R-TS			R-VNS		
	$C_{\max}$	$C_{\max}$	time(s)	$r. d. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$
(large:4;low;uncorrelated)										
1	1494	1235	41.69	-17.34	1101	5904.2	-26.3	1020	5919.5	-31.73
2	1185	1174	39.76	-0.93	965	4147.5	-18.56	964	4161.8	-18.65
3	1389	1259	41.72	-9.36	1038	4250.3	-25.27	999	4265.4	-28.08
4	1332	1148	39.86	-13.81	1049	4429.3	-21.25	1016	4445.3	-23.72
5	1142	1142	41.28	0	1035	5865.8	-9.37	989	5881.2	-13.4
6	1410	1171	39.95	-16.95	983	3490	-30.28	1006	3503.5	-28.65
7	1636	1340	41.86	-18.09	1067	4504.7	-34.78	1100	4518.4	-32.76
8	1405	1226	40.17	-12.74	1010	5218.9	-28.11	989	5232.9	-29.61
9	1415	1231	38.47	-13	902	5098.3	-36.25	909	5110.7	-35.76
10	2018	1167	37.75	-42.17	999	4898.7	-50.49	964	4912.9	-52.23
avg.	1442.6	1209.3	40.25	-16.17	1014.9	4780.8	-29.65	995.6	4795.2	-30.99
(large:4;medium;uncorrelated)										
5	17385	17385	8218.4	0	12502	6732.9	-28.09	12688	6744.2	-27.02
(large:4;low;correlated)										
1	1379	1134	39.37	-17.77	931	4366.6	-32.49	895	4381.7	-35.1
2	1620	1174	39.75	-27.53	922	6406.1	-43.09	976	6418.3	-39.75
3	1403	1151	38.75	-17.96	902	6197.2	-35.71	914	6213	-34.85
4	1358	1278	40.2	-5.89	1055	6329.4	-22.31	1007	6343.8	-25.85
5	1386	1262	41.83	-8.95	1034	5371.6	-25.4	1030	5390.1	-25.68
6	1498	1255	38.58	-16.22	1015	4277.4	-32.24	963	4294.1	-35.71
7	1312	1127	41.72	-14.1	1013	6419.3	-22.79	1010	6432	-23.02
8	1357	1189	40.55	-12.38	986	3805.6	-27.33	1025	3825.3	-24.47
9	1414	1310	42.69	-7.35	979	4448.4	-30.76	1053	4463	-25.53
10	1090	1090	40	0	944	4676.8	-13.39	940	4695.4	-13.76
avg.	1381.7	1197	40.34	-13.37	978.1	5229.8	-29.21	981.3	5245.7	-28.98
(large:4;medium;correlated)										
3	16362	16362	14062.4	0	12223	6401	-25.3	12507	6412.1	-23.56
4	18341	18341	7606.3	0	14021	10079.2	-23.55	14098	10092.6	-23.13
9	17322	16207	31412.4	-6.44	12516	8052.3	-27.74	12245	8060.9	-29.31
avg.	17341.7	16970	17693.7	-2.14	12920	8177.5	-25.5	12950	8188.5	-25.32
(large:8;low;uncorrelated)										
1	2782	2536	126.8	-8.84	2311	10949.5	-16.93	2318	10984.5	-16.68
2	3495	2918	137	-16.51	2620	9405.4	-25.04	2616	9429.1	-25.15
3	3572	3142	137.9	-12.04	2753	6748.9	-22.93	2729	6773.6	-23.6
4	3004	2522	130	-16.04	2338	13085	-22.17	2336	13118.7	-22.24
5	2522	2522	132.4	0	2197	9249.1	-12.89	2142	9272.8	-15.07
6	3479	3240	137.3	-6.87	2907	13232.5	-16.44	2860	13251.1	-17.79
7	2991	2864	133.4	-4.25	2510	9711.4	-16.08	2547	9740.2	-14.84
8	2893	2893	134.4	0	2446	13972	-15.45	2444	14000.9	-15.52
9	2724	2593	130	-4.81	2306	11343.7	-15.34	2349	11369.9	-13.77
10	2959	2959	136.5	0	2505	8764.8	-15.34	2581	8786.9	-12.77
avg.	3042.1	2818.9	133.6	-7.34	2489.3	10646.2	-18.17	2492.2	10672.8	-18.08
(large:8;low;correlated)										
1	3645	2799	135.7	-23.21	2475	13898.8	-32.1	2511	13930.6	-31.11
2	2886	2730	129.6	-5.4	2450	12380.8	-15.11	2450	12409.8	-15.11
3	2673	2522	129	-5.65	2306	10781.3	-13.73	2377	10807.6	-11.07
4	3587	2942	141.2	-17.98	2518	9583.7	-29.8	2532	9615.1	-29.41
5	3376	2748	127.7	-18.6	2429	6015.8	-28.05	2415	6039.8	-28.47
6	3022	2867	134.7	-5.13	2515	11029.1	-16.78	2492	11057.2	-17.54
7	2940	2834	130.9	-3.6	2447	9097.8	-16.77	2423	9129	-17.58
8	2875	2733	131.3	-4.94	2419	12357	-15.86	2289	12377	-20.38
9	3326	2806	130.8	-15.63	2397	11374.8	-27.93	2401	11404.2	-27.81
10	3001	2791	135.7	-7	2517	9418.5	-16.13	2484	9446.2	-17.23
avg.	3133.1	2777.2	132.7	-11.36	2447.3	10593.8	-21.89	2437.4	10621.6	-22.2

Table A.12: R-TS & R-VNS efficiency in improving DRS+DF solutions after re-starting for huge size instances of $\alpha \geq 4$

	DRS+DF	R-DRS+DF			R-TS			R-VNS		
	$C_{\max}$	$C_{\max}$	time(s)	$r. d. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$	$C_{\max}$	time(s)	$r. d. (\%)$
(huge:4;low;uncorrelated)										
1	3820	3203	272.7	-16.15	2726	24597.8	-28.64	2837	24696.2	-25.73
2	3819	2924	280.3	-23.43	2599	25463.7	-31.94	2598	25555.4	-31.97
3	3876	3238	277.9	-16.46	2782	31805.5	-28.22	2597	31873.3	-33
4	3358	3358	278	0	2577	30525.7	-23.26	2707	30625.6	-19.39
5	4106	3317	286.9	-19.22	2651	33733.7	-35.44	2730	33817.9	-33.51
6	4031	3374	279.7	-16.3	2662	37264.6	-33.96	2752	37348.8	-31.73
7	3990	3189	275.1	-20.07	2637	21911.2	-33.91	2648	21977.6	-33.63
8	3522	3092	266.3	-12.21	2537	30279.1	-27.97	2634	30371.8	-25.21
9	4219	3067	276.8	-27.3	2706	30016.4	-35.86	2626	30092	-37.76
10	4038	3202	265.2	-20.7	2749	21336.7	-31.92	2698	21430.2	-33.18
avg.	3877.9	3196.4	275.9	-17.57	2662.6	28693.4	-31.34	2682.7	28778.9	-30.82
(huge:4;low;correlated)										
1	3932	3129	279.7	-20.42	2695	45072.2	-31.46	2742	45180.2	-30.26
2	3466	3466	281.7	0	2664	34354.5	-23.14	2694	34427.1	-22.27
3	3940	3254	274.3	-17.41	2727	23478.3	-30.79	2751	23559.8	-30.18
4	3888	3300	281.8	-15.12	2648	40216.3	-31.89	2656	40309	-31.69
5	4270	3194	279.2	-25.2	2639	29621.5	-38.2	2606	29704.7	-38.97
6	3456	3272	281.4	-5.32	2720	21008.7	-21.3	2770	21086.2	-19.85
7	4027	3220	274	-20.04	2785	25734.2	-30.84	2753	25812.5	-31.64
8	3726	3514	283.2	-5.69	2801	25508.7	-24.82	2828	25594.2	-24.1
9	4376	3124	266.8	-28.61	2523	31486.9	-42.34	2548	31567.8	-41.77
10	3215	3215	266.1	0	2720	30892.6	-15.4	2722	30983.3	-15.33
avg.	3829.6	3268.8	276.8	-14.64	2692.2	30737.4	-29.7	2707	30822.5	-29.31
(huge:8;low;uncorrelated)										
1	7527	7373	1621.8	-2.05	6001	39640.8	-20.27	6008	39805.3	-20.18
2	8518	6877	980.9	-19.26	6341	65575.2	-25.56	6102	65738.6	-28.36
3	8109	7280	963.9	-10.22	6159	42492	-24.05	6025	42635.2	-25.7
4	7846	7108	979.2	-9.41	6550	40127.8	-16.52	6681	40276.3	-14.85
5	8386	7312	965.2	-12.81	6243	48829.9	-25.55	6364	49036.8	-24.11
6	7697	6695	1686.8	-13.02	6234	58487.1	-19.01	6342	58694.5	-17.6
7	9130	7586	992.8	-16.91	6534	61001.2	-28.43	6640	61158.6	-27.27
8	7608	7196	958.3	-5.41	5913	67095.3	-22.28	5932	67280.4	-22.03
9	7527	6886	971.2	-8.52	5968	56013.7	-20.71	6152	56212.5	-18.27
10	7379	6572	932.1	-10.94	5879	59378.4	-20.33	5887	59537.2	-20.22
avg.	7972.7	7088.5	1105.2	-11.09	6182.2	53864.1	-22.46	6213.3	54037.5	-22.07
(huge:8;low;correlated)										
1	7194	6368	1596.9	-11.48	5673	45793	-21.14	6007	45944.8	-16.5
2	7941	7137	973.9	-10.12	6114	65964.3	-23.01	6064	66122.2	-23.64
3	8543	6890	1002.4	-19.35	5929	35841.4	-30.6	6050	36013.2	-29.18
4	7836	7601	976.2	-3	6131	52594.9	-21.76	6263	52763.6	-20.07
5	7682	6679	947.4	-13.06	6206	56741.9	-19.21	6105	56878.3	-20.53
6	7305	6684	1619.9	-8.5	5667	53103.2	-22.42	5840	53280	-20.05
7	7989	6621	948.2	-17.12	5873	66478.6	-26.49	6052	66649.6	-24.25
8	7816	6924	964.3	-11.41	6063	63710.5	-22.43	6154	63871.7	-21.26
9	7425	6836	944.6	-7.93	5953	63229.9	-19.82	6256	63402.4	-15.74
10	8172	7329	987.4	-10.32	6843	43282.6	-16.26	6858	43471.2	-16.08
avg.	7790.3	6906.9	1096.1	-11.34	6045.2	54674	-22.4	6164.9	54839.7	-20.86

Table A.13: TS & VNS solution comparison with M-LS for large size instances of $\alpha \leq 2$ and the high FML

	M-LS	TS		R-TS		VNS		R-VNS	
	$C_{\max}$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$
(large:1;high;uncorrelated)									
1	849	765	-9.89	671	-20.97	783	-7.77	748	-11.9
2	627	666	6.22	605	-3.51	605	-3.51	569	-9.25
3	697	745	6.89	631	-9.47	645	-7.46	640	-8.18
4	802	730	-8.98	677	-15.59	756	-5.74	712	-11.22
5	675	615	-8.89	583	-13.63	637	-5.63	616	-8.74
6	741	784	5.8	713	-3.78	743	0.27	734	-0.94
7	817	778	-4.77	768	-6	702	-14.08	705	-13.71
8	609	621	1.97	608	-0.16	625	2.63	619	1.64
9	678	638	-5.9	613	-9.59	656	-3.24	645	-4.87
10	806	753	-6.58	747	-7.32	780	-3.23	754	-6.45
avg.	730.1	709.5	-2.82	661.6	-9.38	693.2	-5.05	674.2	-7.66
(large:1;high;correlated)									
1	648	643	-0.77	644	-0.62	603	-6.94	566	-12.65
2	707	698	-1.27	656	-7.21	678	-4.1	619	-12.45
3	583	586	0.51	539	-7.55	583	0	538	-7.72
4	785	868	10.57	699	-10.95	677	-13.76	705	-10.19
5	940	804	-14.47	801	-14.79	829	-11.81	805	-14.36
6	729	655	-10.15	719	-1.37	642	-11.93	692	-5.07
7	930	825	-11.29	824	-11.4	814	-12.47	851	-8.49
8	672	648	-3.57	638	-5.06	637	-5.21	642	-4.46
9	603	548	-9.12	529	-12.27	578	-4.15	537	-10.94
10	627	645	2.87	566	-9.73	643	2.55	605	-3.51
avg.	722.4	692	-4.21	661.5	-8.43	668.4	-7.47	656	-9.19
(large:2;high;uncorrelated)									
1	1751	2010	14.79	1668	-4.74	1977	12.91	1675	-4.34
2	2457	2488	1.26	2349	-4.4	2420	-1.51	2363	-3.83
3	1458	1677	15.02	1450	-0.55	1633	12	1391	-4.59
4	1738	1642	-5.52	1672	-3.8	1670	-3.91	1643	-5.47
5	1785	2449	37.2	1739	-2.58	2314	29.64	1744	-2.3
6	1445	1791	23.94	1385	-4.15	1809	25.19	1416	-2.01
7	1723	2135	23.91	1702	-1.22	2121	23.1	1677	-2.67
8	1539	1901	23.52	1497	-2.73	1806	17.35	1509	-1.95
9	1767	1736	-1.75	1714	-3	1642	-7.07	1633	-7.58
10	2360	3502	48.39	2093	-11.31	3265	38.35	2246	-4.83
avg.	1802.3	2133.1	18.35	1726.9	-4.18	2065.7	14.61	1729.7	-4.03
(large:2;high;correlated)									
1	1781	1755	-1.46	1686	-5.33	1769	-0.67	1678	-5.78
2	1387	1696	22.28	1391	0.29	1728	24.58	1388	0.07
3	1727	2698	56.22	1665	-3.59	2598	50.43	1657	-4.05
4	1726	1729	0.17	1680	-2.66	1667	-3.42	1689	-2.14
5	1739	1566	-9.95	1588	-8.68	1558	-10.41	1627	-6.44
6	1556	2827	81.68	1474	-5.27	2890	85.73	1443	-7.26
7	1423	2025	42.3	1378	-3.16	1992	39.99	1412	-0.77
8	1473	1424	-3.33	1472	-0.07	1418	-3.73	1455	-1.22
9	2339	3224	37.84	2151	-8.04	3179	35.91	2220	-5.09
10	1502	1490	-0.8	1468	-2.26	1489	-0.86	1471	-2.06
avg.	1665.3	2043.4	22.7	1595.3	-4.2	2028.8	21.83	1604	-3.68

Table A.14: TS & VNS solution comparison with M-LS for large size instances of $\alpha \geq 4$

	M-LS	TS		R-TS		VNS		R-VNS	
	$C_{\max}$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$
(large:4;low;uncorrelated)									
1	1030	1011	-1.84	1101	6.89	1062	3.1	1020	-0.97
2	1000	931	-6.9	965	-3.5	971	-2.9	964	-3.6
3	992	988	-0.4	1038	4.64	995	0.3	999	0.71
4	1019	995	-2.35	1049	2.94	950	-6.77	1016	-0.29
5	1007	1028	2.08	1035	2.78	927	-7.94	989	-1.79
6	1048	1013	-3.34	983	-6.2	974	-7.06	1006	-4.01
7	1061	1084	2.17	1067	0.56	1077	1.51	1100	3.68
8	989	1055	6.67	1010	2.12	1012	2.33	989	0
9	964	978	1.45	902	-6.43	951	-1.35	909	-5.7
10	957	965	0.84	999	4.39	980	2.4	964	0.73
avg.	1006.7	1004.8	-0.19	1014.9	0.81	989.9	-1.67	995.6	-1.1
(large:4;medium;uncorrelated)									
5	12699	12907	1.64	12502	-1.55	11737	-7.57	12688	-0.09
(large:4;low;correlated)									
1	868	914	5.3	931	7.26	943	8.64	895	3.11
2	1002	1005	0.3	922	-7.98	1008	0.6	976	-2.59
3	926	931	0.54	902	-2.59	896	-3.24	914	-1.3
4	1054	1100	4.36	1055	0.09	1034	-1.9	1007	-4.46
5	1023	1040	1.66	1034	1.07	1004	-1.86	1030	0.68
6	1012	1030	1.78	1015	0.3	1059	4.64	963	-4.84
7	971	1044	7.52	1013	4.32	1058	8.96	1010	4.02
8	956	969	1.36	986	3.14	1018	6.48	1025	7.22
9	1033	989	-4.26	979	-5.23	981	-5.03	1053	1.94
10	954	950	-0.42	944	-1.05	927	-2.83	940	-1.47
avg.	979.9	997.2	1.76	978.1	-0.18	992.8	1.32	981.3	0.14
(large:4;medium;correlated)									
3	13640	11834	-13.24	12223	-10.39	12094	-11.33	12507	-8.31
4	15159	14005	-7.61	14021	-7.51	14256	-5.96	14098	-7
9	12816	12875	0.46	12516	-2.34	12741	-0.58	12245	-4.45
avg.	13871.7	12904.7	-6.97	12920	-6.86	13030.3	-6.06	12950	-6.64
(large:8;low;uncorrelated)									
1	2219	2255	1.62	2311	4.15	2211	-0.36	2318	4.46
2	2648	2559	-3.36	2620	-1.06	2591	-2.15	2616	-1.21
3	2645	2805	6.05	2753	4.08	2759	4.31	2729	3.18
4	2359	2288	-3.01	2338	-0.89	2260	-4.2	2336	-0.97
5	2141	2127	-0.65	2197	2.62	2132	-0.42	2142	0.05
6	2766	2799	1.19	2907	5.1	2817	1.84	2860	3.4
7	2557	2504	-2.07	2510	-1.84	2539	-0.7	2547	-0.39
8	2447	2426	-0.86	2446	-0.04	2456	0.37	2444	-0.12
9	2247	2191	-2.49	2306	2.63	2279	1.42	2349	4.54
10	2587	2703	4.48	2505	-3.17	2614	1.04	2581	-0.23
avg.	2461.6	2465.7	0.17	2489.3	1.12	2465.8	0.17	2492.2	1.24
(large:8;low;correlated)									
1	2509	2587	3.11	2475	-1.35	2552	1.71	2511	0.08
2	2346	2531	7.89	2450	4.43	2499	6.52	2450	4.43
3	2323	2232	-3.92	2306	-0.73	2312	-0.47	2377	2.32
4	2454	2630	7.17	2518	2.61	2592	5.62	2532	3.18
5	2378	2382	0.17	2429	2.14	2317	-2.56	2415	1.56
6	2441	2402	-1.6	2515	3.03	2498	2.33	2492	2.09
7	2481	2610	5.2	2447	-1.37	2505	0.97	2423	-2.34
8	2496	2422	-2.96	2419	-3.08	2498	0.08	2289	-8.29
9	2369	2458	3.76	2397	1.18	2357	-0.51	2401	1.35
10	2415	2673	10.68	2517	4.22	2662	10.23	2484	2.86
avg.	2421.2	2492.7	2.95	2447.3	1.08	2479.2	2.39	2437.4	0.67

Table A.15: TS & VNS solution comparison with M-LS for huge size instances of $\alpha \leq 2$ and the high FML

	M-LS	TS		R-TS		VNS		R-VNS	
	$C_{\max}$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$
(huge:1;high;uncorrelated)									
1	2174	2014	-7.36	1926	-11.41	1852	-14.81	2011	-7.5
2	2224	2478	11.42	2074	-6.74	2242	0.81	2091	-5.98
3	2355	2335	-0.85	2412	2.42	2353	-0.08	2320	-1.49
4	2304	2420	5.03	2446	6.16	2131	-7.51	2289	-0.65
5	2029	2312	13.95	1887	-7	2180	7.44	1957	-3.55
6	2008	2210	10.06	1863	-7.22	2168	7.97	1930	-3.88
7	2195	2072	-5.6	2017	-8.11	1966	-10.43	1972	-10.16
8	2057	1967	-4.37	2044	-0.63	1940	-5.69	1903	-7.49
9	2090	2234	6.89	2080	-0.48	2269	8.56	2118	1.34
10	2045	2266	10.81	2096	2.49	2301	12.52	2097	2.54
avg.	2148.1	2230.8	3.85	2084.5	-2.96	2140.2	-0.37	2068.8	-3.69
(huge:1;high;correlated)									
1	2048	1872	-8.59	2043	-0.24	1966	-4	2001	-2.29
2	2040	2238	9.71	1985	-2.7	2232	9.41	2058	0.88
3	2111	2246	6.39	2239	6.06	2136	1.18	2284	8.19
4	2236	1944	-13.06	2261	1.12	2062	-7.78	2183	-2.37
5	1984	1844	-7.06	1881	-5.19	1870	-5.75	1732	-12.7
6	2042	2057	0.73	1977	-3.18	1984	-2.84	1948	-4.6
7	2017	1988	-1.44	1924	-4.61	1953	-3.17	1894	-6.1
8	2056	2011	-2.19	1976	-3.89	2003	-2.58	1980	-3.7
9	2253	2295	1.86	2138	-5.1	2209	-1.95	2141	-4.97
10	2276	2243	-1.45	2288	0.53	2115	-7.07	2095	-7.95
avg.	2106.3	2073.8	-1.54	2071.2	-1.67	2053	-2.53	2031.6	-3.55
(huge:2;high;uncorrelated)									
1	6101	8156	33.68	5903	-3.24	7927	29.93	5860	-3.95
2	7490	6854	-8.49	7439	-0.68	6954	-7.16	7286	-2.72
3	4039	4453	10.25	4049	0.25	4273	5.79	4118	1.96
4	4336	4379	0.99	4335	-0.02	4325	-0.25	4387	1.18
5	4461	4778	7.11	4258	-4.55	4963	11.25	4392	-1.55
6	4427	4514	1.96	4585	3.57	4562	3.05	4634	4.68
7	5133	7067	37.68	5145	0.23	6948	35.36	5090	-0.84
8	4227	5163	22.14	4780	13.08	5157	22	4811	13.82
9	4669	5002	7.13	4605	-1.37	4940	5.8	4698	0.62
10	4581	4700	2.6	4814	5.09	4690	2.38	4804	4.87
avg.	4946.4	5506.6	11.32	4991.3	0.91	5473.9	10.66	5008	1.24
(huge:2;high;correlated)									
1	4635	5049	8.93	4640	0.11	5141	10.92	4547	-1.9
2	4348	4312	-0.83	4346	-0.05	4369	0.48	4438	2.07
3	4937	8002	62.08	4901	-0.73	8020	62.45	4872	-1.32
4	4710	5709	21.21	4884	3.69	5604	18.98	5018	6.54
5	4928	4921	-0.14	4998	1.42	4874	-1.1	4950	0.45
6	4374	4679	6.97	4459	1.94	4722	7.96	4398	0.55
7	4262	4217	-1.06	4310	1.13	4213	-1.15	4390	3
8	4949	5295	6.99	4902	-0.95	5396	9.03	4939	-0.2
9	4255	4601	8.13	4389	3.15	4457	4.75	4362	2.51
10	4797	5058	5.44	4879	1.71	5102	6.36	4775	-0.46
avg.	4619.5	5184.3	12.23	4670.8	1.11	5189.8	12.34	4668.9	1.07

Table A.16: TS & VNS solution comparison with M-LS for huge size instances of $\alpha \geq 4$

	M-LS	TS		R-TS		VNS		R-VNS	
	$C_{\max}$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$	$C_{\max}$	$r. d. (\%)$
(huge:4;low;uncorrelated)									
1	2552	2606	2.12	2726	6.82	2739	7.33	2837	11.17
2	2576	2733	6.09	2599	0.89	2700	4.81	2598	0.85
3	2704	2631	-2.7	2782	2.88	2640	-2.37	2597	-3.96
4	2717	2737	0.74	2577	-5.15	2589	-4.71	2707	-0.37
5	2786	2880	3.37	2651	-4.85	2860	2.66	2730	-2.01
6	2683	2649	-1.27	2662	-0.78	2619	-2.38	2752	2.57
7	2524	2775	9.94	2637	4.48	2723	7.88	2648	4.91
8	2659	2705	1.73	2537	-4.59	2530	-4.85	2634	-0.94
9	2584	2667	3.21	2706	4.72	2742	6.11	2626	1.62
10	2665	2768	3.86	2749	3.15	2653	-0.45	2698	1.24
avg.	2645	2715.1	2.65	2662.6	0.66	2679.5	1.3	2682.7	1.42
(huge:4;low;correlated)									
1	2739	2614	-4.56	2695	-1.61	2718	-0.77	2742	0.11
2	2544	2541	-0.12	2664	4.72	2592	1.89	2694	5.9
3	2641	2681	1.51	2727	3.26	2619	-0.83	2751	4.16
4	2638	2699	2.31	2648	0.38	2717	2.99	2656	0.68
5	2540	2698	6.22	2639	3.9	2625	3.35	2606	2.6
6	2608	2719	4.26	2720	4.29	2793	7.09	2770	6.21
7	2609	2651	1.61	2785	6.75	2667	2.22	2753	5.52
8	2723	2776	1.95	2801	2.86	2786	2.31	2828	3.86
9	2540	2590	1.97	2523	-0.67	2561	0.83	2548	0.31
10	2567	2666	3.86	2720	5.96	2621	2.1	2722	6.04
avg.	2614.9	2663.5	1.86	2692.2	2.96	2669.9	2.1	2707	3.52
(huge:8;low;uncorrelated)									
1	5700	6063	6.37	6001	5.28	6057	6.26	6008	5.4
2	6232	6106	-2.02	6341	1.75	6357	2.01	6102	-2.09
3	5769	6053	4.92	6159	6.76	6179	7.11	6025	4.44
4	6079	6535	7.5	6550	7.75	6303	3.68	6681	9.9
5	6051	6058	0.12	6243	3.17	5991	-0.99	6364	5.17
6	6123	6399	4.51	6234	1.81	6251	2.09	6342	3.58
7	6553	6907	5.4	6534	-0.29	6874	4.9	6640	1.33
8	5692	5993	5.29	5913	3.88	6102	7.2	5932	4.22
9	5965	5987	0.37	5968	0.05	6076	1.86	6152	3.13
10	5788	5791	0.05	5879	1.57	5799	0.19	5887	1.71
avg.	5995.2	6189.2	3.24	6182.2	3.12	6198.9	3.4	6213.3	3.64
(huge:8;low;correlated)									
1	5619	5742	2.19	5673	0.96	6094	8.45	6007	6.9
2	6115	6241	2.06	6114	-0.02	6271	2.55	6064	-0.83
3	5678	6282	10.64	5929	4.42	6247	10.02	6050	6.55
4	5880	6010	2.21	6131	4.27	6077	3.35	6263	6.51
5	6106	6036	-1.15	6206	1.64	6132	0.43	6105	-0.02
6	5589	5811	3.97	5667	1.4	5884	5.28	5840	4.49
7	5706	6028	5.64	5873	2.93	6068	6.34	6052	6.06
8	6208	6203	-0.08	6063	-2.34	6305	1.56	6154	-0.87
9	5869	5881	0.2	5953	1.43	5833	-0.61	6256	6.59
10	6422	6865	6.9	6843	6.55	6990	8.84	6858	6.79
avg.	5919.2	6109.9	3.22	6045.2	2.13	6190.1	4.58	6164.9	4.15

Appendix B

TS and VNS Parameter Tuning

The TS and VNS algorithms proposed in Chapter 5 have a number of control parameters (or factors) that must be tuned before the performance of each is compared with one another. The objective of parameter tuning is mainly to balance the trade-off between the solution quality and runtime of both TS and VNS meta-heuristics. For example, by increasing the maximum number of non-improving iterations allowed for the TS algorithm, one can expect to find better solutions at the expense of a longer runtime. The question, however, is to what extent the parameter should be increased such that the expected improvement in solution quality is comparable to the extra computation time spent. Parameter tuning answers this question.

As introduced in Chapter 2, a full-factorial design is one of the popular methods to study the effects of several factors and their interactions on a system's response. The drawback with this method is that the number of configurations tested is exponential to the number and levels of the factors defined. Consider the TS algorithm developed in Section 5.3.1 of Chapter 5 with three control parameters. If one defines four levels for each parameter, there are in total $4^3 = 64$ configurations to be tested. If each configuration is run five times¹, then the total number of experiments for each problem instance would

¹Note that due to some stochastic components embedded in the implementation of both TS and VNS algorithms, it is required that the algorithms be run for a few times per problem instance to establish robustness in solution quality.

be $4^3 \times 5 = 320$. Given that the truck scheduling problem studied in this thesis is very complex, an enormous amount of time is needed to direct the experimentation for huge size instances.

The alternative method is a 2^k factorial design which is a special and simplest, but the most widely used, factorial design with k factors, each at two levels. The methodology is to choose a reasonable minimum and maximum value for each control parameter and then test all the combinations of the low and high values of the parameters. A two-level factorial design can substantially reduce the total number of experiments for a given number of factors. For the three-parameter TS algorithm described above, the total number of experiments for each instance reduces from $4^3 \times 5 = 320$ in a full factorial design to $2^3 \times 5 = 40$ in a 2^k design.

Table B.1 lists the control parameters defined for both TS and VNS algorithms together with their considered minimum and maximum values. The TS algorithm has been developed with three factors and VNS with two factors (see Sections 5.3.1 and 5.3.2 of Chapter 5). The values of the factors have been chosen in a way to alter the performance of the algorithms in terms of their search methodology and search space. For example, the two values of the factor A defined for TS in Table B.1(a) represents two situations where in the former, the algorithm uses no tabu list in its search path while in the latter, it uses a rather long tabu list to orient the search. The two values considered for the remaining factors are mainly used to balance a trade-off between solution quality and runtime by confining and expanding the search space, respectively, according to the structure of the given algorithm. This results in having different minimum and maximum values for control parameter $i_{\max}$ for the TS and VNS algorithms.

The standard methodology for a parameter setting involves choosing a representative subset of the problem instances to train parameters on, avoiding over-training the data [67]. For this purpose, a subset of 150 instances (from the total 644 feasible instances) is

Table B.1: TS and VNS factor levels

(a) TS factor levels			
Factor	Description	min.	max.
<i>A</i>	tabu list size (denoted by l)	0	20
<i>B</i>	number of neighbors chosen randomly per iteration (denoted by r)	10	50
<i>C</i>	maximum number of non-improving iterations allowed (denoted by $i_{\max}$)	20	100
(b) VNS factor levels			
Factor	Description	min.	max.
<i>A</i>	number of neighbors chosen randomly per LS iteration (denoted by r)	10	50
<i>B</i>	maximum number of non-improving iterations allowed (denoted by $i_{\max}$)	1	20

chosen. The subset represents all the feasible categories of the experimental data in terms of the problem size, trailer-to-door ratio, freight mixed level, and correlation status. So parameter tuning is carried out for all the various characteristics of the test data. Each data category in the representative subset includes a minimal number of samples that behaved differently² after a deterministic local search was applied to optimize their corresponding initial solution generated by the DRS+DF heuristics. The criteria considered to choose the samples are the amount of improvement, number of moves made to reach local optima, and runtime.

For both TS and VNS algorithms, a factorial ANOVA (analysis of variance) is applied to decide whether and which factors or interactions have significant effect (influence) on the response variable (here, the makespan). It should be noted that the three assumptions on normality, independence and equality of variance are all applicable to the experimental model. Given the generated synthetic data and the stochastic structure of the proposed meta-heuristics, no biasness was observed questioning the validity of the conducted experiments. According to the notation introduced in Table B.2, Tables B.3 and B.4 show the template for the calculation of the ANOVA for two- and three-factor factorial designs, respectively.

²Since the samples have been randomly generated, an arbitrary selection of them would also be justifiable.

Table B.2: Parameters used for the description of ANOVA

Parameter	Description
a	number of levels defined for the factor A
b	number of levels defined for the factor B
c	number of levels defined for the factor C
n	number of runs for each configuration of factors (and their interactions)
v_1	degree of freedom of the numerator of the significance ratio F_0
v_2	degree of freedom of the mean square of error (MS_E)
$F_{0.05, v_1, v_2}$	control limit to decide on the significance of a factor

Table B.3: ANOVA template for a two-factor factorial design

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	Significance Ratio
A	SS_A	$a - 1$	$MS_A = \frac{SS_A}{a-1}$	$F_{0A} = \frac{MS_A}{MS_E}$
B	SS_B	$b - 1$	$MS_B = \frac{SS_B}{b-1}$	$F_{0B} = \frac{MS_B}{MS_E}$
AB	SS_{AB}	$(a - 1)(b - 1)$	$MS_{AB} = \frac{SS_{AB}}{(a-1)(b-1)}$	$F_{0AB} = \frac{MS_{AB}}{MS_E}$
Error	SS_E	$ab(n - 1)$	$MS_E = \frac{SS_E}{ab(n-1)}$	
Total	SS_T	$abn - 1$		

Table B.4: ANOVA template for a three-factor factorial design

Source of Variation	Sum of Squares	Degrees of Freedom	Mean Square	Significance Ratio
A	SS_A	$a - 1$	$MS_A = \frac{SS_A}{a-1}$	$F_{0A} = \frac{MS_A}{MS_E}$
B	SS_B	$b - 1$	$MS_B = \frac{SS_B}{b-1}$	$F_{0B} = \frac{MS_B}{MS_E}$
C	SS_C	$c - 1$	$MS_C = \frac{SS_C}{c-1}$	$F_{0C} = \frac{MS_C}{MS_E}$
AB	SS_{AB}	$(a - 1)(b - 1)$	$MS_{AB} = \frac{SS_{AB}}{(a-1)(b-1)}$	$F_{0AB} = \frac{MS_{AB}}{MS_E}$
AC	SS_{AC}	$(a - 1)(c - 1)$	$MS_{AC} = \frac{SS_{AC}}{(a-1)(c-1)}$	$F_{0AC} = \frac{MS_{AC}}{MS_E}$
BC	SS_{BC}	$(b - 1)(c - 1)$	$MS_{BC} = \frac{SS_{BC}}{(b-1)(c-1)}$	$F_{0BC} = \frac{MS_{BC}}{MS_E}$
ABC	SS_{ABC}	$(a - 1)(b - 1)(c - 1)$	$MS_{ABC} = \frac{SS_{ABC}}{(a-1)(b-1)(c-1)}$	$F_{0ABC} = \frac{MS_{ABC}}{MS_E}$
Error	SS_E	$abc(n - 1)$	$MS_E = \frac{SS_E}{abc(n-1)}$	
Total	SS_T	$abcn - 1$		

As already mentioned, a 2^k factorial design is applied to tune the control parameters of the meta-heuristics. (This means that the number of levels for all the factors is equal to two, i.e., $a = b = c = 2$). The TS algorithm has three factors thus a 2^3 design should be used. This implies that the number of configurations to be tested for TS is eight. For VNS, the number of factors is two so a 2^2 design with four configurations is used. Both meta-heuristics are run five times (i.e., $n = 5$) per configuration for each problem instance in the representative subset. Accordingly, the total number of experiments for TS is $8 \times 5 \times 150 = 6000$ and for VNS is $4 \times 5 \times 150 = 3000$.

An ANOVA is conducted to calculate the ratio F_0 for each factor (or interaction). (The interested reader is referred to [55] for the detailed presentation of ANOVA formulas in Tables B.3 and B.4.) If F_0 is larger than the pre-specified control limit $F_{0.05, v_1, v_2}$, then the factor (or interaction) has significant effect on the response; otherwise, it has no effect. Note that the effect can be either constructive or destructive on the response. For both of the algorithms examined here, the effect of a factor A is constructive if its high value causes the algorithm to significantly improve the makespan and destructive if it degrades it. Likewise, for an interaction AB , the effect is constructive if the high values of A and B together lead to a better makespan than the high value of each individual factor A or B and is destructive if the reverse applies. Also note that since the sum of squares of the observations are used in the calculation of the ratio F_0 , its value is always positive irrespective of whether the effect of the corresponding factor is constructive or destructive.

Tables B.5 and B.6 show the complete ANOVA for the three factors (and their interactions) of the TS algorithm from small to huge size problem instances. (Note that the average results are shown per data category.) The following conclusions can be drawn by analyzing the results:

- The implementation of a tabu list does improve the efficiency of the TS algorithm in finding high-quality solutions for the truck scheduling instances of 32 trailers and less. As the number of trailers is small, it is highly probable that the exchange of two already interchanged trailers, which are now in different positions of the sequence list, leads to cycling back to a previously visited solution. On the other hand, for the problem instances of 64 trailers and more, the tabu list has no significance or even has a destructive effect on the solution quality by blocking some promising moves that can help the algorithm to reach better solutions. As an example, see the ANOVA results for the (huge:2;medium;uncorrelated) data category where the effect of factor A is destructive on the quality of the makespan. The reason for such a behavior is that the methodology to declare a move as tabu is remarkably simple compared to the complexity of the truck scheduling problem. (This is more significant for the difficult instances of the larger size and higher freight mixed levels.) It is thus unlikely that the exchange of two already interchanged trailers in a large sequence of highly interdependent trailers leads to the same solution.
- Factor C , the number of non-improving iterations, has a consistently constructive effect on the solution quality for almost all the categories of the test data. Such a consistency, however, is not observed for factor B , the number of random neighbors evaluated per iteration, except for large and huge size instances where $\alpha = 8$ and $\alpha \geq 4$, respectively. The interaction of the two factors, B and C , has also been destructive for a few samples. This is particularly noticeable for the (medium:2;high;correlated) and (medium:4;medium;correlated) categories in which factor C has a highly constructive effect on the response while factor B has no significance. It seems that for these data categories, the TS meta-heuristic is more successful in choosing non-improving moves and escaping local optima when the number of random neighbors evaluated per iteration is kept low.

Table B.5: ANOVA for the three factors of TS algorithm for small and medium size instances

Data Category	s*	$F_{0.05,v_1=1,v_2=32} = 4.152$										Significance						
		F_{0A}	F_{0B}	F_{0C}	F_{0AB}	F_{0AC}	F_{0BC}	F_{0ABC}	A	B	C	AB	AC	BC	ABC			
(small:1;low;uncorrelated)	1	31.212	0.033	23.585	1.281	4.641	0.033	0.385	1	0	1	0	1	0	0			
(small:1;medium;uncorrelated)	1	23.636	0.495	0.921	0.802	0.409	0.2	0.262	1	0	0	0	0	0	0			
(small:1;high;uncorrelated)	1	22.617	0.078	1.052	0.078	1.052	0.009	0.078	1	0	0	0	0	0	0			
(small:1;low;correlated)	1	11.405	0.02	12.726	0.274	0.057	0.509	1.197	1	0	1	0	0	0	0			
(small:1;medium;correlated)	1	90.961	0	7.622	0	7.622	0.063	0.063	1	0	1	0	1	0	0			
(small:1;high;correlated)	2	7.814	0.614	3.357	0.614	3.357	0.5	0.5	1	0	0	0	0	0	0			
(small:2;low;uncorrelated)	1	3.074	0.063	7.592	0.016	3.529	0.016	0	0	0	1	0	0	0	0			
(small:2;medium;uncorrelated)	2	19.679	2.497	10.64	1.931	1.502	0.076	7.846	1	0	1	0	0	0	1			
(small:2;high;uncorrelated)	1	37.191	0.325	5.007	0.233	0.695	0.556	0.433	1	0	1	0	0	0	0			
(small:2;low;correlated)	1	3.74	1.662	2.597	3.74	5.091	2.597	5.091	0	0	0	0	1	0	1			
(small:2;medium;correlated)	1	0.137	2.981	2.57	0.015	0	0.38	0	0	0	0	0	0	0	0			
(small:2;high;correlated)	1	4.39	0.075	3.66	5.187	0.407	0.207	0.672	1	0	0	1	0	0	0			
(small:4;low;uncorrelated)	2	0.689	2.673	7.509	0.673	0.544	0.124	4.738	0	0	1	0	0	0	1			
(small:4;low;correlated)	3	6.32	12.08	6.844	4.619	1.027	0.85	2.476	1	1	1	1	0	0	0			
(small:4;medium;correlated)	1	19.896	0.132	11.322	0.83	4.622	2.962	0.348	1	0	1	0	1	0	0			
(small:8;low;uncorrelated)	2	0.236	14.23	10.697	4.688	1.558	1.114	0.874	0	1	1	1	0	0	0			
(small:8;low;correlated)	3	1.367	3.198	5.854	0.502	1.722	1.405	1.096	0	0	1	0	0	0	1			
(medium:1;low;uncorrelated)	3	0.989	7.239	7.223	0.24	1.142	0.405	0.787	0	1	1	0	0	0	0			
(medium:1;medium;uncorrelated)	1	0.505	4.842	11.233	0.414	1.101	1.248	0.505	0	1	1	0	0	0	0			
(medium:1;high;uncorrelated)	1	3.425	1.174	4.892	0.736	8.346	0.002	0.399	0	0	1	0	1	0	0			
(medium:1;low;correlated)	3	1.086	2.834	11.799	3.414	3.195	0.821	0.376	0	0	1	0	0	0	0			
(medium:1;medium;correlated)	3	0.997	10.348	19.971	2.503	2.479	3.294	2.167	0	1	1	0	0	0	0			
(medium:1;high;correlated)	3	6.665	2.758	27.824	0.431	1.913	0.055	1.947	1	0	1	0	0	0	0			
(medium:2;low;uncorrelated)	2	1.974	8.093	12.839	0.958	2.7	0.063	1.379	0	1	1	0	0	0	0			
(medium:2;medium;uncorrelated)	2	0.116	2.363	8.202	2.164	0.935	1.039	0.206	0	0	1	0	0	0	0			
(medium:2;high;uncorrelated)	1	3.931	0.25	14.765	1.986	5.645	3.899	2.474	0	0	1	0	1	0	0			
(medium:2;low;correlated)	3	0.615	8.737	4.858	0.698	1.201	1.486	2.413	0	1	1	0	0	0	0			
(medium:2;medium;correlated)	3	0.313	10.63	11.457	0.714	2.401	3.529	0.783	0	1	1	0	0	0	0			
(medium:2;high;correlated)	2	3.939	2.282	35.838	0.583	3.509	19.445	5.361	0	0	1	0	0	1	1			
(medium:4;low;uncorrelated)	1	2.265	4.908	2.265	0.803	0.24	0.048	0.003	0	1	0	0	0	0	0			
(medium:4;medium;uncorrelated)	1	0.449	0.703	5.664	0.289	0.949	0.108	1.383	0	0	1	0	0	0	0			
(medium:4;low;correlated)	3	0.448	8.584	9.072	0.998	0.308	6.644	1.869	0	1	1	0	0	1	0			
(medium:4;medium;correlated)	2	0.802	2.959	13.553	0.617	0.032	6.145	1.184	0	0	1	0	0	1	0			
(medium:8;low;uncorrelated)	2	1.85	3.239	19.904	0.161	0.197	0.367	0.231	0	0	1	0	0	0	0			
(medium:8;low;correlated)	2	2.558	3.562	5.106	0.697	0.442	2.968	0.418	0	0	1	0	0	0	0			

*denotes the number of samples chosen per data category

Table B.6: ANOVA for the three factors of TS algorithm for large and huge size instances

Data Category	s*	$F_{0.05, v_1=1, v_2=32} = 4.152$							Significance						
		F_{0A}	F_{0B}	F_{0C}	F_{0AB}	F_{0AC}	F_{0BC}	F_{0ABC}	A	B	C	AB	AC	BC	ABC
(large:1;low;uncorrelated)	1	0.002	0.057	13.041	1.417	0.455	2.274	0.63	0	0	1	0	0	0	0
(large:1;medium;uncorrelated)	2	0.598	5.327	8.065	1.737	1.932	0.155	0.246	0	1	1	0	0	0	0
(large:1;high;uncorrelated)	2	5.737	1.467	26.949	0.036	0.43	0.023	1.158	1	0	1	0	0	0	0
(large:1;low;correlated)	2	0.205	5.954	10.66	0.416	3.404	2.733	0.325	0	1	1	0	0	0	0
(large:1;medium;correlated)	2	1.313	3.365	10.637	1.878	1.774	4.628	0.89	0	0	1	0	0	1	0
(large:1;high;correlated)	2	1.189	2.321	11.733	0.702	1.492	1.058	0.657	0	0	1	0	0	0	0
(large:2;low;uncorrelated)	1	0.118	1.782	5.775	2.406	0.127	0.983	0.726	0	0	1	0	0	0	0
(large:2;medium;uncorrelated)	2	0.736	0.088	6.597	1.178	1.191	0.077	1.182	0	0	1	0	0	0	0
(large:2;high;uncorrelated)	3	1.423	4.551	17.705	0.256	0.357	1.213	0.629	0	1	1	0	0	0	0
(large:2;low;correlated)	3	0.793	7.506	6.567	0.507	0.699	1.709	1.031	0	1	1	0	0	0	0
(large:2;medium;correlated)	3	0.81	6.088	17.005	0.451	0.787	1.057	1.234	0	1	1	0	0	0	0
(large:2;high;correlated)	3	1.886	4.264	16.809	0.551	1.502	2.217	0.81	0	1	1	0	0	0	0
(large:4;low;uncorrelated)	4	0.805	7.495	11.972	0.465	0.402	0.379	2.692	0	1	1	0	0	0	0
(large:4;medium;uncorrelated)	1	0.144	0.531	20.34	0.949	4.769	1.484	0.149	0	0	1	0	1	0	0
(large:4;low;correlated)	4	0.576	2.456	8.577	0.625	0.97	0.754	1.697	0	0	1	0	0	0	0
(large:4;medium;correlated)	2	3.806	0.162	9.208	0.843	1.643	3.22	0.633	0	0	1	0	0	0	0
(large:8;low;uncorrelated)	2	0.879	6.625	10.143	0.317	0.152	2.075	0.71	0	1	1	0	0	0	0
(large:8;low;correlated)	3	0.683	14.583	6.308	2.42	0.243	1.056	1.006	0	1	1	0	0	0	0
(huge:1;low;uncorrelated)	3	1.637	1.644	2.889	2.741	2.019	1.929	2.541	0	0	0	0	0	0	0
(huge:1;medium;uncorrelated)	1	0.0155	3.818	9.42	0.116	0.089	0.008	0.008	0	0	1	0	0	0	0
(huge:1;high;uncorrelated)	4	0.92	0.36	16.724	0.141	0.625	2.113	0.942	0	0	1	0	0	0	0
(huge:1;low;correlated)	5	1.56	6.394	5.762	1.073	0.493	0.377	0.754	0	1	1	0	0	0	0
(huge:1;medium;correlated)	1	0.781	0.46	3.656	0.747	0.228	0.126	1.553	0	0	0	0	0	0	0
(huge:1;high;correlated)	2	2.613	6.275	5.302	0.45	1.553	0.235	2.297	0	1	1	0	0	0	0
(huge:2;low;uncorrelated)	4	0.316	2.291	12.153	0.31	0.805	0.76	0.393	0	0	1	0	0	0	0
(huge:2;medium;uncorrelated)	2	6.157	6.517	19.232	0.146	1.45	1.543	1.208	1	1	1	0	0	0	0
(huge:2;high;uncorrelated)	3	0.94	10.593	9.795	0.664	1.092	2.135	0.243	0	1	1	0	0	0	0
(huge:2;low;correlated)	3	0.598	5.11	9.279	0.494	1.001	1.185	1.049	0	1	1	0	0	0	0
(huge:2;medium;correlated)	2	4.119	10.506	7.271	2.107	2.299	1.887	0.469	0	1	1	0	0	0	0
(huge:2;high;correlated)	3	1.641	2.898	8.39	1.37	0.734	0.569	0.549	0	0	1	0	0	0	0
(huge:4;low;uncorrelated)	3	0.866	6.324	6.386	1.96	0.555	0.459	0.48	0	1	1	0	0	0	0
(huge:4;low;correlated)	2	0.493	13.824	11.05	0.063	1.241	1.828	0.775	0	1	1	0	0	0	0
(huge:8;low;uncorrelated)	4	0.454	16.373	11.633	2.072	1.389	2.809	0.675	0	1	1	0	0	0	0
(huge:8;low;correlated)	3	2.561	8.171	6.967	1.818	1.074	2.103	0.839	0	1	1	0	0	0	0

* denotes the number of samples chosen per data category

The significance of factors B and C in improving the solution quality is achieved at the expense of a longer runtime. For example, the average time taken for TS to optimize a problem instance in the (huge:8;low;correlated) category is nearly 4.5 hours when both B and C are kept in their low levels. This increases to more than 19 and 20.5 hours when factors B and C are high, respectively. The situation becomes extremely worse when the two factors are set in their high levels simultaneously; the average runtime would be more than 91.5 hours (approximately 4 days!) which might not be reasonable at all from a practical viewpoint of the crossdocking operation. It is thus required to carry out the tuning by assuming middle values for factors B and C so that the runtime can be reduced without degrading the solution quality.

In the first place, the value of factor A , the tabu list size, is adjusted for both small and large size problems. As already verified, the implementation of a tabu list is essential for small size problems to avoid recycling and to escape local optima. However, for large size problems, a tabu list often does not have a constructive effect on the solution quality and even restricts the search on some occasions. The next stage of the tuning is therefore carried out by fixing the size of the tabu list to 10 when the number of trailers is 32 and less, and repeatedly varying the size between 0, 10, and 20 in the five runs of larger instances (i.e., those where the number of trailers is more than 32). In the second place, two middle values of 20 and 50 are assumed for factors B and C , respectively. It is desired to reduce the runtime by applying the middle value of factor B for the (large:8), (huge:4), and (huge:8) data categories and using its low level for the remaining categories. Factor C , on the other hand, has a consistently constructive effect on the solution quality so its middle value is only tested for large and huge size instances.

To demonstrate the applicability of the proposed parameter setting in reducing the runtime without worsening the solution quality, 17 data categories are chosen from Tables B.5 and B.6 for the second stage of the tuning. The data categories (shown in bold in

Table B.7: ANOVA for the significance of the TS factors in their middle values

Data Category	s*	$F_{0.05, v_1=1, v_2=8} = 5.32$	
		$F_{0A'}$	Comment
$A'_-(l=10), A'_+(l=20), r=10, m=100$			
(small:1;medium;correlated)	1	0	insignificant
(small:2;medium;uncorrelated)	2	1.19	insignificant
(small:2;high;uncorrelated)	1	0.212	insignificant
$A'_-(r=10), A'_+(r=50), l=10, m=100$			
(small:4;low;correlated)	3	2.334	insignificant
$A'_-(r=10), A'_+(r=50), l=v^{**}, m=100$			
(small:8;low;uncorrelated)	2	4.179	insignificant
(medium:2;low;correlated)	3	0.333	insignificant
(medium:2;medium;correlated)	3	1.729	insignificant
$A'_-(m=50), A'_+(m=100), l=v^{**}, r=10$			
(large:1;high;uncorrelated)	2	9.747	significant
(large:4;medium;uncorrelated)	1	3.582	insignificant
(huge:1;high;uncorrelated)	4	1.583	insignificant
$A'_-(r=10, m=50), A'_+(r=50, m=100), l=v^{**}$			
(large:2;low;correlated)	3	1.299	insignificant
(large:4;low;uncorrelated)	4	7.539	significant
(huge:2;high;uncorrelated)	3	1.488	insignificant
(huge:2;medium;correlated)	2	1.352	insignificant
$A'_-(r=20, m=50), A'_+(r=50, m=100), l=v^{**}$			
(large:4;low;uncorrelated)	4	2.999	insignificant
(large:8;low;uncorrelated)	2	5.033	insignificant
(large:8;low;correlated)	3	3.385	insignificant
(huge:8;low;uncorrelated)	4	5.969	marginally significant

*denotes the number of samples chosen per data category

**indicates that the size of the tabu list varies between 0, 10, and 20

the tables) are special in that the factors are absolutely significant in their high values. The ANOVA for this new parameter setting is shown in Table B.7. For each phase of the experiment, a dummy factor A' is defined to conduct a simple 2^1 factorial design³ to test the effect of the proposed parameter setting on both solution quality and runtime.

According to the table, a tabu list of size 10 works efficiently for the problem instances of 32 trailers and less. It is also inferred from Table B.7 that a proper adjustment of the tabu list size can maintain the solution quality for small and medium instances if factor B is kept in its low level. Furthermore, the ANOVA shows that the performance of the TS algorithm in terms of solution quality does not deteriorate for the (huge:1) and

³The ANOVA template for a 2^1 design is easily derived from the formulas in Tables B.3 and B.4.

Table B.8: Final values considered for the parameters of the TS algorithm

Problem Size	l	r	$i_{\max}$
small	10 if $\alpha \leq 4$ v^* if $\alpha = 8$	10	100
medium	10 if $\alpha = 1$ v^* if $\alpha \geq 2$	10	100
large	v^*	10 if $\alpha \leq 2$ 20 if $\alpha \geq 4$	100 if $\alpha \leq 2$ 50 if $\alpha \geq 4$
huge	v^*	10 if $\alpha \leq 2$ 20 if $\alpha \geq 4$	50

*indicates that the size of the tabu list varies between 0, 10, and 20

(huge:2) instances if the number of neighbors chosen randomly per iteration is kept to 10 (the low level of factor B) and number of non-improving iterations reduces to 50 (the middle level of factor C). This, however, is not true for the (large:1;high;uncorrelated) and (large:4;low;uncorrelated) categories. (Note that the response value of the former is merely dependent on factor C – see Table B.6). Increasing the number of random neighbors to 20 (the middle level of factor B) while keeping C in its middle level preserves the solution quality for the (large:4), (large:8), and even (huge:8) categories. The significance of the high level of dummy factor A' is marginal for the latter data category, and thus, can be ignored.

The values of the three control parameters of the TS algorithm are finalized in Table B.8 according to the analysis conducted above. In this setting, the average time taken for TS to optimize a problem instance in the (huge:8;low;correlated) category is nearly 18 hours, much shorter than the 91.5 hours required when parameters r and $i_{\max}$ are initialized with their maximum values. In the meantime, the ANOVA showed that the degradation in the solution quality is insignificant.

To tune the VNS meta-heuristic, a 2^2 factorial design is used. Tables B.9 and B.10 show the complete ANOVA for the two factors (and their interactions) of VNS from small to huge size problem instances. (Note that the average results are shown per data category.) Unlike the TS algorithm, both factors A , the number of random neighbors

evaluated by the LS routine per iteration, and B , number of non-improving iterations, have a consistently constructive effect on the solution quality for all the large and huge size instances and most of the small and medium ones. In addition, interaction AB is not destructive implying that both factors independently improve the solution quality when they are set in their high levels.

Like TS, the significance of factors A and B of the VNS algorithm in improving the solution quality is achieved at the expense of a longer runtime. Consider the same problem instance in the (huge:8;low;correlated) data category. The average time taken for VNS to optimize the initial solution is nearly 1.5 hour for the low values of A and B . This increases to more than 7.5 and 23.5 hours when factors A and B are high, respectively. The worst case is when both factors are in their high levels simultaneously; the average runtime would be more than 145 hours, exceeding 6 days. Such a long time is not at all desired in practice.

On the other hand, the further tuning of the two factors by assuming middle values degrades the solution quality. The analysis was carried out by first reducing the number of non-improving iterations to 10 (the middle value considered for factor B) and fixing the number of random neighbors to 10 (which is the low value of factor A). Of the 12 data categories chosen for the experimentation, the ANOVA showed that the high values of the factors were insignificant for only two categories of small instances. In the second trial, the number of random neighbors evaluated by the LS routine per iteration increased to 20 (the middle value considered for factor A). For this new parameter setting, the ANOVA could decide on the insignificance of three more categories of small, medium, and large size instances. This means that for seven data categories of medium, large, and huge size instances, the high values of the two factors should be used to achieve high solution quality. As a result, the evaluation of the VNS algorithm is carried out by finalizing its two control parameters with their maximum values even though this may not be favorable from a practical perspective.

Table B.9: ANOVA for the two factors of VNS algorithm for small and medium instances

Data Category	s*	$F_{0.05, v_1=1, v_2=16} = 4.49$			Significance		
		F_{0A}	F_{0B}	F_{0AB}	A	B	AB
(small:1;low;uncorrelated)	1	0.117	28.135	0.066	0	1	0
(small:1;medium;uncorrelated)	1	0.733	17.219	0.976	0	1	0
(small:1;high;uncorrelated)	1	8.377	53.594	0.029	1	1	0
(small:1;low;correlated)	1	4.278	6.27	6.27	0	1	1
(small:1;medium;correlated)	1	1.263	15.474	0.035	0	1	0
(small:1;high;correlated)	2	2.96	12.098	2.96	0	1	0
(small:2;low;uncorrelated)	1	1.174	6.674	5.84	0	1	1
(small:2;medium;uncorrelated)	2	1.501	29.129	0.282	0	1	0
(small:2;high;uncorrelated)	1	13.94	77.543	18.466	1	1	1
(small:2;low;correlated)	1	7.118	0.529	2.882	1	0	0
(small:2;medium;correlated)	1	0.26	9.984	1.756	0	1	0
(small:2;high;correlated)	1	0.022	0.203	2.253	0	0	0
(small:4;low;uncorrelated)	2	4.334	11.076	0.874	0	1	0
(small:4;low;correlated)	3	4.432	37.499	3.479	0	1	0
(small:4;medium;correlated)	1	0.54	19.653	0.515	0	1	0
(small:8;low;uncorrelated)	2	14.564	18.133	1.42	1	1	0
(small:8;low;correlated)	3	12.06	28.615	1.032	1	1	0
(medium:1;low;uncorrelated)	3	0.834	24.352	0.385	0	1	0
(medium:1;medium;uncorrelated)	1	19.407	23.592	1.033	1	1	0
(medium:1;high;uncorrelated)	1	8.652	8.652	9.012	1	1	1
(medium:1;low;correlated)	3	2.851	27.317	0.233	0	1	0
(medium:1;medium;correlated)	3	6.11	18.891	0.327	1	1	0
(medium:1;high;correlated)	3	5.622	48.241	0.358	1	1	0
(medium:2;low;uncorrelated)	2	6.983	25.512	1.779	1	1	0
(medium:2;medium;uncorrelated)	2	2.908	33.882	0.468	0	1	0
(medium:2;high;uncorrelated)	1	9.196	66.203	0.253	1	1	0
(medium:2;low;correlated)	3	12.005	32.104	0.155	1	1	0
(medium:2;medium;correlated)	3	9.391	10.077	0.276	1	1	0
(medium:2;high;correlated)	2	19.32	114.605	0.193	1	1	0
(medium:4;low;uncorrelated)	1	9.61	8.173	0.011	1	1	0
(medium:4;medium;uncorrelated)	1	46.56	70.886	1.945	1	1	0
(medium:4;low;correlated)	3	4.059	10.207	0.975	0	1	0
(medium:4;medium;correlated)	2	7.14	31.815	0.899	1	1	0
(medium:8;low;uncorrelated)	2	26.794	50.252	1.734	1	1	0
(medium:8;low;correlated)	2	10.129	31.362	0.424	1	1	0

*denotes the number of samples chosen per data category

Table B.10: ANOVA for the two factors of VNS algorithm for large and huge instances

Data Category	s*	$F_{0.05, v_1=1, v_2=16} = 4.49$			Significance		
		F_{0A}	F_{0B}	F_{0AB}	A	B	AB
(large:1;low;uncorrelated)	1	8.015	12.523	0.01	1	1	0
(large:1;medium;uncorrelated)	2	14.415	24.202	0.604	1	1	0
(large:1;high;uncorrelated)	2	13.049	53.519	1.144	1	1	0
(large:1;low;correlated)	2	8.903	23.805	5.235	1	1	1
(large:1;medium;correlated)	2	6.398	28.12	0.547	1	1	0
(large:1;high;correlated)	2	27.903	83.011	2.895	1	1	0
(large:2;low;uncorrelated)	1	13.111	31.123	11.299	1	1	1
(large:2;medium;uncorrelated)	2	18.89	44.99	0.52	1	1	0
(large:2;high;uncorrelated)	3	11.699	34.001	2.155	1	1	0
(large:2;low;correlated)	3	15.612	39.487	0.94	1	1	0
(large:2;medium;correlated)	3	17.856	26.566	2.179	1	1	0
(large:2;high;correlated)	3	9.893	24.871	1.206	1	1	0
(large:4;low;uncorrelated)	4	5.985	23.058	0.281	1	1	0
(large:4;medium;uncorrelated)	1	17.218	49.27	1.951	1	1	0
(large:4;low;correlated)	4	20.885	33.12	2.614	1	1	0
(large:4;medium;correlated)	2	44.761	99.6	9.667	1	1	1
(large:8;low;uncorrelated)	2	10.417	37.496	0.972	1	1	0
(large:8;low;correlated)	3	12.471	37.79	1.156	1	1	0
(huge:1;low;uncorrelated)	3	23.8	47.574	1.694	1	1	0
(huge:1;medium;uncorrelated)	1	7.037	23.417	0.608	1	1	0
(huge:1;high;uncorrelated)	4	15.863	66.693	3.712	1	1	0
(huge:1;low;correlated)	5	29.535	20.066	9.763	1	1	1
(huge:1;medium;correlated)	1	50.254	54.622	0.065	1	1	0
(huge:1;high;correlated)	2	17.128	24.829	0.893	1	1	0
(huge:2;low;uncorrelated)	4	10.04	53.317	2.463	1	1	0
(huge:2;medium;uncorrelated)	2	12.361	25.285	0.228	1	1	0
(huge:2;high;uncorrelated)	3	8.499	33.622	1.243	1	1	0
(huge:2;low;correlated)	3	18.632	38.717	1.186	1	1	0
(huge:2;medium;correlated)	2	48.463	114.706	2.152	1	1	0
(huge:2;high;correlated)	3	10.06	42.241	2.451	1	1	0
(huge:4;low;uncorrelated)	3	13.424	30.737	1.71	1	1	0
(huge:4;low;correlated)	2	10.662	34.772	2.423	1	1	0
(huge:8;low;uncorrelated)	4	30.101	42.064	2.748	1	1	0
(huge:8;low;correlated)	3	16.375	32.204	0.191	1	1	0

*denotes the number of samples chosen per data category

Appendix C

Extended Variable Neighborhood Search

According to the values of the control parameters tuned for both TS and VNS algorithms (see Appendix B), VNS demands a much longer time than TS to terminate its search process. Consequently, the experimentation conducted in Chapter 5 to compare the two algorithms was based on running VNS subject to an external time limit, enforcing the algorithm to run within a time comparable with the TS runtime for each problem instance. This appendix thus conducts an experimentation where VNS is permitted to continue its search till its own termination criterion¹ is met. In that case, the capability of the algorithm in finding better solutions is evaluated at the expense of longer running times. As the optimization of the 644 feasible instances is extremely time consuming, only the representative subset of 150 instances set up for tuning the control parameters of the TS and VNS meta-heuristics in Appendix B is considered for the assessment. (For ease of reference, this extended version of VNS is denoted a E-VNS).

Tables C.1 and C.2 present the average results for the representative subset for small and medium, and large and huge size instances, respectively. The superiority of E-VNS becomes more evident when the number of trailers exceed 32. According to the tables, the

¹The VNS algorithm stops if after a pre-specified number of iterations (i.e., $i_{\max} = 20$), no improvement is achieved by exploring each neighborhood N_k per iteration.

E-VNS algorithm is able to improve the solutions produced by the DRS+DF heuristics for an additional -5% to -6% average relative difference compared to the original TS and VNS algorithms.

The resultant improvement in the solution quality, however, is achieved by an exponential increase in the runtime. For example, the -3% relative difference in the solution quality of the (huge:8;low;correlated) category has been obtained after allowing E-VNS to continue running for additional five days! Given the fact that the original VNS algorithm could return an improved solution in this data category in nearly 19 hours, no convincing justification can be provided for the implementation of the extended version of VNS (E-VNS) in an industrial setting. The application of E-VNS is impractical if truck scheduling has to be carried out in a short-term planning horizon of one day. It is thus required to combine some other approaches with the basic components of the proposed TS and VNS algorithms so that an efficient exploration of the search space is achieved in reasonable running times. One technique is to feed the algorithms with a better initial solution. This was realized in Chapter 5 by combining the re-starting version of the DRS algorithm with the TS and VNS meta-heuristics. The strategy was to feed the best solution produced by the re-starting version of DRS into each algorithm as the initial solution (see Section 5.5.2.2 of Chapter 5 for more information).

Table C.1: Efficiency of E-VNS vs. TS and VNS in improving DRS+DF solutions for small and medium size instances

Data Category	DRS+DF			TS			VNS			E-VNS		
	C_{max}	time(s)		C_{max}	time(s)	$r. d. (%)$	C_{max}	time(s)	$r. d. (%)$	C_{max}	time(s)	$r. d. (%)$
(small:1;low;uncorrelated)	54	0.015		28	27.02	-48.15	28	26.735	-48.15	28	29.55	-48.15
(small:1;medium;uncorrelated)	63	0.016		39	37.48	-38.09	39	37.61	-38.09	39	44.52	-38.09
(small:1;high;uncorrelated)	46	0.001		35	51.25	-23.91	35	49.67	-23.91	35	64.19	-23.91
(small:1;low;correlated)	54	0.016		32	42.66	-40.74	32	25.58	-40.74	32	28.91	-40.74
(small:1;medium;correlated)	63	0.016		30	32.39	-52.38	30	32.25	-52.38	30	37.9	-52.38
(small:1;high;correlated)	60	0.015		46.5	51.41	-22.5	45	50.67	-25	45	54.47	-25
(small:2;low;uncorrelated)	94	0.031		89	97.75	-5.32	90	98.03	-4.25	90	160.42	-4.25
(small:2;medium;uncorrelated)	187	0.023		114.5	195.9	-38.77	122	196.2	-34.76	116.5	303.12	-37.7
(small:2;high;uncorrelated)	265	60.66		213	170.44	-19.62	212	170.8	-20	214	360.84	-19.24
(small:2;low;correlated)	115	0.015		96	107.8	-16.52	95	108.09	-17.39	97	184.56	-15.65
(small:2;medium;correlated)	129	0.031		113	96.34	-12.4	114	96.67	-11.63	112	252.89	-13.18
(small:2;high;correlated)	251	0.031		164	150.98	-34.66	165	151.42	-34.26	167	549.75	-33.47
(small:4;low;uncorrelated)	261	0.054		233.5	235.71	-10.54	231.5	236.21	-11.3	226	1118.6	-13.41
(small:4;low;correlated)	258.3	0.058		218.7	236.79	-15.35	218	237.33	-15.61	213.3	1050	-17.42
(small:4;medium;correlated)	792	121.06		666	439	-15.91	639	439.8	-19.32	617	2031.3	-22.1
(small:8;low;uncorrelated)	938.5	0.156		821.5	814.17	-12.47	833.5	815.38	-11.19	784.5	7236.3	-16.41
(small:8;low;correlated)	1073	1.479		913.7	626.6	-14.85	916.3	627.63	-14.6	859.7	6277.9	-19.88
(medium:1;low;uncorrelated)	169.7	0.078		86.7	266.92	-48.92	85	267.81	-49.9	80	852.99	-52.85
(medium:1;medium;uncorrelated)	177	0.125		115	363.22	-35.03	116	364.2	-34.46	113	1755.4	-36.16
(medium:1;high;uncorrelated)	253	0.203		202	824.83	-20.16	216	826.73	-14.62	209	3961.4	-17.39
(medium:1;low;correlated)	128.3	0.09		75	247.26	-41.56	74.7	248.09	-41.82	69.7	917.06	-45.71
(medium:1;medium;correlated)	168	0.11		114.7	381.63	-31.75	112	382.71	-33.33	106.3	1438	-36.71
(medium:1;high;correlated)	275	0.198		228.3	809	-16.97	220.7	810.57	-19.76	201.3	4519	-26.79
(medium:2;low;uncorrelated)	239.5	0.21		184	497.55	-23.17	186	499.25	-22.34	165.5	4409.8	-30.9
(medium:2;medium;uncorrelated)	390	0.328		290	712.86	-25.64	296	714.52	-24.1	279	3191	-28.46
(medium:2;high;uncorrelated)	907	0.562		603	3055.1	-33.52	640	3057.8	-29.44	559	21328.6	-38.37
(medium:2;low;correlated)	282.3	0.198		171.3	515.16	-39.31	182	516.9	-35.54	164	4369.4	-41.91
(medium:2;medium;correlated)	358	0.323		287	779.9	-19.83	280	781.91	-21.79	278	4208.6	-22.35
(medium:2;high;correlated)	1161.5	0.774		824	1736.2	-29.06	866.5	1739	-25.4	763.5	25550.4	-34.27
(medium:4;low;uncorrelated)	475	0.516		416	1116.8	-12.42	421	1120.2	-11.37	395	10368.8	-16.84
(medium:4;medium;uncorrelated)	3414	4708.6		2658	1680.1	-22.14	2585	1683.66	-24.28	2519	20556.6	-26.22
(medium:4;low;correlated)	497.7	0.573		401	1404.1	-19.42	393	1407.3	-21.03	385	9548.5	-22.64
(medium:4;medium;correlated)	3220	454.94		2436	2038.6	-24.35	2405	2042	-25.31	2119	23118.7	-34.19
(medium:8;low;uncorrelated)	1212.5	1.539		1092.5	3069.9	-9.9	1082.5	3075.6	-10.72	1061	28148.3	-12.49
(medium:8;low;correlated)	1436	1.703		1171	2858.2	-18.45	1158.5	2865	-19.32	1122	28271.5	-21.87

Appendix D

List of Publications

This research has led to the following publications:

Journal

Shakeri, M., Low, M. Y. H., Turner, S. J., and Lee, E. W., “A robust two-phase algorithm for the truck scheduling problem in a resource-constrained crossdock,” *Computers & Operations Research* 39(11), 2564–2577 (2012) (2011 Impact Factor: 1.720).

Conference

Shakeri, M., Low, M. Y. H., and Lee, E. W., “Development of a novel resource-constrained crossdocking model for the truck scheduling problem,” in *IEEE International Conference on Emerging Technologies and Factory Automation*, 1–9 (2010).

Shakeri, M., Low, M. Y. H., Li, Z., and Lee, E. W., “Two efficient constructive heuristics for scheduling trucks at crossdocking terminals,” in *IEEE International Conference on Service Operations and Logistics, and Informatics*, 177–182 (2010).

Shakeri, M., Low, M. Y. H., and Li, Z., “A generic model for crossdock truck scheduling and truck-to-door assignment problems,” in *IEEE International Conference on Industrial Informatics (INDIN08)*, 857–864 (2008).

Submitted Journal

Shakeri, M., Low, M. Y. H., Turner, S. J., and Lee, E. W., “Robust hybrid meta-heuristics for the truck scheduling optimization in a resourced-constrained crossdock,” *Computers & Operations Research* [under review] (2011 Impact Factor: 1.720).

Working Paper

Shakeri, M., Low, M. Y. H., Turner, S. J., and Lee, E. W., “An efficient fractional fitness evaluation mechanism for the truck scheduling optimization in a crossdocking system.”