Minimising the total travel distance to pick orders on a unidirectional picking line

Anton Pierre de Villiers

Thesis presented in partial fulfilment of the requirements for the degree of
Master of Science (Operations Research)
in the Faculty of Science at Stellenbosch University

Supervisor: Prof SE Visagie

December 2011

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

December 1, 2011

Abstract

Order picking is the most important activity in distribution centres. It involves the process of retrieving products from storage in response to a specific customer request. The order picking system in a distribution centre used by Pep Stores Ltd. (Pep), located in Durban, South Africa, is considered. The order picking system in Pep utilises picking lines. The system requires that the pickers move in a clockwise direction around the picking line. The planning of picking lines may be divided into three tiers of decisions. The first tier determines which Stock Keeping Units (SKUs) should be allocated to which picking line and is known as the SKU to Picking Line Assignment Problem (SPLAP). The second tier, the SKU Location Problem (SLP), considers the positioning of the various SKUs in a picking line. The final tier considers the sequencing of the orders for pickers within a picking line and is referred to as the Order Sequencing Problem (OSP). Collectively, these three tiers aim to achieve the objective of picking all the SKUs for all the orders in the shortest possible time. The decisions associated with each tier are made sequentially during the planning of a picking line. Each problem therefore relies on the information generated by its predecessing tier(s).

Initially the OSP is addressed. A number of heuristic and metaheuristic approaches are presented, together with an exact formulation to solve this tier. The size of the problem is reduced by using a relaxation of the problem that may be solved exactly. A number of greedy tour construction heuristics, a scope and ranking algorithm, methods based on awarding starting locations with respect to preference ratios and a modified assignment approach was used to solve the OSP. Furthermore, a tabu search, simulated annealing, genetic algorithm and a generalised extremal optimisation approach are used to solve the OSP. The solution quality and computational times of all the approaches are compared for the data provided by Pep, with the generalised extremal optimisation approach delivering the best solution quality.

Two methods from the literature was used to model the SLP, whereafter an ant colony system was used to maximise the number of orders in common between adjacent SKUs. A number of agglomerative clustering algorithms were used from which dendrograms could be constructed. Two novel heuristic clustering algorithms were considered. The first heuristic calculates a distance between two clusters as the set of orders that have to collect all the SKUs in both clusters, whereas the second method is based upon the frequency of SKUs within a cluster. Little or no improvement was achieved in most cases.

The SPLAP was introduced by means of a number of possibilities of how to formulate objectives. A possible exact formulation is presented, followed by a nearest neighbour search, which was initially used to construct new picking lines based on all data sets. A different approach was then taken by means of a tabu search where the waves of two or three picking lines were altered. Significant savings may be incurred for large data sets.

Opsomming

Die opmaak van bestellings is die belangrikste aktiwiteit in 'n distribusiesentrum. Dit behels dat geskikte hoeveelhede produkte uit stoorplekke opgespoor en herpak moet word om aan kleinhandeltakke gestuur te word. Die bedrywighede binne een van Pep Stores Ltd. (Pep) se distribusiesentrums in Durban, Suid-Afrika, word beskou. Die sisteem vereis dat die werkers in 'n kloksgewyse rigting om 'n uitsoeklyn beweeg. Die beplanning van die uitsoeklyne kan verdeel word in drie besluite/probleme. Die eerste besluit is watter voorraadeenhede (VEs) toegewys moet word aan watter uitsoeklyn. Die tweede besluit is in watter vakkies in die uitsoeklyn die VEs geplaas moet word, en word die VE-plasings probleem (VLP) genoem. Die finale besluit is in watter volgorde bestellings opgemaak moet word in 'n uitsoeklyn, en staan bekend as die volgorde-van-bestellings-probleem (VBP). Die doel van al drie hierdie probleme is om al die bestellings in 'n uitsoeklyn in die kortste moontlike tyd af te handel. Die besluite wat verband hou met elke vlak van besluit word opeenvolgend gedoen tydens die beplanning van 'n uitsoeklyn. Die oplossing van elke subprobleem berus op die inligting van die voorafgaande probleme.

Aanvanlik word die VBP beskou. 'n Aantal heuristiese en metaheuristiese benaderings word aangebied saam met 'n eksakte formulering om die derde vlak op te los. Die grootte van die probleem is verminder deur die gebruik van 'n verslapping van 'n eksakte formulering. 'n Aantal toerkonstruksie heuristieke, 'n omvang en rangorde algoritme, metodes wat gebaseer is op die toekenning van beginpunte met betrekking tot voorkeurverhoudings en 'n veralgemeende toewysingsprobleem is gebruik om die VBP op te los. 'n Tabu-soektog, gesimuleerde tempering, genetiese algoritme en 'n veralgemeende-ekstreme-optimering-benadering word ook gebruik om die VBP op te los. Die oplossingsgehalte en berekeningstye van al die benaderings word vergelyk vir werklike data wat verskaf is deur Pep. Die veralgemeende-ekstreme-optimering-benadering lewer die beste oplossingsgehalte.

Twee metodes uit die literatuur is gebruik om die VLP te modelleer, waarna 'n mier kolonie stelsel gebruik word om die aantal bestellings wat aangrensende VEs in gemeen het te maksimeer. 'n Aantal groeperingsalgoritmes word gebruik wat dendrogramme kan lewer. Twee heuristiese groeperingsalgoritmes word oorweeg. Die eerste heuristiek bereken die afstand tussen twee groepe as die aantal bestellings wat al die VEs in beide groepe moet versamel, terwyl die tweede metode gebaseer is op die frekwensie van VEs binne 'n groep. Min of geen verbeterings is in die meeste gevalle gevind.

Die eerste besluit word bekend gestel na aanleiding van 'n aantal moontlike maniere om die doelwitte te formuleer. 'n Moontlike eksakte formulering word aangebied. 'n Alternatiewe benadering is geneem deur middel van 'n tabu-soektog waar die golwe van twee of drie uitsoeklyne gewysig word. Beduidende besparings word gerealiseer vir groot datastelle.

Acknowledgements

The author hereby wishes to express his deepest gratitude towards those who played a significant role during the progress of work towards this thesis:

- Prof Stephan E Visagie for his valuable insight, tireless work and support during the compilation of this thesis;
- Jason Matthews for assistance and patient guidence;
- My colleagues (fellow GoreLab peers);
- My friends and family;
- My parents.

The Department of Logistics at Stellenbosch University are hereby thanked for the use of their computing facilities and office space. The financial support of Pep Stores Ltd., in the form of a Bursary, and of Stellenbosch University in the form of two Merit Bursaries towards this research is hereby acknowledged. Any opinions or findings in this dissertation are those of the author and do not necessarily reflect the views of Stellenbosch University or of Pep Stores Ltd.

Table of Contents

List of Figures **xiii**

List of Tables **xvii**

List of Algorithms **xix**

List of Acronyms **xxi**

List of Reserved Symbols **xxiii**

1 Introduction **1**

1.1 A supply chain 1

1.2 A distribution centre 2

1.3 The Pep DCs 3

1.4 Order picking and picking lines 4

1.5 Problem description and project scope 4

1.5.1 The order sequencing problem 5

1.5.2 SKU location problem 5

1.5.3 SKU to picking line assignment problem 6

1.6 Project objectives 6

1.7 Project organisation 6

2 Literature review **9**

2.1 Introduction to the supply chain 9

2.2 Warehousing 10

2.2.1 The nature and importance of warehousing 10

2.2.2 Warehouse operations 10

2.3 Order picking systems 11

2.4 Order batching systems 14

2.5 Sequencing and routing policies . . . 15
2.5.1 Sequencing and routing for conventional single-block warehouses . . . 16
2.5.2 Sequencing and routing for conventional multi-parallel-aisle systems . . . 17
2.5.3 Sequencing and routing AS/RS . . . 18
2.6 Carousels . . . 18
2.7 Picking area zoning . . . 20
2.8 Chapter summary . . . 21

3 The Pep DCs 23
3.1 Organisational background . . . 23
3.2 The Durban and Kuilsrivier DC . . . 24
3.3 Layout of a picking line . . . 25
3.4 Building of a picking line . . . 26
3.5 The pickers . . . 29
3.6 Assigning SKUs to locations . . . 30
3.7 Control procedures . . . 31
3.8 Rules imposed by Pep . . . 33
3.9 Chapter summary . . . 33

4 Order sequencing problem 35
4.1 Introduction . . . 36
4.2 Exact formulations . . . 36
4.2.1 An exact formulation . . . 37
4.3 Notation for the OSP . . . 38
4.3.1 A branch and bound approach . . . 42
4.3.2 A lower bound . . . 43
4.4 Tour construction heuristics . . . 45
4.4.1 Next shortest order . . . 46
4.4.2 Next shortest order relative to minimum span . . . 47
4.4.3 Next shortest order relative to its number of picks . . . 47
4.4.4 Results for the tour construction heuristics . . . 47
4.5 Scope and ranking algorithms . . . 48
4.5.1 Ranked preference span algorithm . . . 49
4.5.2 Minimum span preserving algorithm . . . 50
4.5.3 Results of the scope and ranking algorithms . . . 52
4.6 Awarding spans according to preference ratios . . . 52

4.7 Generalised assignment approach . . . 58
4.7.1 Heuristic solution methods to solve the generalised assignment problem . 61
4.7.2 Implementation of the modified assigment problem . . . 68
4.7.3 Results of the generalised assignment problem . . . 70
4.8 Metaheuristic approaches to order sequencing . . . 71
4.8.1 Data structure and improvement of a solution . . . 72
4.8.2 Greedy starting heuristic . . . 75
4.8.3 Tabu search . . . 76
4.8.4 Simulated annealing . . . 79
4.8.5 Genetic algorithm . . . 84
4.8.6 Extremal Optimisation . . . 95
4.9 Results summary . . . 97
4.10 Chapter summary . . . 104

5 SKU location problem 105
5.1 A lower bound . . . 105
5.2 SKU location in literature . . . 107
5.2.1 Organ-pipe arrangement . . . 107
5.2.2 Greedy ranking and partitioning . . . 108
5.3 Ant colony algorithm for locating similar SKUs . . . 110
5.3.1 Behaviour of social insects . . . 110
5.3.2 Ant system . . . 111
5.3.3 Extensions of the ant system . . . 112
5.4 (Agglomerative) Clustering algorithms . . . 114
5.4.1 The single-link method . . . 115
5.4.2 The average-link method . . . 115
5.4.3 The centroid method . . . 117
5.4.4 Ward's algorithm . . . 117
5.4.5 When to stop the clustering process . . . 118
5.4.6 Implementation of clustering algorithms . . . 119
5.5 Heuristic clustering methods . . . 119
5.5.1 Clustering according to orders . . . 120
5.5.2 Clustering according to SKU frequency . . . 120
5.6 Computational results . . . 121
5.7 Random SLP configurations . . . 122

5.8 Chapter summary 124

6 Ideas and preliminary results on the SPLAP **127**

6.1 Managerial considerations 127

6.2 Nearest neighbour search 128

6.3 A tabu search approach 130

6.4 Chapter summary 134

7 Conclusion **137**

7.1 Thesis summary and achievement of goals 137

7.2 Thesis contributions 138

7.2.1 Quick heuristic and metaheuristic approaches for the OSP 138

7.2.2 Novel clustering algorithms for the SLP 139

7.2.3 Preliminary investigation into the SPLAP 139

7.3 Possible future work 139

7.3.1 Developing criteria for measuring performance of the SPLAP 139

7.3.2 Finding bounds on improvements of the SLP 139

7.3.3 Measuring the trade-offs for using non-exact approaches to solve the SPLAP 139

7.3.4 Optimal number of pickers in a picking line 140

7.3.5 Optimal mix of picking line sizes 140

7.3.6 The effect of KPIs on the DC operations 140

7.3.7 Permanent picking line 140

References **141**

A An exact formulation for the SPLAP **149**

List of Figures

1.1 A schematic representation of the physical layout of a picking line containing m locations. . . . 4

2.1 The predominant flow in a typical supply chain. . . . 10
2.2 Typical warehouse functions and flows [104]. . . . 11
2.3 Examples of picker-to-parts systems. . . . 12
2.4 Examples of parts-to-picker systems. . . . 12
2.5 A distribution of how a picker's time is spent [104]. . . . 13
2.7 Diagrams depicting the five different routing methods of pickers between aisles for order picking within a single-block warehouse. . . . 17
2.8 Examples of carousel systems (Source: [35]). . . . 19
2.9 A carousel consisting of 12 distinct locations and 1 order. . . . 20

3.1 A view of the Pep DC in Durban. . . . 24
3.2 The layout of the Pep DC in Durban. . . . 25
3.3 The receiving process at the Durban DC. . . . 25
3.4 The movement of goods within the Durban DC. . . . 26
3.5 Examples of possible picking line configurations in the Pep DC in Durban. . . . 27
3.6 An example of the picking line configuration in the Pep DC in Kuilsrivier. . . . 27
3.7 Building picking lines at the Durban DC. . . . 28
3.8 Operational picking lines at the Durban DC. . . . 28
3.9 The gate at the centre of a picking line in the Durban DC. . . . 29
3.10 Pickers operating in a picking line. . . . 30
3.11 Information contained in the system that is used to track operations within the DC. . . . 32
3.12 Stickers used to identify the location of cartons, its destination and its contents. 32

4.1 A schematic representation of the layout of a picking line containing 20 locations. 39
4.2 A carousel illustrating a picking line consisting of 12 distinct locations and 5 orders. 41

4.3 A carousel illustrating a picking line consisting of 12 distinct locations and 5 orders with lines indicating an optimal sequence when starting at the first location. 41

4.4 A carousel illustrating a picking line consisting of 12 distinct locations and 5 orders with lines indicating a non-optimal sequence when starting at the first location. 42

4.5 A bar chart displaying the results when the scope varied for values between 1 and 30 locations. 50

4.6 A bar chart displaying the results when the ranked preference span algorithm is used when varying the preference spans and the scope. 51

4.7 A schematic representation of the layout of a picking line containing 20 locations and 3 orders. 51

4.8 A bar chart displaying the results for the minimum span preserving algorithm. . 52

4.9 The trade-off between the number of cycles travelled and number of possible spans awarded to each order, when solving the RS_1 and RS_2 when varying the lengths of the preference list. 60

4.10 The cost matrix used when solving the OSP using a generalised assigment problem. 62

4.11 The cost matrix used to solve the OSP by means of the GAP for $n = 3$ and $Q = 2$. 63

4.12 Graphical illustration of the Greedy RTB heuristic. 64

4.13 Graphical illustration of the Greedy RBT heuristic. 65

4.14 Graphical illustration of the Greedy RR heuristic. 66

4.15 Graphical illustration of the Greedy CLR heuristic. 67

4.16 Graphical illustration of the Greedy CRL heuristic. 67

4.17 Graphical illustration of the Greedy CR heuristic. 68

4.18 An example of three subtours that have to be connected. 69

4.19 The manner in which three subtours may be connected to form a single subtour. 70

4.20 A bar chart displaying the results when the number of preference spans used in the generalised assignment problem is varied. 71

4.21 Example of an order that may reduce the maximal cut when a different span is selected. 74

4.22 Example of when the span of an order is altered that reduces the maximal cut. . 74

4.23 Example of an order that may shift and reduce the number of maximal cuts when a different span is selected. 75

4.24 Example when the preference span of an order is altered shifts and reduces the number of maximal cuts. 76

4.25 Example of an order that may be altered to increase the number of maximal cuts. 77

4.26 Example of an order that is altered to increase the number of maximal cuts when a different span is selected. 78

4.27 An example of a uniform order-based crossover where both parent solutions contain 6 orders. 87

4.28 An example of a 50/50 crossover. 92
4.29 A bar chart displaying the results (cycles travelled) obtained in Table 4.21. . . . 102
4.30 A bar chart displaying the results (cycles travelled) obtained in Table 4.22. . . . 103

5.1 An example of using OPA to solve the SLP. The frequency of a SKU refers to the number of orders requiring that SKU. 108
5.2 An example of using greedy ranking and partitioning algorithm to solve the SLP. The frequency of a SKU refers to the number of orders requiring a SKU. 109
5.3 A graphical representation of the solution quality of the ant colony system for data set K, when the value of β is varied at 1, 2, 5 and 10. 114
5.4 Dendrogram of the clusters formed by the single-link method as a function of their threshold distances for data set K. 115
5.5 Schematic illustration of the mechanism of the average-link method for data set K. 116
5.6 Dendrogram of the clusters formed by the average-link method as a function of their threshold distances. 116
5.7 Schematic illustration of the mechanism of the centriod method. 117
5.8 Dendrogram of the clusters formed by the average-link method as a function of their threshold distances for data set K. 118
5.9 Dendrogram of the clusters formed by Ward's algorithm to orders as a function of their threshold distances for data set K. 119
5.10 Dendrogram of the clusters formed by heuristic manner of clustering according to orders as a function of their threshold distances for data set K. 120
5.11 Dendrogram of the clusters formed by heuristic manner of clustering according to SKU frequency as a function of their threshold distances for data set K. . . . 121
5.12 The number of picks at a location that is picked most often for each data set, compared to the number of cycles travelled using Pep's configuration. 124

6.1 Frequency of each SKU before and after the tabu search is implemented on picking line A and B. 132
6.2 Frequency of each SKU before and after the tabu search is implemented on picking line A and B. 133
6.3 Frequency of each SKU before and after the tabu search is implemented on picking line H and L. 133
6.4 Frequency of each SKU before and after the tabu search is implemented on picking line A, B and E. 134

List of Tables

2.1 Order closeness metrics for batching systems [50]. 15

4.1 Results obtained from the maximal cut algorithm used to solve the OSP. 46

4.2 Total number of cycles obtained from the tour construction algorithms used to solve the OSP. 48

4.3 Computational times in milliseconds of the tour construction algorithms used to solve the OSP. 49

4.4 Results indicating the number of cycles travelled when using the ranked preference span algorithm, when $p = 3$ and $s_c = 6$, as well as the minimum span preserving algorithm when the $s_c = 4$. 53

4.5 Results comparing the best results obtained for the RS_1 and RS_2 heuristics. . . . 57

4.6 Results comparing the RS_1 and RS_2 heuristics when altering the size of the preference list. 58

4.7 Comparing computational times of the RS_1 and RS_2 heuristics when altering the size of the preference list. 59

4.8 The preference lists that deliver the best overall results for the RS_1 and RS_2 respectively. 60

4.9 The number of cycles travelled for the generalised assignment problem when the 2 best possible spans of each order are considered. 72

4.10 Computational times in seconds for the generalised assignment problem when the 2 best possible spans of each order are considered. 73

4.11 Results for the tabu search. 80

4.12 Computational times for the tabu search. 81

4.13 Results for the simulated annealing. 85

4.14 Computational times for the simulated annealing. 86

4.15 Results for the genetic algorithm modelling chromosomes as the sequence of orders. 88

4.16 Computational times for the genetic algorithm modelling chromosomes as the sequence of orders. 89

4.17 Results for the genetic algorithm. 93

4.18 Computational times for the genetic algorithm. 94

4.19 Results for the extremal optimisation. 98
4.20 Computational times for the extremal optimisation. 99
4.21 Summary of the number of cycles travelled when the OSP is solved with various heuristic algorithms. 100
4.22 Summary of the number of cycles travelled when the OSP is solved with various metaheuristic algorithms. 101

5.1 Ant colony parameter ranges for the SLP data sets. 114
5.2 Results obtained when implementing the various SLP algorithms. 122
5.3 Best performing configurations obtained when implementing the various SLP algorithms. 123
5.4 The number of picks at a location that is picked most often (most frequent location) for each data set. 123
5.5 Percentage of the shortest spans used when solving 200 instances of random SLP configurations for the 22 data sets considered. 125

6.1 Average number of cycles travelled by using a tabu search when considering two picking lines. 131
6.2 The 10 best pairs of picking lines that resulted in the largest savings in cycles travelled when using the tabu search when considering two picking lines. 132
6.3 The 10 best groups of three picking lines that resulted in the largest savings in cycles travelled when using the tabu search when considering two picking lines. . 136

List of Algorithms

1 A branch and bound approach. . . . 44
2 Subtour generation heuristic. . . . 45
3 Next shortest order heuristic (NSO). . . . 47
4 Preference ratio RS_1 54
5 Preference ratio RS_2 55
6 Greedy RTB 64
7 Greedy RBT 65
8 Greedy RR 66
9 Greedy CLR 68
10 Greedy CRL 69
11 Greedy CR 69
12 Greedy maximal cut reducing heuristic. . . . 76
13 Tabu search. . . . 79
14 Simulated annealing. . . . 84
15 Genetic algorithm. . . . 91
16 Generalised extremal optimisation. . . . 97

17 Organ pipe algorithm. . . . 109
18 Greedy ranking and partitioning algorithm. . . . 110
19 Ant System. . . . 112

List of Acronyms

ACO	Ant colony optmisation
ACS	Ant colony system
AM	Average-link method
AS/RS	Automated storage and retrieval systems
AS	Ant system
AGVS	Automated guided vehicle systems
CAPS	Computer aided picking system
CBT	Column bottom-top
CM	Centroid method
COG	Centre of gravity
CTB	Column top-bottom
CR	Column random
CSCMP	Counsil of supply chain management professionals
DC	Distribution centre
E-GTSP	Equality generalised travelling salesman problem
EO	Extremal optmisation
FIFO	First-in-first-out
GA	Genetic algorithm
GEO	Generalised extremal optimisation
GRP	Greedy ranking and partitioning
GTSP	Generalized travelling salesman problem
KPI	Key performance indicator
L	Locations
LB	Lower bound
MAP	Modified assignment problem
MOP	Multiple order pick sequencing
NN	Nearest neighbour
NSO	Next shortest order
NSOM	Next shortest order relative to minimum span
NSOP	Next shortest order relative to number of picks
O	Orders
OPA	Organ pipe arrangement
OSP	Order sequencing problem
RBT	Row bottom-top
RTB	Row top-bottom
RR	Row random
RS	Ratio spans
SA	Simulated annealing

SCOR	Supply chain operation reference
SKU	Stock keeping unit
SLP	SKU location problem
SM	Single-link method
SOC	Self-organised criticality
SOPS	Single order pick sequencing
SPLAP	SKU to picking line assignment problem
TS	Tabu search
TSP	Travelling salesman problem
VRP	Vehicle routing problem
W	Ward's algorithm

List of Reserved Symbols

Symbols in this dissertation conform to the following font conventions:		
$\mathcal{A}$	Symbol denoting a **set**	(Calligraphic capitals)
$\mathbf{A}$	Symbol denoting a **matrix**	(Boldface capitals)
$\boldsymbol{a}$	Symbol denoting a **vector**	(Boldface lower case letters)

Symbol	Meaning
Indices	
i	An index spanning locations.
j	An index spanning locations.
k	An index spanning orders.
ℓ	An index spanning orders.
p	An index spanning preference spans.
q	An index spanning preference spans.
Sets	
$\mathcal{O}_k$	The set of locations that have to be visited for order k.
$\mathcal{N}$	A set of duples (i, k).
$\mathcal{I}_k$	A proper partition of set $\mathcal{N}$.
$\mathcal{S}$	A set of starting locations.
$\mathcal{E}$	A set of ending locations.
$\mathcal{T}$	A set of subtours.
$\mathcal{A}$	A set of costs used in the assignment problem.
$\mathcal{S}_k$	The set of the lengths of the spans in order k.
$\mathcal{P}$	An ordered set of the spans assigned to order k.
$\mathcal{U}$	An ordered set containing the sequence in which SKUs should be placed into the picking line.
$\mathcal{T}^+$	The set of arcs from the incumbent solution.
$\mathcal{P}_{X,Y}$	The set of orders that have to collect all the SKUs in cluster X and Y.
$\mathcal{P}_X$	The set of orders that have to collect a SKU in cluster X.
Variables	
$x_{k\ell}^{ij}$	A binary variable equal to 1 if order k is picked from location i, completed and then travels to the starting location of order ℓ at location j, and 0 otherwise.
$d_{k\ell}$	A variable equal to the distance travelled when order ℓ is to be completed after order k was picked, and 0 otherwise.

x_{ik}	A binary variable equal to 1 if order k starts at location i and 0 otherwise.
d_{ikj}	A binary variable equal to 1 if order k starts at location i and passes location j and 0 otherwise.
e_{ikj}	A binary variable equal to 1 if order k starts at location i and is completed location j and 0 otherwise.
$x_{k\ell}$	A binary variable equal to 1 if supply point k is assigned to demand point ℓ, and 0 otherwise.
$c_{k\ell}$	The cost of assigning supply point k to demand point ℓ.
c_{ki}	Equal to 1 if order k passes location i and 0 otherwise.
x_i^t	Equal to 1 if SKU t is positioned at location i and 0 otherwise.
s_{ki}	Equal to 1 if order k starts at location i and 0 otherwise.
s_k	The location at which order k starts.
e_k	The location at which order k ends.
O_{ki}	The i^{th} location that order k has to visit.
u_s	The total number of cycles travelled.
d_s	The total distance travelled for the orders in $\boldsymbol{v}_s$.
d_p	The minimum distance that has to be travelled by the orders in $\boldsymbol{v}_p$.
d_m	The maximum of the sum of the minimum spans of the orders in $\boldsymbol{v}_p$.
d_c	The value of the location that is visited by the largest number of orders multiplied by the number of locations in the picking line.
d_t	The total distance travelled by the picker.
p_s	The location of the picker, once the last order in $\boldsymbol{v}_s$ is completed.
d_t	The total distance travelled by a picker to complete all the orders.
c_t	The number of cycles travelled by the picker.
u_k	The number of preference spans considered in order k.
r_k	The ratio of the number of locations traversed when order picking is done on the minimum span divided by the number of locations traversed when order picking is done on the ${u_k}^{\text{th}}$ span.
t_s	An element of a test interval such that $t_s \in \{0, 1\}$.
L_k	Ratio for order k used to construct a preference list.
N_k^p	Ratio for the p^{th} best span of order k used to construct a preference list.
N_C	The number of locations containing the maximal cut.
E_x	The energy for a state x.
N_C^x	The number of locations containing the maximal cut for a state x.
ΔE	Variation in solution quality.
$\overline{\Delta E}$	The average variation in solution quality.
T_y	The temperature after the y^{th} alteration.
c_k^i	Scoring the k^{th} starting location of order i.
p_k^i	Normalising the score c_k^i, resulting in a probability of starting order k at location i.
r	A random number between 0 and 1.
f_z	The fitness of chromosome z.
C_z	The maximal cut of chromosome z.
N_C^z	The number of locations containing the maximal cut in chromosome z.
Δf_x	The change in performance of gene x.
r_x	The rank of gene x.
a	The index representing an ant.
η_{ij}	The visibility between two SKUs i and j.
τ_{ij}	The intensity of the trial between SKUs i and j.

p_{ij}^a	The rule of displacement between SKU i and j for ant a.
J_i^a	The list of SKUs that have already been visited, when ant a is currently at SKU i.
$\Delta\tau_{ij}(t)$	The total amount of pheromones left by all the ants between location i and j during iteration t.
c_h	The number of cycles travelled on a picking line h.
p_h	The number of picks in a picking line h.
Ω	The total number of cycles currently travelled within a picking line.
$p_{h\omega}$	The number of picks in each cycle within picking line h.
s_h	The number of SKUs in picking line h.
μ_c	The average number of cycles travelled for all picking lines considered.
σ_c	The standard deviation of μ_c.
μ_p	The average picks per cycle for all picking lines considered.
σ_p	The standard deviation of μ_p.
P_{kh}^t	Equal to 1 if SKU t is in order k in picking line h.
$z_{k\ell rh}^{ij}$	Equal to 1 if a picker r in picking line h travels from location i in order k to location j in order j.
$y_{k\ell h}^{ij}$	Equal to 1 if the inactive picker in picking line h travels from location i in order k to location j in order ℓ.
x_{ih}^t	Equal to 1 if SKU t in picking line h is in location i.
w_{ih}^k	Equal to 1 if location i in picking line h must be visited in order k.
C^{TT}	The length of a nearest neighbour tour of length T.
L^+	The size (length) of $\mathcal{T}^+$.
$x_{\max}$	The largest element in a distance matrix.
$d_{X,Y}$	The distance between cluster X and Y.
n_X	The number of elements in cluster X.
Z_X	The centroid of cluster X.

Vectors

$\boldsymbol{b}_s$	The current best sequence of orders.
$\boldsymbol{v}_s$	Keeps track of the orders that have been inserted to a new sequence of orders to be picked.
$\boldsymbol{v}_p$	All the orders that have not been added to the sequence of executable orders.
$\boldsymbol{s}$	A vector of starting locations.

Parameters

m	Number of locations.
n	Number of orders.
$d_{k\ell}^{ij}$	The distance travelled when order k is picked from location i, completed and then travels to the starting location of order ℓ at location j.
s_c	The scope ahead of the current location of the picker's vision.
$Q+1$	The number of preference spans from which to select an appropriate candidate span.
τ_0	The initial rate of acceptance.
T	The number of SKUs assigned to a picking line.
A	The total number of ants.
α	A parameter used for relative importance of the trail intensity.
β	A parameter used for relative importance of the trail intensity.
W	A fixed parameter $\in (0, \infty)$.

ϱ	A parameter used to raise the power of the Euclidean distance between the centres of clusters.
ρ	The evaporation rate.
$t_{\max}$	The maximum number of iterations.
D	A distance matrix.
q_0	A parameter for quanitifying the rule of transition.
τ_1	The initial pheromones on the trial.
R	The number of pickers.
R_h	The number of pickers assigned to picking line h.
H	The number of picking lines in the DC.
ζ	A parameter between 0 and 1.
$D^{ij}_{k\ell h}$	The number of locations that must be picked from, when starting at location i in order k and travelling to location j in order ℓ in picking line h.

Other	
$\lvert\mathcal{O}_k\rvert$	The number of elements in the set $\mathcal{O}_k$.
(i,k)	A duple representing order k starting at location i.
M	A large constant.
S^i_k	Represents order k starting at location i and ending at the closest possible ending position.
e^i_k	The closest possible ending position given order k starting at location i.
C	Maximal cut.
$S^{\min}_k$	The minimum span of order k.
P^p_k	The p^{th} preference span of order k.
$P(r_x)$	The probability of modifying gene x with fitness rank r_x.
$\Delta\tau^a_{ij}(t)$	The quantity of pheromones ant a leaves on the course when travelling from location i to location j.
$T^a(t)$	The path traversed by ant a during iteration t.
(i,j)	An arc from location i to location j.
$C(x)$	The x coordinates of the cluster centroid.
X	A cluster of SKUs.
Y	A cluster of SKUs.
$\text{ESS}(X)$	The error sum of squares of a cluster X.

CHAPTER 1

Introduction

Contents

1.1 A supply chain 1
1.2 A distribution centre 2
1.3 The Pep DCs 3
1.4 Order picking and picking lines 4
1.5 Problem description and project scope 4
1.5.1 The order sequencing problem *5*
1.5.2 SKU location problem *5*
1.5.3 SKU to picking line assignment problem *6*
1.6 Project objectives 6
1.7 Project organisation 6

Order picking is known as the most important activity in warehouses [122]. It involves the process of clustering and scheduling orders, assigning stock to locations in picking lines, releasing orders to the floor, picking the pallets from storage locations and the disposal of picked products [28]. Poor performance in order picking may lead to unsatisfactory output and high costs incurred by a warehouse.

1.1 A supply chain

The Council for Supply Chain Management Professionals (CSCMP) defines supply chain management as encompassing:

> "The planning and management of all activities involved in sourcing and procurement, conversion, and all Logistics Management activities. Importantly, it also includes coordination and collaboration of channel partners, which can be suppliers, intermediaries, third-party service providers, and customers. In essence, Supply Chain Management integrates supply and demand management within and across companies" [47].

The supply chain encompasses every effort involved in producing and delivering a final product, from the supplier's supplier to the customer's customer. Four basic processes: plan, source,

make and deliver broadly define these efforts, which include managing supply and demand, sourcing raw materials and parts, manufacturing and assembly, warehousing and inventory tracking, order entry and order management and distribution across all channels and delivery to the customers [71]. There are four key elements in the supply chain that deliver the product to the final customer, namely the suppliers, manufacturers, distributors and retail stores.

The supply chain is initially provided with raw materials from the suppliers. The suppliers are at the root of the supply chain. The raw materials provided by the suppliers must be used to produce products for the final user.

Manufacturers process the raw materials into goods that is in a form or condition requested by the customers. Manufacturers may use a series of processes to transform the raw material into a useful product. Manufacturers make use of human activity and technology to create finished goods on a large scale. A series of manufacturers, each adding different processes, may be used to work on the raw materials to create the final goods.

After the manufacturers have delivered the final product, the distributors (or wholesalers), maintain a warehouse of stock. Stock is usually received in bulk from the manufacturers, after which the stock is reassigned and redistributed to the retailers [100]. A distributor often makes use of a distribution centre (DC) which is a warehouse that permits the functions of receiving, transfer and storage, order picking, shipping and sometimes cross-docking [28]. The receiving activity includes unloading the goods from the transportation vehicle, inspection of the goods to determine if the correct products are delivered, the delivered products are in a good condition with no damages and that the correct quantities thereof have arrived. The inventory records are updated accordingly, after which the new inventory is transferred into storage locations. Before a product is stored, it may be repackaged into a different (standard) size container used in the warehouse. The process of order picking is an operation in which quantities of the correct products are selected for each customer (retail) order. The shipping process entails a final quality check of outgoing orders and the distribution thereof.

The retailers are the shops or stores which sell the completed products received from the distributors in smaller quantities directly to consumers, who are the users/owners of these final products.

1.2 A distribution centre

Distribution centres are warehouses used for storing and buffering products before products are dispatched when needed downstream in the supply chain. A distribution centre has a vital role to play in the supply chain of most companies [100]. The distribution centre contribute to a significant cost within a supply chain. Warehouses and DCs are not identical in general, but in the case of Pepstores Ltd. (Pep) the DC and the warehouse refer to the same part of the logistics system, since traditional activities taking place in a warehouse and a DC is done at the same location. For the purpose of this thesis, a DC and a warehouse refer to the same part of the logistics system. This facility controls the operations from its boundaries as goods are moved into the warehouse, handled and stored, and moved from the warehouse to the customers [113]. Warehouses often involve large investments and operating costs [28].

Products typically arrive packaged on a larger scale and leaves packaged on a smaller scale. An important function of a warehouse is to break down large chunks of product and redistribute it in smaller quantities [12]. Warehouses are used to support manufacturing operations, where products in bulk are mixed and consolidated into shipments for the following users.

Operations within a distribution centre are usually more labour intensive than other operations within the supply chain. The reordering of incoming products may be done by means of two processes: *inbound processes* and *outbound processes* [12]. Inbound processes include receiving and put-away of goods, while outbound processes include order-picking, checking, storage and shipping of goods.

The efficiency of warehouse operations is influenced by facility design, storage and replenishment methods and picking policies [49]. In order to reduce unnecessary handling of products, a general rule is that products should flow continuously through this sequence of processes [12]. To meet customer demands for shorter replenishment times and lower prices, warehouse processes have to be examined for productivity and cost improvements [26]. A prominent design criterion is the maximum throughput, to be reached at minimum investment and operational cost [96].

1.3 The Pep DCs

Pep is a retail company based in South Africa. Founded in 1965, Pep operates in 11 countries in Southern Africa. In 2010, Pep sold over 500 million items through over 220 million customer transactions [69]. Pep serves at least 100 customers every five seconds [85]. Currently Pep has more than 1 500 stores and is the biggest single brand chain store in Southern Africa and employs more than 15 000 people.

This thesis focuses on Pep as the distributor. The DC in Durban is classified as a retail distribution centre, as its direct customers are the Pep retail stores [100]. A large and continuous flow of products in the DC is maintained to satisfy the demands of more than 1 200 retail outlets that the Durban DCs serves.

A DC receives a notification that an arrival for cartons of product will be arriving in the near future. The DC then schedules accordingly for the receipt and unloading of the cartons upon arrival. Once products arrive at the DC via a truck, the cartons are unloaded and moved to the *Goods Received* area. A control check is performed where the quality of the products, the number of products and the various sizes of the products are investigated to determine whether the goods may be stored or set aside to be returned to the suppliers. If the cartons clear the control check, the cartons receive a label (sticker) that contains the details of the contents of the carton. The labels are scanned to register the arrival of the products to the computer system. The cost associated with receiving accounts for approximately 10% of the operating cost in a DC [28].

The manner in which the cartons are handled are of great importance, since the locations where stock keeping units (SKUs) are stored has a significant influence on how much time will be spent to collect products when needed. The goods are handled by means of two different ways in the *Goods Received* area. Cartons that need not be opened before shipment are moved to the *Full Carton Area*. These cartons are distributed to retailers without being repackaged. Cartons that contain more products than needed by a single retail outlet need to be repackaged into smaller cartons, to meet the demand of retail outlets. These cartons will be used during picking to make cartons specially packed with various products needed by retail outlets. The cartons may be moved to the *Storage Racks* or is immediately moved to a picking area that is organised in a picking line. The cost associated with storage is approximately 15% of operating costs in a DC [28].

When a picking line is built, people (called pickers) walk along the picking line picking SKUs which are placed into a carton destined for a particular retail outlet. The requests of a retail

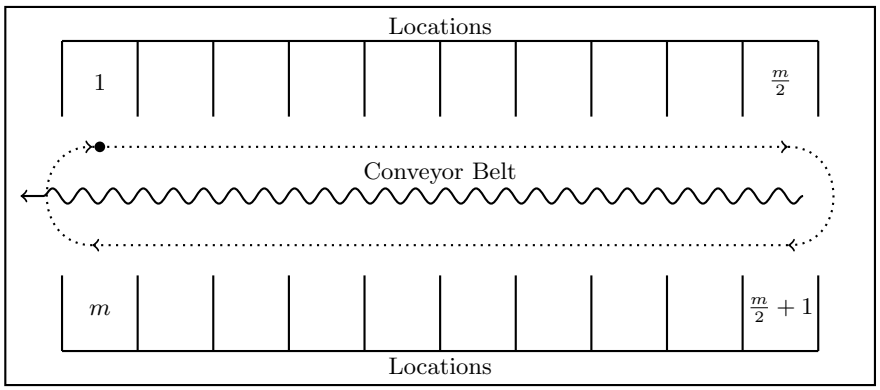


Figure 1.1: *A schematic representation of the physical layout of a picking line containing m locations.*

outlets is referred to as an *order*. When an order from a retail outlet is completed, the picker places the carton on a conveyor belt. The carton is moved to the *Distribution Area* where the cartons will be collected to be shipped to the various retail outlets once final quality and quantity checks are completed. The cost associated with order picking is approximately 55% of operating costs in a DC [28, 30].

1.4 Order picking and picking lines

Order picking is defined as the process by which appropriate quantities of products are retrieved from storage locations to fulfil customer orders [63]. Order picking is done on a picking line. A picking line is the layout of various SKUs in such a way that pickers may easily acquire various SKUs for retail outlets. Order picking is a labour-intensive task in warehousing — improving the performance of order picking may lead to huge savings in warehousing costs [58]. The efficiency of order picking is dependable on factors such as the storage racks, warehouse layout and control mechanisms and philosophies of companies.

A picking line resembles a flow line in which each SKU has a unique location or set of locations on the picking line. Pickers move along the picking line collecting items for locations. Each picker has a set of orders assigned to him/her. Usually an automated system communicates instructions to the picker *i.e.* to which location to move, the number of items to collect, when a order is completed and when a new order may be started. A picking line may be constructed in some form of a cycle, where a picking line's end position is connected to the starting position. Figure 1.1 contains a schematic representation of a typical picking line used in Pep's DC.

1.5 Problem description and project scope

Pep is interested in improving the efficiency of the picking lines within their DCs and consequently lowering its operating costs. The scope of this thesis is to improve picking operations in Pep's DC in Durban, South Africa. Initially, it is necessary to determine which SKUs must be sent to the available picking lines. SKUs within picking lines then have to be assigned to various locations within that picking line. Finally, orders within a picking line have to be sequenced with the aim of minimising the total distance travelled by all pickers, while assigning an equal workload to each picker.

Pep has identified a bottleneck in its supply chain to be the operations within the picking

lines. The picking line schedulers construct SKUs on a picking line based on past experiences and "rules-of-thumb". Picking line schedulers rely on the feedback from pickers on the picking lines [80] to generate these rules. The only measurement of the effectiveness of the picking lines available to the schedulers, is the time required to complete operations within a picking line.

The planning of picking lines may be divided into three tiers of decisions. The first tier determines which SKUs should be allocated to which picking line. The problem of assigning scheduled SKUs to available picking lines is referred to as the SKU to Picking Line Assignment Problem (SPLAP). The second tier is called the SKU to Location Problem (SLP) and considers the positioning of the various SKUs in a picking line. The final tier considers the sequencing of the orders for pickers within a picking line and is referred to as the Order Sequencing Problem (OSP). All of these subproblems aim to achieve the objective of picking all the orders in the shortest possible time.

The decisions associated with each tier are made sequentially during the planning of a picking line. Firstly, it has to be decided which SKUs are assigned to which picking lines, then within each picking line each SKU has to be assigned to a specific location, and finally the sequence in which the orders should be picked must be determined.

Each tier therefore relies on the information generated by its predecessor. For example, the OSP will rely on the SKU locations generated by the SLP. Similarly, to solve a tier the solution to the successive tier must be calculated. For example, to evaluate a candidate solution for an instance of the SLP will require the sequencing of orders generated in the OSP. Due to this exchange of information between subproblems, the first problem that needs to be solved is the OSP.

1.5.1 The order sequencing problem

The DC uses an order picking system which is based on the concept of a *wave*. A wave may be described as the set of SKUs in conjunction with a set of branches requiring at least one of these SKUs. All the orders, or requests by the branches for that SKUs for that wave are picked in a single operation. The OSP may be described as the sequencing of all the orders, for each picker, given a wave of SKUs assigned to distinct locations in a picking line, such that the total picking time is minimised.

Each order requires a number of distinct SKUs in various quantities. A picker must visit each location containing the SKUs required by that order and collect the requested number of SKUs. A picker may only commence a new order once all the SKUs have been collected from the current order. Pickers are required to move in a clockwise direction when collecting SKUs. Since each subproblem relies on its predecessor, the OSP relies on which SKUs are assigned to the picking line and where the SKUs are situated within that picking line.

1.5.2 SKU location problem

One tier up it should be investigated whether improvement in the time needed to pick all the orders may be achieved by reassigning the SKUs to other locations within a picking line. It is assumed that a set of SKUs are fixed to a picking line for a given wave of picking. It is important that SKUs may not be duplicated in a picking line. Each location contains a distinct SKU within a picking line.

1.5.3 SKU to picking line assignment problem

The SPLAP combines the OSP and the SLP and relies on the solution approaches of both the OSP and the SLP to be solved. The SPLAP divides all the SKUs available from distribution among the picking lines in a DC in such a way to minimise the time necessary to complete all the orders over all the picking lines. A SKU may only be assigned to a single picking line during a wave of picking.

1.6 Project objectives

This thesis is aimed at optimising the current state of the operations in the Pep DC directly concerned with order picking. Towards realising this aim, seven objectives are persued throughout the thesis.

Objective I: To present an exact formulation for each problem, in order to better understand the complexity of the problems at hand.

Objective II: To examine the complexities of each problem, identify a level of difficulty in finding good solutions and find various approaches of addressing each problem.

Objective III: To formulate and solve the OSP by means of heuristics and metaheuristic approaches.

Objective IV: To address the possibilities of improvements within the SLP.

Objective V: To determine various measures for defining a balanced picking line in terms of the balance of work amoungst the workers, minimising the distance travelled by the pickers and adhering to managerial requests.

Objective VI: To compare the performance of the solution approaches for each problem with respect to the solution quality and execution time.

Objective VII: To comment on open questions and to suggest further studies that may have an impact on the current state of the Pep DC operations.

1.7 Project organisation

In Chapter 2 a brief introduction to operations within supply chains are presented. Basic concepts of logistics management is discussed. Logistics processes are examined as well as the impact of these processes on the entire supply chain. Emphasis is placed on the importance of order picking systems. A large range of order picking configurations are presented that are used to support the handling of various SKUs that may be found in practise. Order batching systems are introduced as well as conventional routing policies. The notion of carousels are considered since the picking lines within the Pep DC may be modelled as unidirectional carousels, which is relatively unexplored and rarely used in practise. Picking area zoning strategies are discussed, which cannot be employed by the Pep DC, due to constraints by the system and the layout of the picking lines.

In Chapter 3 the operations within the Pep DC are examined. The operations within the Durban DC is investigated as well as some references to the DC in Kuilsrivier. The physical layout and

building of a picking line is sketched. The interaction between the system and the pickers is explained as well as the manner in which SKUs are assigned to picking lines. Control procedures for coordinating activities within the DC and the rules imposed by Pep are presented.

Chapter 4 contains all the aspects of the OSP. An exact formulation is presented, however, the computational complexities of this formulation is too excessive to solve real-life problems. A number of definitions are introduced to simplify the description of the OSP, which results in approaches that may be used to create a relaxation of the initial formulation. Matthews [77] presents a tight lowerbound for the OSP. Heuristic as well as metaheuristic approaches are developed to deliver good solutions that requires less computational time. All approaches used to address the OSP are compared and recommendations are made on the suitability of each approach.

Chapter 5 addesses the SLP. A formulation is presented that delivers a lowerbound. Approaches from the literature are used to address the SLP. A metaheuristic approach and well-known agglomerative clustering algorithms are also adapted to be implemented. Two novel clustering approaches are contructed for the SLP. These approaches are compared with one another.

Chapter 6 contains a nearest neighbour approach and a tabu search methodology is implemented to solve the SPLAP. The SPLAP is addressed where a set of *waves* are clustered and new *waves* are constructed.

Finally Chapter 7 contains the conclusions and closing remarks of the thesis. The chapter closes with a number of ideas and recommendations with respect to further work.

CHAPTER 2

Literature review

Contents

2.1 Introduction to the supply chain 9
2.2 Warehousing 10
2.2.1 The nature and importance of warehousing 10
2.2.2 Warehouse operations 10
2.3 Order picking systems 11
2.4 Order batching systems 14
2.5 Sequencing and routing policies 15
2.5.1 Sequencing and routing for conventional single-block warehouses 16
2.5.2 Sequencing and routing for conventional multi-parallel-aisle systems . . 17
2.5.3 Sequencing and routing AS/RS 18
2.6 Carousels 18
2.7 Picking area zoning 20
2.8 Chapter summary 21

Order picking is the most labour-intensive operation in a warehouse with manual systems [46]. The organisation of order picking operations impacts on the distribution centres and thereby the supply chain's performance [28]. Warehouse design, storage assignment and planning the routes of pickers may be used to enhance the operating efficiency and the space utilisation and reduce order picking costs [57].

2.1 Introduction to the supply chain

Supply chains may be explained by means of the product life cycle processes comprising physical, information, financial and knowledge flows whose purpose is to satisfy end-user requirements with physical products and services from multiple, linked suppliers [5]. Figure 2.1 displays a typical supply chain. A supply chain may be altered in order to suit the needs of a specific business.

Supply chain management (SCM) consists of businesses collaborating in order to leverage strategic positioning and to improve operating efficiency [21]. All the firms involved in a particular supply chain operation is part of a relationship based on strategic choice. Any supply chain strategy is dependant on collaboration between the parties involved.

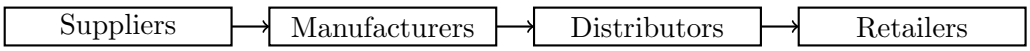


Figure 2.1: *The predominant flow in a typical supply chain.*

2.2 Warehousing

Warehousing may be defined as being part of a firm's logistics system that stores products (raw materials, parts, goods-in-process and finished goods) at and between the point of origin and point of consumption, and provides information to management on the status, condition and disposition of items being stored [47].

Cromier & Gunn [27] classifies warehousing problems into three major categories: *throughput capacity models, storage capacity models* and *warehouse design models*. Throughput capacity models refer particularly to storage assignment policies, in which incoming products are matched with available storage locations. Focus is placed on reducing material handling cost, inventory holding cost and reordering cost. Storage capacity models aim to find the warehouse size which may either minimise total discounted cost or allow a required service level. Warehouse design models primarily optimise the external warehouse configuration, while taking the internal capacities of the warehouse into consideration.

2.2.1 The nature and importance of warehousing

Warehousing usually provided storage of products during all the phases of the logistics process. Two basic types of inventory may be placed into storage, namely the *physical supply* and the *physical distribution*. Physical supply comprises of raw materials, components and other parts, while physical distribution refers to finished goods.

Warehouses may be used to support manufacturing, to mix products from multiple production facilities for shipment to a single customer, to breakbulk or subdivide a large shipment of product into smaller shipments to satisfy the needs of many customers and to combine or consolidate a number of small shipments into a single higher-volume shipment [47].

2.2.2 Warehouse operations

The main warehouse activities include: *receiving*, *transfer and put away*, *order picking*, *accumulation/sortation*, *cross-docking* and *shipping* [28]. Figure 2.2 displays a typical layout of a warehouse with its functional areas and flows.

Receiving is the process of unloading products from the trucks, inspecting the unloaded products for quality or quantity errors and updating the inventory record. The transfer and put away activity involves the moving of the received products to storage. Order picking is the process of retrieving the correct quantity of the right product for customer (branch) orders. Cross-docking is performed only when products are transferred directly to the shipping area. These products require no order picking and storage and remains in the warehouse for a short period of time. The shipping process is concerned with traversing products to retail outlets.

A good warehouse system should ensure easy and efficient access of merchandise, properly use the storage location to find the shortest path, and finally to deliver the merchandise in a

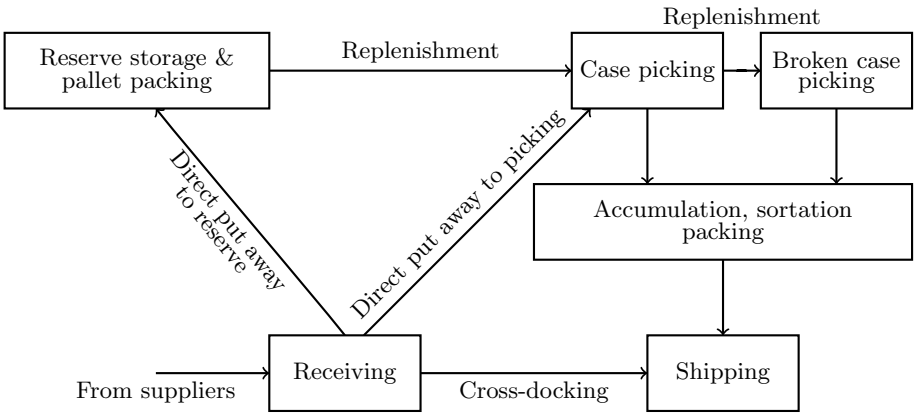


Figure 2.2: *Typical warehouse functions and flows [104].*

reasonable time [57]. Companies are required to cope with uncertain changes in production volume, product mix, and product life cycle [55].

2.3 Order picking systems

Order picking involves the process of clustering and scheduling orders, assigning stock on locations to picking lines, releasing orders to floor, picking the pallets from storage locations and the disposal of picked products [28]. There are usually more than one order picking system employed within a warehouse. These order picking systems may be fully automated or handled by humans, but most systems employ humans as order pickers. In a typical warehouse approximately 65% of operating expenses are consumed by order picking [97].

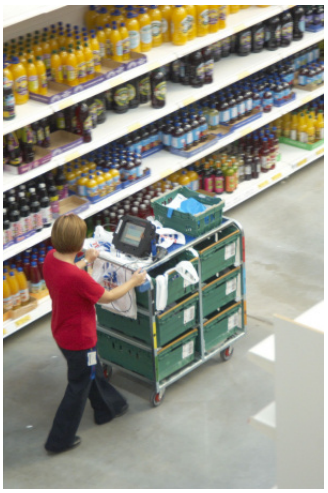
In the *picker-to-parts* system the picker moves along the aisles to pick items. There are two distinguished picker-to-parts systems, the *low-level* and *high-level* picking. The low-level picking systems entails that the pickers picks the requested items from a bay (storage rack), while travelling along the storage aisles. The high-level picking system uses high storage racks and the picker is moved by a crane or a truck to the destination of the bay from which the picker has to collect items. This system is also known as a *man-aboard* picking system [28]. Ruben and Jacobs [97] refers to this system as a *person-aboard automatic storage/retrieval* systems.

For information support system, a computer aided picking system (CAPS) or electronic paperless *pick-to-light* system may be implemented in practise in a *pick-and-pass* system [86]. A pick-to-light system uses a hands free terminal to inform pickers on instructions. Benefits of this system include quick and accurate order picking. The pick-and-pass system consists of a conveyor connecting all bays (locations) located along the conveyor line. Figure 2.3 displays examples of typical picker-to-parts systems.

Parts-to-picker systems include automated storage and retrieval systems (AS/RS). This system is isle-bound and retrieves pallets from storage to a picking position for the picker to collect via a crane. Once the picker has completed the required picks, the crane returns the pallets and collects another for order picking. Figure 2.4 displays examples of typical parts-to-picker systems.

Other technological options include micro stackers and automated carousels, powered conveyors and Automated Guided Vehicle Systems (AGVS), automated sorting packages, pick list generation software, and bar coding [49]. These technologies usually required a substantial capital

(a) A picker collecting a product (Source: [106]).



(b) A picker travelling with a trolley (Source: [62]).

Figure 2.3: *Examples of picker-to-parts systems.*

investment, since these systems have to be installed in an integrated way. Benefits of AVGS include lower handling costs, reduced handling-related product damage, improved safety, ability to interface with other automated systems and increased reliability [47]. Robots may also be used in many processes of materials handling. Robots have developed due to robotics technology and have expanded their use to a larger number of applications. Robots are usually automatic guided vehicles capable of storing and retrieving products and in some instances may also build pallet loads of product as well as stacking pallets of product.

(a) A crane retrieving products (Source: [116]).

(b) A typical carousel system (Source: [102]).

Figure 2.4: *Examples of parts-to-picker systems.*

A *put-system* or *distribution system* uses either a parts-to-picker or a picker-to-parts to retrieve items from storage. A picker then sorts the pre-picked items into customer orders. This systems delivers good results for customer orders with short time periods.

Order picking systems may also be done by automated processes, which is usually used in special cases. Automated process is used when handling valuable items or small and delicate items or when the cost of labour is too high [28].

The design of real order picking systems is often complicated, due to the wide spectrum of external and internal factors which impact on design choices [28]. Factors that influence external considerations of the order picking system include marketing channels, customer demand

patterns, supplier replenishment patterns and inventory levels, the overall demand for product and the state of the economy [46]. Internal factors include system characteristics, organisation and the operational policies of order picking systems [28].

Different order picking methods may be employed in a warehouse, *e.g.* single-order picking, batching and sort-while-pick, batching and sort-after-pick, single-order picking with zoning, and batching with zoning [121]. Each order picking method consists of some or all of the following basic steps: batching, routing and sequencing, and sorting [50].

The objective of the order picking system is to minimise the order retrieval time of all orders on a picking line *i.e.* the time that is needed to pick all orders assigned to a picking line. The sooner an order is retrieved, the sooner that order may be shipped to the retail outlets and the sooner other orders may be completed. In the case of manual-pick order picking systems, the travel time is an increasing function with respect to travel distance [28].

Thompkins *et al.* [104] indicates that the travel time of a typical picker in a picker-to-parts system, is approximately 50% of the total time a picker spends in the picking line during order picks. Bartholdi and Hackman [12] states "travel time is waste. It costs labour hours but does not add value." Figure 2.5 displays the distribution in which a picker's time is spent on average in a picker-to-parts order picking system.

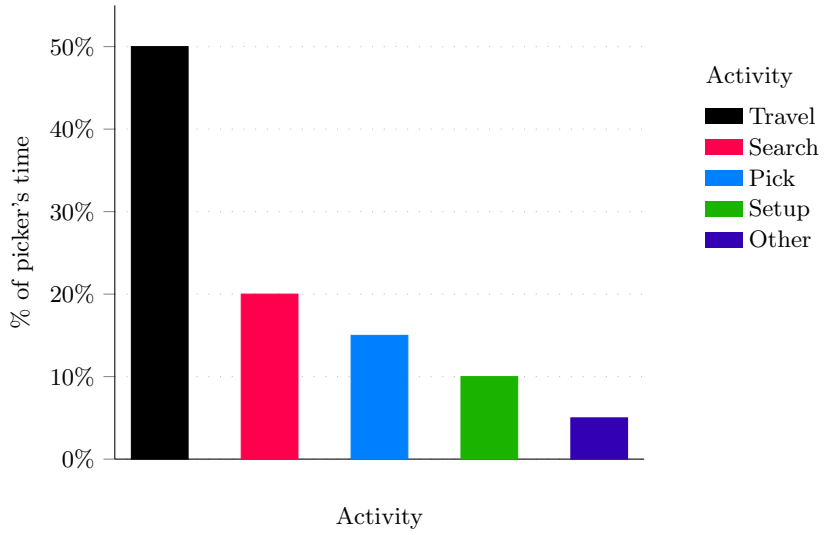


Figure 2.5: *A distribution of how a picker's time is spent [104].*

It is clear that minimising the average distance travelled by all pickers should be an objective of the distribution centre. Other important objectives might include minimising the total cost associated with the order picking process. Objectives which are often taken into consideration in warehouse design and optimisation include minimising the throughput time of an order, minimising the overall throughput time, maximising the use of space, maximising the use of equipment, maximising the use of labour and maximising the accessibility to all items [28].

Management decisions concerning warehousing and order picking systems are classified into two categories: *strategic management decisions* and *control decisions* [76]. Strategic management decisions are the long-term decisions that concern broad policies and plans for a company's resources in order to implement a competitive strategy. Control decisions influence short-term and operational aspects of operations within a company — the routing, sequencing, planning and batching problems.

2.4 Order batching systems

Given a set of released orders, the problem is to partition the set into batches, where each batch will be picked and accumulated for packing and shipping during a specific time window, or "pick wave" [50]. A wave may be described as the set of products in conjunction with a set of orders requiring at least one of the products. All the orders for the products for that wave are picked as a single operation. Each product is therefore completely picked for all orders during one wave. The problem of partitioning orders amoung the pickers is a variation of the classical vehicle routing problem (VRP), in which "stops" are assigned to routes and the objective is to minimise the total route distance or time [50].

Batching is used to minimise the impact of idle-time during the time it takes to complete the processing of all orders [4]. Order batching is the method of grouping a set of customer orders into a number of sub-sets, such that multiple customer orders are completed simultaneously rather than completing one at a time [28]. It is usually easier to assign one picker to attend to large orders. When orders are small, travel times may potentially be reduced by picking a set of orders in a single picking tour.

There are two criteria for batching: the *proximity of pick locations* and *time windows* [28]. Proximity batching is the assignment of each order to a batch based on the proximity of its storage location to those of other customer orders. The difficulty then is to calculate ways in which to measure the proximity's among orders. Time window batching entails that orders arriving within the same time interval, are grouped as a batch. This batch may be processed in the following stages of operations. Realistic time windows need to be established in order to process the orders arriving during a time window.

Gu *et al.* [50] describes the batching problem as given, a warehouse configuration, pick wave schedule and set of orders to pick during a shift, determine the partition of orders for assignment to waves and pickers, subject to performance criteria and constraints such as picker effort, imbalances amoung workers, time slots, picker capacity and order due dates.

This may be reduced to two levels of considerations: partitioning work into time slots and partitioning work amoung pickers within a zone or picking line. Partitioning work into time slots may be described as a *bin-packing problem*, where the objective is to balance the workload amoung the workers. The difficulty is that the time (or distance travelled) in order to pick a batch may only be determined once the batch has been determined, partitioned amoung the individual pickers and routed through the warehouse (or picking line).

There are two major types of batching heuristics that attempt to minimize total picking effort and are based on VRP heuristics [50]. A *seed algorithm* selects an initial seed order to form a batch. Additional orders are added to this batch according to an order closeness metric, until a capacity constraint prohibits more orders from being placed into a batch. *Saving algorithms* starts by assigning each order to a unique batch. The algorithm iteratively combines pairs of batches until no more batches may be combined due to the capacity constraints.

A number of order closeness metrics, found in literature, have been summarised by Gu *et al.* [50] and are displayed in Table 2.1. Closeness metric **(1)** calculates the total distance in number of locations traversed when performing order picking. Closeness metric **(2)** determines the number of locations used to serve two or multiple orders, while closeness metric **(3)** measures the distance between adjacent orders, while taking the distance of each order into consideration. Closeness metric **(4)** uses a two-dimensional spacefilling curve (SFC), which is a continuous mapping from a point, θ, on the unit square, where $0 \leq \theta \leq 1$. This measures the clockwise revolutions

Index	Closeness metric	Example
(1)	Number of locations between two orders	Elsayed [36]
(2)	Combined number of locations for multiple orders	Elsayed & Stern [37]
(3)	Sum of the distance between each location of one order and the closest location on the other order	Elsayed & Stern [37]
(4)	Difference of the order theta values of two orders based on space-filling curves	Gibson & Sharp [43]
(5)	The number of additional aisles to travel when two orders are combined	Rosenwein [95]
(6)	Savings in travel when two orders are combined	Elsayed & Unal [38]
(7)	Centre of gravity metric	Rosenwein [95]
(8)	Economic convex hull base metric	Hwang & Lee [59]
(9)	Common covered regions or areas	Hwang *et al.* [60]

Table 2.1: *Order closeness metrics for batching systems [50].*

removed from a fixed reference point. The value $\theta = 1$ corresponds to 360°. For different values of θ the SFC traces a tour of all the points in a unit square [13]. Closeness metric **(5)** compares the distance required to be traversed to pick two particular orders if they are picked together on a picking tour with the distance required to pick each order by itself, while closeness metric **(6)** determines the savings in time travelled when orders are combined and picked in a single sequence. The centre of gravity (COG) metric **(7)** aims to maximise the total distance saved when a subset of orders are considered to be picked simultaneously. The number of aisles traversed is considered in this case. Closeness metric **(8)** uses a convex hull based on finding boundary points by means of the locations of an order or set of orders. This is used then to cluster orders, by analysing the intersection of the convex hulls of two orders. Closeness metric **(9)** relies on cluster analysis techniques to cluster (batch) orders to minimise the total distance travelled.

2.5 Sequencing and routing policies

The purpose of picker routing planning is to reduce the unnecessary picking distance that in turn results in the shortest and most efficient picking [57]. Most order picking systems do not sequence a picker's path optimally during order picking, since most picking systems does not know the layout and dimensions of the warehouse. Industry mainly applies heuristic methods to solve the problem of routing pickers through a warehouse [100]. The order picking in a warehouse is a special case of the travelling salesman problem (TSP) since travel is restricted by aisles of products, and due to this structure it is possible to quickly find optimal solutions by computer [12, 50].

Numerous methods have been proposed for multiple aisle picking lines to minimise the overall travel time, and thus travel distance, within the operation of order picking [100]. Products are stored in longitudinal pick aisles. Cross aisles are located perpendicular on the pick aisles and allow for an efficient movement from one pick aisle to the next [103]. A graphical representation of a typical multi-parallel-aisle layout is displayed in Figure 2.6.

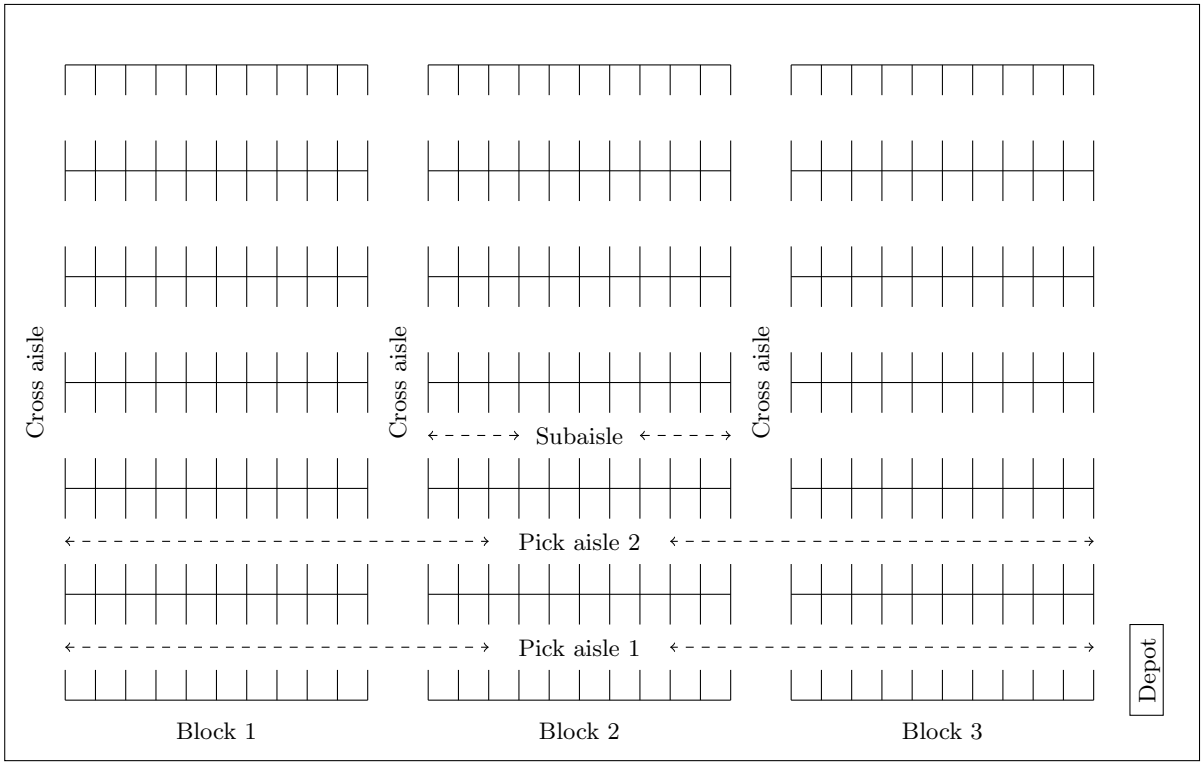


Figure 2.6: *Example of a warehouse layout with multiple cross aisles [94].*

2.5.1 Sequencing and routing for conventional single-block warehouses

Industry mainly applies heuristic methods to solve the problem of routing order pickers through a warehouse during the picking process. These are typically applied to a single-block warehouse. Schematics of these are found in Figure 2.7. For each method, the same customer order is considered. The simplest of these methods is the *S-shape* heuristic as shown in Figure 2.7(a). This heuristic operates by traversing an entire aisle if at least one product has to be picked within an aisle. If no product has to be picked in an aisle, the aisle is not traversed.

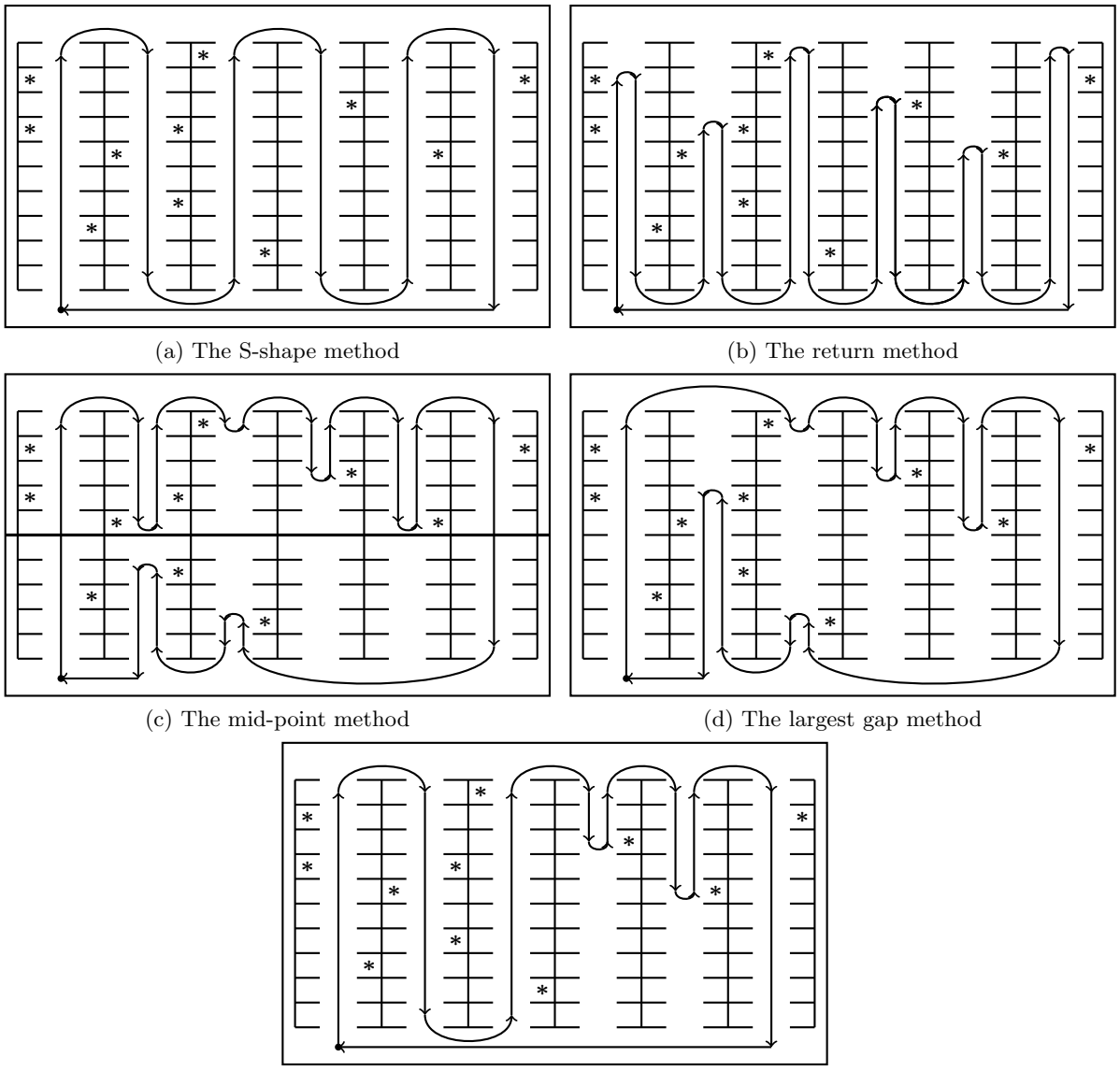
The *return* heuristic, as depicted in Figure 2.7(b) is another example of a simple routing heuristic where a picker enters only those aisles containing at least one pick and leaves them from the same end at which the picker entered the aisle.

The *mid-point* heuristic is one which delivers better results than the S-shape heuristic and is displayed in Figure 2.7(c) [28]. Here, the aisles are divided into two areas where picks in the front half are accessed from the front cross aisle, and those in the back half, from the back cross aisle. The last or first aisle visited is used by the picker to move between the two areas.

The *largest gap* heuristic is displayed in Figure 2.7(d) and is similar to the mid-point method. In this strategy the picker will enter an aisle as far as the largest gap instead of the midpoint. The gap is the separation between any two adjacent picks, the first pick and the front cross aisle or the last pick and the back cross aisle. The picker will return from both ends of the aisle if the largest gap is between two adjacent picks. Otherwise the return route will be either from the front cross aisle or the back cross aisle. Thus the portion of the aisle that is not traversed by the picker is the largest gap.

The *combined* or *composite* heuristic merges these concepts, as shown in Figure 2.7(e). A picker may either traverse an aisle containing picks entirely, or enter and leave at the same end, the

choice is made by means of dynamic programming [28].



(a) The S-shape method

(b) The return method

(c) The mid-point method

(d) The largest gap method

(e) The composite method

Figure 2.7: *Diagrams depicting the five different routing methods of pickers between aisles for order picking within a single-block warehouse. The same order is considered for all methods. The asterisks indicate the locations that need to be picked from and the arrows depict how the picker will move between the isles. (Source: [100]).*

2.5.2 Sequencing and routing for conventional multi-parallel-aisle systems

In a conventional multi-parallel-aisle system, the aisle structure limits the TSP state space, which greatly simplifies its solution [50]. Ratliff and Rosenthal [91] formulated a dynamic programming algorithm to optimally solve the problem. The algorithm may solve the problem in polynomial-time. A number of assumptions are made in order to utilise the algorithm. Firstly, the aisles are parallel, narrow and equal in size. There is a single depot for the picker. The aisles are connected by a cross aisle at each end. The locations of the products are known in advance.

Others have considered variations of the multi-parallel-aisle systems by relaxing some assumptions. Although it is possible to construct optimal routing algorithms efficiently, simple heuristics such as the *traversal* and *return policies* are widely used in practice because they are simple to implement and the resulting routes are consistent [50]. With the traversal policies, the picker will travel across an entire aisle if that aisle contain at least one pick. The picker therefore enters the isle at the one end and leaves the aisle at the other end. For the return policy, the picker always enters and exits at the same end of the aisle, and the same aisle may be entered twice from both its ends.

2.5.3 Sequencing and routing AS/RS

The routing problem for man-on-board AS/RS is a TSP with a Chebyshev distance metric. The problem may be formulated by dividing the rack into two equal height horizontal bands; the points in the lower band are visited in the increasing x-coordinate direction, while the points in the upper band are visited in the opposite direction [50]. If the tour must visit many points, the rack may be divided into pairs of horisontal bands. A man-on-board AS/RS is an automatic storage and retrieval machine that may performs the same actions as an order picker and may sometimes operate as an industrial truck when outside an aisle. Chebyshev distance is also called *maximum value distance*. It examines the absolute magnitude of the differences between coordinates of a pair of objects [65]. The Chebyshev distance between two vectors or points p and q, with standard coordinates p_i and q_i, respectively, is

$$D_{\text{Chebyshev}}(p, q) := \max_i(|p_i - q_i|). \tag{2.1}$$

The AS/RS routing are usually used for storage racks containing multiple levels. The routing problem for unit-load AS/RS pairs a storage operation with a retrieval operation for a combined command cycle. Graves *et al.* [48] show that travel distance may be minimised when reducing unproductive travel between storage and retrieval locations. The algorithms in the literature are usually either static or dynamic. Static algorithms consider a group of storage and retrieval requests, sequence the requests in the group and execute the resulting schedule ignoring new storage and retrieval requests. The dynamic approach is used to re-sequence the storages and retrievals when new requests are presented.

2.6 Carousels

Carousel storage and retrieval systems are often used in warehouses for the storing and order picking of small, light, and highly demanded items [110]. Instead of the order picker travelling to the storage location, the storage location travels to the order picker [12]. A carousel conveyor is essentially a length of shelf that is designed into a closed loop that may be rotated clockwise or counterclockwise [42]. A carousel system may hold many different products stored in bins (locations) that rotate in a closed loop. The system is served by an operator (either human or robotic) who occupies a fixed position at the front of the carousel. To retrieve a product, the carousel system automatically rotates the bin with the requested product to the position of the operator [108].

Three types of carousel systems are used in practise: the horizontal carousel, the vertical carousel and the rotary rack [108]. The horizontal carousel consists of a number of shelves (carriers) that are rotated horizontally by a chain in a closed loop. Each shelve may contain multiple bins for

different items. Vertical carousels are similar to horizontal carousels with the exception of being rotated by a chain in a vertical closed loop. The rotary rack is an extension of the horizontal carousel in that every bin may rotate independently, reducing the waiting time of the operator (picker). Since the product rotates to the person, there is no need for an aisle by which to access product. This means that carousels may be installed side-by-side, which increases space utilisation and also provides security for the product [14]. Hassini [53] has also identified a twin-bin carousel, which is similar to horizontal carousels that allows the picker to pick from two adjacent locations during a given instance. Figure 2.8 displays examples of typical carousel systems.

(a) An example of horisontal carousels

(b) An example of a vertical carousel

Figure 2.8: *Examples of carousel systems (Source: [35]).*

Carousels have similar layouts to the picking lines used by Pep. Instead of a rotating carousel, the picker merely travels around a horizontal carousel. Since the picker may only move in a single direction, the carousel becomes a unidirectional carousel. Hassini [53] presents a general model that includes the one-dimensional carousel storage location problem as well as other well-known combinatorial optimisation problems as special cases.

Bartholdi and Platzman [14] considered the Single Order Pick Sequencing (SOPS) problem. The sequence of the orders are fixed and the time necessary to move between bins by the picker is neglected. The problem is then reduced to find the shortest Hamiltonian path on a circle. Wen and Chang [117] have considered a SOPS and incorporated the time necessary to move between bins on a carousel. Bartholdi and Platzman [14] also considered the Multiple Order Pick Sequencing (MOPS) problem. Three heuristics are presented to complete all the orders, when the orders may be arbitrarily sequenced. Van den Berg [108] presents an efficient dynamic programming algorithm for MOPS that finds the optimum sequence of the orders. Ghosh and Wells [42] present further algorithms for retrieval of goods from a carousel. Vickson and Fujimoto [110] formulated an optimal storage and retrieval system for carousels.

Figure 2.9 displays a carousel containing one order. Assume the locations that has to be visited by the order is indicated by the squares. Also assume that the picker is situated at location 1. The shortest distance travelled to complete the order is indicated by the arrows in the centre of the carousel. The carousel should move such that the picker is positioned (relative to the carousel) from location 1 to 2. The carousel should then change direction and move such that the picker is positioned (relative to the carousel) to location 8.

The operation of carousels may be compared to the *Patrolling Repairman problem*. The problem may be classified as a priority queueing problem. A single operator (picker) is assinged a group of N semiautomatic machines (locations), which fail intermittenly and are repaired by the

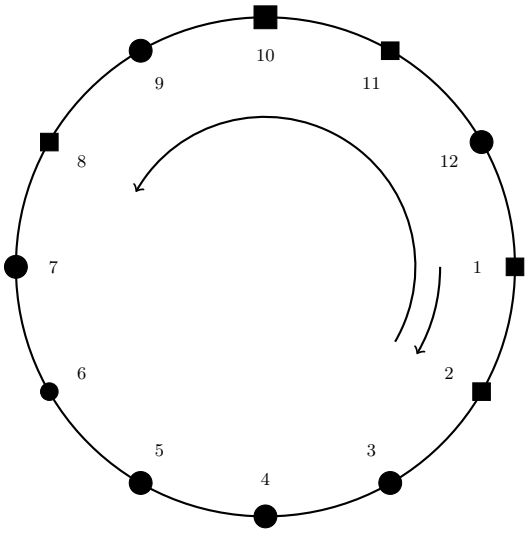


Figure 2.9: *A carousel consisting of 12 distinct locations and 1 order.*

operator (have to be picked for an order by the picker) [83]. The patrolling repairman problem differs from typical carousel systems where failures in machines may be predicted (usually by a Poisson distribution), whereas carousel systems may know which locations a picker has to visit in advance.

Koenigsberg and Mamer [70] studied the efficiency of a single carousel server serving a single carousel system. This research has lead to assessing a wide range of carousel configurations that may arise. Kim and Koenigsberg [67] studied the efficieny when a single robot (picker) serves two or more carousels simultaneously.

Recently, focus have been placed on the performance of pods[1] of carousels. Park *et al.* [87] considers the case where a pod contains two carousels. Vlasiou and Adan [112] extends the results by Park *et al.* [87] by considering more general distributions for the pick times. Meller and Klote [79] considered the efficiency when the number of pods in a carousel is equal to two or greater.

2.7 Picking area zoning

Zoning is the problem of dividing the whole picking area into a number of smaller areas (zones) and assigning order pickers to pick requested items within the zone [29]. Zoning is a strategy useful for a warehouse with a predetermined layout, storage strategy and routing policy [86, 100]. The order picking area is divided into smaller zones. A picker is assigned to a zone picking orders only from his zone. Zoning may be classified as *synchronised* zoning or *progressive* zoning [29]. In progressive zoning, an order is processed at a single zone at a time and then passed on to the next zone. Synchronised zoning is where multiple pickers work on the same customer order, each picker staying in his/her respective zone. The difficulty associated with zoning is to find the correct trade-offs between the number of zones, the size of each zone and the number of pickers assigned to each zone in such a way to balance the workload evenly over all the zones in order to minimise the throughput time of each customer order.

[1]Pods refer to a single rod that contains a series of attached carousels, each carousel usually having similar characteristics.

The *bucket brigade* system is based on a dynamic zoning system. A bucket brigade is a linear picking line in which each worker picks up a job and processes it at each station until he gets "bumped" by a downstream worker. Whenever the last worker in the line finishes a job, he initiates a reset of the production line — each worker takes over his predecessor's job and the first worker starts a new job [3]. Each picker follows a rule of "carry your work forward from station to station until someone takes over your work; then go back for more" [10]. The last order picker, after completing the customer order, will walk back upstream and take over the work of his predecessor, who will then walk back and take over the work of his predecessor and so on until the first order picker starts a new customer order at the beginning of the picking line [100]. Pickers are not restricted to any stations — each picker carries his/her work forward as far as possible, except that workers may not overtake one another. This means that a picker might catch up to his/her successor and is then blocked. Pickers must also be sequenced from slowest to fastest along the picking line. Work is paced by the fastest worker, who triggers each successive series of walk-backs — the result is a pure pull system [10].

Bucket brigades are constructed in a way such that no picker has a restricted set of pick locations, each picker simply picks as much of the customer order as possible, until another picker takes over the customer order. A distinctive feature of bucket brigades is that they are self-balancing — a balanced partition of the work will emerge spontaneously. Another advantage is that bucket brigades creates a balanced system in which the maximum possible rate of production is realised [10, 11].

Pep does not employ a zoning strategy for their picking lines. The voice-automated software system that Pep uses is unable to handle any zoning strategy. Each picker has to finish one customer order at a time, multiple pickers are unable to work on the same customer order. Another problem with zoning is that if the orders differ considerably, zoning systems and bucket brigades never reach a convergence condition. This might lead to a chaotic picking line [11]. Zoning and bucket brigades models contain weaknesses, since assumptions are made on picker velocities and teamwork between pickers [9]. Bartholdi *et al.* [11] describes that chaos has implications external to the picking line as "a chaotic assembly line will appear to start and to complete products at random, even though the assembly line is completely deterministic."

2.8 Chapter summary

The aim of this chapter has been to introduce the concept of a supply chain, together with the importance of an integrated logistics system. Order picking forms a crucial part of any DC operation and must be investigated for efficient planning and implementation throughout the supply chain. Initially the notion of a supply chain is described in §2.1. Focus is then shifted to the importance and necessity of warehousing (§2.2). The process of order picking as well as processes and systems associated with it is considered in §2.3. Various order picking systems are presented that are used in a variety of warehouses for different types of order picking configurations. In §2.4 batching of orders are considered together with measures in literature that may be used to evaluate closeness between orders. Sequencing and routing policies are considered in §2.5 for conventional single-block warehouses and multi-parallel-aisle warehouses, as well as AS/RS systems. In §2.6 the developments in carousel systems are presented. The notion of picking area zoning is discussed in §2.7.

CHAPTER 3

The Pep DCs

Contents

3.1 Organisational background 23
3.2 The Durban and Kuilsrivier DC 24
3.3 Layout of a picking line 25
3.4 Building of a picking line 26
3.5 The pickers 29
3.6 Assigning SKUs to locations 30
3.7 Control procedures 31
3.8 Rules imposed by Pep 33
3.9 Chapter summary 33

This chapter considers the various aspects of how the Pep DCs operate. Two Pep DCs owned by Pep is considered, one in Durban and another in Kuilsrivier. Operations within these DCs are explained.

3.1 Organisational background

A central team plan and manages stock levels within the DCs. Pep utilises a Supply Chain Operation Reference-model (SCOR-model) to aid in management consulting issues. Pep uses the SCOR-model to mainly measure their performance relative to competitors and to learn best practises from other companies [69].

Pep utilises a system where approximately 50% of its stock is required for replenishment considerations and the remaining 50% is for seasonal requirements [69]. Pep imposes both *push* and *pull* systems. The pull systems are frequently used in conjunction with *replenishment stock*, while the push system is used in collaboration with the *seasonal stock*. Replenishment stock refers to stock sold throughout the year, not subject to seasonal change, trend or changes in style. These items include underwear, linen and school clothing. Seasonal stock refers to products that are only sold during a specific period, after which it is usually replaced by a new fashion. Shirts, trousers and dresses form a small part of the seasonal stock.

Pep's biggest cost component is the transport cost. The average distance between the DCs and retail outlets are 1 300 km. Pep does not own any transport vehicles — transport is outsourced

to an outside company. However, it is Pep's responsibility to specify the number of trucks needed and it is Pep's responsibility to use the trucks efficiently.

Pep's stock is manufactured locally and abroad. Local production forms one third of Pep's stock and the rest is imported from the Far East. Approximately 70% of the imported stock arrives from China, where three or four shipping lines are used to move all the stock from abroad.

3.2 The Durban and Kuilsrivier DC

Operations within two Pep DCs, the Durban DC and Kuilsrivier DC, will be investigated. The Durban DC is the largest DC of Pep, utilising a floor space of more than 50 $000m^2$ [69]. The Kuilsrivier DC is considerably smaller than the Durban DC, but also plays a critical part in the distribution of goods within the supply chain employed by Pep. Figure 3.1 displays a general view of the Pep DC in Durban.

Figure 3.1: *A view of the Pep DC in Durban.*

Each of the Durban and Kuilsrivier DCs contain four main areas, namely the receiving, storage, order picking and dispatch areas. The layout for the Durban DC is displayed in Figure 3.2. The Kuilsrivier DC has a similar layout to the Durban DC.

Goods arrive in cartons at the receiving docks displayed in Figure 3.3(a), and then goes to the storage racks, displayed in Figure 3.3(b), or the full carton area. Some products are distributed as full cartons to the retail outlets, which are assigned to the full carton area. The cartons that need to be opened and repackaged for smaller shipments are sent to the storage racks. Cranes operate between the storage racks to store and collect stock as needed. These cranes have the ability to easily move between the storage racks and ascend and descend to store or collect pallets of products.

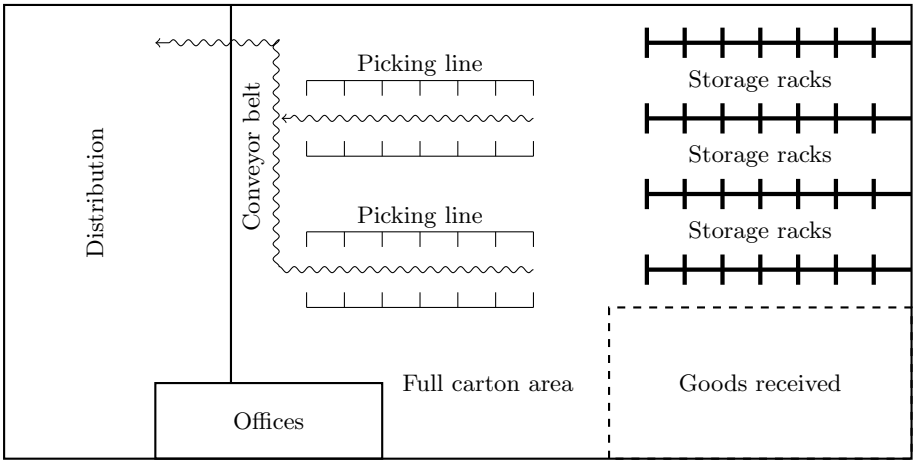


Figure 3.2: *The layout of the Pep DC in Durban.*

(a) A receiving dock at the goods received area.

(b) Storage racks used to store products for later use.

Figure 3.3: *The receiving process at the Durban DC.*

Since the DCs receive and send many products in various carton sizes, pallets are used to ease the movements of goods within the DC. Pallets are portable platforms used for storing or moving cargo or freight. The general pallet is 1 000mm wide, 1 200mm in breadth and 140mm in height. Figure 3.4(a) displays how cartons may be stacked on a pallet, in order to move products in bulk within the DC. Figure 3.4(b) displays how a forklift may be used to move pallets.

Since Pep predominantly stocks clothing, the weight of products is not the main consideration. Pallets are able to store many cartons stacked on top of one another. Pallets may also be stored on top of one another, if the cartons are strong enough to support the weight.

3.3 Layout of a picking line

In the Durban DC all order picking is done on one level. All the picking lines are constructed next to one another. In contrast to this, the Kuilsrivier DC employs a layout where picking lines are constructed on different levels on top of each other.

Pep employs isle picking lines. Pickers walk in a clockwise direction around the picking line. In the Durban DC, there are 112 locations on a picking line — 56 locations on each side of the conveyor belt. The picking lines at the Kuilsrivier DC contain 72 locations on each side of the

(a) Cartons of product stacked on pallets.

(b) A forklift moving pallets with the DC.

Figure 3.4: *The movement of goods within the Durban DC.*

conveyor belt — a total of 144 locations per picking line. Each location contains a different set of identical products. Figure 3.5 displays the configurations of a picking line used in the Pep DC in Durban. Figure 3.5(a) depicts the scenario where the whole picking line is used (*i.e.* all 112 locations are utilised for order picking). This scenario is seldomly used in the Durban DC. Each picking line in the Durban DC has a gate in the middle. The gate may be utilised when a picker does not need to make a full cycle around the picking line. Two smaller picking lines may be constructed using the gate. Each picking line then has only 56 locations allocated to it. Figure 3.5(b) depicts a scenario where one picking line is split in two smaller picking lines. This layout is used frequently at the Durban DC. The last scenario is where products on one side of the conveyor belt is mirrored onto the other side of the conveyor belt displayed in Figure 3.5(c). Figure 3.6 displays how a picking line is constructed in the Kuilsrivier DC. The Kuilsrivier DC uses this single layout, since the picking lines does not have gates in the middle.

The products are fixed in their respective locations until the pickers in the picking line have completed all the orders requested by the retail outlets. The length of a picking line in the Durban DC is approximately 75 meters and pickers travel approximately 150 meters to complete a cycle. The length of a picking line in the Kuilsrivier DC is approximately 95 meters and pickers travel approximately 190 meters to complete a cycle. The picking lines in the Kuilsrivier DC is longer than the picking lines in the Durban DC, since the Kuilsrivier DC have more locations in a picking line. The area between the locations and the conveyor belt is large enough to let pickers overtake one another when necessary [81].

3.4 Building of a picking line

The Durban DC is made up of 6 parallel picking lines, 5 picking lines as the standard single isle picking lines and a 6^{th} picking line using flow racks. This picking line is used to pick items that are requested by the majority of retail outlets — these products are *high intensity* products. The picking lines are located alongside one another on the ground level of the DC. The building process of picking lines starts with the forklifts collecting the necessary pallets, stacked with cartons of identical product, from storage and placing it in specific locations. Each location may hold up to five pallets at any time. The cartons containing products remain stacked on the pallet during the picking process. A location is the same concept as a bin, which is referred to in literature by Koster *et al.* [28] or product locations as used in Bartholdi & Hackman [12].

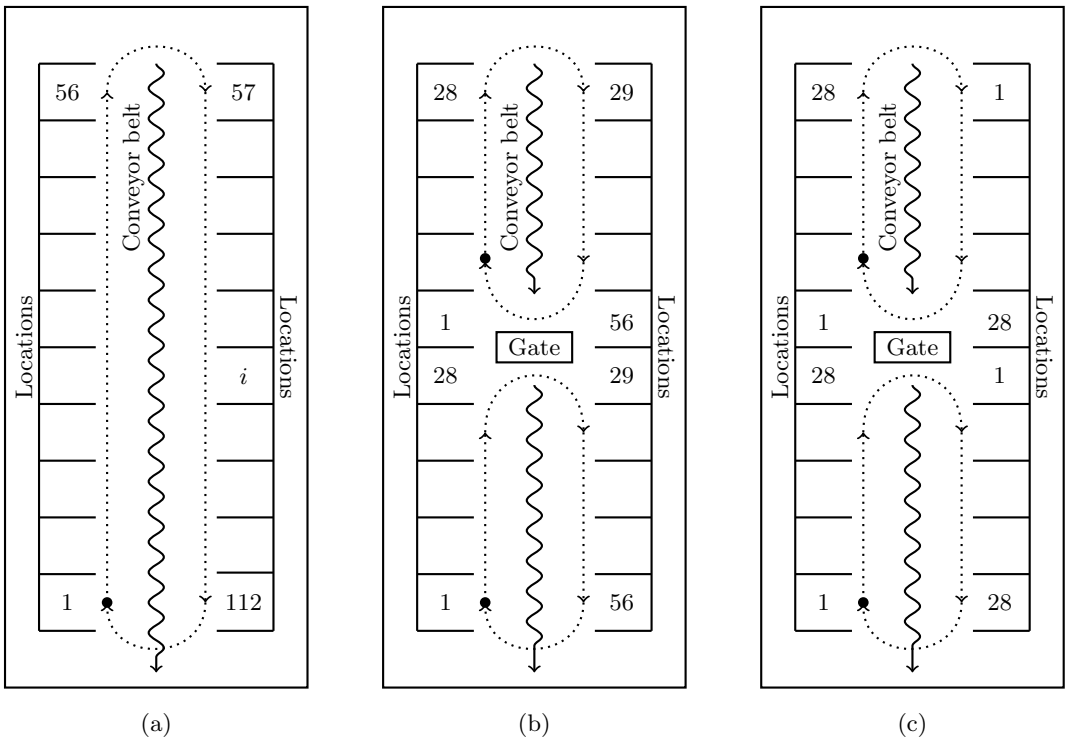


Figure 3.5: *Examples of possible picking line configurations in the Pep DC in Durban. The arrows indicate the direction that the pickers may travel. In (b) and (c) the pickers use the gate in the centre of the conveyor belt.*

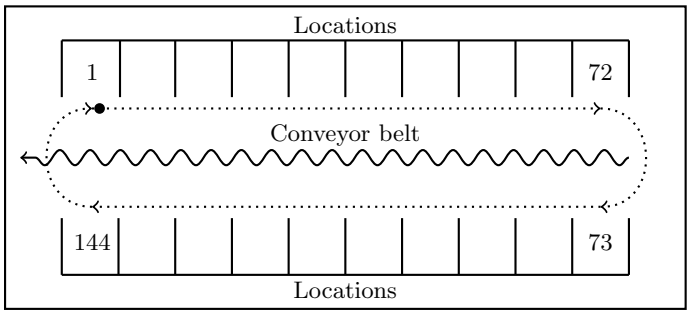


Figure 3.6: *An example of the picking line configuration in the Pep DC in Kuilsrivier. The arrows indicate the allowable direction that a picker may travel.*

The Kuilsrivier DC is made up of 8 picking lines. In contrast to the Durban DC, each location in a picking line may only contain 2 pallets of products. The picking lines in the Kuilsrivier DC are situated on different levels. There are 10 levels in the DC. Picking lines are situated on the second, third, fourth and fifth level, each level containing two parallel picking lines.

The Pep DC builds picking lines according to lists of distribution requests that are received from central office [69]. A list contains a product with the quantities required by the various retail outlets. The retail outlets become the "customers" that are serviced by the DC — customer orders refers to the requests of the retail outlets. Picking lines must then be built in such a way that all the products in their correct quantities may be collected for distribution to the retailers. The building process entails that each location or set of locations in a picking line receives a unique product from storage.

The DCs use an order picking system which is based on the concept of a *wave*. A wave may

be described as the set of products in conjunction with the set of retail outlets requiring at least one of the products. All the orders for that wave are picked as a single operation. All the products in a wave are therefore completely picked for all retail outlets during that wave.

Figure 3.7(a) displays an empty picking line with no pallets. Figure 3.7(b) displays a picking line being built. Note that the location at the back contains five pallets of product, while the front two contain no pallets.

(a) An empty picking line currently not in operation.

(b) A picking line being built with one location containing pallets of product.

Figure 3.7: *Building picking lines at the Durban DC.*

Figure 3.8 displays how the conveyor belt moves the cartons to the dispatch area, where the cartons are sealed and labelled appropriately. Both picking lines are in full operation. To the right of the conveyor belt a picker is busy completing an order.

(a) Conveyor of picking line moving cartons to the dispatch area.

(b) A picking line in full operation with pickers on either side of the conveyor belt.

Figure 3.8: *Operational picking lines at the Durban DC.*

When only half a picking line is used, the gate has to be opened to enable pickers to move around the smaller picking line. Figure 3.9(a) displays an open gate used to construct two smaller picking lines within a picking line, while Figure 3.9(b) displays a closed gate.

All products in storage are classified into stock-keeping-units (SKUs). SKUs are unique components held in stock [47], for example, two different sizes of the same shirt (or product) will be two different SKUs. The picking line manager has the duty of placing SKUs in locations [32, 80, 81]. The high intensity SKUs have to be separated by *lower intensity* SKUs in order to avoid congestion of the pickers within a picking line. Similarly, SKUs that are difficult to pick have to be

(a) An open gate used to move to the opposite side of the conveyor belt.

(b) A closed gate with cartons moving along the conveyor belt.

Figure 3.9: *The gate at the centre of a picking line in the Durban DC.*

alternated with those that are easier to pick. Once SKUs have been allocated to locations, the software system stores this configuration in memory.

3.5 The pickers

A picking line usually contains eight pickers but this number may vary in accordance with the number of pickers available or the layout or contents of the picking line. Pickers pick orders in *serial, i.e.* each picker picks one order at any time, as opposed to *parallel* order picking where multiple pickers pick one order simultaneously [12]. Each picker has a trolley which contains a carton in which SKUs are placed as the picker moves along the picking line collecting SKUs for an order. A picker only leaves his/her trolley when picking a SKU, collecting a new carton or when placing a carton on the conveyor belt. Pickers are assigned to a picking line and do not leave their picking line until the pickers have completed all the orders assigned to that picking line. The SKUs needed by a retail outlet may span over more than one picking line. This simply means that the required SKUs for a retail outlet may be spread over more than one picking line.

All cartons are sealed before leaving a picking line. Whenever a carton is full, it is placed on the conveyor belt, sealed, and moved to the distribution area. An order may require more than one carton. If a picker is unable to complete his/her order by only using a single carton, a full carton may be placed onto the conveyor belt, collecting a new carton for the rest of the picks. This may be done as many times as needed until the customer order is completed.

A voice-automated software system is used to communicate with the pickers. Each picker has a headset with which the computer system instructs the picker and the picker may also communicate with the system if needed. The system directs the picker to locations from which SKUs are obtained and placed into cartons. Once a picker has completed picking from a particular location, the picker notifies the system with his/her voice via the headset. The system then directs the picker to the next location. Once a picker has finished an order, the carton is placed on the conveyor belt. The picker now collects a new carton and receives a new set of locations to visit for a different retail outlet. Figure 3.10 displays pickers collecting SKUs in a picking line.

SKUs in the DC are classified into one of three categories, A-, B- and C-type of picks. An A-type pick is a SKU picked with a minimal effort. A-type picks are usually picked using a


Figure 3.10: *Pickers operating in a picking line.*

single hand. *B*-type picks are picked with medium effort. A picker has to use both his/her hands to pick a *B*-type SKU. *C*-type SKUs requires a great deal of effort to pick. The picker has to use his/her entire body in an effort to collect the SKU. The picker may crouch, use both hands to grip the SKU and use his/her body to lift the SKU. *C*-type SKUs are rarely built into picking lines, instead they are often shipped as full carton shipments to the retail outlets [80]. Typically, a picking line consists only of SKUs requiring *A*- and *B*-type picks.

3.6 Assigning SKUs to locations

Each location may only contain a set of identical SKUs. The software is unable to recognise more than one location containing the same SKU on a picking line. The locations containing the same SKUs must be in consecutive locations. The systems operator will then locate the SKU in one location and leave consecutive locations empty on the system. The system records these locations as containing no SKUs, but the forklifts do place additional SKUs for order picking in them. The pickers are informed that they may collect SKUs from the "empty" locations if necessary.

High frequency SKUs usually need more than one location on a picking line, since these pallets may deplete quickly, which makes the process of replenishment difficult. Pep management also prefer building picking lines in such a manner that no SKUs have to be replenished during a wave of order picking [80]. This enables constant, uninterrupted flow of work during order picking. If a location has to be replenished, the entire picking line comes to a stand-still. If, however, the SKUs in a location have to be replenished, the picking line manager is required to inform a forklift to retrieve additional pallets of SKUs from storage. The driver of the forklift will place this request at the bottom of his/her list of priorities, possibly making replenishment a tedious process.

If more than one location is allocated to a SKU, the SKUs have to be placed in sequence, *i.e.* only consecutive locations may share the identical SKUs. For example, if product x is placed in locations i, $i+1$ and $i+2$, the system will instruct the picker to collect product x from location i every time product x has to be picked, ignoring location $i+1$ and $i+2$. If location i contains no more SKUs, the picker will know that he/she should move to the next location ($i+1$ or $i+2$) to collect the product from the "reserve" locations[1]. The locations i, $i+1$ and $i+2$ will form a single SKU location, since they contain identical SKUs. Since each location in the Kuilsrivier

[1]The pickers working in the picking will be informed of "reserve" locations before order picking commences.

DC may only have two pallets at a time, most SKUs have more than one location allocated to it. High frequency SKUs may have in excess of 6 locations allocated to it. This is done to reduce replenishment during order picking. The DC in Durban rarely allocates more than two locations for a SKU, since each location can hold 5 pallets. The picking lines in the Durban DC is built in such a way that pallets never have to be restocked during order picking.

A *picking line manager* has the duty of addressing all elements of construction of picking lines. This staff member uses his/her knowledge from years of experience. No mathematical optimisation is done when constructing the picking lines, except lessons learnt from past experiences.

The planning of a picking line is completed in approximately 30 minutes [81]. After planning, the picking line is built for picking. A picking line containing less than 30 000 SKUs may usually be completed within one day. A picking line usually contains no more than 60 000 SKUs and such picking lines usually does not exceed two days of picking [80].

All products owned by Pep are divided in a hierarchical order. At the first level of the hierarchy stock is received as a *business unit*. The business unit is divided into *groups* after which it is sorted into *divisions*, specifically to suit the operations within a DC [75]. Products are then divided into *departments* indicating the gender or age group of the products (*e.g.* men's or women's clothing). Departments are divided into *classes* relating different items of clothing in each department (*e.g.* shirts or trousers). Classes may be divided into *subclasses* (*e.g.* short sleeved shirts and long sleeved shirts). Each subclass contains different *styles* between products (*e.g.* "v-neck shirts" and "polo shirts"). Styles are grouped according to different sizes.

3.7 Control procedures

All the operations in the Pep DCs are captured and controlled by a computerised system. The system contains all the relevant information needed to operate a DC. Figure 3.11 displays the data captured from an active picking line. An entry is added to the system once a picker has successfully collected a number of identical SKUs for a specific order. All the data is retrieved from picking line 9997. The first recorded pick was by Nomusa, collecting SKU 877835, at location C101, for branch (order) 549. She was required to collect 2 packs of this SKU, one pack containing 6 SKUs. She collected these SKUs during the early morning of the 21st of May 2010, at 03:33:13.

Pickers are reawarded for how quick they pick, a picker with more orders completed receives a bonus. The pickers in a picking line are also rewarded for how efficient a picking line operates. These incentives promote fast and thorough work done by the pickers. It also motivates a sense of team participation among pickers working in the same picking line.

Once a carton is initially received at the *Goods received* area, it receives a sticker to describe it. When a carton is sealed after order picking, it receives a new sticker indicating the destination of the carton. Figure 3.12 (a) displays a sticker that a carton receives when the carton is received initially. The sticker indicates the location where the carton should be stored, the order number which acts as a counter for the carton, the SKU is the identification number of the contents within the carton, the quantity of SKUs within the carton, the number of cartons containing the same SKUs and the packsize of the SKU. Figure 3.12 (b) displays the sticker used to indicate the destination of the carton. The sticker contains the invoice number, the hub of the destination, the route this carton should take and where to deliver the cartons.

After the cartons are packed, sealed and moved to the distribution area, a small fraction of

PEP

Report name: rfpickdetail.rdf
Report run on: 24-May-2010 17:15:49
Attention: TJAARTL
From: 19-MAY-10 To: 24-MAY-10

Total picks for a picker for a given period

Total Branches picked for this period: 0

Picked Date 21-May-10
Line No 9997

Picker	Branch	Location	Sku No	Req Qty	Pick Qty	Status	Date&Time Picked	PSize	Carton
NOMUSA	549	C101	877835	12	12	Uploaded	21-MAY-10 03:33:03	6	588178682
FISIWE	552	C101	877835	6	6	Uploaded	21-MAY-10 03:33:06	6	588215390
NOMUSA	549	C102	832077	1	1	Uploaded	21-MAY-10 03:33:13	1	588178682
NOMUSA	549	C103	864812	6	6	Uploaded	21-MAY-10 03:33:14	6	588178682
BUSISIWE	402	C101	877835	24	24	Uploaded	21-MAY-10 03:33:21	6	588120372
NOMUSA	549	C106	832078	2	2	Uploaded	21-MAY-10 03:33:21	1	588178682
GRACE	309	C101	877835	6	6	Uploaded	21-MAY-10 03:33:23	6	588165690
FISIWE	552	C103	864812	18	18	Uploaded	21-MAY-10 03:33:27	6	588215390
BUSISIWE	402	C103	864812	6	6	Uploaded	21-MAY-10 03:33:32	6	588120372
GRACE	309	C102	832077	2	2	Uploaded	21-MAY-10 03:33:32	1	588165690
NOMUSA	549	C112	864811	6	6	Uploaded	21-MAY-10 03:33:34	6	588178682
GRACE	309	C103	864812	18	18	Uploaded	21-MAY-10 03:33:40	6	588165690
FISIWE	552	C110	613402	1	1	Uploaded	21-MAY-10 03:33:45	1	588215390
FISIWE	552	C112	864811	6	6	Uploaded	21-MAY-10 03:33:50	6	588215390

Figure 3.11: *Information contained in the system that is used to track operations within the DC.*

orders are selected for a control check. Each of these cartons are opened, each item is removed and counted to check that the picker made no mistakes. If the correct quantities of the correct SKUs are found in the carton, the SKUs are repacked into a new carton, sealed and sent to the distribution area. If there is a problem with the order, the correct SKUs are retrieved, repacked into a new carton, sealed and sent to the distribution area. Pickers are rewarded with incentive bonuses if the correct quantities of the correct SKUs are packed into these cartons [80, 81].

The type of SKU is not specified on the system. It is unknown whether SKU 877835 is an *A*-, *B*- or *C*-type product. Various DCs may therefore categorise certain SKUs differently.

(a) Sticker indicating the contents of a carton

(b) Sticker indicating the destination of a carton

Figure 3.12: *Stickers used to identify the location of cartons, its destination and its contents.*

3.8 Rules imposed by Pep

The Pep DC builds picking lines according to the list of distributions (DBNs) that is received from central office. A list on the SKU level contains all the products that each branch requires, along with the respective quantities thereof.

Pep operates on a first-in-first-out (FIFO) system. DBNs that are received today have to be completed before tomorrow's DBNs may be started. The only exception to this rule, is when a very urgent request is sent. The floor manager will be informed of the urgent request, and the floor manager will shift that specific DBN to be scheduled in a picking line as soon as possible.

A DBN must be dispatched within seven days after it has been issued. This rule is imposed as a key performance indicator (KPI). Pep believes in a system where stock is continuously moved through the supply chain to the retail outlets.

All the orders requested by the Botswana retail outlets are picked first. Each item of product that has to be shipped to Botswana has to receive a new price tag at a seperate work station. Employees have to open each carton containing requested products, remove the old price tag, add a new price tag and repackage the products.

Apart form the restrictions mentioned in §3.4 on the placing of SKUs in locations, the various DCs impose rules to reduce errors during the picking process. The Kuilsrivier DC imposes a rule that similar SKUs of the same colour may not be placed next to one another (*e.g.* "boys maroon school jerseys" may not be placed next to "girls maroon school jerseys"). The Durban DC does not impose this rule used in the Kuilsrivier DC. However, the Durban DC imposes that similar products of different sizes are not be allowed to be placed next to one another (*e.g.* "boys maroon school jerseys (medium)" may not be placed in a successive location to "boys maroon school jerseys (large)"). In contrast to this rule, the Kuilsrivier DC insists that similar products of different sizes are placed next to one another. Pickers in the Kuilsrivier DC are informed that the sizes of similar products are placed next to one another.

3.9 Chapter summary

This chapter provides a general framework for the operations present in the Pep DCs. The layout of the Pep DCs are discussed in §3.3, while the picking line configurations and operations are considered in §3.4 and §3.5. The decisions regarding the construction of a picking line is also discussed, and finally some control procedures and rules are presented in §3.7 and §3.8.

CHAPTER 4

Order sequencing problem

Contents

4.1 Introduction . . . 36
4.2 Exact formulations . . . 36
4.2.1 An exact formulation . . . *37*
4.3 Notation for the OSP . . . 38
4.3.1 A branch and bound approach . . . *42*
4.3.2 A lower bound . . . *43*
4.4 Tour construction heuristics . . . 45
4.4.1 Next shortest order . . . *46*
4.4.2 Next shortest order relative to minimum span . . . *47*
4.4.3 Next shortest order relative to its number of picks . . . *47*
4.4.4 Results for the tour construction heuristics . . . *47*
4.5 Scope and ranking algorithms . . . 48
4.5.1 Ranked preference span algorithm . . . *49*
4.5.2 Minimum span preserving algorithm . . . *50*
4.5.3 Results of the scope and ranking algorithms . . . *52*
4.6 Awarding spans according to preference ratios . . . 52
4.7 Generalised assignment approach . . . 58
4.7.1 Heuristic solution methods to solve the generalised assignment problem *61*
4.7.2 Implementation of the modified assigment problem . . . *68*
4.7.3 Results of the generalised assignment problem . . . *70*
4.8 Metaheuristic approaches to order sequencing . . . 71
4.8.1 Data structure and improvement of a solution . . . *72*
4.8.2 Greedy starting heuristic . . . *75*
4.8.3 Tabu search . . . *76*
4.8.4 Simulated annealing . . . *79*
4.8.5 Genetic algorithm . . . *84*
4.8.6 Extremal Optimisation . . . *95*
4.9 Results summary . . . 97
4.10 Chapter summary . . . 104

This chapter will provide context for the OSP and the implications of solving this problem. Various solution techniques are considered to solve the OSP in reasonable time. Each section will address the OSP with a different approach. The chapter concludes with computational results obtained from the performance of the various algorithms used.

4.1 Introduction

The OSP for a picking line may be modelled in a variety of ways. It may be considered as a single picker working on a picking line. The sequence of orders is then scheduled for a single picker. Since it might be more efficient to use more than one picker in a picking line, this schedule may then be partitioned into smaller sections or schedules for each picker in the picking line. A dynamic approach may be used to assign orders to a picker whenever a picker has completed an order. It is assumed that when orders are scheduled in a picking line, all the SKUs have already been assigned to locations.

The time needed to pick a SKU from a location is considered as constant. The time needed to travel between orders and between locations within orders are considered as variables. Unnecessary travelling time may be eliminated by sequencing orders to minimise the total distance travelled.

The following assumptions are made to model the OSP. These assumptions were made in cooperation with the management of Pep.

1. A picker must complete an order before starting a new order.
2. The time to pick a SKU is constant over all orders.
3. Pickers move at a constant speed.
4. An order may commence at any location.
5. The time required to switch between orders is negligible.
6. A picker may not collect the first SKU of a new order at the same location as the last location of a previous order.

4.2 Exact formulations

The time that pickers travels between locations is directly proportional to the distance travelled by the picker. Thus the objective is to sequence a set of orders in such a way as to minimise the number of (pick) cycles required to complete all the orders.

Due to the size of the OSP, exact formulations cannot be solved in a realistic timeframe. If a picking line contains n orders and each order k contains $|\mathcal{O}_k|$ locations, a total of $n! \prod_{k=1}^{n} |\mathcal{O}_k|$ possible combinations of sequences exists to pick all orders in the worst case. This solution space may be reduced to $n!$ possible combinations. All test cases were solved using an Intel(R) Core(TM)2 Duo 3 GHz with 3.7 GB ram using LINUX UBUNTU. The programming was done in either JAVA [101] or LINGO 11 [73].

In the following subsection an exact formulation for solving the OSP is considered. A branch and bound approach is then presented in §4.3.1. The maximal cut formulation is presented in §4.3.2 which is a strong lower bound for the OSP. The solutions of the maximal cut formulation act as a benchmark for all heuristic approaches presented here to solving the OSP.

4.2.1 An exact formulation

Each order may begin at any location even if the order may not necessarily require the SKU at that particular starting location. The number of locations passed to complete any single order is therefore determined by the starting position assigned to it. Let the duple (i, k) represent order k starting at location i, which may be considered as a solution of picking order k. Let $\mathcal{N}$ be a set of all duples (i, k). Let $\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_n$ be a proper partition of the set $\mathcal{N}$, where $\mathcal{I}_k = \{(1, k), (2, k), \ldots, (m, k)\}$, if m locations are present on a single picking line[1]. The set $\mathcal{N}$ may be interpreted as the vertices on a digraph with edges representing the connection between orders. The problem may be viewed as a variant of the generalized travelling salesman problem (GTSP). The objective of the GTSP is to find a cycle of edges that visits exactly one vertex, or duple, in each set $\mathcal{I}_k$.

The OSP may thus be interpreted as an equality generalised travelling salesman problem (E-GTSP), where only one vertex must be visited in each set $\mathcal{I}_k$. One vertex in each set $\mathcal{I}_k$ has to be connected to one vertex in a different set, in an attempt to minimise the total arc lengths of the selected edges [17, 39, 51, 52]. The E-GTSP is an $\mathcal{NP}$-hard problem [51, 52].

Each edge of the E-GTSP used to solve the OSP, represents the distance of starting an order at a particular location contained in that order, picking all the locations within that order and then travelling to the starting position of another order. The goal is to minimise the distance travelled by a picker.

To formulate the OSP input data is required. Let

$d_{k\ell}^{ij}$ be the distance when order k is picked from location i, completed and then travels to the starting location of order ℓ at location j,
n be the number of orders,
m be the number of locations,
M be a large constant in the model.

Two sets of variables are required in modelling this problem. Let

$x_{k\ell}^{ij}$ be equal to 1 if order k is picked from location i, completed and then travels to the starting location of order ℓ at location j, and 0 otherwise,
$d_{k\ell}$ be equal to the distance travelled when order ℓ is to be completed after order k was picked, and 0 otherwise.

[1] The number of locations are equal to the number of distinct products on a single picking line.

The objective is to

$$\text{minimise } \frac{1}{m}\sum_{k=0}^{n}\sum_{\ell=0}^{n} d_{k\ell} \tag{4.1}$$

$$\text{subject to } \sum_{i=0}^{m}\sum_{j=0}^{m}\sum_{\ell=0}^{n} x_{k\ell}^{ij} = 1 \qquad k = 0,1,\ldots,n, \tag{4.2}$$

$$\sum_{i=0}^{m}\sum_{j=0}^{m}\sum_{k=0}^{n} x_{k\ell}^{ij} = 1 \qquad \ell = 0,1,\ldots,n, \tag{4.3}$$

$$x_{0\ell}^{0j} = 1 \qquad \begin{array}{l}\ell = 0,1,\ldots,n,\\ j = 1,2,\ldots,m,\end{array} \tag{4.4}$$

$$\sum_{i=0}^{m}\sum_{j=0}^{m} d_{k\ell}^{ij} x_{k\ell}^{ij} = d_{k\ell} \qquad \begin{array}{l}k = 0,1,\ldots,n,\\ \ell = 0,1,\ldots,n,\end{array} \tag{4.5}$$

$$u_k - u_\ell + n\sum_{i=0}^{m}\sum_{j=0}^{m} x_{k\ell}^{ij} \leq n-1 \qquad \begin{array}{l}k \neq \ell,\\ k = 0,1,\ldots,n\\ \ell = 0,1,\ldots,n,\end{array} \tag{4.6}$$

$$x_{k\ell}^{ij} \in \{0,1\} \qquad \begin{array}{l}k = 0,1,\ldots,n,\\ \ell = 0,1,\ldots,n,\\ i = 1,2,\ldots,m,\\ j = 1,2,\ldots,m,\end{array} \tag{4.7}$$

$$u_k \geq 0 \qquad k = 0,1,\ldots,n. \tag{4.8}$$

The objective function (4.1) minimises the total number of cycles travelled to complete all the orders. Constraint set (4.2) ensures that each order has a starting location and constraint set (4.3) ensures that each order has an ending location. Constraint set (4.4) ensures that the picker initially starts order picking at a location 0 which acts as a depot. Order 0 only has to visit the depot. Constraint set (4.5) calculates the distance travelled between orders k and ℓ, if order picking for order ℓ starts once order picking is completed for order k. Constraint set (4.6) ensures that no subtours are formed. This problem contains $n(nm^2+m+1)$ variables of which n^2m^2 are binary variables and $n(n+m+2)$ constraints. For a typical real life instance $n \approx 1\,200$ and $m \approx 56$, yielding a number of variables in excess of $4,5 \times 10^9$ and a number of constraints in excess of $2,9 \times 10^6$.

This formulation is computationally too expensive to solve. Even the input data becomes too large to handle, when implementation is done using LINGO 11 [73] for small sized problems faced by Pep.

4.3 Notation for the OSP

This section provides concise notation to represent the distance travelled by a picker on a picking line. A number of definitions have to be made to simplify the description of determining desirable starting locations for an order.

Definition 1 *The* **span** *of an order is the smallest set of locations passed to complete the order given a starting location.*

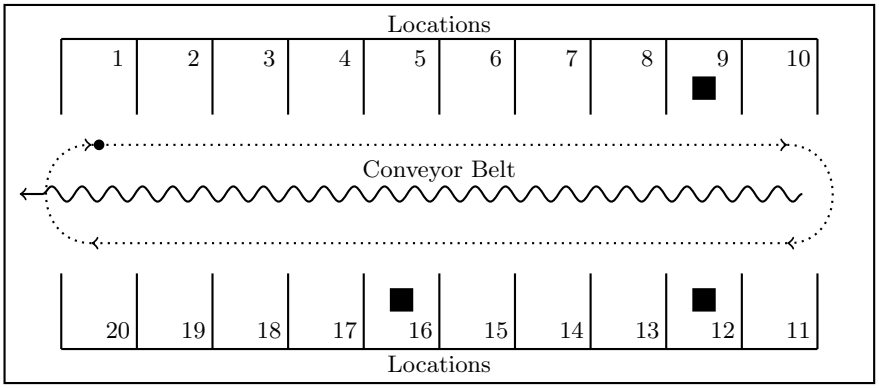


Figure 4.1: *A schematic representation of the layout of a picking line containing* 20 *locations.*

A span for an order k starting at location i may be represented by $S_k^i = \langle i, e_k^i \rangle$, where i is the starting location and e_k^i is the closest ending position of order k. Following Definition 1, any starting location for an order has an unique span associated with it, since an order must be completed once it is started.

Definition 2 *The* **size of a span** *is the number of locations that have to be traversed to complete the order if a starting position is specified.*

Each order may be assigned one starting point from all the possible locations within an order. The size of a span S_k^i for an order k may be represented by

$$|S_k^i| = |\langle i, e_i^k \rangle| = \begin{cases} e_i^k - i & \text{if } i < e_i^k \\ m + e_i^k - i & \text{if } i \geq e_i^k, \end{cases}$$

where m is the total number of locations, i is the starting location of the span and e_k^i is the ending location of the span.

Consider an example of a picking line in Figure 4.1 containing 20 locations. Let $\mathcal{O}_k$ be the set of different locations that has to be visisted to complete order k. Assume order k requires SKUs from locations 9, 12 and 16 (indicated by the squares in Figure 4.1). Thus $\mathcal{O}_k = \{9, 12, 16\}$. If a picker, currently at location 6, is assigned to pick order k a distance of $|S_k^6| = |\langle 6, 16 \rangle| = 10$ locations are travelled. The picker may now start a new order at location 17.

If, however, a picker currently at location 13, is assigned to pick order k a distance of $|S_k^{13}| = |\langle 13, 12 \rangle| = 19$ locations are travelled.

Definition 3 *The* **minimum span** *of an order is a span of smallest size for an order.*

Let $|S_k^{\min}|$ denote the length of the minimum span of order k. From the example in Figure 4.1, $|S_k^{\min}| = |\langle 9, 16 \rangle| = 7$.

Real life instances of the exact formulations cannot be solved using LINGO 11, due to memory issues with the large input data sets, as well as the number of decision variables and constraints. Thus methods of finding lower bounds are considered.

Definition 4 *The* **cut** *of a location is the number of spans passing that location.*

The cut at each location represents a lower bound for the number of cycles needed to pick a set of spans since it represents the minimum number of times a location must be passed to complete the picking of all the spans considered.

Definition 5 *The* **maximal cut** *is the number of spans passing the location containing the largest cut.*

Since the number of cycles travelled in a picking line to complete all the orders is dependant on the number of cuts at each location, minimising the cuts of each location, by assigning a starting location (a span) to each order, may reduce the number of cycles travelled to complete a set of orders. By reducing the maximal cut C, the possible number of cycles needed to complete orders in a picking line may be reduced.

The idea of a cut evolved from the realisation that in some picking line layouts, it is not beneficial to pick orders on their minimum spans. Figure 4.2 represents a picking line, displayed in the form of a carousel, consisting of 12 distinct locations, each location containing a unique product, and 5 orders. The first order is represented by three squares, $\mathcal{O}_1 = \{1, 2, 8\}$. The second order is represented by circles, $\mathcal{O}_2 = \{3, 4, 9\}$. The third, fourth and fifth orders are represented by pentagons, diamonds and triangles, respectively. Every order contains a unique minimum span. The length of the minimum span of the first order is $|S_1^{\min}| = |\langle 8, 2\rangle| = 6$. Similarly, the length minimum span of the second order is $|S_2^{\min}| = |\langle 3, 9\rangle| = 6$. Figure 4.2 represents an example where orders should not be picked on their minimum spans. Each location i is assigned a label $\langle \bar{s}, s\rangle_i$ referring to the optimal number of cycles travelled when orders are not picked on their minimum spans ($\bar{s}$) and the number of cycles necessary when orders are picked on their minimum spans (s) when order picking commences at location i. All $\langle \bar{s}, s\rangle_i$ values in Figure 4.2 are tested by means of a brute force manner, where all possible sequences are tested for each case. This example illustrates that unidirectional carousels (like the picking lines considered here) are more complex than bi-directional carousels found in literature. For bi-directional carousels it is always optimal to pick all orders on their minimum spans [14].

Figure 4.3 displays the distance and cycles travelled by a picker when orders do not have to be picked on their minimum spans. Assume the picker starts order picking at location 1. The picker may start order picking for the first order on S_1^1. The second order is then picked on S_2^9. The third order is then picked on S_3^5. The fourth order is then completed followed by the last remaining order, ending at location 12. Three cycles were travelled in order to complete all the orders. Notice that each location has a cut of three, *i.e.* each location is traversed exactly three times. This result remains the same when starting at any other location, other than location 1.

Figure 4.4 displays the distance and cycles travelled by a picker when orders have to be picked on their minimum spans. The arrows in the centre of the carousel indicate the minimum spans of all the orders. Assume the picker starts order picking at location 1. The picker may start order picking for the second order at location 3 and end at location 9. The third order is then started at location 5 and ends at location 10. The fourth order is then commenced at location 6 and ends at location 11. The last remaining order is then completed followed by the first order, ending at location 1. Five cycles was travelled in order to complete all the orders. Each location has a cut of five. Location 1 is traversed exactly five times, while the remaining locations are traversed four times each. This experiment remains the same when starting at any other location other than location 1.

It is evident form the examples presented that the starting position of an order may reduce the number of cycles travelled by a picker. By assigning starting positions to orders it is possible to

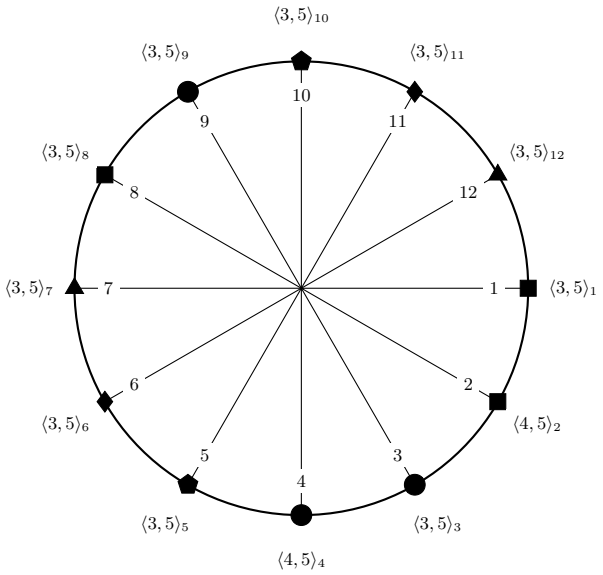


Figure 4.2: *A carousel illustrating a picking line consisting of 12 distinct locations and 5 orders, where each order contains a unique set of shapes, namely, squares, circles, pentagons, diamonds and triangles. Each location is assigned a label $\langle \bar{s}, s\rangle_i$ referring to the optimal number of cycles travelled when orders are not picked on their minimum spans ($\bar{s}$) and the number of cycles necessary when orders are picked on their minimum spans (s), when order picking commences at location i.*

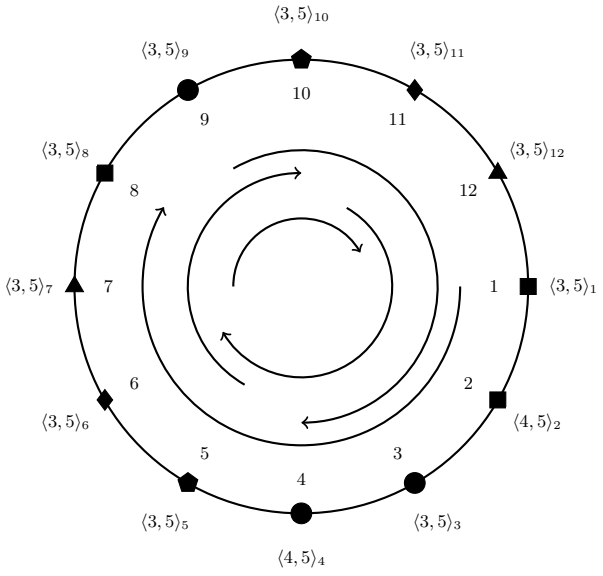


Figure 4.3: *A carousel illustrating a picking line consisting of 12 distinct locations and 5 orders, where each order contains a unique set of shapes, namely, squares, circles, pentagons, diamonds and triangles. Each location is assigned a label $\langle \bar{s}, s\rangle_i$ referring to the optimal number of cycles travelled when orders are not picked on their minimum spans ($\bar{s}$) and the number of cycles necessary when orders are travelled on their minimum spans (s) when order picking commences at location i. A sequence in which orders are picked in the least number of cycles is presented by the arrows, when order picking starts at location 1.*

reduce the maximum cut of a picking line reducing the bound on the number of cycles travelled by a picker in a picking line.

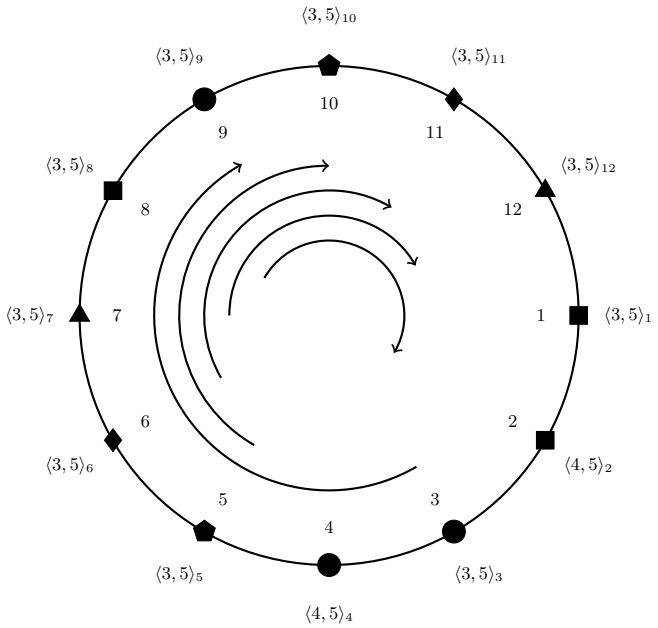


Figure 4.4: *A carousel illustrating a picking line consisting of 12 distinct locations and 5 orders, where each order contains a unique set of shapes, namely, squares, circles, pentagons, diamonds and triangles. Each location is assigned a label $\langle \bar{s}, s \rangle_i$ referring to the optimal number of cycles travelled when orders are not picked on their minimum spans ($\bar{s}$) and the number of cycles necessary when orders are travelled on their minimum spans (s) when order picking commences at location i. A sequence in which orders are picked when each order may only be picked on its minimum span, is presented by the arrows, when order picking starts at location 1.*

4.3.1 A branch and bound approach

The branch and bound method may be described as a set of enumerative methods applied to solving discrete optimisation problems [64]. The original problem, or *root problem*, is bounded from below and above. If the bounds are equal the optimal solution has been found. If this is not the case the feasible region is to be explored, by dividing the feasible region into subregions. The subregions amount to feasible regions for subproblems, which become children of the root problem in a search tree.

Before the branch and bound algorithm presented here is explained, a heuristic procedure is used to get an initial upper bound to the problem. The sequence of orders presented by the heuristic is presented by the vector, $\boldsymbol{b}_s$, which is the best current solution. The total number of cycles travelled when using the results obtained from the heuristic is denoted by u_s. The value of u_s is an upper bound. Two vectors are then created from which the first is to keep track of the orders that have been inserted to a new sequence of orders, $\boldsymbol{v}_s$. The inserted vector contains the sequence of the orders that will be executed consecutively. The total distance for the orders in $\boldsymbol{v}_s$ is denoted by d_s. The second vector contains all the orders that have not been added to the sequence of executable orders, $\boldsymbol{v}_p$. The minimum distance that has to be travelled by the orders in $\boldsymbol{v}_p$ is denoted by d_p. The value of d_p is calculated as the maximum of either the sum of the minimum spans d_m of the orders in $\boldsymbol{v}_p$ or as the value of the location that is visited by the largest number of orders multiplied by the number of locations in the picking line, d_c. Let D_c be the number of times location d_c has to be visited. An order cannot be completed by visiting less locations than the number of locations on its minimum span. Also, if a location has to be visited by n orders, at least n cycles have to be travelled to complete the remaining orders.

The branch and bound method commences by means of a depth-first search. At each branch an upper bound is calculated. The total distance travelled by a picker is $d_t = d_s + d_p + 1$. Since the number of cycles travelled is of importance, the total distance is then translated into cycles. The total number of cycles is then calculated as $c_t = \left\lceil \frac{d_t}{m} \right\rceil$.

The value of d_c may be influenced by the location of the picker, once the last order in $\boldsymbol{v}_s$ is completed. Assume this position is denoted by p_s. Assume then b_n is the location in $\boldsymbol{v}_p$ with the largest number of orders that has to be visited, then

$$d_c = \begin{cases} (D_c - 1)m + (b_n - p_s) & \text{if } p_s \leq b_n \\ D_c m - (p_s + b_n) & \text{if } p_s > b_n. \end{cases} \tag{4.9}$$

The minimum distance is determined as $d_p = \max\{d_c, d_m\}$. If $\boldsymbol{v}_p$ is empty and $d_t < u_s$ a better solution has been found. The value of u_s is then updated as well as $\boldsymbol{b}_s$.

Consider an example of a picking line containing 10 locations. Assume the vector $\boldsymbol{v}_p$ contain 3 orders. These orders are $\mathcal{O}_1 = \{1, 3, 6\}$, $\mathcal{O}_2 = \{5, 8\}$ and $\mathcal{O}_3 = \{2, 3, 7, 8, 10\}$. Each of these orders have a unique minimum span. The length of the minimum span in order 1 is $|S_1^{\min}| = |\langle 1, 6\rangle| = 5$, $|S_2^{\min}| = |\langle 5, 8\rangle| = 3$ in order 2 and $|S_3^{\min}| = |\langle 7, 3\rangle| = 6$ in order 3. The sum of the minimum spans in $\boldsymbol{v}_p$ is 14 ($d_m = 14$). Now it is necessary to determine the location that is visited by the largest number of orders. Both orders 1 and 3 have to visit location 3. Location 3 is thus visited by the most orders in $\boldsymbol{v}_p$ ($b_n = 3$, $C = 2$). The value of d_c is dependant on the current location at which the picker is positioned. If the picker is currently positioned at location 2 ($p_s = 2$), $d_c = (2 - 1)10 + (3 - 2) = 11$ ($p_s \leq b_n$). If the picker is positioned at location 5 ($p_s = 5$), $d_c = (2 \times 10) - (5 - 3) = 18$, since three cycles would at least be travelled.

The pseudo code listing of the branch and bound algorithm may be found in Algorithm 1. This branch and bound approach was coded, but could not solve the OSP in a reasonable amount of time.

4.3.2 A lower bound

An integer programming formulation may be used to formulate the problem of minimising the maximum cut. This problem was formulated by Matthews & Visagie [77].

A number of variables required in modelling this problem are defined. Let

x_{ik}	be equal to 1 if order k starts at location i and 0 otherwise,
C	be the maximum cut of all the locations.

A set of parameters are also defined. Let

n	be the number of orders,
m	be the number of locations,
d_{ikj}	be equal to 1 if order k starts at location i and passes location j and 0 otherwise,
e_{ikj}	be equal to 1 if order k starts at location i and is completed location j and 0 otherwise.

The objective then is to minimise the maximum cut, *i.e.* to

Algorithm 1: A branch and bound approach.

Data: The values of $\boldsymbol{b}_s$ and u_s obtained by means of a heuristic.
Result: The values of $\boldsymbol{b}_s$ and u_s.
initialise $\boldsymbol{v}_s$;
$\boldsymbol{v}_p \leftarrow \boldsymbol{b}_s$;
sort $\boldsymbol{v}_p$ in decreasing order;
insert the first element in $\boldsymbol{v}_p$ into $\boldsymbol{v}_s$ and remove that element from $\boldsymbol{v}_p$;
calculate c_t according to the solution of the heuristic;
while *every branch resulting from a "dummy" root node has not been investigated* **do**
  calculate d_t;
  if $d_t \leq u_s$ **then**
    if $\boldsymbol{v}_s$ *contains* n *elements* **then**
      remove the last element in $\boldsymbol{v}_s$ and add to $\boldsymbol{v}_p$;
      find the smallest element in $\boldsymbol{v}_p$ larger than the last element in $\boldsymbol{v}_s$ and add to $\boldsymbol{v}_s$;
      if no such element exists or if this branch has been tested, repeat the above steps;
    else
      insert the last element in $\boldsymbol{v}_s$ into $\boldsymbol{v}_p$ and remove that element from $\boldsymbol{v}_s$;
    end
  else
    if $d_t < u_s$ **then**
      $\boldsymbol{b}_s \leftarrow \boldsymbol{v}_s$;
      $c_t \leftarrow \left\lceil \frac{d_t}{m} \right\rceil$;
      test whether switching the last two elements in $\boldsymbol{v}_s$ yields a better result;
      remove the last three elements in $\boldsymbol{v}_s$ and add to $\boldsymbol{v}_p$;
      if all the orders in $\boldsymbol{v}_p$ has been tested, remove the last element in $\boldsymbol{v}_s$ and add to $\boldsymbol{v}_p$ until a new order is reached;
    else
      remove the last element in $\boldsymbol{v}_s$ and add to $\boldsymbol{v}_p$;
      find the smallest element in $\boldsymbol{v}_p$ larger than the last element in $\boldsymbol{v}_s$ and add to $\boldsymbol{v}_s$;
      if no such element exists or if this branch has been tested, repeat the above steps;
    end
  end
end

$$\text{minimise } C \tag{4.10}$$

$$\text{subject to } \sum_{i=1}^{n} x_{ik} = 1 \qquad k = 1, 2, \ldots, n, \tag{4.11}$$

$$\sum_{k=1}^{n}\sum_{i=1}^{m} d_{ikj} x_{ik} \leq C \qquad j = 1, 2, \ldots, m, \tag{4.12}$$

$$\sum_{k=1}^{n} x_{j+1,k} - \sum_{k=1}^{n}\sum_{i=1}^{m} x_{ik} e_{ikj} = 0 \qquad j = 1, 2, \ldots, m-1, \tag{4.13}$$

$$\sum_{k=1}^{n} x_{1k} - \sum_{k=1}^{n}\sum_{i=1}^{m} x_{ik} e_{ikm} = 0, \tag{4.14}$$

$$x_{ik} \in \{0, 1\} \qquad \begin{matrix} k = 1, 2, \ldots, n, \\ i = 1, 2, \ldots, m. \end{matrix} \tag{4.15}$$

The objective function (4.10) minimises the maximal cut. Constraint set (4.11) assigns a starting position to each order. Constraint set (4.13) calculates the maximum cut. Equation sets (4.14) pairs every starting position with an ending position over all the orders.

The solution to the maximal cut model is described in total by two sets $\mathcal{S}$ and $\mathcal{E}$ as the starting and ending positions respectively, of each order and the objective function value. The set $\mathcal{E}$ is determined by the set $\mathcal{S}$, since the starting location of an order determines the ending location. Each starting position is paired with an ending position within one location. The solution may not be feasible as the starting and ending positions of all the orders may not allow for the generation of a complete tour [77].

A subtour generation heuristic was presented by Matthews & Visagie [77] and is supplied in Algorithm 2. This heuristic may be used to find a set of $\mathcal{T}$ subtours containing all the orders. It was then proven by Matthews that a feasible solution may be generated by linking the starting and ending positions in such a way that the cost with linking the subtours will result in at most one cycle travelled.

Algorithm 2: Subtour generation heuristic.

Data: A set of starting positions $\mathcal{S}$ and ending positions $\mathcal{E}$ obtained from formulation (4.10)–(4.15).
Result: A set $\mathcal{T}$ of subtours that link up all the orders.
while *All orders have not been allocated to a subtour* **do**
  Generate a new subtour s_t with the first available unallocated order;
  Let the current ending position of s_t be location i;
  if *An unallocated order exists which has a starting location corresponding to* $i + 1 \pmod m$ **then**
    Add this order to the end of the uncompleted subtour;
  end
  else
    Close the subtour by connecting the last order to the first order;
  end
end

Table 4.1 displays the results obtained for the maximal cut formulation when considering 22 data sets provided by Pep. Each of the data sets contains a number of orders (O) and a number of locations (L). The solutions are indicated in the number of cycles that have to be traveled in order to complete the OSP. Computational times are also indicated in seconds. The data sets are partitioned into three groups: *large, medium* and *small*. The large data sets contain more than 1 000 orders, medium data sets contain more than 200 orders but less than 1 000 orders and small data sets contain fewer than 200 orders.

The results in Table 4.1 will be used as a benchmark for all other OSP algorithms used further in this thesis. Heuristic approaches will be investigated in order to determine reasonable solutions that may be obtained in computational times that outperform those of the maximal cut formulation. For average data sets the maximal cut formulation does not exceed computational times in excess of 10 minutes. However, data set E requires an excess of 4 hours to be solved, using the maximal cut formulation. In a real life scenario a picking line manager would not have this amount of time to his/her disposal to construct a sequence in which orders may be executed. Alternative algorithms may therefore be used to reduce computational time and simultaneously providing a good quality solutions.

4.4 Tour construction heuristics

Heuristic algorithms for the TSP may be broadly divided into two classes: *tour construction procedures*, which build a tour by successively adding a new node at each step; and *tour improvement procedures*, which start from an initial tour and seek a better one by iteratively moving from one solution to another, according to adjacency relationships defined by a given

Data set	Size (O, L)	Maximal cut approach (cycles)	Computational time (seconds)
A	(1262,49)	1232	133.31
B	(1264,54)	1226	243.83
C	(1265,51)	1161	193.45
D	(1263,56)	1072	464.02
E	(1264,51)	1069	14733.61
F	(1258,55)	1025	253.48
G	(1258,53)	1005	223.53
H	(1244,54)	992	253.66
I	(1260,56)	955	303.78
J	(1264,56)	947	563.61
K	(943,63)	259	202.77
L	(846,56)	232	71.87
M	(728,51)	152	775.51
N	(733,55)	125	51.55
O	(396,63)	90	31.44
P	(574,48)	80	50.92
Q	(242,64)	45	20.69
R	(158,55)	14	10.34
S	(89,42)	9	10.12
T	(82,51)	8	10.14
U	(90,48)	7	10.17
V	(80,56)	6	10.2

Table 4.1: *Results obtained from the maximal cut algorithm used to solve the OSP. A total of 22 data sets are considered where number of orders (O) and locations (L) are displayed for each data set. The solution of the maximal cut algorithm is displayed as the number of cycles travelled by a picker and the computational time is indicated in seconds.*

neighbourhood structure [51]. The following tour construction heuristics present possible ways in which to sequence orders.

4.4.1 Next shortest order

The next shortest order (NSO) heuristic considers a single picker in a picking line. The picker considered starts at the first location in a picking line. If there are any orders that has to visit the first location, these orders are considered. The order with the shortest span is selected as the first order of the sequence. When the picker is finished with the first order, the current location is updated as well as the number of cycles travelled by the picker.

If, however, there are no orders starting at the current location, the current location of the picker is incremented, *i.e.* the picker travels to the next location. If there are any orders that has to visit this location, these orders are considered and the order with the shortest picking distance is selected. This process is repeated iteratively until all orders are allocated to the sequence. Algorithm 3 displays the pseudo code for the NSO heuristic.

The NSO heuristic determines the next order k to be picked, as

$$k = \arg\min_{k \in \mathcal{O}_i} |S_k^i|. \tag{4.16}$$

The nearest order may be interpreted as the order which may thus be completed within the smallest number of cycles from the current location.

Algorithm 3: Next shortest order heuristic (NSO).

Data: A set of orders, a set of SKU locations within a picking line and which SKUs must be picked by which order.
Result: A sequence in which the orders may be picked and the total number of cycles traversed.
Set the current location to location 1;
while *All orders have not been sequenced* **do**
  Determine all the orders containing a SKU at the pickers current location;
  if *no such order exists* **then**
    Increment the current location by one;
    Update the number of cycles traversed;
  end
  else
    Determine the best order (or most desirable order) which picks from the current location;
    Add this order to the sequence;
    Update the current location;
    Update the number of cycles traversed;
  end
end

4.4.2 Next shortest order relative to minimum span

The next shortest order relative to its minimum span (NSOM) heuristic is similar to the NSO heuristic — it only varies in the manner in which orders are selected to enter the sequence.

The NSOM heuristic determine the next order k to be picked, by

$$k = \arg\min_{i \in \mathcal{O}_k} \frac{|S_k^i|}{|S_k^{\min}|}, \tag{4.17}$$

where i is the current location of the picker. If no order is found, the current location i is increased by one. If $\frac{|S_k^i|}{|S_k^{\min}|} = 1$ the length of order k is at a minimum. This order will receive preference over other orders that do not pick on their minimum spans. This encourages the process of picking an order on its minimum span or an alternative span that does not increase the number of locations traversed considerably.

4.4.3 Next shortest order relative to its number of picks

The next shortest order relative to its number of picks (NSOP) heuristic is similar to the NSO heuristic — it only varies in the manner in which orders are selected to enter the sequence.

The NSOP heuristic determines the next order k to be picked, by

$$k = \arg\min_{i \in \mathcal{O}_k} \frac{|S_k^i|}{|\mathcal{O}_k|}, \tag{4.18}$$

where i is the current location of the picker. If $\frac{|S_k^i|}{|\mathcal{O}_k|} = 1$ the length of order k is at a minimum and there is a pick at each location on the minimum span.

4.4.4 Results for the tour construction heuristics

A number of test cases are used to evaluate the results obtained from the various models used to solve the OSP. The test cases consisted of actual order requests and actual layouts used by

Pep in practice. Ten out of the 22 test cases contain an excess of 1 200 orders with an average of approximately 56 locations used within the picking line. Table 4.2 displays the results (in terms of the number of cycles travelled to complete order picking on a picking line) obtained when the tour construction algorithms are used to solve the OSP for each of the data sets. The first column lists all the data sets, the second column displays the number of orders (O) and locations (L) used in each data set. Results from the maximal cut formulation is then presented.

	Size	Maximal cut	Total number of cycles		
Data set	(O, L)	approach	NSO	NSOM	NSOP
A	(1262,49)	1232	**1253**	1255	1254
B	(1264,54)	1226	**1243**	1247	1248
C	(1265,51)	1161	**1200**	1229	1226
D	(1263,56)	1072	**1120**	1203	1207
E	(1264,51)	1069	**1132**	1209	1199
F	(1258,55)	1025	**1072**	1142	1141
G	(1258,53)	1005	**1076**	1186	1185
H	(1244,54)	992	**1056**	1181	1178
I	(1260,56)	955	**1018**	1163	1161
J	(1264,56)	947	**999**	1163	1163
K	(943,63)	259	**367**	444	434
L	(846,56)	232	**322**	346	347
M	(728,51)	152	260	**248**	260
N	(733,55)	125	**209**	215	218
O	(396,63)	90	**200**	216	218
P	(574,48)	80	148	**139**	150
Q	(242,64)	45	110	109	**101**
R	(158,55)	14	28	**27**	28
S	(89,42)	9	**12**	13	**12**
T	(82,51)	8	**15**	**15**	**15**
U	(90,48)	7	**14**	**14**	**14**
V	(80,56)	6	10	**9**	10
Total		11711	12864	13773	13769

Table 4.2: *Total number of cycles obtained from the tour construction algorithms used to solve the OSP. The next shortest order (NSO), next shortest order relative to minimum span (NSOM) and the next shortest order relative to number of picks (NSOP) are tested against the lower bound obtained from the maximal cut formulation for each of the 22 data sets. The number of orders (O) and locations (L) are displayed for each data set. The best solutions are printed in boldface.*

The results in Table 4.2 indicate that NSO outperforms NSOM and NSOP in most test cases. The total number of cycles travelled in the NSO for the 22 data sets considered only differs by 9.85% from the total number of cycles travelled for the maximal cut formulation. However, the average deviation between the NSO and the maximal cut formulation is considerably larger. The NSO, as well as the NSOM and NSOP, fail to deliver good solutions in data sets containing less than 1 000 orders.

These tour construction algorithms are implemented in JAVA [101] and is able to solve an average problem in a short period of time. Computational times of the various algorithms are displayed in Table 4.3 in milliseconds.

4.5 Scope and ranking algorithms

The heuristic algorithms that have been considered sequence orders by only considering orders that start closest to the picker. Also, orders are not ranked in any manner in terms of their

Data set	Size (O, L)	Computational times NSBO	NSBOM	NSBOP
A	(1262,49)	712	693	771
B	(1264,54)	372	340	376
C	(1265,51)	236	235	187
D	(1263,56)	186	183	179
E	(1264,51)	183	175	180
F	(1258,55)	161	134	137
G	(1258,53)	251	184	175
H	(1244,54)	160	158	206
I	(1260,56)	180	184	184
J	(1264,56)	141	143	150
K	(943,63)	52	53	90
L	(846,56)	46	47	45
M	(728,51)	32	33	34
N	(733,55)	29	32	30
O	(396,63)	24	22	25
P	(574,48)	21	37	23
Q	(242,64)	26	10	13
R	(158,55)	6	6	6
S	(89,42)	4	4	5
T	(82,51)	5	5	5
U	(90,48)	4	7	5
V	(80,56)	5	7	5
Total		2836	2692	2831

Table 4.3: *Computational times in milliseconds of the tour construction algorithms used to solve the OSP. The next shortest order (NSO), next shortest order relative to minimum span (NSOM) and the next shortest order relative to number of picks (NSOP) are tested against the lower bound obtained from the maximal cut formulation for each of the 22 data sets. The number of orders (O) and locations (L) are displayed for each data set.*

respective importance to be picked. A number of heuristic approaches are considered where the picker has a *scope* of locations ahead of the current location of the picker, from which a possible order must be selected. This scope may be varied. The scope may then be combined by using measures to determine preferable orders during the picking process.

A further definition is needed to introduce the concept of ranking preferable orders.

Definition 6 *The* **preference spans** *of an order are the spans of that order, that starts at a location at which the order requires a SKU.*

Let P_k^p denote the p^{th} shortest preference span of order k. Referring to Figure 4.1, $P_k^1 = S_k^9, P_k^2 = S_k^{16}$ and $P_k^3 = S_k^{12}$ are the only three preference spans of order k.

Initially each order is ranked in terms of its minimum span relative to its p^{th} best preference span.

4.5.1 Ranked preference span algorithm

In contrast to the heuristics considered thus far, the greedy scope algorithm may consider picking orders that do not necessarily pick from the pickers current location. Instead the picker considers orders starting within a certain scope of the current location. This scope, s_c, has to be specified in advance. The range specifies the number of locations the picker considers ahead

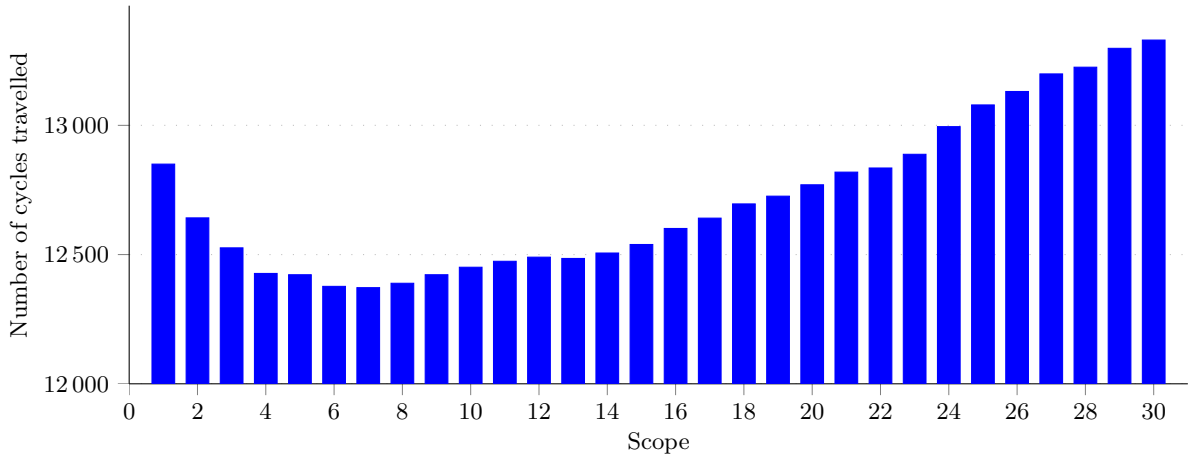


Figure 4.5: *A bar chart displaying the results (cycles travelled) obtained for all of the 22 data sets considered when setting a scope of possible starting locations. The scope is varied from 1 to 30 locations ahead of the current location of the picker.*

of the current location as possible starting locations for a new order to be picked. All orders that have preference span starting within the scope are considered as possible orders to choose from.

Figure 4.5 displays the total number of cycles travelled for all 22 data sets, when the picker's scope is varied at different values. The lowest total number of cycles are found when the scope is set at 7.

A second measure may be considered where orders are ranked in terms of the length of their spans. Orders may be ranked according to a selected span for order picking relative to a different span of that order. The orders are initially ranked according to $\frac{|S_k^{\min}|}{|P_k^p|}$ in increasing order, which refers to the p^{th} shortest preference spans of order k. The order selected during every iteration of the algorithm is an order starting at the current location for the order that is ranked best.

These two heuristic approaches may be combined, where orders are ranked and the picker has a certain scope to select orders ahead of the current location. The span selected for comparison with the minimum span of each order is varied as the second best span to the fifth best span. The scope is varied for all integer values between and including 1 and 30. Figure 4.6 displays a bar chart where the scope as well as the preference span is varied. The lowest total number of cycles are realised when $s_c = 6$ and $p = 3$.

4.5.2 Minimum span preserving algorithm

It might be beneficial to attempt to pick all orders on their minimum spans. The greedy heuristics in §4.4 often pick the initial orders on their minimum spans, however, when fewer orders remain to sequence, many orders are picked on span much longer than their minimum span(s).

This algorithm starts by determining for each location, the number of orders that start at that location if the order is picked on its minimum span. If an order has multiple minimum spans, each location that is the start of a minimum span is taken into account. The aim is to sequence orders in such a way that the ending location of the span of a selected order is followed by the starting location of the minimum spans of a large number of orders.

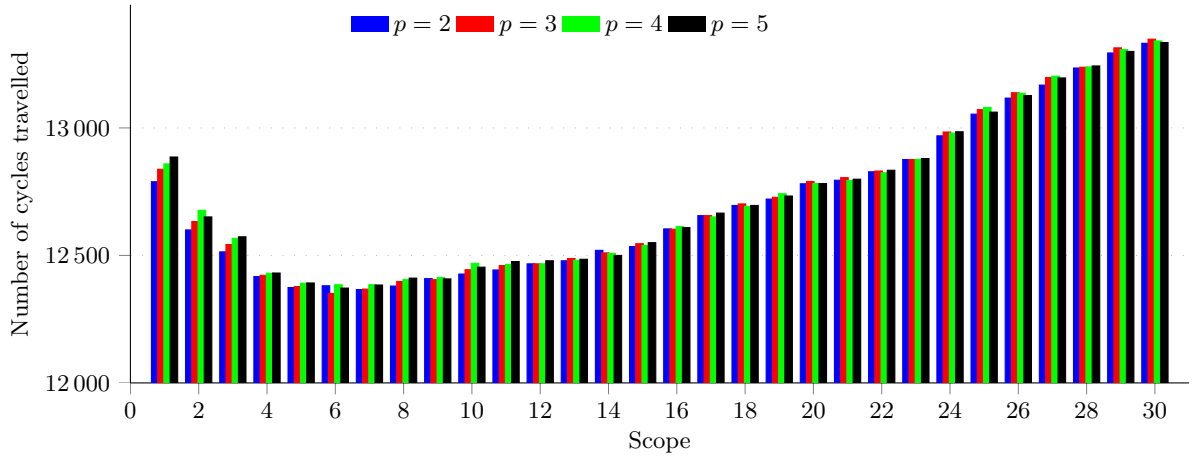


Figure 4.6: *A bar chart displaying the results (cycles travelled) obtained for all of the 22 data sets considered when setting a scope of possible starting locations and altering the rank of orders based on various spans. The score for each order is varied by considering the p^{th} best span relative to the minimum span of that order. The scope is varied from 1 to 30 locations ahead of the current location of the picker.*

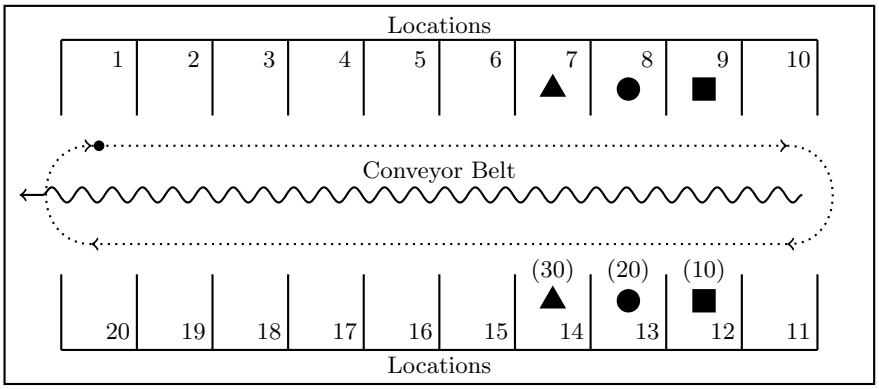


Figure 4.7: *A schematic representation of the layout of a picking line containing 20 locations and 3 orders.*

This algorithm is also combined with a scope. Consider the example in Figure 4.7 of three orders. A picker is currently positioned at location 6 and requires a new order. If the scope is set at three locations the picker may be awarded one of three orders, namely the triangles, the circles or the squares. The ending location of each order contains a number which indicates the number of orders that start on their minimum span at the location following the final location of that order. In Figure 4.7 the squares contain 10 such orders, the circles 20 and the triangles contain 30. In this case it would be beneficial to pick the triangles, since the picker would pick the triangles on its minimum span (the same goes for the circles and the squares), but a large number of orders await that may be picked on their minimum span (relative to the circles and squares). In this manner orders are still picked on their minimum spans, but each location is likely to be the starting location of the minimum span of multiple orders. The minimum spans are preserved such that orders may be picked on their minimum spans during the entire order picking process.

The scope of this algorithm may be varied for better performance of the algorithm. During order picking, if no order, within the scope, is found, orders that do not start at their minimum spans are considered. If no such order is found, the scope is temporarily increased until an order is found that picks from a location within the scope.

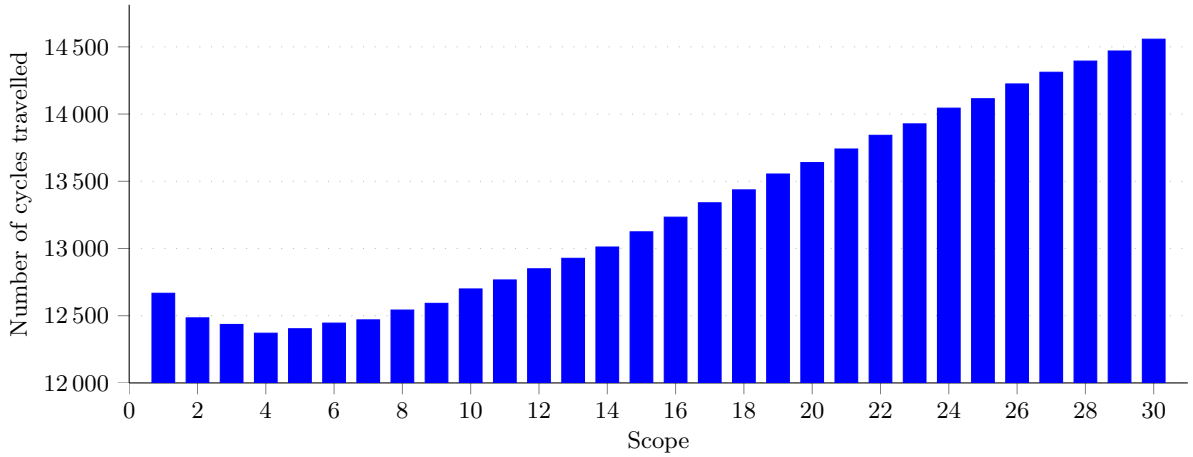


Figure 4.8: *A bar chart displaying the results (cycles travelled) obtained for all of the 22 data sets considered when setting a scope of possible starting locations for the minimum span preserving algorithm. The scope is varied from 1 to 30 locations ahead of the current location of the picker.*

Figure 4.8 displays a bar chart of the total number of cycles travelled for all 22 data sets considered when the scope is varied from 1 to 30. The best results are obtained for $s_c = 4$.

4.5.3 Results of the scope and ranking algorithms

From the results in Figure 4.6 concerning the ranked preference span algorithm, the span selected to determine a score for each order does not have a significant influence on the solution quality, however, the scope may have a direct effect on the solution quality. The least number of cycles may be achieved when $p = 3$ and $s_c = 6$. The best results are obtained when $s_c = 6$ irrespective of the value of p. Similarly, the minimum span preserving algorithm, displays the best overall results when $s_c = 4$.

Table 4.4 displays the results for the best solution obtained for the ranked preference span algorithm and the minimum span preserving algorithm. The ranked preference span algorithm and the minimum span preserving algorithm achieve similar results, however, the ranked preference span algorithm outperforms the minimum span preserving algorithm for medium and smaller data sets.

4.6 Awarding spans according to preference ratios

All the algorithms considered thus far have considered all the possible spans of an order. A number of algorithms are developed to reduce the number of possible preference spans for each order, to eliminate preference spans that will potentially reduce the solution quality, forcing orders to be picked on shorter preference spans.

This heuristic procedure awards a certain number of possible preference spans to an order. The preference spans awarded to each order, are ranked in increasing order in terms of the number of locations that have to be traversed, *i.e.* if an order k is awarded u_k possible preference spans, these spans are the u_k shortest possible preference spans of order k.

One span, from a number of possible preference spans, have to be selected for each order to solve the OSP within a picking line. Initially the maximum number of possible preference spans,

Data set	Size (O, L)	Maximal cut approach	Ranked preference span		Minimum span preserving	
			Number of cycles	Computational time (ms)	Number of cycles	Computational time (ms)
A	(1262,49)	1232	1250	390	**1247**	518
B	(1264,54)	1226	1240	248	**1236**	264
C	(1265,51)	1161	**1180**	169	1181	209
D	(1263,56)	1072	1113	196	**1103**	207
E	(1264,51)	1069	**1125**	175	1133	218
F	(1258,55)	1025	1095	174	**1063**	660
G	(1258,53)	1005	**1060**	185	1066	213
H	(1244,54)	992	**1029**	167	1036	261
I	(1260,56)	955	1015	186	**1005**	204
J	(1264,56)	947	**983**	167	986	200
K	(943,63)	259	300	46	**299**	57
L	(846,56)	232	**261**	34	**261**	48
M	(728,51)	152	208	33	**194**	39
N	(733,55)	125	**150**	29	160	34
O	(396,63)	90	**124**	47	145	34
P	(574,48)	80	**111**	21	121	37
Q	(242,64)	45	**54**	12	67	15
R	(158,55)	14	**17**	7	22	7
S	(89,42)	9	**10**	7	12	5
T	(82,51)	8	**11**	7	13	7
U	(90,48)	7	**8**	6	11	5
V	(80,56)	6	**7**	8	9	6
Total		11711	12351	2314	12370	3248

Table 4.4: *Results indicating the number of cycles travelled when using the ranked preference span algorithm, when $p = 3$ and $s_c = 6$, as well as the minimum span preserving algorithm when the $s_c = 4$. The results are compared to the lower bound obtained from the maximal cut formulation for each of the 22 data sets. The number of orders (O) and locations (L) are displayed for each data set. Computational time are displayed in milliseconds. The best performing algorithm is indicated in boldface for each data set.*

$Q+1$, have to be determined. No order may have more than $Q+1$ preference spans from which to select an appropriate candidate span to be used in the OSP. Let Q_k be the maximum number of distinct preference spans that an order may have. Consider an order $\mathcal{O}_k = \{\alpha, \beta, \gamma\}$. Order k has three preference spans, namely $S_k^\alpha = \langle \alpha, \gamma \rangle$, $S_k^\beta = \langle \beta, \alpha \rangle$ and $S_k^\gamma = \langle \gamma, \beta \rangle$. The number of preference spans within an order k is equal to the number of locations that order k have to visit ($|\mathcal{O}_k| = Q_k$).

An order can never contain more preference spans than the number of locations it has to visit. Therefore, the number of preference spans considered in each order must be the minimum of either the maximum number of possible preference spans considered ($Q+1$) or the total number of locations within an order ($|\mathcal{O}_k|$). Let u_k be the number of preference spans considered in order k ($u_k = \min\{Q+1, |\mathcal{O}_k|\}$). Let $\mathcal{S}_k = \{q_k^1, q_k^2, \ldots, q_k^s, \ldots, q_k^{u_k}\}$ be the set of the lengths of the spans in order k, listed in decreasing order in terms of distance travelled. In this case q_k^s refers to the length of the s^{th} best preference span in order k.

Intervals are specified to determine the number of preference spans that may be awarded to each order. The number of preference spans assigned to each order is determined by the ratio, r_k, of the number of locations traversed when order picking is done on the minimum span divided by the number of locations traversed when order picking is done on the ${u_k}^{\text{th}}$ best preference span of order k. Thus $r_k = \frac{q_k^1}{q_k^{u_k}}$.

The ratio, r_k, is used to determine the number of preference spans assigned to order k. If the maximum number of possible preference spans are $Q+1$, *test intervals* $\{t_1, t_2, \ldots, t_s, \ldots, t_Q\}$ is constructed such that $t_1 > t_2 > \cdots t_s \cdots > t_Q$, where each $t_s \in \{0, 1\}$, $s = 1, 2, \ldots, Q$. If $r_k > t_1$, Q possible preference spans are awarded to order k. When order $t_{p+1} < r_k \leq t_p$, a total of $Q + 1 - p$ possible preference spans are awarded to order k. If $r_k < t_Q$, only the minimum span is awarded to order k. Let $\mathcal{P}_k$ be an ordered set of the spans assigned to order k.

The reason for assigning the preference spans to orders in this manner is that if an order contains a large ratio ($r_k \approx 1$), the number of locations that have to be traversed in the minimum span does not differ considerably in comparison to the ${u_k}^{\text{th}}$ best preference span. A small cost (in terms of locations traversed) is incurred when the minimum span of an order is not selected, when that order has a large ratio. However, if an order contains a small ratio, many more locations than the minimum span have to be traversed to complete the order when the minimum span is not selected. In this case the algorithm will only award a single span or few spans, to ensure that unnecessary travelling is avoided.

Once the number of preference spans have been awarded to all the orders, each span is *ranked* according to one of two criteria. According to the first criteria, each order is ranked by means of the ratio $L_k = Q + 1 - u_k + r_k$ for $k = 1, 2, \ldots, n$. A *preference list* is constructed where each selected span of an order is listed. The preference list is ranked in decreasing order. In this manner if order k is ranked higher than order ℓ (*i.e.* $L_k > L_\ell$), all of the u_k spans awarded to order k will be ranked higher than all of the u_ℓ spans that is awarded to order ℓ. The pseudo code for the algorithm utilising this criteria is supplied in Algorithm 4.

Algorithm 4: Preference ratio RS_1

Data: The maximum number of preference spans Q, as well as a test interval $\{t_1, t_2, \ldots, t_{Q-1}\}$.
Result: An ordered list in which the orders should be picked.
Let $\mathcal{B} = \emptyset$ and $|\mathcal{B}| = 0$;
$i \leftarrow 1$;
Calculate the ordered set $\mathcal{S}_k$ for all $k = 1, 2, \ldots, n$;
Calculate r_k for all $k = 1, 2, \ldots, n$;
Construct the set $\mathcal{P}_k$ for each order $k = 1, 2, \ldots, n$;
Calculate the ratio L_k for all $k = 1, 2, \ldots, n$;
Create an orderned preference list $\mathcal{V}_1$ containing all orders,where each order k according to the value of L_k for all $k = 1, 2, \ldots, n$, where higher values of L_k constitues a higher rank;
while $|\mathcal{B}| \neq n$ **do**
  while *an order is not found* **do**
    Find the first preference span $p_k \in \mathcal{P}_k$ for $k \in \mathcal{V}_1$ that starts at location i;
    if *no order is found* **then**
      $i \leftarrow i + 1 \,(\text{mod } m)$;
    end
    $\mathcal{V}_1 \leftarrow \mathcal{V}_1 \setminus \mathcal{P}_k$;
  end
  $\mathcal{B} \leftarrow \mathcal{B} \cup p_k$;
  $i \leftarrow$ endpoint of $Q_k + 1$;
end

According to the second criteria, each span of an order receives a unique preference. The preference of the p^{th} span of an order k is calculated as $N_k^p = u_k + \frac{q_k^p}{|\mathcal{O}_k|}$ for $k = 1, 2, \ldots, n$ and $p = 0, 1, \ldots, u_k$. This preference list is ranked in increasing order. In this situation orders that have to visit locations on a picking line that are sparsely distributed will receive more spans. Orders that may contain a cluster of locations on particular part of the picking line, will receive a small amount spans.

Once the preference list is constructed, a sequence in which orders may be picked is constructed. The sequence starts with order picking commencing at the first location. If the first criteria is considered, the preference list is searched starting at the first entry in the list. The preference list is searched until a preference span is reached that starts at the current location of the picker. If such a span exists, this preference span is entered to the sequence, the pickers position is updated and all the preference spans of that order in the preference list is eliminated. However, if no preference span starts at the current position of the picker, the picker moves to the next location. In this manner all orders will be included in a sequence.

The maximum number of possible preference spans, $Q+1$, that may be selected, as well as the test interval, must be determined prior to the execution of this algorithm a series of simulations are constructed to determine the relevent number of preference spans and the intervals that yields the lowest number of spans traversed. The pseudo code containing this procedure is supplied in Algorithm 5.

Algorithm 5: Preference ratio RS_2

Data: The maximum number of preference spans Q, as well as a test interval $\{t_1, t_2, \ldots, t_{Q-1}\}$.
Result: An ordered list in which the orders should be picked.
Let $\mathcal{B} = \emptyset$ and $|\mathcal{B}| = 0$;
Let i be the current location;
$i \leftarrow 1$;
Calculate the ordered set $\mathcal{S}_k$ for all $k = 1, 2, \ldots, n$;
Calculate r_k for all $k = 1, 2, \ldots, n$;
Construct the set $\mathcal{P}_k$ for each order $k = 1, 2, \ldots, n$;
Calculate the ratio N_k^p for all $p \in \mathcal{P}_k$ and $k = 1, 2, \ldots, n$;
Create an orderened preference list $\mathcal{V}_2$ for all orders, where every order k receives a rank according to the value of N_k^p for all $p \in \mathcal{P}_k$ and $k = 1, 2, \ldots, n$, where lower values of N_k^p constitutes a higher rank;
while $|\mathcal{B}| \neq n$ **do**
  while *an order is not found* **do**
    Find the first preference span $p_k \in \mathcal{V}_2$ that starts at location i;
    if *no order is found* **then**
      $i \leftarrow i + 1 \,(\mathrm{mod}\ m)$;
    end
    $\mathcal{V}_2 \leftarrow \mathcal{V}_2 \setminus \mathcal{P}_k$;
  end
  $\mathcal{B} \leftarrow \mathcal{B} \cup p_k$;
  $i \leftarrow$ endpoint of $Q_k + 1$;
end

In order to illustrate the algorithm an example is presented. Consider two orders namely, $\mathcal{O}_1 = \{1, 3, 7, 8\}$ and $\mathcal{O}_2 = \{2, 4, 9\}$, that have to be picked from a picking line containing 10 locations. Order 1 contain four preference spans namely, $|S_1^1| = |\langle 1, 8\rangle| = 7$, $|S_1^3| = |\langle 3, 1\rangle| = 8$, $|S_1^7| = |\langle 7, 3\rangle| = 6$ and $|S_1^8| = |\langle 8, 7\rangle| = 9$. Order 2 contain three preference spans namely, $|S_2^2| = |\langle 2, 9\rangle| = 7$, $|S_2^4| = |\langle 4, 2\rangle| = 8$ and $|S_2^9| = |\langle 9, 4\rangle| = 5$. Let the maximum number of possible preference spans assigned to a picking line be 3 ($Q = 2$).

The ranked sizes of the preference spans in an increasing manner for order 1 and 2 are $\{\langle 7, 3\rangle$, $\langle 1, 8\rangle$, $\langle 3, 1\rangle$, $\langle 8, 7\rangle\}$ and $\{\langle 9, 4\rangle$, $\langle 2, 9\rangle$, $\langle 4, 2\rangle\}$. However, since $Q = 2$, $u_1 = u_2 = 3$. Only three shortest preference spans of order 1 may be considered. Therefore, $\mathcal{P}_1 = \{\langle 7, 3\rangle, \langle 1, 8\rangle, \langle 3, 1\rangle\}$ and $\mathcal{P}_2 = \{\langle 9, 4\rangle$, $\langle 2, 9\rangle$, $\langle 4, 2\rangle\}$.

Consider the case where the test interval is specified as $\{0.8, 0.6\}$. Preference spans are now ranked according to the first criteria. The ratio of order 1 is $R_1 = \frac{|\langle 7,3\rangle|}{|\langle 3,1\rangle|} = \frac{6}{8} = 0.75$. Since $0.6 < R_1 \leq 0.8$, the two shortest preference spans ($\langle 7, 3\rangle$ and $\langle 1, 8\rangle$) are considered for order

1. The ratio of order 2 is $R_2 = \frac{|\langle 9,4\rangle|}{|\langle 4,2\rangle|} = \frac{5}{8} = 0.625$. Since $0.6 < R_2 \leq 0.8$, the two shortest preference spans ($\langle 9,4\rangle$ and $\langle 2,9\rangle$) are considered for order 2.

All the selected preference spans are ranked in a final list. $L_1 = 3 - 3 + 0.8 = 0.8$ and $L_2 = 3 - 3 + 0.625 = 0.625$. Both preference spans of order 2 will be considered before the preference spans of order 1 will be considered. Order picking may now be started. Order picking starts at location 1. None of the preference spans of order 2 starts at location 1, however order 1 contains a preference span starting at location 1 ($\langle 1,8\rangle$). This span is selected for order 1 and the current position of the picker is updated. The picker is now positioned at location 9. Order 2 contains a preference span starting at location 9 ($\langle 9,4\rangle$). Order 2 is picked on this span and the picker is positioned at location 4. A total of 14 locations are travelled to complete order 1 and 2 using the first criteria.

Consider the case where spans are ranked according to the second criteria. Each preference span of each order k receives a ratio, N_k^p, for $p = 1, \ldots, u_k$ and $k = 1, 2, \ldots, n$. Both orders contain 2 preference spans. Thus $R_1^1 = 2 + \frac{6}{8} = 2.75$, $R_1^2 = 2 + \frac{7}{8} = 2.875$, $R_2^1 = 2 + \frac{5}{8} = 2.625$ and $R_2^2 = 2 + \frac{7}{8} = 2.875$. All of these ratios are used to rank the spans of each order in a preference list. The preference list will be $\{\langle 9,4\rangle_2, \langle 7,3\rangle_1, \langle 1,8\rangle_1, \langle 2,9\rangle_2\}$. The first and the last span corresponds to the spans of order 2 while the remaining two spans correspond to order 1 as indicated by the subscript. Since $R_1^2 = R_2^2$, the third and the fourth elements in the preference list may be swapped. This will, however, make no difference when executing the algorithm as the two spans start at different locations. Order picking may now be started. Order picking starts at location 1. The preference list is searched until a span is found that starts at location 1. The third preference span in the preference list is selected. This preference span corresponds to order 1. Order 1 is picked first and the current location of the picker is updated. The picker is now positioned at location 8. All the preference spans in the preference list corresponding to order 1 is removed. The preference list now becomes $\{\langle 9,4\rangle_2, \langle 2,9\rangle_2\}$. The list is searched for a preference span starting at location 9. The first preference span in the preference list is selected. The current position of the picker is updated upon completing order picking for order 2. The picker has completed order picking and is positioned at location 4. The picker has travelled 14 locations in order to complete both orders.

For the data presented in this example both criteria reaches the same sequence. However, when a large number of orders are considered, the results of the two crieria may differ.

Table 4.5 displays the results obtained from the selecting spans according to ratio tests. The number of ratios awarded according to the first criteria is indicated by RS_1 (as displayed in Algorithm 4) and the second criteria for determining the number of ratios is indicated by RS_2 (as displayed in Algorithm 5). The minimum number of cycles travelled for each criteria is displayed for each data set. Each result is accompanied by the preference list that is able to achieve this number of cycles.

Entries in the preference list are varied in permutations. Initially the preference list is constructed as $\{0.9\}$ and decreases in intervals of 0.1 units. When the preference list is $\{0.1\}$, the preference list is awarded another ratio and the preference list becomes $\{0.9, 0.8\}$. All permutations where each ratio is a multiple of 0.1 and where the first ratio is larger than the second ratio are tested, upon which the preference list is awareded another ratio. This process is repeated iteratively until all possible permutations are considered.

Results in Table 4.5 indicate that RS_2 outperforms RS_1 for eight of the twelve data sets and RS_1 only outperforms RS_2 for one data set. It is therefore better to rank each span of an order individually according to RS_2 when considering a large variation in the length and entries

Data set	Size (O, L)	Maximal cut approach	RS_1	RS_1 preference list	RS_2	RS_2 preference list
A	(1262,49)	1232	**1233**	$\{0.9, 0.8, 0.7\}$	**1233**	$\{0.9, 0.8, 0.7\}$
B	(1264,54)	1226	**1226**	$\{0.8, 0.6, 0.5\}$	**1226**	$\{0.8, 0.6, 0.5\}$
C	(1265,51)	1161	1181	$\{0.7, 0.6, 0.5, 0.4\}$	**1177**	$\{0.8, 0.7, 0.4, 0.3, 0.2, 0.1\}$
D	(1263,56)	1072	1164	$\{0.9, 0.6, 0.4, 0.3, 0.2, 0.1\}$	**1121**	$\{0.9, 0.8, 0.7, \ldots, 0.2, 0.1\}$
E	(1264,51)	1069	1138	$\{0.4, 0.3\}$	**1121**	$\{0.9, 0.8, 0.7, 0.6, 0.5\}$
F	(1258,55)	1025	1099	$\{0.6, 0.4, 0.3\}$	**1065**	$\{0.7, 0.5, 0.2, 0.1\}$
G	(1258,53)	1005	1085	$\{0.6, 0.3\}$	**1065**	$\{0.9, 0.8, 0.6, 0.2\}$
H	(1244,54)	992	1060	$\{0.8, 0.5\}$	**1041**	$\{0.9, 0.6, 0.5\}$
I	(1260,56)	955	1041	$\{0.9, 0.7, 0.4\}$	**1021**	$\{0.9, 0.8, 0.7, 0.5\}$
J	(1264,56)	947	1022	$\{0.8, 0.7\}$	**995**	$\{0.8, 0.7, 0.5\}$
K	(943,63)	259	284	$\{0.7\}$	**278**	$\{0.7\}$
L	(846,56)	232	**244**	$\{0.8\}$	**244**	$\{0.8\}$
M	(728,51)	152	**180**	$\{0.9\}$	191	$\{0.9\}$
N	(733,55)	125	139	$\{0.9\}$	**137**	$\{0.9\}$
O	(396,63)	90	**109**	$\{0.7\}$	**109**	$\{0.9\}$
P	(574,48)	80	**95**	$\{0.6\}$	**95**	$\{0.8\}$
Q	(242,64)	45	54	$\{0.5\}$	**51**	$\{0.5\}$
R	(158,55)	14	**16**	$\{0.6\}$	**16**	$\{0.5\}$
S	(89,42)	9	**10**	$\{0.9\}$	11	$\{0.9\}$
T	(82,51)	8	**10**	$\{0.9\}$	11	$\{0.9\}$
U	(90,48)	7	**8**	$\{0.9\}$	**8**	$\{0.9\}$
V	(80,56)	6	**7**	$\{0.9\}$	**7**	$\{0.9\}$
Total		11711	12405		12223	

Table 4.5: *The number of ratios awarded according to the first criteria is indicated by RS_1 and the second criteria for determining the number of ratios is indicated by RS_2. One preference list is displayed that may be used to achieve the minimum number of cycles for each data set. The criteria that achieved the lowest number of cycles for each data set is displayed in boldface. The number of orders (O) and locations (L) are displayed for each data set.*

of a preference list. In order to test the robustness of both criteria, a series of instances are constructed where the length of the preference list is varied.

A series of tests are considered to perform a sensitivity analysis on the length of the preference list. The reduction in the size of the preference list entails that fewer solutions are calculated, which decreases computing time. Table 4.6 displays results obtained from varying the length of the preference list. Both criteria are considered in this table, where the length of the preference list is indicated in brackets. Once again the preference list is constructed as $\{0.9\}$ and decreases in intervals of 0.1 units. When the preference list is $\{0.1\}$, the preference list is awarded another ratio and the preference list becomes $\{0.9, 0.8\}$. All permutations where each ratio is a multiple of 0.1 and where the first ratio is larger than the second ratio are tested, upon which the preference list is awareded another ratio. This process is repeated iteratively until all possible permutations are considered for each case displayed Table 4.6. The results indicate that savings in cycles travelled are possible when increasing the length of the preference list. On average the solutions obtained for the second criteria outperform the solutions of the first criteria. It is clear that the solution quality is not very sensitive to a change in the length of the preference list.

The computational times of the results obtained in Table 4.6 are displayed in Table 4.7. A large increase in total computational time is incurred, when increasing the length from 3 possible spans to 6 possible spans. The computational times to solve instances by using the second criteria are higher on average than the first criteria.

Data set	Size (O, L)	Maximal cut approach	RS_1 (10)	RS_1 (6)	RS_1 (3)	RS_1 (2)	RS_2 (10)	RS_2 (6)	RS_2 (3)	RS_2 (2)
A	(1262,49)	1232	1233	1233	1234	1234	1233	1233	1234	1234
B	(1264,54)	1226	1226	1226	1227	1232	1226	1226	1227	1232
C	(1265,51)	1161	1181	1181	1198	1207	1177	1178	1183	1202
D	(1263,56)	1072	1164	1169	1196	1201	1121	1174	1191	1194
E	(1264,51)	1069	1138	1138	1138	1147	1121	1121	1128	1146
F	(1258,55)	1025	1099	1099	1102	1153	1065	1065	1083	1133
G	(1258,53)	1005	1085	1085	1085	1094	1065	1065	1068	1083
H	(1244,54)	992	1060	1060	1060	1085	1041	1041	1048	1066
I	(1260,56)	955	1041	1041	1043	1058	1021	1021	1023	1034
J	(1264,56)	947	1022	1022	1022	1026	995	995	1001	1015
K	(943,63)	259	284	284	284	284	278	278	278	278
L	(846,56)	232	244	244	244	244	244	244	244	244
M	(728,51)	152	180	180	180	180	191	191	191	191
N	(733,55)	125	139	139	139	139	137	137	137	137
O	(396,63)	90	109	109	109	109	109	109	109	109
P	(574,48)	80	95	95	95	95	95	95	95	95
Q	(242,64)	45	54	54	54	54	51	51	51	51
R	(158,55)	14	16	16	16	16	16	16	16	16
S	(89,42)	9	10	10	10	10	11	11	11	11
T	(82,51)	8	10	10	10	10	11	11	11	11
U	(90,48)	7	8	8	8	8	8	8	8	8
V	(80,56)	6	7	7	7	7	7	7	7	7
Total		11711	12405	12410	12461	12593	12223	12277	12344	12497

Table 4.6: *The number of ratios awarded according to the first and second criteria are indicated by RS_1 and RS_2 respectively. The minimum number of cycles travelled for each criteria is displayed for each data set. The number of orders (O) and locations (L) are displayed for each data set.*

It is evident that a trade-off exists between solution quality and the length of the preference list to solve the OSP using the preference list. Figure 4.9 contains a bar chart of solution quality versus the maximum number of spans awarded to each order.

The preference list has to be determined before any of the algorithms may be used. A preference list is considered and solutions are obtained for each data set. The preference list yielding the lowest total number of cycles travelled are listed in Table 4.8 in increasing order. These preference lists deliver stable solution qualities. The preferred preference list for both criteria is $\{0.6\}$. However, RS_2 outperforms RS_1 when the criteria is $\{0.6\}$. It appears that awarding fewer spans yields better results when all data sets are considered.

4.7 Generalised assignment approach

Assignment problems deal with assigning a number of sources (supply points) to destinations (demand points). Each source must be assigned to a destination, where a cost is incurred for each of the source-destination combinations. It is required to assign exactly one source to one destination in such a way that the total cost of the assignment is minimised [51, 120]. The

Data set	Size (O, L)	RS_1 (10)	RS_1 (6)	RS_1 (3)	RS_1 (2)	RS_2 (10)	RS_2 (6)	RS_2 (3)	RS_2 (2)
A	(1262,49)	75.552	52.949	8.039	3.145	96.025	69.652	9.632	3.798
B	(1264,54)	91.781	63.940	8.359	2.872	125.104	89.308	10.499	3.681
C	(1265,51)	47.530	34.029	6.036	2.264	65.146	47.346	6.926	2.569
D	(1263,56)	64.886	50.665	7.658	2.510	95.586	73.585	9.428	3.065
E	(1264,51)	39.396	27.490	4.731	2.252	54.220	37.371	5.327	2.445
F	(1258,55)	53.200	42.726	7.419	2.301	57.774	44.279	8.305	2.661
G	(1258,53)	29.882	21.600	4.448	1.886	40.031	27.849	4.744	2.008
H	(1244,54)	26.675	18.770	3.588	1.756	32.945	22.979	3.916	1.823
I	(1260,56)	25.795	18.574	3.909	1.775	33.945	23.992	4.074	1.864
J	(1264,56)	20.880	14.604	3.201	1.648	26.266	17.814	3.258	1.978
K	(943,63)	9.016	6.874	1.876	1.359	11.423	8.404	1.888	1.628
L	(846,56)	7.852	6.166	1.839	1.505	11.117	8.121	1.866	1.360
M	(728,51)	6.740	5.479	2.014	1.619	8.504	6.325	1.883	1.392
N	(733,55)	4.785	3.802	1.808	1.309	6.021	4.595	1.776	1.459
O	(396,63)	4.616	3.607	1.453	1.291	5.524	4.049	1.435	1.523
P	(574,48)	4.085	3.447	1.483	1.455	4.760	4.146	1.476	1.364
Q	(242,64)	2.232	1.703	1.470	1.572	2.110	1.815	1.344	1.256
R	(158,55)	1.429	1.305	1.507	1.255	1.533	1.323	1.396	1.404
S	(89,42)	1.287	1.358	1.375	1.240	1.474	1.371	1.139	1.561
T	(82,51)	1.442	1.463	1.303	1.381	1.245	1.731	1.147	1.246
U	(90,48)	1.557	1.199	1.527	1.498	1.226	1.254	1.300	1.219
V	(80,56)	1.299	1.206	1.538	1.241	1.346	1.205	1.389	1.382
Total		521.917	382.956	76.581	39.134	683.325	498.514	84.148	42.686

Table 4.7: *Computational times in seconds when assigning preference to ratios for data sets when solving the OSP. The number of ratios awarded according to the first and second criteria are indicated by RS_1 and RS_2. The number of orders (O) and locations (L) are displayed for each data set.*

mathematical model for the classic assignment problem may be given as

$$\text{minimise} \sum_{k=1}^{n}\sum_{\ell=1}^{n} c_{k\ell}x_{k\ell} \tag{4.19}$$

$$\text{subject to} \sum_{k=1}^{n} x_{k\ell} = 1 \qquad \ell = 1, 2, \ldots, n, \tag{4.20}$$

$$\sum_{\ell=1}^{n} x_{k\ell} = 1 \qquad k = 1, 2, \ldots, n, \tag{4.21}$$

$$x_{k\ell} \in \{0, 1\} \qquad \begin{matrix} k = 1, 2, \ldots, n, \\ \ell = 1, 2, \ldots, n. \end{matrix} \tag{4.22}$$

If $x_{k\ell} = 1$ supply point k is assigned to demand point ℓ, and 0 otherwise, and $c_{k\ell}$ is the cost of assigning supply point k to demand point ℓ [89]. Constraint set (4.20) ensures that only one supply point is assigned to a demand point, while constraint set (4.21) ensures that only one demand point is assigned to a supply point.

Matthews & Visagie [77] proved that a number of orders grouped into a set of $\mathcal{T}$ subtours may be linked to create a feasible solution with the addition of at most one cycle travelled (see §4.3.2). It is therefore possible to convert a solution to the assignment problem to obtain a feasible solution of the OSP. The result obtained from the assignment problem may contain subtours. Linking these subtours will result in an increase of at most one cycle.

The assignment problem is used to link two orders resulting in the minimum distance travelled.

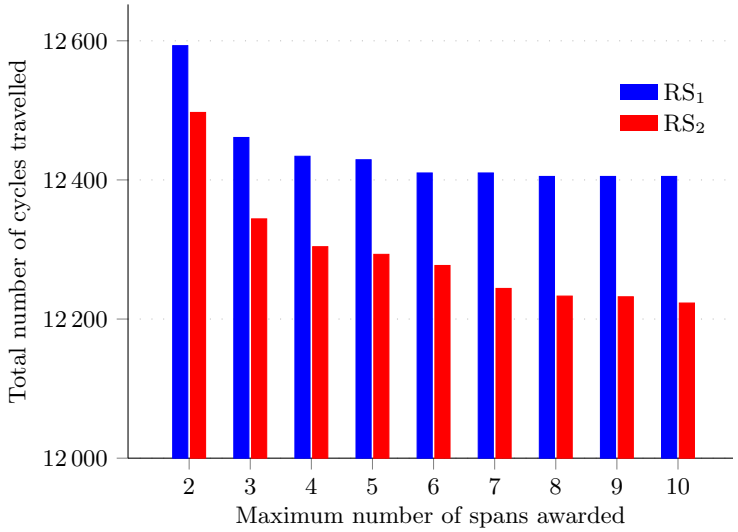


Figure 4.9: *The trade-off between the number of cycles travelled and number of possible spans awarded to each order, when solving the RS_1 and RS_2 when varying the lengths of the preference list.*

RS_1		RS_2	
Preference list	Total cycles	Preference list	Total cycles
$\{0.6\}$	12663	$\{0.6\}$	12601
$\{0.7\}$	12683	$\{0.5\}$	12608
$\{0.8\}$	12710	$\{0.4\}$	12615
$\{0.5\}$	12721	$\{0.7\}$	12616
$\{0.4\}$	12747	$\{0.3\}$	12624
$\{0.3\}$	12782	$\{0.6, 0.5\}$	12628
$\{0.9\}$	12787	$\{0.6, 0.4\}$	12634
$\{0.7, 0.5\}$	12795	$\{0.7, 0.5\}$	12642
$\{0.6, 0.5\}$	12795	$\{0.8, 0.4\}$	12643
$\{0.8, 0.5\}$	12807	$\{0.5, 0.4\}$	12647

Table 4.8: *The preference lists that yield the best solution for all 22 data sets considered. Results are indicated in the number of cycles travelled when the preference list is used. Results are displayed for both criteria RS_1 and RS_2. The numbers in brackets indicate the length of the maximum length of the preference list. The number of orders (O) and locations (L) are displayed for each data set.*

The assignment problem must, however, be adapted for the situations in which an order k would be assigned to order ℓ starting at the location associated with its p^{th} best preference span. In this case order ℓ must start at the location associated with its q^{th} best span.

Since each order may request a different set of SKUs to pick, each order may have a different number of preference spans. Each order has a number of preference spans equal to the number of various SKUs requested by that order. Assume that a picking line contains n orders and $Q+1$ preference spans are considered for each order. Let $c_{np+k,nq+\ell}$ be the distance to pick order k on its p^{th} best span, and to then travel to the location preceding the starting location of the q^{th} best span of order ℓ. Also let $x_{np+k,nq+\ell}$ be equal to 1 if order k starts picking on its p^{th} best span, is assigned to the location preceding the starting location of the q^{th} best span of order ℓ, and 0 otherwise.

Let $c_{np+k,nq+k} = M$, where M is a very large number (relative to the actual distances in the problem). Setting $c_{np+k,nq+k} = M$ ensures that we will not go to order i immediately after leaving order i. If an order contains fewer products than the value of $Q+1$, the corresponding

entries in the cost matrix is also set to M.

Then the generalised assignment problem (GAP) may be formulated to

$$\text{minimise} \sum_{k=1}^{n}\sum_{\ell=1}^{n}\sum_{p=0}^{Q}\sum_{q=0}^{Q} c_{np+k,nq+\ell} \cdot x_{np+k,nq+k} \tag{4.23}$$

subject to

$$\sum_{k=1}^{n}\sum_{p=0}^{Q}\sum_{q=0}^{Q} x_{np+\ell,nq+k} = 1 \qquad \ell = 1,2,\ldots,n, \tag{4.24}$$

$$\sum_{\ell=1}^{n}\sum_{p=0}^{Q}\sum_{q=0}^{Q} x_{np+k,nq+\ell} = 1 \qquad k = 1,2,\ldots,n, \tag{4.25}$$

$$\sum_{\ell=1}^{n}\sum_{q=0}^{Q} x_{np+k,nq+\ell} = \sum_{\ell=1}^{n}\sum_{q=0}^{Q} x_{nq+\ell,np+k} \qquad \begin{array}{l} k = 1,2,\ldots,n, \\ p = 0,1,\ldots,Q, \end{array} \tag{4.26}$$

$$x_{np+k,nq+\ell} \in \{0,1\} \qquad \begin{array}{l} k = 1,2,\ldots,n, \\ \ell = 1,2,\ldots,n, \\ p = 0,1,\ldots,Q, \\ q = 0,1,\ldots,Q. \end{array} \tag{4.27}$$

The objective function (4.23) gives the total length of the assignments. The constraint set (4.24) ensures that each order is assigned to following order only once. The constraints set (4.25) ensures that each order is assigned to a prior order only once. The cost matrix is displayed in Figure 4.10. The constraints in (4.26) ensures that if a subtour is generated, the beginning of the subtour is *connected* to the end of the subtour. This implies that all the ending locations of an order is connected to the starting position of another order.

This problem contains n^2Q^2 variables and $2n + nQ$ constraints. Due to the size of the problem and the size of data required as input variables, this problem could not be solved for typical sized problems faced by Pep, when using LINGO 11 [73].

Figure 4.11 represents a general cost matrix to solve the GAP for a problem containing three orders ($n = 3$), where a maximum of three preference spans may be selected ($Q = 2$). Constraint set (4.24) ensures that only one element is selected for each row $k = 1$, $k = 2$ and $k = 3$. Each order may only start on a single preference span. Similarly, constraint set (4.25) ensures that only one element is selected for each column $\ell = 1$, $\ell = 2$ and $\ell = 3$. Each order may be preceded by a single order. Assume c_{15} is selected (indicated in red in Figure 4.11). This means that order 1 is picked on its shortest preference span and then moves to the starting location of order 2, if order 2 starts on its second shortest preference span. Constraint set (4.25) ensures that order 2 starts on its second shortest preference span (*i.e.* one element c_{5x} is selected for a value of x where $x \in \{1,2,\ldots,9\}$ indicated in blue in Figure 4.11).

Due to the size of the problem, heuristic approaches are used to solve the GAP. A number of greedy heuristics are adapted to solve the assignment problem and are considered in the following subsection.

4.7.1 Heuristic solution methods to solve the generalised assignment problem

Six variations of a greedy heuristic are discussed in this section. In the first three variations the rows of the cost matrix are searched for a minimum entry in (respectively) a top-down fashion

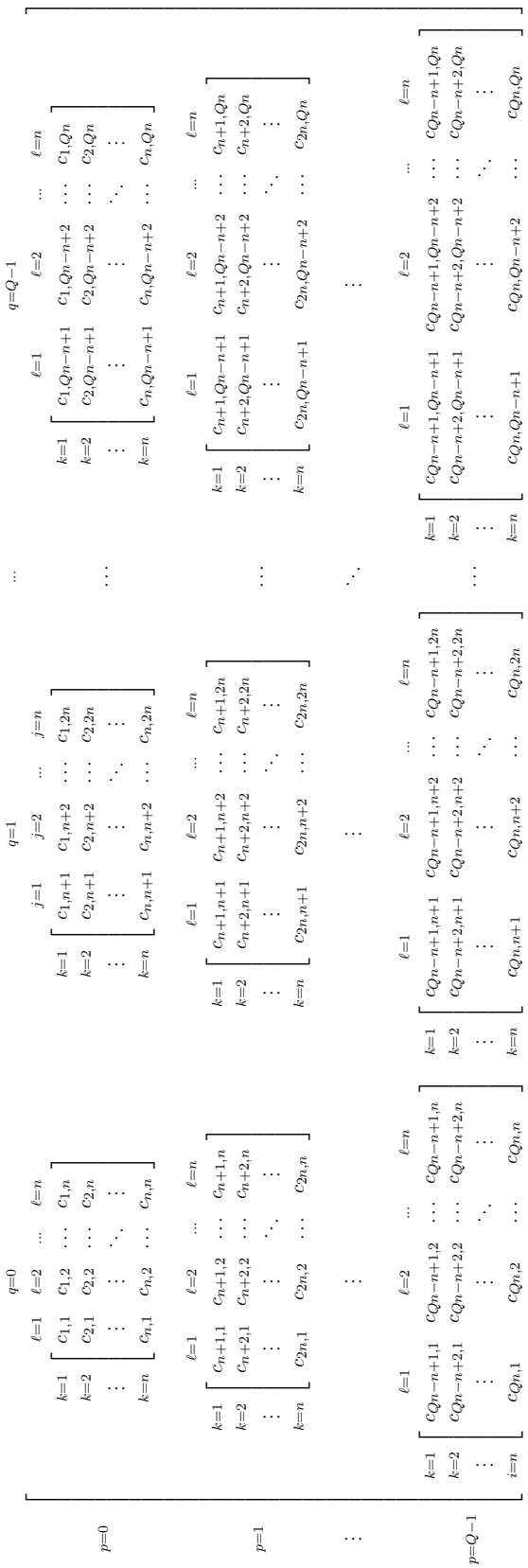


Figure 4.10: *The cost matrix used when solving the OSP using a generalised assigment problem.*

$$
\begin{array}{c}
 \\
p{=}0 \\
 \\
 \\
p{=}1 \\
 \\
 \\
p{=}2 \\
 \\
\end{array}
\begin{bmatrix}
\begin{array}{c} q{=}0 \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \end{array}
\begin{array}{c} q{=}1 \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{14} & c_{15} & c_{16} \\ c_{24} & c_{25} & c_{26} \\ c_{34} & c_{35} & c_{36} \end{bmatrix} \end{array}
\begin{array}{c} q{=}2 \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{17} & c_{18} & c_{19} \\ c_{27} & c_{28} & c_{29} \\ c_{37} & c_{38} & c_{39} \end{bmatrix} \end{array} \\
\begin{array}{c} \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{41} & c_{42} & c_{43} \\ c_{51} & c_{52} & c_{53} \\ c_{61} & c_{62} & c_{63} \end{bmatrix} \end{array}
\begin{array}{c} \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{44} & c_{45} & c_{46} \\ c_{54} & c_{55} & c_{56} \\ c_{64} & c_{65} & c_{66} \end{bmatrix} \end{array}
\begin{array}{c} \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{47} & c_{48} & c_{49} \\ c_{57} & c_{58} & c_{59} \\ c_{67} & c_{68} & c_{69} \end{bmatrix} \end{array} \\
\begin{array}{c} \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{71} & c_{72} & c_{73} \\ c_{81} & c_{82} & c_{83} \\ c_{91} & c_{92} & c_{93} \end{bmatrix} \end{array}
\begin{array}{c} \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{74} & c_{75} & c_{76} \\ c_{84} & c_{85} & c_{86} \\ c_{94} & c_{95} & c_{96} \end{bmatrix} \end{array}
\begin{array}{c} \\ \\ k{=}1 \\ k{=}2 \\ k{=}3 \end{array}
\begin{array}{c} \begin{array}{ccc} \ell{=}1 & \ell{=}2 & \ell{=}3 \end{array} \\ \begin{bmatrix} c_{77} & c_{78} & c_{79} \\ c_{87} & c_{88} & c_{89} \\ c_{97} & c_{98} & c_{99} \end{bmatrix} \end{array}
\end{bmatrix}
$$

Figure 4.11: *The cost matrix used to solve the OSP by means of the GAP for $n = 3$ and $Q = 2$.*

(*Greedy RTB*), bottom-up fashion (*Greedy RBT*) and randomly (*Greedy RR*). In the last three variations the columns of the cost matrix are searched for a minimum entry in (respectively) a left-to-right fashion (*Greedy CLR*), a right-to-left fashion (*Greedy CRL*) and randomly (*Greedy CR*). These heuristics were introduced in an unpublished report by Van Dieman [107]. Due to the nature of the generalised assignment problem, only these six variations heuristics are considered. These six variations were implemented for the classical assignment problem and may be modified for the generalised assignment problem.

The Greedy RTB starts with the first row and progresses in a top–down fashion. The mimimum entry c_{ij} in first row is selected and is recorded into a set $\mathcal{A}$. Now the row and column that corresponds to the minimum entry c_{ij} is deleted and this procedure is continued until all rows and columns are deleted at which time the set $\mathcal{A}$ has cardinality n, where n is the dimension of the problem. The objective function is obtained by summing up the entries in the set $\mathcal{A}$. The Greedy RBT and the Greedy RR works in a similar way, but the Greedy RBT begins with the last row and progresses to the top row and for the Greedy RR, the rows are permuted according to a permutation P.

Consider the example where the distance matrix is

$$
C = \begin{bmatrix} 5 & 4 & 5 & 2 & 3 \\ 6 & 4 & 3 & 5 & 3 \\ 2 & 1 & 6 & 5 & 4 \\ 0 & 2 & 4 & 5 & 6 \\ 6 & 0 & 4 & 2 & 3 \end{bmatrix}. \tag{4.28}
$$

The minimum entry in row 1 of the distance matrix is $c_{14} = 2$, row 1 and column 4 are eliminated (Figure 4.12 (a)). Now row 2 is examined, the minimum uncovered entries in row 2 are $c_{23} = 3$ and $c_{25} = 3$; c_{23} is chosen arbitrary, and row 2 and column 3 are eliminated (Figure 4.12 (b)). The next row to be examined is row 3 in which $c_{32} = 1$ is the minimum uncovered entry; row 3 and column 2 are therefore eliminated (Figure 4.12 (c)). The minimum uncovered entry in

row 4 is $c_{41} = 0$; row 4 and column 1 are thus eliminated (Figure 4.12 (d)). Now entry $c_{55} = 3$ is the only uncovered entry; row 5 and column 5 are therefore eliminated, since all the rows and columns have been eliminated the algorithm is terminated. The objective function value obtained by this algorithm is $z = 9$, which is optimal. Algorithm 6 displays the pseudo code of the Greedy RTB algorithm.

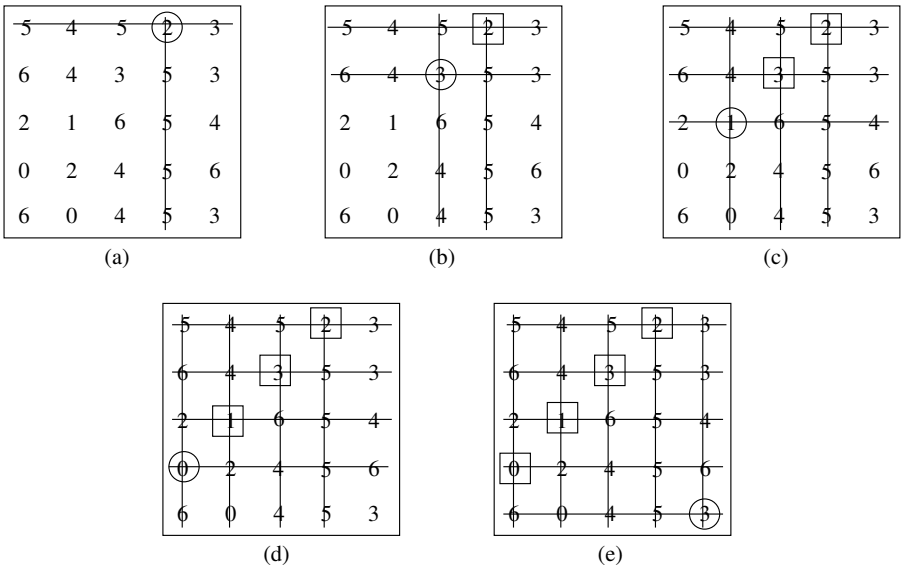


Figure 4.12: *Graphical illustration of the steps followed by the greedy RTB heuristic.*

Algorithm 6: Greedy RTB

Data: A $n \times n$ cost matrix C.
Result: A low cost assignment.
Let $i \leftarrow 1$, $\mathcal{A} = \emptyset$;
while $|\mathcal{A}| \neq n$ *and* $C \neq \boldsymbol{0}$ **do**
  Find minimum c_{ij} for $j = 1, \ldots, n$;
  $\mathcal{A} \leftarrow \mathcal{A} \cup c_{ij}$;
  Remove column j;
  $i \leftarrow i + 1$;
end
$z = \sum_{c_{ij} \in \mathcal{A}} c_{ij}$;

The Greedy RBT is similar to the Greedy RTB, however, it starts with the last row of the distance matrix as displayed in Figure 4.13. Algorithm 7 displays the pseudo code for the Greedy RBT. The Greedy RR uses a permutation P for the selection of the rows.

Consider the permutation $P = \{3, 1, 5, 4, 2\}$ for the selection of the rows. The first row to be examined is row 3 with a minimum uncovered entry, $c_{32} = 1$, thus row 3 and column 2 are eliminated (Figure 4.14 (a)). The minimum uncovered entry in row 1 is $c_{14} = 2$; and row 1 and column 4 are therefore eliminated (Figure 4.14 (b)). The next row to examine is row 5 with a minimum uncovered entry of $c_{55} = 3$; thus row 5 and column 5 are eliminated (Figure 4.14 (c)). Now the minimum entries for rows 4 and 2 is $c_{41} = 0$ and $c_{23} = 3$ respectively and rows 4 and 2 and columns 1 and 3 are eliminated (Figure 4.14 (d) and Figure 4.14 (e)). An objective function value of $z = 9$ (optimal solution) is obtained by the Greedy RR heuristic. Algorithm 8 displays the pseudo code of the Greedy RR algorithm.

In the next variations of the greedy heuristic the columns, instead of the rows, are examined.

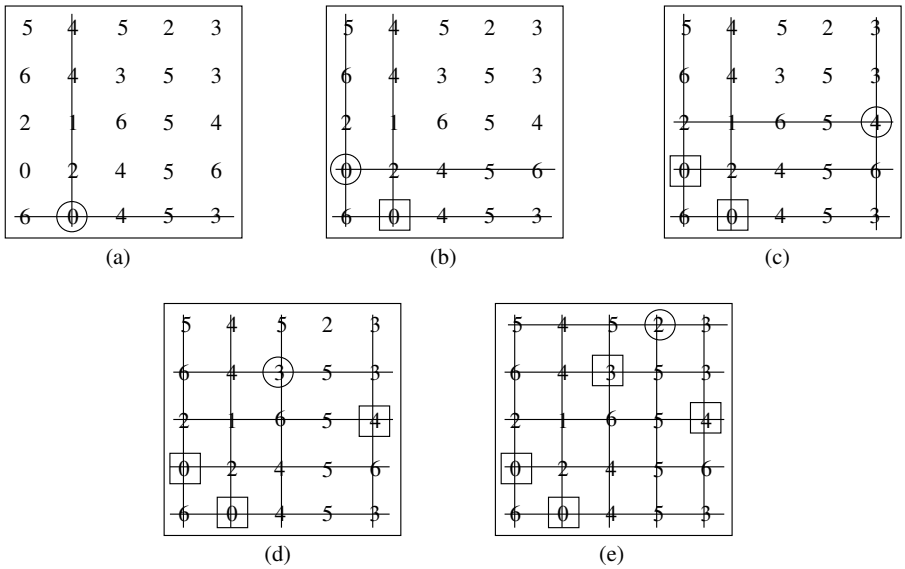


Figure 4.13: *Graphical illustration of the steps followed by the Greedy RBT heuristic.*

Algorithm 7: Greedy RBT

Data: A $n \times n$ cost matrix C.
Result: A low cost assignment.
Let $i \leftarrow n$, $\mathcal{A} = \emptyset$;
while $|\mathcal{A}| \neq n$ *and* $C \neq \boldsymbol{0}$ **do**
  Find minimum c_{ij} for $j = 1, \ldots, n$;
  $\mathcal{A} \leftarrow \mathcal{A} \cup c_{ij}$;
  Remove column j;
  $i \leftarrow i - 1$;
end
$z = \sum_{c_{ij} \in \mathcal{A}} c_{ij}$

The Greedy CLR starts with the first column and progresses in a left to right fashion. The minimum entry c_{ij} in each column is sorted and is recorded in a set $\mathcal{A}$. Now the row and column that corresponds to the minimum entry c_{ij} is deleted and this procedure is continued until all rows and columns are deleted at which time the set $\mathcal{A}$ has cardinality n, where n is the dimension of the problem. The objective function is obtained by summing up the entries in the set $\mathcal{A}$. The Greedy CRL and the Greedy CR works in a similar way, but the Greedy CRL begins with the last column and progresses from the most right column to the left and for the Greedy CR the columns is permuted according to a permutation P. The pseudo codes of these three variations may be found in Algorithms 9, 10 and 11.

The minimum entry in column 1 to 5 of the distance matrix is $c_{41} = 0$, $c_{52} = 0$, $c_{23} = 3$, $c_{14} = 2$ and $c_{35} = 4$ respectively and a objective function value of $z = 9$ (optimal solution) is obtained by Greedy CLR (Figure 4.15). The minimum entries in columns 5 to 1 in the distance matrix is $c_{15} = 3$, $c_{24} = 5$, $c_{43} = 4$, $c_{52} = 0$ and $c_{31} = 2$ respectively and an objective function value of $z = 14$ is obtained by Greedy CRL (Figure 4.16). Consider the permutation $P = \{2, 5, 1, 4, 3\}$. The first column to be selected by the Greedy CR heuristic will be chosen according to the permutation P. The minimum entries for columns 2, 5, 1, 4 and 3 are $c_{32} = 1$, $c_{15} = 3$, $c_{41} = 0$, $c_{24} = 5$ and $c_{53} = 4$ respectively and an objective function value of $z = 13$ is obtained (Figure 4.17).

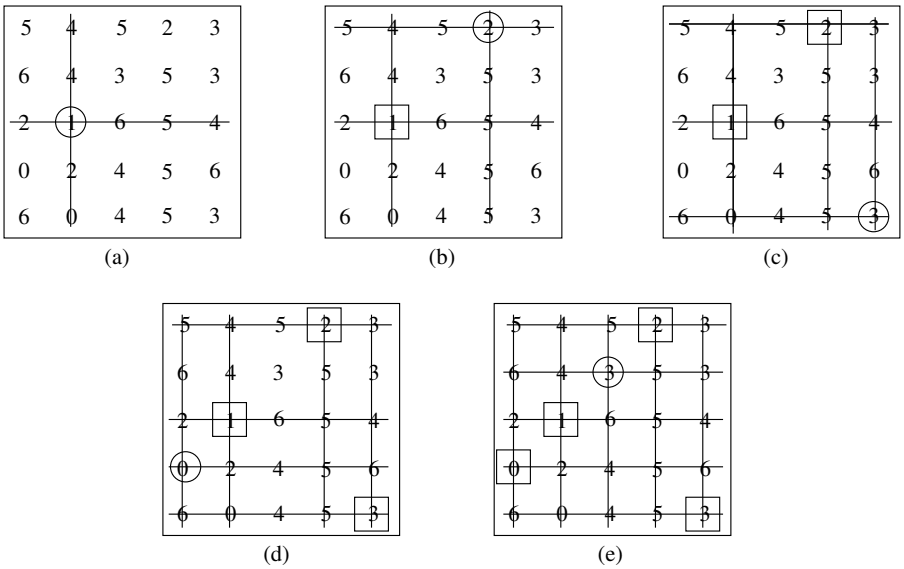


Figure 4.14: *Graphical illustration of the steps followed by the Greedy RR heuristic.*

Algorithm 8: Greedy RR

Data: A $n \times n$ cost matrix C.
Result: A low cost assignment.
Let $\mathcal{A} = \emptyset$ and $|\mathcal{A}| = 0$;
$j \leftarrow 1$;
Let P be a permutation of $\{1, \ldots, n\}$;
while $|\mathcal{A}| \neq n$ *and* $C \neq \boldsymbol{0}$ **do**
    Find minimum c_{ij} for $i \in P$ and $j = 1, \ldots, n$;
    $\mathcal{A} \leftarrow \mathcal{A} \cup c_{ij}$;
    $j \leftarrow j + 1$;
    $P \leftarrow P \setminus \{i\}$;
end
$z = \sum_{c_{ij} \in \mathcal{A}} c_{ij}$;

The elementary operations of the greedy heuristic is scan a row/column and delete a row and column. The maximum number of steps required is $2n$ (n rows/columns must be scanned and n rows and columns must be deleted), thus the time complexity of the greedy heuristic is $\mathcal{O}(n)$.

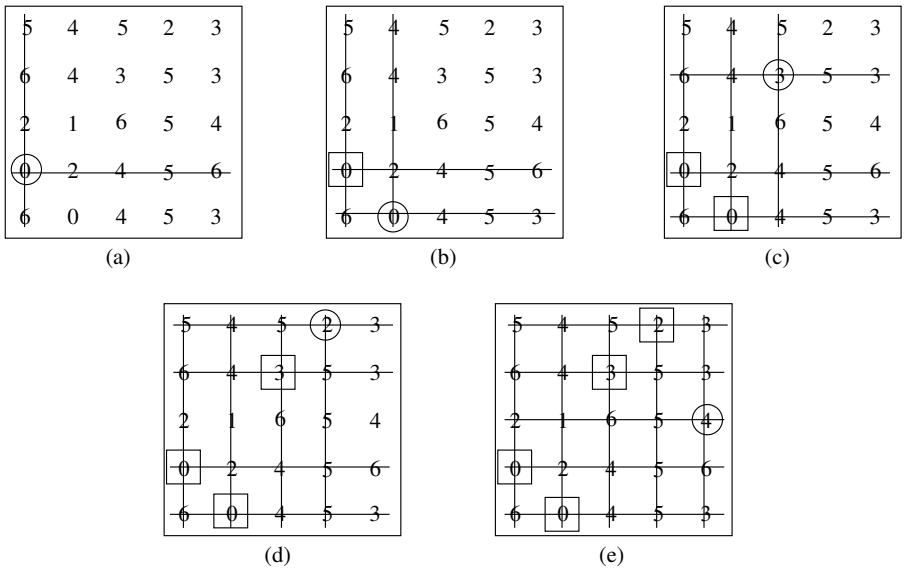


Figure 4.15: *Graphical illustration of the steps followed by the Greedy CLR heuristic.*

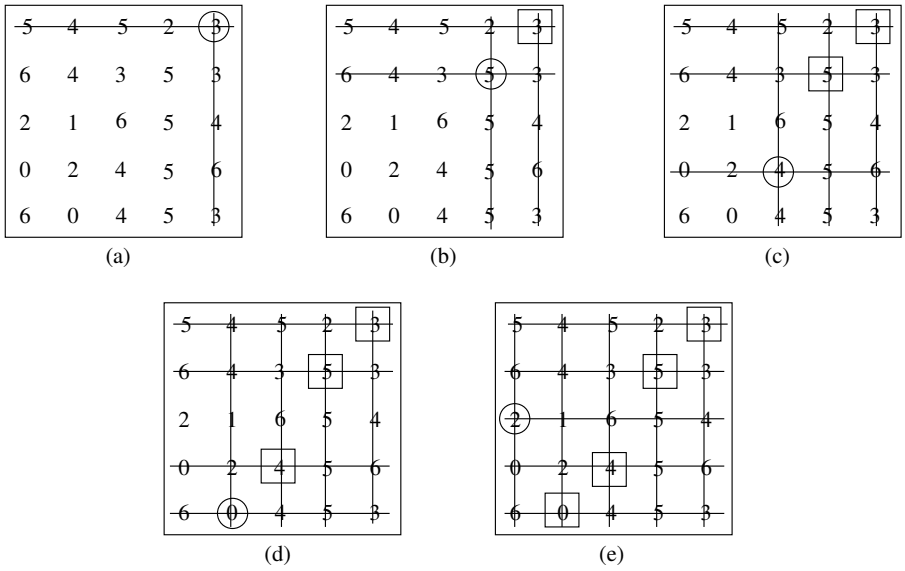


Figure 4.16: *Graphical illustration of the steps followed by the Greedy CRL heuristic.*

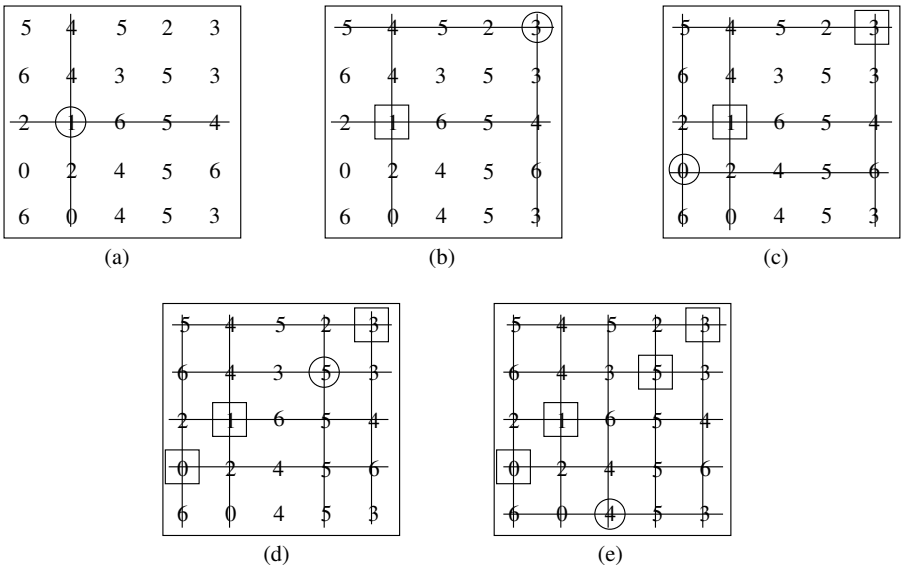


Figure 4.17: *Graphical illustration of the steps followed by the Greedy CR heuristic.*

Algorithm 9: Greedy CLR

Data: A $n \times n$ assignment of minimum cost.
Result: A low cost assignment.
Let $\mathcal{A} = \emptyset$;
$j \leftarrow 1$;
while $|\mathcal{A}| \neq n$ *and* $C \neq 0$ **do**
    Find minimum c_{ij} for $i = 1, \ldots, n$;
    $\mathcal{A} \leftarrow \mathcal{A} \cup c_{ij}$;
    Remove row i $j \leftarrow j + 1$;
end
$z = \sum_{c_{ij} \in \mathcal{A}} c_{ij}$

4.7.2 Implementation of the modified assigment problem

The heuristic approaches to solving the assigment problem in §4.7.1 may be used to solve the generalised assignment problem, where each order may contain a number of possible starting locations.

When implementing the assignment problem, orders may be connected in such a way that subtours are created. If a number of subtours are generated, a new assignment problem may be constructed where each subtour contains only one starting location and one ending location. The starting and ending location may be determined by considering the most number of locations travelled to connect any two orders in a single subtour. Consider the example in Figure 4.18 where three subtours exist, each subtour containing four orders.

Within the first subtour, the longest arc connecting any two orders, is the arc connecting order 8 and 3. The starting position of this subtour starts at the location where order 8 starts. Similarly, the ending position of this subtour ends at the location where order 3 ends. The arc connecting order 8 and 3 is then removed.

This process may be implemented for each subtour. Each subtour will contain a unique span, with a single starting and ending location. A classical assignment problem may be implemented to connect subtours.

Algorithm 10: Greedy CRL

Data: A $n \times n$ assignment of minimum cost.
Result: A low cost assignment.
Let $\mathcal{A} = \emptyset$;
$j \leftarrow n$;
while $|\mathcal{A}| \neq n$ *and* $C \neq \boldsymbol{0}$ **do**
    Find minimum c_{ij} for $i = 1, \ldots, n$;
    $\mathcal{A} \leftarrow \mathcal{A} \cup c_{ij}$;
    Remove row i;
    $j \leftarrow j - 1$;
end
$z = \sum_{c_{ij} \in \mathcal{A}} c_{ij}$

Algorithm 11: Greedy CR

Data: A $n \times n$ cost matrix C.
Result: A low cost assignment.
Let $\mathcal{A} = \emptyset$;
$j \leftarrow 0$;
Let P be a permutation of $\{1, \ldots, n\}$;
while $|\mathcal{A}| \neq n$ *and* $C \neq \boldsymbol{0}$ **do**
    Find minimum c_{ij} for $j \in P$ and $i = 1, \ldots, n$;
    $\mathcal{A} \leftarrow \mathcal{A} \cup c_{ij}$;
    Remove row i;
    $P \leftarrow P \setminus \{j\}$;
end
$z = \sum_{c_{ij} \in \mathcal{A}} c_{ij}$

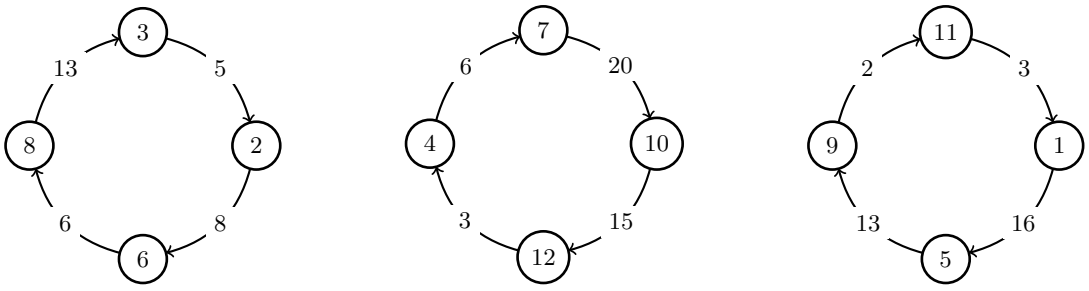


Figure 4.18: *An example of three subtours that have to be connected. Each subtour contains four orders. The connecting cost is indicated on each arc. The connecting cost is the number of locations travelled from the ending location of an order to the starting location of the following order.*

If the cost matrix for the example is

$$\boldsymbol{c} = \begin{bmatrix} M & 4 & 16 \\ 10 & M & 7 \\ 3 & 5 & M \end{bmatrix},$$

the result would be to connect subtour 1 to subtour 2, subtour 2 to subtour 3 and subtour 3 to subtour 1, resulting in a single tour. Figure 4.19 displays the final tour containing all the subtours. The number of locations traversed in this tour is used to determine the number of cycles that may be travelled by a picker.

If the classical assignment problem does not create a single subtour, the longest arc within each subtour is removed, and a unique starting and ending location is calculated for each subtour.

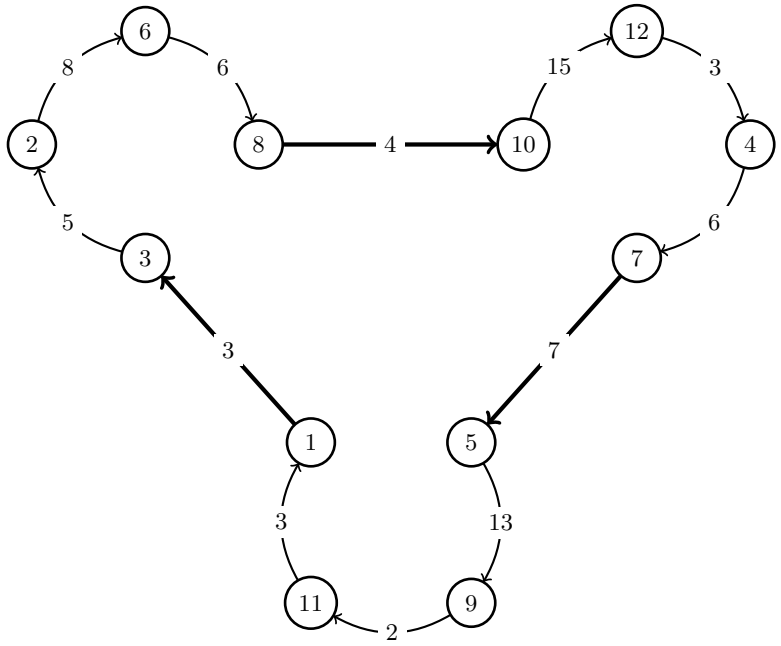


Figure 4.19: *The manner in which three subtours may be connected to form a single subtour. The arcs in bold connect two orders from different between subtours. The connecting cost is indicated on each arc. The connecting cost is the number of locations travelled from the ending location of an order to the starting location of the following order.*

The classical assignment problem is used once more to generate a single tour. This process is iteratively implemented until a single tour is generated.

All the greedy heuristics used in §4.7.1 are used to calculate a single tour containing all the orders assigned to a picking line. Each greedy heuristic is used to solve the generalised assignment problem as well as the classical assignment problem for each data set.

4.7.3 Results of the generalised assignment problem

All the data sets are considered for implementation of the generalised assignment problem. The solution techniques of the generalised assignment problem are tested by varying the number of possible spans assigned to all the orders within a picking line. Figure 4.20 displays number of cycles travelled for the generalised assignment problem when the 10 best possible preference spans of each order are considered when all 22 data sets are considered. The best results for the RBT, RR, CLR, CRL and CR may be achieved when only two preference spans are awarded to each order. The best results for the RTB heuristic is achieved when only a single preference span is awarded to each order. The results for the RR heuristic is poor compared to the other heuristics for any number of preference spans considered.

The results for each individual data set is considered when the number of preference spans are fixed at two. All of the six greedy heuristics used to solve the GAP provided the best average solutions when two preference spans were used. Table 4.9 displays the results for each data set when only two preference spans are considered for each order.

Due to implementation issues a maximum of 10 possible spans for each order is considered. The greedy heuristics used to solve the generalised assignment problem is coded in JAVA [101], which is unable to process cost matrices for average sized problems ($n \approx 1\,200$) when the number of

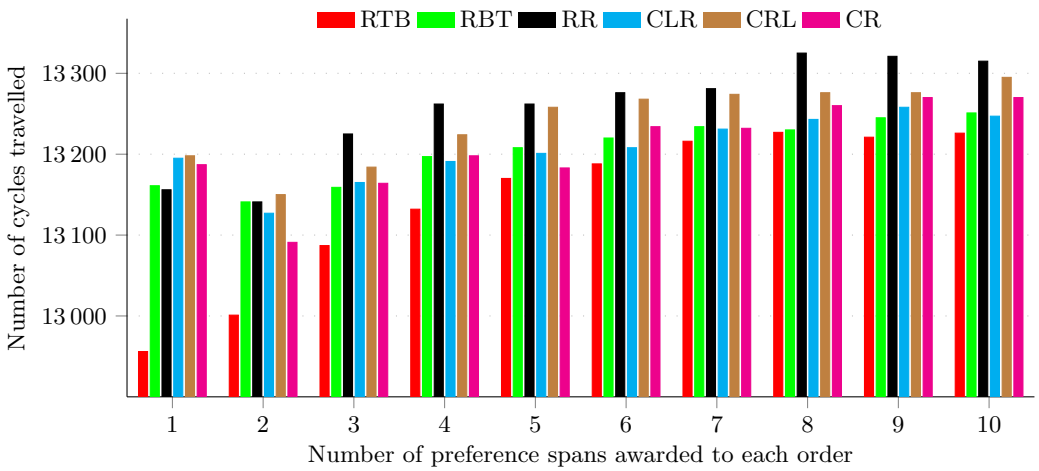


Figure 4.20: *A bar chart displaying the results (cycles travelled) obtained for all of the 22 data sets considered when the number of preference spans used in the generalised assignment problem is varied for values ranging from 1 to 10. Results from six heuristics used to solve the GAP include the RTB, RBT, RR, CLR, CRL and CR heuristics.*

possible spans exceeds 10. The dimensions of the cost matrix in this case is 12 000 × 12 000, which equates to 144 000 000 entries in this matrix.

Table 4.10 displays the computational times for the results displayed in Table 4.9. The times are similar for all the heuristic algorithms considered. The algorithms where random sequences (*Greedy RR* and *Greedy CR*) use slightly more computational time to solve any data set considered.

4.8 Metaheuristic approaches to order sequencing

Metaheuristics are procedures that coordinate simple heuristics and rules to find good (and even optimal) solutions to computationally difficult optimisation problems [2]. Metaheuristics are among the most effective solution strategies for solving combinatorial optimisation problems in practice and have been applied to a wide variety of problems [19]. The general objective of metaheuristics is to effectively and efficiently explore the solution space [92].

Two important components in metaheuristics are intensification and diversification strategies [45, 92]. Intensification is the exploitation of the accumulated search experience — when concentrating the search effort in a confined, small area of the solution space. Intensification aims to explore the "most promising" areas in the solution space to ensure that the best solutions in these areas are indeed found [41]. Intensification procedures are based on some *intermediate-term memory* which evaluates the number of consecutive iterations that certain components in the solution space have been present without interruption. The solution space that contains these components may be explored thoroughly in order to improve the solution quality.

Diversification is an exploration into the larger unexplored regions of the solution space. It is essentially an algorithmic mechanism that forces the solution space into previously unexplored areas of the solution space [41]. *Long-term memory* may be used to evaluate the total number of iterations that various solution components have been involved in the current solution. Diversification aims to essentially restart the search from an area currently unexplored within the solution space.

Data set	Size (O, L)	Maximal cut approach	RTB	RBT	RR	CLR	CRL	CR
A	(1262,49)	1232	1243	1237	1241	**1235**	1237	1240
B	(1264,54)	1226	**1257**	1269	1266	1263	1271	1270
C	(1265,51)	1161	1231	1222	1228	1209	**1208**	1209
D	(1263,56)	1072	1218	1211	1205	1202	1201	**1197**
E	(1264,51)	1069	1176	1175	1184	1176	1176	**1173**
F	(1258,55)	1025	**1164**	1191	1189	1180	1186	1183
G	(1258,53)	1005	1153	1182	1186	1157	**1147**	1157
H	(1244,54)	992	**1114**	1119	**1114**	1128	1132	1121
I	(1260,56)	955	**1094**	1099	1116	1096	1113	**1094**
J	(1264,56)	947	**1041**	1091	1075	1049	1060	1050
K	(943,63)	259	325	**304**	312	337	348	334
L	(846,56)	232	**265**	278	268	284	276	278
M	(728,51)	152	**200**	214	218	211	213	217
N	(733,55)	125	154	**152**	160	182	185	170
O	(396,63)	90	148	**131**	132	137	135	132
P	(574,48)	80	**102**	113	111	112	112	113
Q	(242,64)	45	**55**	66	62	89	68	70
R	(158,55)	14	**19**	33	23	23	24	27
S	(89,42)	9	14	15	**11**	15	16	15
T	(82,51)	8	**9**	15	16	17	15	15
U	(90,48)	7	**11**	14	14	14	16	13
V	(80,56)	6	**8**	10	10	11	11	13
Total		11711	13001	13141	13141	13127	13150	13091

Table 4.9: *The number of cycles travelled for the generalised assignment problem when the 2 best possible spans of each order are considered. Results from each of the six greedy heuristics are considered, namely the Greedy RTB, Greedy RBT, Greedy RR, Greedy CLR, Greedy CRL and the Greedy CR heuristics. The heuristic that performs the best for each data set is indicated in boldface. The number of orders (O) and locations (L) are displayed for each data set.*

A tabu search, simulated annealing, genetic algorithm and an extremal optimisation approach is presented to address the OSP. The OSP may be simplified by considering the preference spans of various orders and electing one of these spans for each order. The subtour generation heuristic in §4.3.2 may then be used to create a set of subtours that may be linked with at most one additional cycle travelled.

4.8.1 Data structure and improvement of a solution

The size of the problem may be reduced by only considering the preference spans of an order. When an order is not started on a preference span a number of locations are traversed that does contribute to the picking process in that order. In this case more distance is travelled for an order than may actually be required. The subtour generation heuristic may be used to obtain a feasible solution that may alter the spans of orders to deliver a feasible tour.

The metaheuristics considered to solve the OSP are modelled by assigning a starting location to each order. The length of the preference span for each order may be calculated when the starting location is known for each order. The cut at each location is then determined. The maximal cut(s) are also identified.

A number of situations may present itself where improvement may be achieved. To reduce the maximal cut(s), orders that pass the maximal cut(s) but do not pick from that/those locations are selected to possibly reduce the maximal cut(s).

Data set	Size (O, L)	RTB	RBT	RR	CLR	CRL	CR
A	(1262,49)	780	884	851	651	700	908
B	(1264,54)	432	401	461	348	369	473
C	(1265,51)	313	308	357	280	303	333
D	(1263,56)	328	329	374	282	296	330
E	(1264,51)	342	350	388	295	310	343
F	(1258,55)	289	293	344	245	250	295
G	(1258,53)	313	310	346	281	302	327
H	(1244,54)	311	314	367	270	287	324
I	(1260,56)	388	392	440	356	376	400
J	(1264,56)	315	348	367	276	305	318
K	(943,63)	139	144	159	132	136	147
L	(846,56)	132	127	137	115	110	132
M	(728,51)	78	78	81	70	70	76
N	(733,55)	78	74	74	63	70	86
O	(396,63)	103	78	79	23	24	24
P	(574,48)	39	43	44	93	83	100
Q	(242,64)	13	13	13	11	11	11
R	(158,55)	6	8	7	7	7	6
S	(89,42)	7	4	4	4	5	4
T	(82,51)	5	6	5	5	5	5
U	(90,48)	4	6	5	4	7	5
V	(80,56)	5	4	5	5	6	12
Total		4420	4514	4908	3816	4032	4659

Table 4.10: *Computational times in seconds for the generalised assignment problem when the 2 best possible spans of each order are considered. Results from each of the six greedy heuristics are considered, namely the Greedy RTB, Greedy RBT, Greedy RR, Greedy CLR, Greedy CRL and the Greedy CR heuristics. The number of orders (O) and locations (L) are displayed for each data set.*

An example is displayed in Figure 4.21 where three orders are considered. Each location contained on the preference span of the order is representated by either a filled circle or square. Suppose that a filled square represents a location that the order has to pick from, while a filled circle represents a location passed by the order which is not picked from. The first order ends at location i, while the second order starts at location i. The third order starts at location $i-1$ and ends at location $i+1$. Assume that the current maximal cut is located at location i. Since orders 1 and 2 have to pick from location i, altering the spans of these orders will not decrease the maximal cut C. However, order 3 passes location i without picking from it. By altering the preference span of order 3 in such a way that the preference span starts at location $i+1$ and ends at location $i-1$, the maximal cut may be reduced if any location preceding location $i-1$ and any location preceding location $i+1$ does not contain a cut $C-1$ or more. If such an order exists, the span of the order is altered and the maximal cut is decreased.

This altering of the preference span of an order, where the maximal cut of a picking line is reduced may be referred to as a *reducing move*. Whenever a reducing move is performed the solution quality is improved by reducing the total number of cycles travelled by one pick cycle. Figure 4.22 displays the resulting picking line configuration when the preference span of order 3 is changed. Each of the locations on the section of the illustrated picking line now contains a cut of $C-1$.

If no improvement may be incurred, by altering preference spans of the orders to minimise the maximal cut(s), an effort may be made to change the location(s) containing the maximal cut(s). Figure 4.23 displays a situation in which the maximal cut cannot be reduced. It is, however,

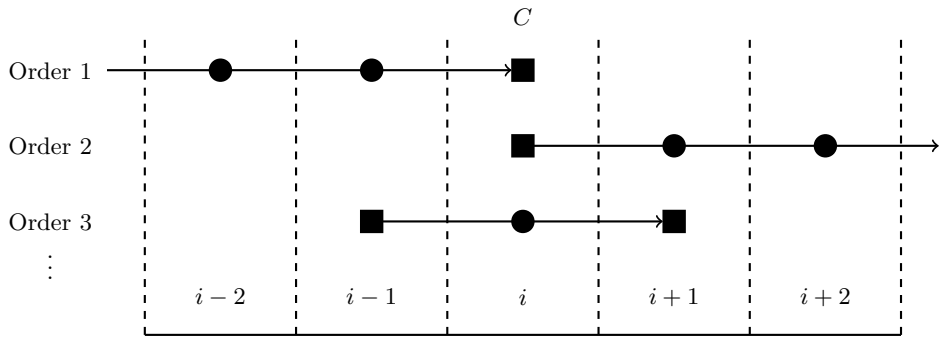


Figure 4.21: *Example of an order that may reduce the maximal cut when a different span is selected. Only three orders are displayed, namely* 1, 2 *and* 3. *The locations are indicated by* $i-2$, $i-1$, i, $i+1$ *and* $i+2$, *which represents a section of a picking line. Filled squares indicate locations where orders have to pick, while filled circles indicate locations that are traversed without being picked from. The maximal cut is located at location* i. *The arrows indicate the current spans of each order.*

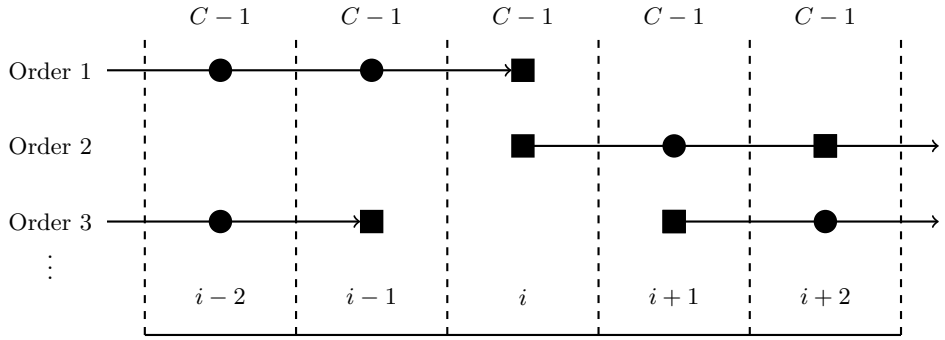


Figure 4.22: *Example of when the span of an order is altered that reduces the maximal cut. Only three orders are displayed, namely* 1, 2 *and* 3. *The locations are indicated by* $i-2$, $i-1$, i, $i+1$ *and* $i+2$, *which represents a section of a picking line. Filled squares indicate locations where orders have to pick, while filled circles indicate locations that are traversed without being picked from. The maximal cut is located at all the locations considered on the section of the picking line displayed. The arrows indicate the current spans of each order.*

possible to alter the location(s) containing the maximal cut(s). Order 4 starts order picking at location $i-2$ and ends at location $i+1$. Assume that order 4 only has to visit locations $i-2$ and $i+1$. By changing the preference span of this order, to start at location $i+1$ and end at location $i-2$, the number of locations containing the maximal cut(s) are reduced and changed if all the locations from $i+1$ to $i-2$ contains a cut of $C-1$ or less. Before the alteration, locations $i-1$ and i contained the maximal cut C, but when the preference span of order 4 is changed, only location $i+2$ contains a maximal cut. These alterations act as intermediate actions that may contribute to decreasing the maximal cut. This move may be referred to as a *shifting move*. Figure 4.24 displays the situation in which the span of order 4 is altered to reduce the number of maximal cut locations as presented in Figure 4.23.

Care should be taken not to increase the maximal cut if some of the locations have a cut of $C-1$. The locations containing a cut of one less than the maximal cut $(C-1)$ must also be examined. Figure 4.25 displays a situation which may seem similar to the situation in Figure 4.21. Four orders are present in the picking line. The current maximal cut C is at location $i-1$. Orders 1 and 2 have to pick from location $i-1$, however, order 4 does not require a SKU from location $i-1$. It may seem beneficial to alter the preference span of order 4 by starting at location i and ending at location $i-2$, decreasing the cut at location $i-1$ to $C-1$, but the cut is increased

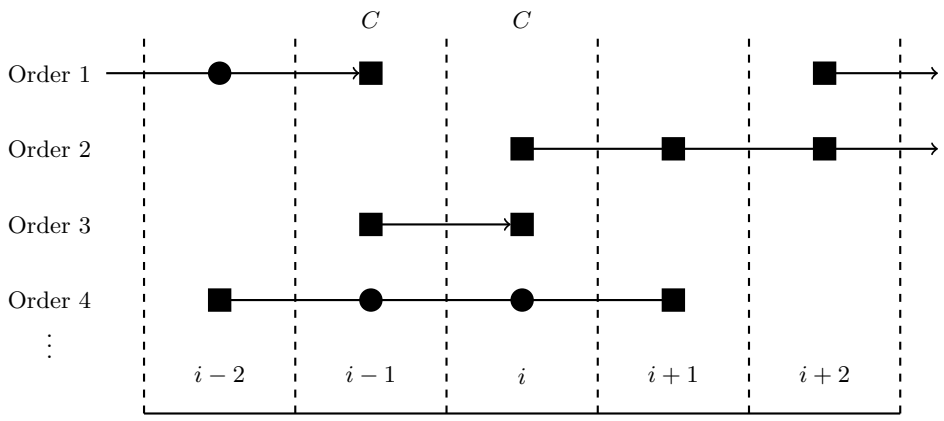


Figure 4.23: *Example of an order that may shift and reduce the number of maximal cuts when a different span is selected. Only four orders are displayed, namely 1, 2, 3 and 4. The locations are indicated by $i-2$, $i-1$, i, $i+1$ and $i+2$, which represents a section of a picking line. Filled squares indicate locations where orders have to pick, while filled circles indicate locations that are traversed without being picked from. The maximal cuts are located at location $i-1$ and i. The arrows indicate the current spans of each order.*

at locations $i+1$ and $i+2$ as shown in Figure 4.26.

These three considerations have to be considered when metaheuristic procedures are investigated to solve the OSP. The next section will present a number of metaheuristics used to solve the OSP by assigning a starting location to each order.

4.8.2 Greedy starting heuristic

A greedy heuristic to reduce the maximal cut(s) is initialised by iteratively assigning starting locations to all the orders on a picking line. Only preference spans of an order are considered. When a starting location is awarded to an order, the maximal cut(s) may be calculated. The starting location of the following order is chosen to minimise the maximal cut and furthermore reduce the number of maximal cuts as far as possible. The first criteria is to reduce the maximal cut, followed by reducing the number of maximal cuts.

When all the orders on a picking line have received a starting location, the subtour generation heuristic is used to create a feasible solution. The subtour generation heuristic may effectively alter the starting locations of some orders. However, each order will still be picked on its preference span identified by the greedy maximal cut(s) reducing heuristic.

Due to the lack of connectivity between orders, the solution structure moves away from an $n \times n$ connectivity matrix (connecting orders to one another) to an $n \times m$ assignment matrix (assigning a starting location to an order). In addition, the sequencing of orders are done once each order have been awarded a starting location associated with a preference span.

Let C represent the maximal cut for a given configuration, where the preference spans are awarded to orders in a picking line, and N_C be the number of locations containing the maximal cut. Let the solution structure be a vector $\boldsymbol{s}$ of starting locations, where the k^{th} element s_k of the vector $\boldsymbol{s}$ represents the starting location of order k, with respect to a single preference span of order k. From a given solution $\boldsymbol{s}$ both C and N_C may be calculated.

The greedy heuristic in Algorithm 12 may be used to find good starting solutions for metaheuristics in the following sections.

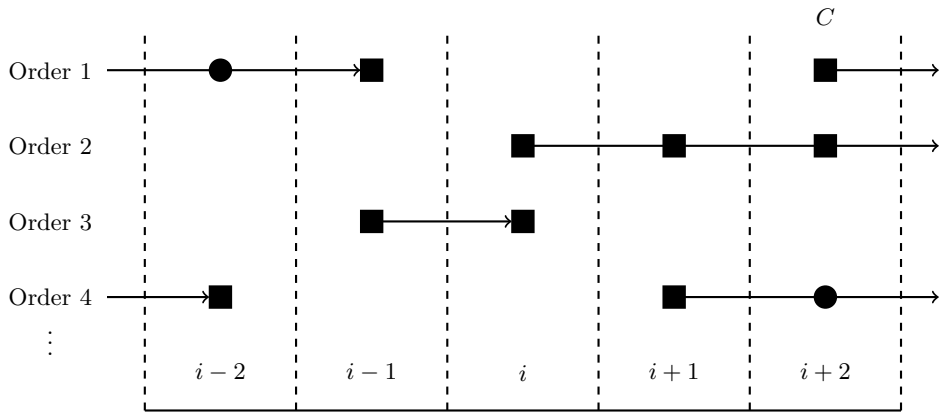


Figure 4.24: *Example of an order that shifts and reduces the number of maximal cuts when the preference span is altered. Only four orders are displayed, namely* 1, 2, *3 and* 4. *The locations are indicated by* $i-2$, $i-1$, i, $i+1$ *and* $i+2$, *which represents a section of a picking line. Filled squares indicate locations where orders have to pick, while filled circles indicate locations that are traversed without being picked from. The maximal cut is located at location* $i+2$. *The arrows indicate the current spans of each order.*

Algorithm 12: Greedy maximal cut reducing heuristic.

Data: A set of all the order $\mathcal{S}$. All the preference spans for each order.
Result: A vector $\boldsymbol{s}$ of starting locations awarded to each individual order.
Let $C \leftarrow 0$ and $N_C \leftarrow 0$;
while $\mathcal{S} \neq \emptyset$ **do**
  Randomly select an order k such that $k \in \mathcal{S}$;
  if *the selected order contains a preference span that will not increase the current value of* C **then**
    Assign that starting location to the k^{th} entry in vector $\boldsymbol{s}$;
  end
  else
    Select a preference span of order k that results in the lowest value of N_C;
  end
  Update the value of C and N_C;
  $\mathcal{S} \leftarrow \mathcal{S} \backslash k$;
end

4.8.3 Tabu search

Tabu search (TS) was first introduced by Fred Glover in an article in 1986 [44]. Although many of the ideas that Glover used in tabu search was suggested earlier, he proposed the majority of the principles we currently use. The fundamental principle that distinguishes tabu search from the local search technique, is that tabu search incorporates "intelligence". The tabu search directs an iterative search in a "good" direction so that the search for the optimal solution is not determined solely by chance.

Implementing tabu search techniques poses two challenges. The first challenge is to create an effective search engine. The search engine is responsible for evaluating neighbourhood solutions. Secondly, all pieces of knowledge available should be included to the search procedure, as to avoid bad regions of the solution space. With this in mind, the search procedure should be directed while avoiding the mistake of repeating solutions already visited.

Tabu search is based on the premise of solving a problem intelligently. For this to be possible *adaptive memory* and *responsive exploration* is necessary. Adaptive memory refers to the property that the solution space is searched economically and effectively. Local choices are thus

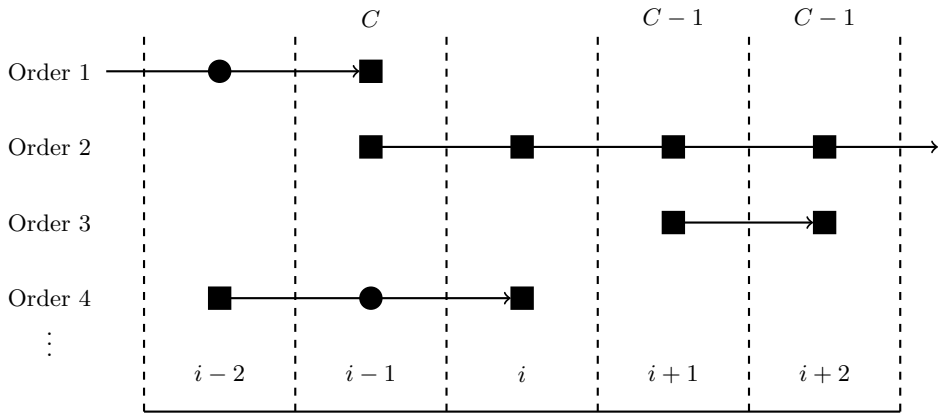


Figure 4.25: *Example of an order that may be altered to increase the number of maximal cuts when a different span is selected. Four orders are displayed, namely* 1, 2, 3 *and* 4. *The locations are indicated by* $i-2$, $i-1$, i, $i+1$ *and* $i+2$, *which represents a section of a picking line. Filled squares indicate locations where orders have to pick, while filled circles indicate locations that are traversed without being picked from. The maximal cut is located at location* $i-1$. *The arrows indicate the current spans of each order.*

guided by information that are retrieved during the search. This is in contrast with a genetic algorithm search which incorporates a "memory-less" search.

An emphasis on responsive exploration is derived from the notion that a strategic choice that yields a poor result contributes more information than a random choice that yields a good result. In a system that incorporates memory, a bad strategic choice may give hints that may be used to alter the strategy profitably. Responsive exploration is the backbone of intelligent search in that the good solutions always surface, while the solution space is simultaneously explored.

In most tabu search problems the problem may be expressed as optimising (maximising or minimising) a function $f(\boldsymbol{s})$ subject to $\boldsymbol{s} \in \mathcal{S}$, where the set $\mathcal{S}$ summarises constraints on the vector of decision variables $\boldsymbol{s}$ [92]. It must be noticed that not all tabu search problems can be formulated in this way. Tabu search begins in the same way as a local search method, by utilising an iterative search from one point to another until a specified termination criterion is reached. Here each $\boldsymbol{s} \in \mathcal{S}$ has a neighbourhood $N(\boldsymbol{s}) \in N$. Each new solution $\boldsymbol{s}' \in N(\boldsymbol{s})$ that is reached from $\boldsymbol{s}$, is called a *move*. It is often useful to choose an $\boldsymbol{s}'$ at each iteration that yields a "good" result of $f(\boldsymbol{s}')$ [34].

Tabu search has to make an "intelligent" choice from the solutions in $N(\boldsymbol{s})$. Reducing the size of the neighbourhood is a good way to speed up the process of evaluating the neighbourhood. This might also direct the search. This may be implemented in the form of a data structure known as a *candidate list*. Here it should be noticed that a good quality move will remain good for solutions not too different. This may be implemented by ordering the entire set of feasible moves by decreasing quality, in a given iteration. During the later iterations only the moves that are classified as the best will be considered.

Tabus may become too powerful in the sense that they may prohibit attractive moves, even when the danger of repeating prior states are not present. This may lead to the stagnation of the searching process. It is therefore necessary to use algorithmic devices that may allow the system to *revoke* (cancel) tabus. These are called *aspiration criteria* [41]. The most commom used aspiration criterion consists of allowing a tabu move, if it results in a solution with an objective value better than the current best solution. This is done since the new best solution has not been visited in previous iterations. The basic local search is usually called iterative improvement, since each move is only performed if the resulting solution is better than the

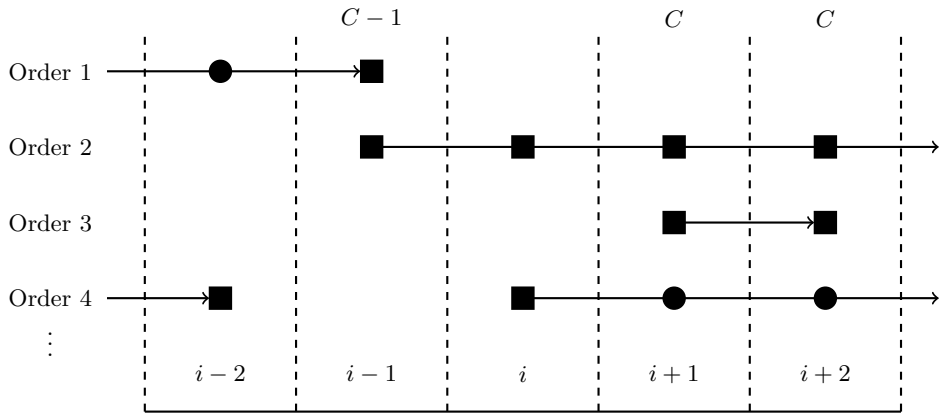


Figure 4.26: *Example of an order that is altered to increase the number of maximal cuts when a different span is selected. Four orders are displayed, namely* 1, 2, 3 *and* 4. *The locations are indicated by* $i-2$, $i-1$, i, $i+1$ *and* $i+2$, *which represents a section of a picking line. Filled squares indicate locations where orders have to pick, while filled circles indicate locations that are traversed without being picked from. The maximal cut is located at location* $i+1$ *and* $i+2$. *The arrows indicate the current spans of each order.*

current solution [19]. The algorithm stops as soon as it finds a local minimum.

When using memory in an iterative process, a check should be included that verifies whether a solution in the neighbourhood was already visited. This concept is called *short-term memory*. The data structure used to implement this is called a *hashing table*. Hashing tables might, however, prove to be very ineffective in certain situations [34]. Hashing tables prohibit a solution whose value was already obtained during the last t iterations. Thus a cycle of length t or less may be prohibited.

Another way to eliminate cycles and to direct the search, is to use a *tabu list*. To implement a tabu list, $N(\boldsymbol{s})$ has to have two characteristics, namely *connectivity* and *reversibility*. Connectivity refers to the characteristic that the optimal solution may be reached from any feasible solution. Reversibility refers to the property that a move $\boldsymbol{m}$ applied to a solution $\boldsymbol{s}$, may be reversed by a move $\boldsymbol{m}^{-1}$ such that $(\boldsymbol{s} \oplus \boldsymbol{m}) \oplus \boldsymbol{m}^{-1} = \boldsymbol{s}$. Since it makes no sense to apply move $\boldsymbol{m}^{-1}$ straight after move $\boldsymbol{m}$, we need to limit the moves when we have $\boldsymbol{s} \oplus \boldsymbol{m}$ to all possible moves at this position, except $\boldsymbol{m}^{-1}$. By prohibiting the reverse of a move during several iterations, one can prevent other cycles from intermediate solutions.

The implementation of the tabu search commences by iteratively selecting a starting location for each order to minimise the cut of each location. Orders are not sequenced in this process. Once an order has been assigned a starting position, the span of that order is calculated. The cut of each location is then updated accordingly and the location(s) containing the maximal cut(s) are determined.

The greedy heuristic in §4.8.1 is considered to deliver an initial solution for any instance. The tabu search is implemeted by considering reducing moves placed into a candidate list. If the candidate list contains at least one starting location, any one of these starting locations are selected. The starting location of the corresponding order is updated and that order becomes tabu for a number of predetermined iterations.

If no starting locations are identified that will reduce the maximal cut (*i.e.* the candidate list is empty) a shifting move is attempted. All the starting locations of orders that does not increase the maximal cut(s) but shifts the number of maximal cuts is considered, effectively creating

a new candidate list containing all shifting moves). A single starting location is then selected which shifts the maximal cut(s). The order corresponding to the starting location then becomes tabu. If no such starting location exists, no alteration in the number of starting locations may be achieved that reduces the number of maximal cut(s) or ensures that the number of maximal cuts remain the same. In this case the algorithm is terminated and the subtour generation heuristic is used to find a feasible tour with the current starting locations of the tabu search. Algorithm 13 contains the psuedo code of the tabu search implementation.

Algorithm 13: Tabu search.

Data: A set of all the order $\mathcal{S}$, all the preference spans for each order considered and, a tabu list length ℓ, the termination number t and two boolean variables ReducingMove and ShiftingMove.
Result: A vector $\boldsymbol{s}$ of starting locations awarded to each individual order.
Find an initial solution $\boldsymbol{s}$ by means of the greedy reducing heuristic. Calculate the values of C and N_C;
Create a tabu list $\mathcal{L}$ such that $|\mathcal{L}| = \ell$;
ReducingMove $\leftarrow$ `false`;
ShiftingMove $\leftarrow$ `false`;
repeat
  repeat
    Select an order k to be examined at random such that $k \in \mathcal{S}$;
    if *order k contains a preference span that will not increase the current value of C* **then**
      ReducingMove $\leftarrow$ `true`;
      Assign that preference spans starting location to the k^{th} entry in vector $\boldsymbol{s}$;
    end
  until *all orders $k \in \mathcal{S}$ has been examined*;
  if ReducingMove $\leftarrow$ *`false`* **then**
    repeat
      Select an order k to be examined at random such that $k \in \mathcal{S}$;
      if *order k contains a preference span that results in the lowest value of N_C^* ($N_C^* \leq N_C$)* **then**
        ShiftingMove $\leftarrow$ `true`;
        Assign that preference spans starting location to the k^{th} entry in vector $\boldsymbol{s}$;
      end
    until *all orders $k \in \mathcal{S}$ has been examined*;
    Update the value of C and N_C;
    Update the tabu list $\mathcal{L}$ and set $\mathcal{L} \leftarrow \mathcal{L} \cup k$;
  end
until *no improvement for t iterations* **or** ReducingMove $\leftarrow$ *`false`* **and** ShiftingMove $\leftarrow$ *`false`*;

Table 4.11 displays the results of the tabu search. The tabu serach was tested by varying the tabu list length at 10%, 20%, 30% and 40% of the number of orders considered. A second parameter was introduced which terminated the algorithm if the maximal cut was not reduced after 50, 100 or 200 iterations. The best results are realised when the tabu list is fixed at 30% of the number of orders considered. Insignificant savings are achieved when the algorithm is terminated at 200 iterations without improvement or 100 iterations without improvement. However, the solution quality decreases substancially when the termination criteria is set at 50. Thus only 100 iterations without a reduction of the maximal cut is considered, which requires significantly less computational time. The computational times are displayed in Table 4.12.

4.8.4 Simulated annealing

The simulated annealing algorithm (SA) is based on the analogy between the simulation of the annealing of solids and the problem of solving large combinatorial optimisation problems [109]. Annealing is the process of physically heating a solid material to impart high energy to it. At a high temperature, the solid becomes a liquid. The particles of the solid randomly rearrange

data set	Size (O, L)	Maximal cut approach	Stopping criteria of 50				Stopping criteria of 100				Stopping criteria of 200			
			$T = 10\%$	$T = 20\%$	$T = 30\%$	$T = 40\%$	$T = 10\%$	$T = 20\%$	$T = 30\%$	$T = 40\%$	$T = 10\%$	$T = 20\%$	$T = 30\%$	$T = 40\%$
A	(1262,49)	1232	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233
B	(1264,54)	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226
C	(1265,51)	1161	1178	1178	1179	1178	1178	1178	1178	1178	1178	1178	1178	1178
D	(1263,56)	1072	1114	1114	1114	1114	1114	1114	1114	1114	1114	1114	1114	1114
E	(1264,51)	1069	1125	1125	1125	1125	1125	1125	1124	1125	1125	1125	1124	1126
F	(1258,55)	1025	1062	1063	1062	1062	1063	1063	1062	1062	1062	1063	1062	1063
G	(1258,53)	1005	1069	1069	1070	1069	1070	1069	1069	1069	1069	1069	1070	1070
H	(1244,54)	992	1049	1049	1049	1049	1049	1050	1049	1050	1049	1049	1049	1049
I	(1260,56)	955	1018	1018	1019	1017	1018	1017	1017	1017	1017	1018	1017	1017
J	(1264,56)	947	1005	1006	1005	1005	1005	1006	1005	1005	1005	1005	1005	1005
K	(943,63)	259	294	293	294	296	293	294	293	293	293	293	293	293
L	(846,56)	232	247	247	247	247	247	248	247	247	247	247	247	247
M	(728,51)	152	170	170	171	170	170	171	171	170	171	170	170	170
N	(733,55)	125	135	134	134	134	134	135	134	134	134	134	134	134
O	(396,63)	90	100	100	100	100	100	100	101	100	100	100	100	100
P	(574,48)	80	84	84	84	84	84	84	84	84	84	84	84	84
Q	(242,64)	45	53	53	53	53	53	53	53	53	54	53	54	55
R	(158,55)	14	17	16	16	16	17	16	16	16	16	17	16	16
S	(89,42)	9	10	10	10	10	10	10	10	10	10	10	10	10
T	(82,51)	8	8	8	8	8	8	8	8	8	8	8	8	8
U	(90,48)	7	8	8	8	8	8	8	8	8	8	8	8	8
V	(80,56)	6	7	7	6	7	7	7	6	7	6	7	7	6
Total		11711	12212	12211	12213	12211	12212	12215	12208	12209	12209	12211	12209	12212

Table 4.11: *Results (expressed as the number of cycles traversed) when the tabu search was applied to the OSP. The length of the tabu list is varied at 10%, 20%, 30% and 40% of the number of orders considered. The algorithm is terminated if the maximal cut was not reduced after 50, 100 or 200 iterations. The number of orders (O) and locations (L) are displayed for each data set.*

Data set	Size (O, L)	Stopping criteria of 50				Stopping criteria of 200				Stopping criteria of 200			
		$T = 10\%$	$T = 20\%$	$T = 30\%$	$T = 40\%$	$T = 10\%$	$T = 20\%$	$T = 30\%$	$T = 40\%$	$T = 10\%$	$T = 20\%$	$T = 30\%$	$T = 40\%$
A	(1262,49)	605	717	709	695	653	657	672	655	863	669	849	754
B	(1264,54)	190	207	221	190	311	273	333	265	330	327	356	303
C	(1265,51)	163	179	175	195	263	268	273	224	257	248	255	306
D	(1263,56)	234	233	234	239	143	145	148	145	241	210	307	247
E	(1264,51)	113	131	116	114	150	146	147	152	186	192	188	216
F	(1258,55)	132	149	137	119	171	171	173	173	195	186	198	216
G	(1258,53)	114	115	114	115	146	138	144	140	252	257	200	203
H	(1244,54)	161	162	157	172	175	149	150	148	211	243	216	232
I	(1260,56)	110	117	111	110	179	175	169	165	216	215	197	220
J	(1264,56)	123	131	146	131	132	134	124	123	186	191	195	210
K	(943,63)	55	49	59	51	100	101	89	99	133	127	132	140
L	(846,56)	42	43	48	41	64	63	57	58	88	89	95	108
M	(728,51)	34	37	37	32	44	48	41	44	78	84	84	85
N	(733,55)	33	36	37	34	44	47	53	63	78	82	73	96
O	(396,63)	37	41	37	35	40	42	43	36	48	46	63	61
P	(574,48)	30	30	28	30	41	40	44	39	63	63	55	60
Q	(242,64)	17	16	19	16	22	20	21	21	30	28	28	46
R	(158,55)	10	9	10	12	13	11	11	12	17	20	20	22
S	(89,42)	5	7	7	6	6	9	9	7	9	9	11	11
T	(82,51)	8	8	6	7	8	7	7	10	13	11	11	16
U	(90,48)	7	6	11	7	7	7	7	9	10	10	12	14
V	(80,56)	7	9	8	8	9	9	8	11	11	13	14	13
Total		2230	2432	2427	2359	2721	2660	2723	2599	3515	3320	3559	3579

Table 4.12: *Computational times (in milliseconds) to execute the tabu search when the length of the tabu list is varied at 10%, 20%, 30% and 40% of the number of orders considered. A second parameter was introduced which terminated the algorithm if the maximal cut was not reduced after 50, 100 or 200 iterations. The number of orders (O) and locations (L) are displayed for each data set.*

themselves in a liquid phase. A cooling phase follows where the temperature is slowly lowered. Decreasing the temperature in a controlled manner leads to the material forming a crystalised solid state. This is a stable state of an absolute minimum energy.

Quencing the opposite technique of annealing. It consists of quickly lowering the temperature. This can lead to an amorphous structure[2], a metastable state that corresponds to a local minimum of energy [1]. If the decrease in temperature is too fast, it may cause defects which can be eliminated by local reheating [34].

The notion is to use this annealing technique to address optimisation problems by means of controlling the annealing process by altering the temperature for the process. The temperature must have the same effect as in the physical system: it must condition the number of possible states and lead towards the optimal state, if the temperature is lowered gradually in a slow and well-controlled manner (as in the annealing technique) and leads towards a local minimum if the temperature is lowered abruptly (as in the quenching technique) [34].

In the annealing process, when thermodynamic balance is reached at a given temperature, the relative probability of a physical system to have an energy E in a multi-state system in thermodynamic equilibrium is proportional to the Boltzmann factor of $\exp\left(\frac{-E}{k_B T}\right)$, where k_B denotes the Boltzmann constant and T is the temperature of the system [1].

The Metropolis algorithm is used to enable the solid to reach a thermal equilibrium whenever the temperature is lowered. Given the structure of the system (preference span awarded to each order), the system is subject to a modification (altering preference spans of the orders). If this transformation improves the objective function (Reduce the maximal cut and reduce the number of locations containing a maximal cut) of the system, it is accepted. If the alteration reduces the solution quality by ΔE of the objective function, it may be accepted with a probability of $\exp\left(\frac{-\Delta E}{T}\right)$. Following this criterion the system eventually evolves into a thermal equilibrium at the temperature considered — this leads to a Boltzmann distribution of energy states [1, 34, 109].

Initially, at a high temperature, $\exp\left(\frac{-E}{k_B T}\right)$ is close to 1. The majority of exchanges are considered, and the algorithm becomes a random walk within the solution space. Many exchanges may be accepted to deteriorate the state of the system. At low temperatures $\exp\left(\frac{-E}{k_B T}\right)$ is close to 0. Most of exchanges that would reduce the solution quality are rejected. The algorithm is now similar to an iterative improvement heuristic. At intermediate temperatures the system explores local minima, probing desirable solution spaces.

The annealing process may be terminated when the system is *solidified* (the temperature reaches 0 or no moves are able to improve the solution quality). The system may also be reheated, essentially applying the annealing process again if the solidified state of the current system is not desirable.

An initial solution is generated using the greedy algorithm described in §4.8.1. The choice of valueing the exchange of starting locations is measured by a combination of the maximal cut and the number of locations containing the maximal cut. The energy E_x for a given state x is

$$E_x = C_x + \frac{N_C^x}{m}, \tag{4.29}$$

where C_x denotes the maximal cut, N_C^x denotes the number of locations containing the maximal cut and m is the total number of locations in the picking line for a given state x.

[2] An amorphous solid is any noncrystalline solid in which the atoms and molecules are not organized in a definite lattice pattern. Such solids include glass, plastic, and gel [22].

The initial temperature T_0 is calculated by assigning a random starting location to each order. A total of 100 alterations are made at random calculating the average $\overline{\Delta E}$ of the corresponding ΔE variations. The initial rate of acceptance τ_0 that is associated with the quality of the initial configuration must be selected. According to Dréo *et al.* [34] for poor quality solutions to be considered $\tau_0 = 0.5$ (starting at a high temperature), while good quality solutions are considered when $\tau_0 = 0.2$ (starting at a low temperature). The value of T_0 is deduced from the relation

$$\exp\left(\frac{-\overline{\Delta E}}{T_0}\right) = \tau_0. \tag{4.30}$$

The Metropolis acceptance rule is used to address exchanges that decreases the solution quality. If $\overline{\Delta E} > 0$, a number $r \in [0, 1]$ is randomly generated. The exchange is accepted if $r < \exp\left(\frac{-\Delta E+t}{T_0}\right)$, where T indicates the current temperature, and t is the current number of iterations at a given temperature. The Metropolis rule is adapted, since the greedy heuristic in §4.8.1 serves as a good general point of departure. Few moves are able to improve the solution quality. If too many moves are selected that do not improve the solution quality, the algorithm tends towards a random local search.

During an iteration a number of orders are randomly selected. The starting locations of these orders may be altered. Each starting location is awarded with a certain probability of being elected. If an order k requests $|\mathcal{O}_k|$ different SKUs, each of the $|\mathcal{O}_k|$ possible starting locations associated with various preference spans is awarded a probability relative to the minimum span of order k. Each starting location i of an order k is awarded a score of

$$c_k^i = \frac{|S_k^{\min}|}{|S_k^i|}. \tag{4.31}$$

This score is then normalised, by awarding a probability p_k^i for each starting location i of an order k by using

$$p_k^i = \frac{c_k^i}{\sum_{i \in \mathcal{O}_k} c_k^i}. \tag{4.32}$$

When an order k is selected for an alteration of its starting location, the probability p_k^i constitutes the probabilty of starting order k on location i.

Changes in temperature may occur at the end of an iteration where a certain percentage of orders are elected. The change in temperature occurs when either a number of exchanges are accepted or when a number of exchanges are attempted. The change in temperature is altered according to the geometric law, where

$$T_{y+1} = 0.9 \cdot T_y. \tag{4.33}$$

An iteration is ended after 5 successive temperature decreases without acceptance of any exchanges. The program is terminated after 5 successive iterations without an improvement in the solution quality, where a maxmimum of 100 iterations are awarded to the algorithm. Algorithm 14 displays the pseudo code for the simulated annealing process.

The SA is implemented by altering the percentage of preference spans during each iteration at 5%, 7,5%, 10% and 11%. The value of τ_0 is varied at 0.02, 0.05 and 0.1. The best results are produced when τ_0 is equal to 0.02 and when 10% of the preference spans are considered to be altered during every iteration. Considerable computational time increases are incurred when

Algorithm 14: Simulated annealing.

Data: A set of all the order $\mathcal{S}$, all the preference spans for each order considered and the number of orders to be altered during each iteration s_o and the value of τ_0.
Result: A vector $\boldsymbol{s}$ of starting locations awarded to each individual order.
Find an initial solution $\boldsymbol{s}$ by means of the greedy reducing heuristic in Algorithm 12. Calculate the values of C and N_C;
Evaluate $\overline{\Delta E}$;
Calculate T_0 according to (4.30);
repeat
  Select s_o orders at random and an alternative preference span for each order based on (4.32);
  for $x \leftarrow 0$ *to* s_o **do**
    Use Metropolis rule of acceptance;
    Update the fitness E_x;
    if *12 perturbations are accepted* **or** *100 perturbations attempted* **then**
      $T_{y+1} = 0.9 \cdot T_y$;
    end
  end
until *no improvement in the fitness is found in 5 iterations* **or** *until 100 iterations have been completed*;

the number of preference spans altered are shifted from 5% to 10% and even more from 10% to 11% and neither of these time increases justify the minor or no increase in solution quality. Table 4.13 displays the results obtained for the SA. Table 4.14 displays the computational time in milliseconds for the results in Table 4.13. More computational time is required when more orders are considered during an iteration, but better quality solutions are not necessarily attained.

4.8.5 Genetic algorithm

During the 1990s genetic programming became a popular search technique due to work by Koza [68]. Fogel [40] describes genetic programming as a two-step iterative process, consisting of random variation followed by selection. The link between this description of evolution and the optimising algorithms that are the hallmark of evolutionary computation is conceptually simple. Hirsh [56] explains genetic programming by progressively breeding a population of computer programs over a series of generations using the Darwinian principles of evolution and natural selection. It extends the genetic algorithm to the arena of computer programs.

Genetic Algorithms (GAs) and Evolutionary Strategies (ES) are two types of algorithms which try to imitate the mechanism of natural evolution [6]. GAs are search techniques inspired by the biological evolution of species in nature. The principles of a GA commences with an *initial population*, where each individual has a certain fitness level, which measures the level of *adaptation* to the objective aimed. Adaptation is the evolutionary process whereby a population (or individual within the population) becomes better suited to its habitat [34]. The GA gradually evolves in successive *generations*.

First an initial population of possible solutions (*chromosomes*) must be generated. Chromosomes may be any feasible solution to the problem and is usually expressed as a binary string. A GA must generate many chromosomes as an initial population. A chromosome is formed by *genes*. The genes are the elements that described the chromosome in its entirety. The chromosomes are observed and evaluated according to their fitness function. A fitness function is the objective function that specifies the goodness of the solution (chromosome).

Within a population, the chromosomes are then ranked according to their fitness. All the

Data set	Size (O, L)	Maximal cut approach	Percentage to swap is 5%			Percentage to swap is 7.5%			Percentage to swap is 10%			Percentage to swap is 11%		
			$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$	$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$	$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$	$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$
A	(1262,49)	1232	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233
B	(1264,54)	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226
C	(1265,51)	1161	1168	1166	1166	1165	1165	1167	1169	1168	1168	1169	1168	1168
D	(1263,56)	1072	1091	1090	1089	1090	1091	1092	1087	1089	1092	1087	1088	1090
E	(1264,51)	1069	1087	1082	1085	1086	1086	1085	1083	1083	1081	1082	1083	1083
F	(1258,55)	1025	1039	1038	1045	1041	1038	1038	1040	1038	1040	1041	1041	1040
G	(1258,53)	1005	1050	1054	1046	1048	1045	1044	1047	1042	1047	1045	1045	1039
H	(1244,54)	992	1027	1022	1024	1026	1022	1020	1019	1021	1021	1022	1022	1020
I	(1260,56)	955	993	990	998	990	997	994	995	988	990	991	988	994
J	(1264,56)	947	984	982	992	985	985	986	982	982	988	983	983	985
K	(943,63)	259	276	275	276	275	273	276	276	276	279	275	272	279
L	(846,56)	232	244	243	242	240	242	242	242	241	241	242	243	243
M	(728,51)	152	155	157	155	155	156	155	155	155	154	155	154	156
N	(733,55)	125	130	133	132	131	131	131	133	132	133	129	132	133
O	(396,63)	90	93	94	93	94	95	93	93	93	93	93	94	94
P	(574,48)	80	82	83	82	83	83	82	82	82	82	82	83	82
Q	(242,64)	45	46	45	45	45	45	45	45	45	45	45	45	45
R	(158,55)	14	14	14	14	14	14	14	14	14	14	14	14	14
S	(89,42)	9	9	9	9	9	9	9	9	9	9	9	9	9
T	(82,51)	8	8	8	8	8	8	8	8	8	8	8	8	8
U	(90,48)	7	8	8	8	8	8	8	8	8	8	8	8	8
V	(80,56)	6	6	6	6	6	6	6	6	6	6	6	6	6
Total		11711	11969	11958	11974	11958	11958	11954	11952	11939	11958	11945	11945	11955

Table 4.13: *Results (in number of cycles traversed) when the simulated algorithm was applied to the OSP. The percentage of preference spans to be altered during each iteration is varied at 5%, 7.5%, 10% and 11%. The value of τ_0 is varied at 0.02, 0.05 and 0.1. The number of orders (O) and locations (L) are displayed for each data set.*

Data set	Size (O, L)	Percentage to swap is 5%			Percentage to swap is 7.5%			Percentage to swap is 10%			Percentage to swap is 11%		
		$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$	$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$	$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$	$\tau_0 = 0.02$	$\tau_0 = 0.05$	$\tau_0 = 0.1$
A	(1262,49)	733	733	804	840	840	947	6226	5380	5751	22481	20440	6792
B	(1264,54)	394	394	361	525	525	599	5375	5625	6269	22647	25294	6779
C	(1265,51)	596	596	500	863	863	1269	10995	16754	19841	80810	80979	21811
D	(1263,56)	433	433	757	1784	1784	2171	30971	24692	25111	187991	122638	34229
E	(1264,51)	966	966	678	2383	2383	1665	31451	33093	24952	285235	103622	51077
F	(1258,55)	401	401	418	1283	1283	2078	31912	29430	22189	101831	119443	30154
G	(1258,53)	607	607	833	1563	1563	2864	34789	27299	50665	193001	138934	51093
H	(1244,54)	596	596	683	1869	1869	2295	32653	48500	38003	147623	143175	39599
I	(1260,56)	487	487	904	2161	2161	1596	37687	42712	36960	166043	156838	32099
J	(1264,56)	779	779	751	1599	1599	1674	42010	26845	46311	199479	197267	42001
K	(943,63)	226	226	206	446	446	974	2140	5110	3763	5623	7233	1926
L	(846,56)	144	144	199	315	315	312	1282	969	1204	2958	3081	1111
M	(728,51)	133	133	125	223	223	185	492	619	466	1086	1205	463
N	(733,55)	102	102	122	193	193	234	344	416	541	820	761	501
O	(396,63)	82	82	66	108	108	93	179	152	111	132	274	164
P	(574,48)	88	88	93	133	133	75	131	143	282	231	284	163
Q	(242,64)	44	44	49	54	54	59	102	67	62	63	67	59
R	(158,55)	17	17	26	18	18	23	22	24	30	23	22	20
S	(89,42)	9	9	9	12	12	10	11	14	12	11	11	11
T	(82,51)	10	10	11	11	11	10	14	13	12	11	13	13
U	(90,48)	11	11	12	11	11	10	15	12	12	13	11	11
V	(80,56)	10	10	12	12	12	11	12	13	12	16	12	14
Total		6868	6868	7619	16406	16406	19154	268813	267882	282559	1418128	1121604	320090

Table 4.14: *Computational times, in milliseconds, for the simulated annealing when the percentage of preference spans to be altered during each iteration is varied at 5%, 7.5%, 10% and 11%. The value of τ_0 is varied at 0.02, 0.05 and 0.1. The number of orders (O) and locations (L) are displayed for each data set.*

chromosomes that produce a good fitness, or even the chromosomes with weaker fitnesses, are then allowed to create offspring [90]. This process is called the *crossover*. The crossover is the breeding process of the population. The reproduction process generates diversity in the gene pool [88]. The existing chromosomes breed to create new chromosomes (offspring), which will enter the following generation. The genes from two or even multiple parents may produce offspring.

When an offspring improves itself, the chromosome has created a *hybrid* of itself. *Mutation* is similar to hybrid but with mutation a chromosome does not have control over the changes that happens to it. This is usually used to achieve *genetic diversity*. Genetic diversity maintain diversity from one generation to another.

A GA is applied to each picking line individually. Solutions (chromosomes) are presented as the sequence in which orders must be executed. The fitness of each gene is calculated as the number of cycles travelled in each picking line.

The initial population was constructed either randomly or in a greedy manner. The greedy manner of finding a gene entails that an order is selected at random. All the other orders in that picking line is then considered and the order resulting in the least number of locations travelled is then selected. This process is iteratively done until all orders are sequenced to form a chromosome. A ranked based selection is used to rank the genes. Genes are ranked in terms of their fitness in increasing order. A fraction of the genes may be eliminated if their fitness is poor compared to the other genes. A uniform order-based crossover is used. An offspring inherits a combination of the orders existing in two "parent" sequences. This crossover has the advantage of being simple to implement and effective. The crossover follows three stages [34]:

- A binary template is generated at random.
- Two parents are mated. The 0 (respectively 1), of the binary string indicates the preserved positions in the sequence of parent 1 (respectively parent 2).
- Non-preserved elements are permuted to generate offspring 1 (respectively offspring 2).

Figure 4.27 displays an example of an uniform order-based crossover, where each parent contains six orders.

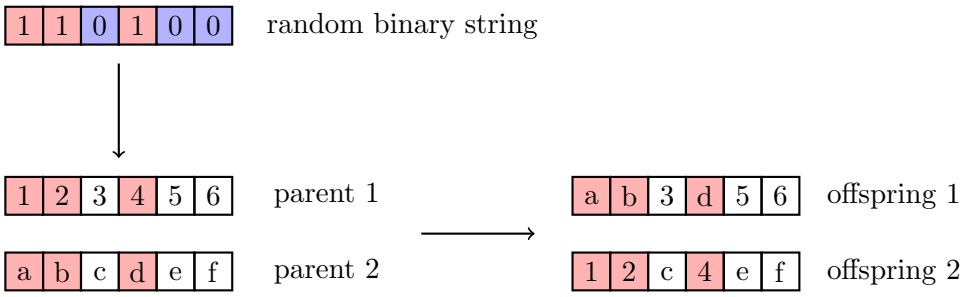


Figure 4.27: *An example of a uniform order-based crossover where both parent solutions contain 6 orders. A random binary string is used to aid the crossover where genes are exchanged to create offspring.*

Data set	Size (O, L)	Maximal cut approach	Number of chromosomes = 100				Number of chromosomes = 200				Number of chromosomes = 500			
			t =40	t =60	t =80	t =100	t =40	t =60	80	t =100	t =40	t =60	t =80	t =100
A	(1262,49)	1232	1234	1234	1234	1234	1234	1234	1234	1234	1234	1234	1234	1234
B	(1264,54)	1226	1230	1231	1229	1230	1229	1230	1230	1229	1229	1229	1229	1228
C	(1265,51)	1161	1216	1216	1215	1214	1214	1214	1215	1216	1215	1214	1214	1213
D	(1263,56)	1072	1195	1195	1193	1193	1195	1195	1194	1190	1194	1193	1192	1193
E	(1264,51)	1069	1200	1200	1199	1200	1198	1197	1199	1201	1199	1199	1200	1198
F	(1258,55)	1025	1156	1157	1158	1157	1157	1156	1157	1157	1156	1155	1156	1156
G	(1258,53)	1005	1181	1182	1179	1181	1179	1181	1180	1180	1179	1179	1178	1180
H	(1244,54)	992	1166	1165	1165	1165	1162	1165	1164	1165	1162	1165	1163	1165
I	(1260,56)	955	1158	1159	1158	1158	1156	1157	1158	1159	1157	1155	1155	1158
J	(1264,56)	947	1158	1157	1159	1155	1157	1156	1156	1157	1157	1156	1156	1157
K	(943,63)	259	601	596	602	605	603	600	604	603	597	598	599	600
L	(846,56)	232	526	530	529	525	526	527	527	525	526	526	522	525
M	(728,51)	152	421	421	423	421	417	423	420	416	417	419	419	419
N	(733,55)	125	409	413	413	409	412	408	409	413	408	406	410	401
O	(396,63)	90	229	226	230	228	230	227	229	229	226	225	230	228
P	(574,48)	80	305	308	304	304	306	304	302	306	302	300	302	299
Q	(242,64)	45	130	131	132	132	130	129	130	129	129	129	128	127
R	(158,55)	14	74	72	72	73	73	72	72	72	69	71	70	71
S	(89,42)	9	39	37	38	38	37	38	37	39	37	37	37	34
T	(82,51)	8	35	35	36	36	34	36	36	35	35	33	34	34
U	(90,48)	7	39	37	36	39	38	35	37	37	36	37	36	37
V	(80,56)	6	35	33	35	34	34	33	35	32	33	34	33	34
Total		11711	14737	14735	14739	14731	14721	14717	14725	14724	14697	14694	14697	14691

Table 4.15: *Results (in the number of cycles traversed) when the genetic algorithm models chromosomes as the sequence of orders, when the number of chromosomes are varied at either 100, 200, or 500. The number of iterations are varied at either 40, 60, 80 or 100. The number of orders (O) and locations (L) are displayed for each data set.*

Data set	Size (O, L)	Number of iterations = 100				Number of iterations = 200				Number of iterations = 500			
		t =40	t =60	t =80	t =100	t =40	t =60	t =80	t =100	t =40	t =60	t =80	t =100
A	(1262,49)	1530	1990	2524	2721	2526	3666	4770	5939	11203	16860	23420	29105
B	(1264,54)	901	1225	1546	1961	1927	2809	3721	4681	12029	15565	21594	34795
C	(1265,51)	1019	1376	1761	2115	2092	3045	3921	4975	12144	16344	22489	42438
D	(1263,56)	963	1387	1829	2282	2175	3156	4236	5423	11954	16728	25309	44261
E	(1264,51)	1043	1447	1833	2131	2096	3108	4072	5057	12189	15573	24821	44527
F	(1258,55)	919	1296	1689	2069	2083	3008	3975	4930	11204	15549	21727	43508
G	(1258,53)	961	1359	1739	2159	2091	3105	4014	5130	11150	15956	22123	42201
H	(1244,54)	1001	1433	1861	2334	2429	3390	4447	5490	11143	17745	22880	43705
I	(1260,56)	1080	1457	1905	2228	2223	3287	4248	5479	11041	22004	23393	43675
J	(1264,56)	920	1312	1751	2159	2158	3130	4154	5099	11124	15639	23162	36422
K	(943,63)	527	763	996	1270	1295	1907	2505	3116	7116	8378	13338	13773
L	(846,56)	538	677	886	1109	1142	1703	2261	2758	5280	7300	9928	12298
M	(728,51)	1173	1295	896	1969	1251	1836	2778	3365	7882	12155	9322	11304
N	(733,55)	385	540	733	917	954	1428	1916	2356	4798	10992	14026	9989
O	(396,63)	239	332	425	542	598	864	1145	1368	2574	3848	4815	5925
P	(574,48)	293	418	547	674	724	1064	1445	1759	3302	4890	6497	7720
Q	(242,64)	222	255	341	395	367	550	734	870	1438	2141	2821	3449
R	(158,55)	101	108	143	176	195	285	380	469	871	1292	1717	2163
S	(89,42)	54	64	80	99	113	162	216	269	505	738	986	1203
T	(82,51)	48	58	88	91	144	150	200	245	453	675	895	1099
U	(90,48)	51	62	80	98	110	161	218	264	491	732	986	1190
V	(80,56)	49	58	73	90	101	150	199	240	449	669	900	1077
Total		14017	18912	23726	29589	28794	41964	55555	69282	150340	221773	297149	475827

Table 4.16: *Computational times, in milliseconds, for the genetic algorithm models chromosomes as the sequence of orders. when the number of iterations are varied at either 40, 60, 80 or 100. The number of chromosomes are varied at either 100, 200, or 500. The number of orders (O) and locations (L) are displayed for each data set.*

Modelling chromosomes as the sequence for orders

The implementation of the GA commences with the creation of the initial population. A number of chromosomes are generated, each containing a sequence in which orders must be executed. All the genes within the population are evaluated and the number of cycles travelled by a picker when picking the orders in that sequence is the fitness associated with that gene. The genes are then ranked in decreasing order according to the fitness of that gene.

Two common fitness assignment methods in GAs are the *ranked-based fitness* and the *proportional fitness assignment*. The ranked-based fitness sorts the chromosomes in the population in decreasing order in terms of their fitness value. The proportional fitness assignment calculates an actual objective value for each gene. The ranked-based fitness assignment is used, since it tends to be more robust than proportional fitness assignment and is thus selected for implementation of this GA [6].

The chromosomes that do not produce good fitnesses are then neglected from performing crossovers. A set of binary strings are then generated for the genes that may perform crossovers — one binary string for every two genes. A uniform order-based crossover is performed by every pair of these genes. When crossovers have been completed, the genes are ranked according to their fitness. Only the best performing genes are used for a mutation process, where alterations are made to genes with the aim of improving the fitness of the genes.

The mutation commences with identifying the location(s) containing the maximal cut within a gene. Orders that pass the maximal cut(s) but do not pick from these location(s) are then considered for mutation. Within each gene, the considered orders are swapped with all other orders in the gene. Swaps that decrease the maximal cut at one or more locations, are ranked according to the number of maximal cuts decreased and the best swap is selected as a mutation. If no swaps are able to reduce the maximal cut or decrease the location(s) containing the maximal cut, a swap that alters the location(s) containing the maximal cut are chosen as the mutation. This entails the process of reducing the maximal cut of the location considered, but forcing one other location to have a cut equal to the maximal cut. This is identical to a swap in the tabu search metaheuristic explained in §4.8.3. The mutation will never increase the maximal cut or the number of locations containing the maximal cut. It may, however, shift the locations containing the maximal cut.

The number of chromosomes are varied at 100, 200 and 500, while the number of iterations (t) are varied at 40, 60, 80 and 100. The number of chromosomes that are used for the mutation process is the top 2% of the number of chromosomes considered. The mutation process is computationally expensive and is therefore only reserved for the best genes during each iteration. After each iteration 20% of the worst performing chromosomes are replaced by a set of random chromosomes.

Table 4.15 displays the results obtained from a series of parameters used to solve the GA. Table 4.16 displays the computational times of the results obtained from a series of parameters used to solve the GA.

Modelling chromosomes as starting locations

Instead of modelling chromosomes as the sequence in which orders must be picked, chromosomes may also indicate the starting location of each order. When the starting location of an order is known, the size of the span may be calculated. Furthermore, the maximal cut(s) may be identified for each chromosome.

Initially starting locations for all the orders are iteratively awarded to each chromosome according to the heuristic in §4.8.1. The sequence in which orders are awarded preference spans is altered for each chromosome to obtain a diverse range of starting solutions in an attemp to diversify the gene pool.

The fitness of each chromosome z is calculated as

$$f_z = C_z + \frac{N_C^z}{m}, \tag{4.34}$$

where m is the total number of locations, C_z denotes the maximal cut and N_C^z denotes the number of locations containing the maximal cut.

Three types of crossovers may be used to create offspring between two parents. The crossovers entail the process of swapping starting locations for the various orders between two chromosomes. Before the crossover commences, chromosomes are ranked in decreasing order in terms of its fitness. All the chromosomes are allowed to create offspring in a ranked-based manner.

Algorithm 15: Genetic algorithm.

Data: A set of all the order $\mathcal{S}$, all the preference spans for each order considered and population size P and the desired crossover approach.
Result: A vector $\boldsymbol{s}$ of starting locations awarded to each individual order.
Find an initial solution $\boldsymbol{s}$ by means of the greedy reducing heuristic, together with the values of C^z and N_C^z for each chromosome $z = 1, 2, \ldots, Z$;
Evaluate f_z for each chromosome $z = 1, 2, \ldots, Z$;
repeat
  Rank the chromosomes in a decreasing manner in terms of their fitness;
  for $z \leftarrow 0$ *to* $P - 1$ *(with step size 2)* **do**
    Perform a crossover between chromosomes z and $z + 1$ according to the desired crossover approach;
    Update the fitness f_z and f_{z+1};
  end
  Rank the chromosomes in a decreasing manner in terms of their fitness;
  Perform mutation on the best 1% of the chromosomes;
  Rank the chromosomes in a decreasing manner in terms of their fitness;
  Perform elitism of the top 2% of chromosomes;
  Replace 30% of the worst performing chromosomes with solutions by means of the greedy reducing heuristic;
until *no improvement in the fitness is found in 5 iterations*;

The first type of crossover is a random swapping starting locations of orders between two chromosomes, called the 50/50 *crossover*. A vector of random numbers are generated for each order within a chromosome. Two parents are considered. If the random number for an order is larger than 0.5, the starting locations (genes) of that order is swapped. Figure 4.28 displays an example of two chromosomes, each containing six genes. A vector containing six random numbers are generated. Three genes are swapped to create two offspring. A second manner of approaching crossovers is to award a higher probability of one offspring receiving the shorter preference span, called the *ranked crossover*. When considering two parents, the probability of swapping two starting locations (genes), i and j, for an order k by generating a random number r. If

$$r < \frac{|S_k^i|}{|S_k^i| + |S_k^j|}, \tag{4.35}$$

the two starting locations are swapped. In this manner a higher likelihood exists of creating an offspring containing the majority of the shorter preference spans and one offspring containing the longer preference spans. A final crossover is considered where the maximal cut(s) of the

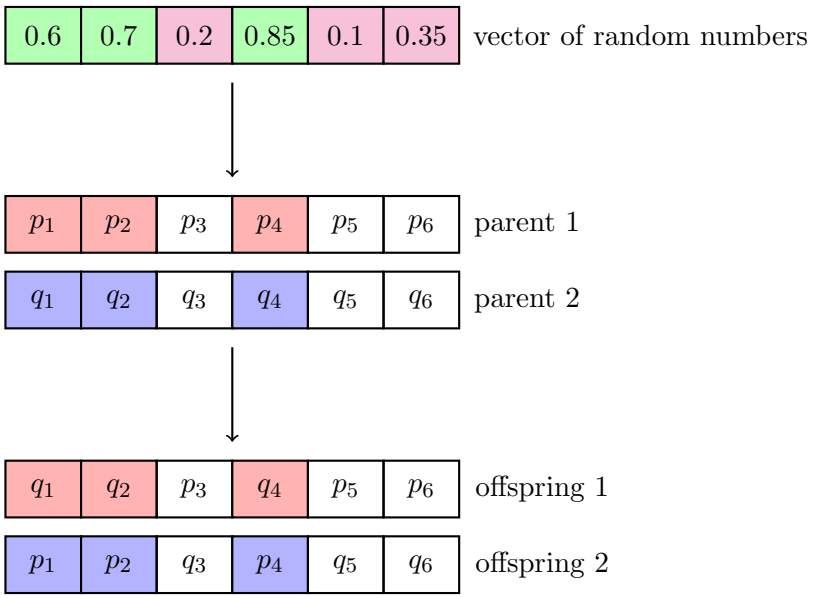


Figure 4.28: *An example of a 50/50 crossover, where the starting locations of two parents are swapped if the random number is larger than 0.5. A total of six orders are considered in this example.*

first parent is identified, called the *maximal cut traverse crossover*. A crossover is only made between two starting locations if the starting location of an order in the first parent currently crosses all the current maximal cut(s) associated with the fitness of the first parent. In this case an effort is only made to improve the quality of one of the parents.

Elitism is also used in the genetic algoritm. Only 1% of the best solutions will enter the next generation as is. This allows the algorithm to always alter the current best performing chromosome in an attempt to improve its solution quality. Also 1% of the best performing chromosomes in an iteration are selceted for a mutation phase, where each gene is individually examined for any alteration in starting location (preference span) that may improve the solution quality. During each iteration a total of 30% of the worst performing chromosomes are replaced with new chromosomes generated by the heuristic procedure in §4.8.1. Algorithm 15 shows the pseudo code for the implementation of the GA.

Table 4.17 displays the results obtained for the GA when the three approaches used to do crossovers are considered when the number of chromosomes are varied at 40, 60, 80 and 100 for each crossover approach. The number of chromosomes are relatively little compared to the number suggested in literature, but these parameters are selected, since the computational times associated with increasing the number of chromosomes does not yield significant improvements. Also, the other metaheuristics considered to solve the OSP require little computational time, compared to the GAs, to solve an instance. To compare the metaheuristics considered, the computational times have to be of a similar nature. Table 4.18 displays the computational times in milliseconds for the results in Table 4.17.

Data set	Size (O, L)	Maximal cut approach	50/50 crossover				Ranked crossover				Maximal cut traverse crossover			
			$Cr = 20$	$Cr = 40$	$Cr = 60$	$Cr = 80$	$Cr = 20$	$Cr = 40$	$Cr = 60$	$Cr = 80$	$Cr = 20$	$Cr = 40$	$Cr = 60$	$Cr = 80$
A	(1262,49)	1232	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233
B	(1264,54)	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226
C	(1265,51)	1161	1164	1166	1164	1163	1166	1165	1163	1162	1168	1165	1167	1163
D	(1263,56)	1072	1100	1091	1097	1099	1099	1097	1098	1097	1098	1098	1096	1096
E	(1264,51)	1069	1088	1091	1095	1087	1092	1085	1085	1089	1102	1098	1096	1093
F	(1258,55)	1025	1054	1053	1054	1051	1052	1053	1053	1054	1055	1053	1053	1054
G	(1258,53)	1005	1059	1058	1060	1060	1064	1060	1058	1059	1064	1063	1063	1059
H	(1244,54)	992	1033	1035	1034	1033	1031	1034	1028	1031	1033	1035	1032	1034
I	(1260,56)	955	1000	1000	1004	1002	1002	1002	1002	1002	1002	1002	1002	1002
J	(1264,56)	947	996	996	992	991	986	988	993	993	995	993	992	991
K	(943,63)	259	278	278	278	275	271	269	269	267	271	271	274	267
L	(846,56)	232	238	240	240	240	239	236	236	236	241	238	237	238
M	(728,51)	152	157	154	157	157	156	154	154	155	159	155	154	154
N	(733,55)	125	129	128	128	130	129	128	128	129	129	132	128	128
O	(396,63)	90	95	92	92	93	94	93	93	93	92	93	93	93
P	(574,48)	80	82	81	81	81	82	81	82	82	83	83	82	81
Q	(242,64)	45	46	45	45	45	45	45	45	45	45	45	45	45
R	(158,55)	14	14	14	14	14	14	14	14	14	14	14	14	14
S	(89,42)	9	9	9	9	9	9	9	9	9	9	9	9	9
T	(82,51)	8	8	8	8	8	8	8	8	8	8	8	8	8
U	(90,48)	7	8	8	8	8	8	8	8	8	8	8	8	8
V	(80,56)	6	6	6	6	6	6	6	6	6	6	6	6	6
Total		11711	12023	12012	12025	12011	12012	11994	11991	11998	12041	12028	12018	12002

Table 4.17: *Results (in the number of cycles traversed) when the genetic algorithm when the 50/50 crossover, ranked crossover and the maximal cut traverse approaches are tested with the number of chromosomes, Z, varied at 40, 60, 80 and 100 for each approach. The number of orders (O) and locations (L) are displayed for each data set.*

Data set	Size (O, L)	Percentage to swap is 2.5%				Percentage to swap is 5%				Percentage to swap is 7.5%			
		$Cr = 20$	$Cr = 40$	$Cr = 60$	$Cr = 80$	$Cr = 20$	$Cr = 40$	$Cr = 60$	$Cr = 80$	$Cr = 20$	$Cr = 40$	$Cr = 60$	$Cr = 80$
A	(1262,49)	1758	2196	2618	3000	1879	2172	2547	3106	1923	2323	2554	3067
B	(1264,54)	1421	1712	2023	2401	1438	1740	2012	2448	1474	1759	2122	2358
C	(1265,51)	2340	2141	4780	4273	2531	3951	6478	4620	1563	1546	4152	3484
D	(1263,56)	2492	1656	4812	2272	3876	5572	2312	2588	1418	1654	1934	4991
E	(1264,51)	3033	3860	2401	6379	3568	8139	6679	4910	1305	2685	1748	3762
F	(1258,55)	963	1467	5945	2623	1244	1508	2331	1924	2144	1451	2048	1936
G	(1258,53)	1281	2127	4475	2785	1260	1572	4497	5441	1294	2101	1811	3903
H	(1244,54)	2163	1663	2694	2623	6211	1653	1954	2207	1467	1653	2380	2610
I	(1260,56)	3603	2285	1913	2189	1354	4908	2267	2567	1696	1595	1906	2552
J	(1264,56)	3270	1923	1866	5780	1308	3231	5490	2108	2526	1565	1869	3247
K	(943,63)	1316	1541	1539	3125	1790	2273	1521	3688	1204	1587	1009	3349
L	(846,56)	629	1551	876	1089	1542	957	2339	1376	750	809	1007	1070
M	(728,51)	503	710	770	637	658	396	414	1579	476	549	770	1216
N	(733,55)	883	1021	429	471	1709	969	812	560	479	672	527	664
O	(396,63)	228	186	296	429	196	227	381	305	186	507	505	249
P	(574,48)	344	527	303	329	256	441	888	271	256	274	303	552
Q	(242,64)	68	103	144	190	74	90	145	190	85	104	106	168
R	(158,55)	27	33	39	59	26	33	42	49	29	33	39	57
S	(89,42)	10	14	16	21	12	13	18	20	10	13	16	20
T	(82,51)	13	14	19	24	11	15	21	23	13	16	18	22
U	(90,48)	12	22	27	28	12	15	17	21	11	18	18	26
V	(80,56)	14	16	20	27	15	16	22	25	13	17	20	25
Total		26371	26768	38005	40754	30970	39891	43187	40026	20322	22931	26862	39328

Table 4.18: *Computational times, in milliseconds, for the genetic algorithm when the 50/50 crossover, ranked crossover and the maximal cut traverse approaches are tested with the number of chromosomes, Z, varied at 40, 60, 80 and 100 for each approach. The number of orders (O) and locations (L) are displayed for each data set.*

4.8.6 Extremal Optimisation

Extremal Optimisation (EO) is an optimisation heuristic inspired by the property of self-organized criticality (SOC) from the field of statistical physics[3]. EO is based on the dynamics of non-equilibrium processes and in particular those exhibiting SOC, where better solutions emerge dynamically without the need for parameter tuning [16]. The Bak-Sneppens model of co-evolution of species exhibits SOC and was the inspiration for EO [118].

In contrast to genetic algorithms which consider the entire gene pool, EO improves a single candidate solution by considering each component as co-evolving according to Darwinian principles [15]. EO uses a non-equilibrium approach which requires few parameters that have to be altered, unlike simulated annealing. It may be modelled by having a single parameter, while maintaining competitive, and even superior, results with respect to elaborate optimisation algorithms. In contrast to simulated annealing EO takes the system far from equilibrium as it applies no decision criteria and all new configurations are accepted indiscriminately.

In EO the search spaces are always spaces of structured tuples $\boldsymbol{g} = (g_1, g_2, \ldots, g_x, \ldots, g_X)$. A single solution is considered and is examined to determine the contributions of its genes to the overall fitness. Analogously to the Bak-Sneppens model, each gene g_x has its own fitness contribution $\lambda(g_x)$. The higher the value of $\lambda(g_x)$, the better the contribution. The EO algorithm proceeds as follows [118]:

1. Create an initial solution (individual) $\boldsymbol{h}$ with a random gene construction $\boldsymbol{g}$ and set the current best known solution candidate f^*, *i.e.* $f^* = \boldsymbol{h} \cdot \boldsymbol{g}$.
2. Sort all al the genes $h_x \cdot g_x$ of $\boldsymbol{h} \cdot \boldsymbol{g}$ in a list in ascending order according to their fitness contributions $\lambda(h_x \cdot g_x)$.
3. The gene with the lowest fitness contribution is selected and modified randomly, leading to a new solution candidate $f = \boldsymbol{h} \cdot \boldsymbol{g}'$.
4. If f is better than f^*, set $f^* = \boldsymbol{h} \cdot \boldsymbol{g}'$.
5. If the termination criterion has not yet been met, return to Step 2.

Instead of always picking the weakest part, Boettcher & Percus [15] selected a gene to be modified ramdomly in order to ensure that the algorithm does not become stuck in a local optima. The probability

$$P(y_x) \propto y_x^{-\tau} \tag{4.36}$$

is awarded to a gene g_x with a fitness rank y_x. The higher the value of τ, the less prone it is to alter genes that are ranked lower.

The major concern with EO is the manner of determining the fitness contributions $\lambda(h_x \cdot g_x)$ for the elements $h_x \cdot g_x$. De Sousa *et al.* [31] extended EO to the Generalised Extremal Optimisation (GEO), by examining each gene in the search space, according to the following procedure:

1. Create an initial solution (individual) $\boldsymbol{h}$ with a random gene construction $\boldsymbol{g}$ and set the current best known solution candidate f^*, *i.e.* $f^* = \boldsymbol{h} \cdot \boldsymbol{g}$.

[3]The theory of SOC states that large interactive systems evolve to a state where a change in one single of its elements may lead to domino effects that may reach any other element in the system [118].

2. For each gene g_x, do the following:
   (a) Toggle the value of each gene g_x to retrieve a solution $f'_x = h_x \cdot g'_x$.
   (b) Set the change in fitness at $\Delta f_x = f^* - f'_x$.
   (c) Return the gene to its origional form.
3. Rank the genes in decreasing order in terms of Δf_x for $x = 1, 2, \ldots, X$.
4. Use (4.36) as *mutating probability* to determine which gene will be mutated. A random number r is uniformly generated in the range $[0, 1]$. If r is smaller than the mutating probablity, the mutation is confirmed, otherwise the process is repeated until a gene is confirmed to mutate. This yields a fitness of f'.
5. If f' is better than f^*, set $f^* = \boldsymbol{h} \cdot \boldsymbol{g}'$.
6. If the termination criterion has not yet been met, continue with Step 2.

The EO algorithm is implemented for the OSP by considering the starting locations of each order. Once again, to reduce the size of the problem, only preference spans of orders are considered. An initial solution to the problem is based on the heuristic in Algorithm 12. This initial solution enables the EO algorithm to reach better solutions in fewer iterations. The implementation of the EO algorithm is based on the GEO approach used by De Souse *et al.* [31]. Each gene (starting location of an order) is considered and toggled. By toggling the starting location of an order an attempt is made to find a more suitable starting location that will benefit the solution quality by considering implications that may arise as the algorithm progresses.

Before genes are altered, the maximal cut C of the current configuration is determined together with the number of maximal cuts in the current system N_C. Similarly, if a location exists containing one less span than the maximal cut it is also identified, say $C-1$, and the number of locations that contain a cut of $C-1$, say N_{C-1}. For each gene the current starting location is compared to all other starting locations (preference spans) of that order. An alternative starting location for an order is identified that may improve the solution quality or slightly reduce the solution quality based on a number of considerations.

During an iteration genes that will be considered for alteration are identified. Orders that do not currently traverse any of the N_C locations with maximal cut, are not considered. These orders do not influence the maximal cut — the maximal cut C and the number of locations containing the maximal cut N_C cannot be reduced by altering these genes. Thus only genes that currently traverse maximal cut(s) are considered. Each of these genes g_x are awarded with a change in performance Δf_x, where the current preference span of the order g_x is compared with all its other preference spans and the best alternative preference span is identified.

The change in performance is influenced by a number of factors. If an alternative preference span for a gene g_x is considered the locations that currently traverse and not traverse the maximal cuts(s) are considered. The fitness Δf_x receives a score (with a numerical value of 1) for each maximal cut that the current preference span has to traverse and the proposed preference span does not have to visit. Similarly, the fitness function receives a score of -1 when the current preference span does not traverse a maximal cut but the proposed preference span does.

The fitness is also influenced by the possibility of creating maximal cuts at other locations. If the preference span of a gene g_x currently does not traverse a location containing one less span than the maximal cut (cut of $C-1$), but does traverse this location with the proposed preference span, the fitness is awarded with a score of -1, since the cut of this location will

increase to the current maximal cut (C). This is done in order to avoid creating maximal cuts at other locations.

Once the change in performance is calculated for each considered gene x, the genes are ranked in decreasing order of Δf_x. Each gene g_x is awarded a probability of

$$P(r_x) = r_x^{-\tau}, \tag{4.37}$$

where r_x denotes the rank of gene x. These probabilities are then normalised to create a probablity distribution where a higher likelihood exists of altering a gene that will improve the solution quality. A random number is used to determine the alteration. When the alteration is made, the cuts at each location is updated.

Algorithm 16: Generalised extremal optimisation.

Data: All the preference spans for each order considered and a set of all the order $\mathcal{S}$, the value of τ and the stopping criteria s_o.
Result: A vector $\boldsymbol{s}$ of starting locations awarded to each individual order.
Find an initial solution $\boldsymbol{s}$ by means of the greedy reducing heuristic and f_x^*;
repeat
  for $x \leftarrow 0$ *to* n **do**
    Toggle the value of each gene g_x and retireve the solution $f'_x = \boldsymbol{h} \cdot \boldsymbol{g}'$;
    Calculate Δf_x;
  end
  Rank the genes in a decreasing order in terms of Δf_x for $x = 1, 2, \ldots, n$;
  Use the mutating probalilty distribution in (4.37) to determine a gene for mutation;
  if $f^* < f'$ **then**
    Let $f^* = \boldsymbol{h} \cdot \boldsymbol{g}'$ be the new best solution;
  end
until *no improvement in the fitness is found in* s_o *iterations*;

When an alteration is made with a possitive fitness, the solution may remain unchanged. In some cases the solution quality may even decrease. These situations exist when a picking line contains multiple maximal cuts. Some maximal cuts are reduced but some are increased (the GEO algorithm will ensure that more maximal cuts are reduced than the number of maximal cuts increased). Algorithm 16 displays the pseudo code implementation of the GEO implementation used.

Table 4.19 diplays the results for the GEO algorithm when the value of τ is tested at 1, 1.25, 1.5 and 1.75. The stopping criteria is tested for situations where no improvement was found in 100, 200 and 300 iterations. The best results are found for the GEO when $\tau = 1.5$ and when the stopping criteria is set at 200 iterations without improvement. Table 4.20 displays the computational time in milliseconds for the results in Table 4.19. More computational time is required when the stopping criteria is increased and when the value of τ is increased.

4.9 Results summary

In this section all the results obtained from the exact and heuristic approaches to solve the OSP are summarised. Table 4.21 displays the summarised results for non-metaheuristic approaches used to solve the OSP. Results from the NSO are displayed, since the NSO outperformed the NSOM and NSOP for all the data sets considered. The best results from the scope and ranking algorithms are also displayed (Scope). Solutions of the algorithm that assigns spans according

Data set	Size (O, L)	Maximal cut approach	Stopping criteria 100				Stopping criteria 200				Stopping criteria 300			
			$\tau = 1$	$\tau = 1.25$	$\tau = 1.5$	$\tau = 1.75$	$\tau = 1$	$\tau = 1.25$	$\tau = 1.5$	$\tau = 1.75$	$\tau = 1$	$\tau = 1.25$	$\tau = 1.5$	$\tau = 1.75$
A	(1262,49)	1232	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233	1233
B	(1264,54)	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226
C	(1265,51)	1161	1169	1168	1167	1166	1169	1168	1167	1167	1169	1168	1168	1167
D	(1263,56)	1072	1079	1082	1080	1079	1077	1079	1080	1080	1080	1080	1079	1081
E	(1264,51)	1069	1075	1073	1073	1074	1075	1073	1073	1073	1074	1073	1073	1073
F	(1258,55)	1025	1044	1048	1044	1046	1044	1042	1040	1043	1041	1040	1042	1044
G	(1258,53)	1005	1032	1032	1035	1042	1022	1027	1026	1032	1021	1021	1027	1029
H	(1244,54)	992	1014	1021	1014	1018	1011	1009	1008	1009	1012	1009	1008	1010
I	(1260,56)	955	983	978	983	987	981	976	977	981	977	977	973	977
J	(1264,56)	947	989	981	976	984	975	975	971	976	970	973	975	971
K	(943,63)	259	267	267	269	275	264	264	263	263	265	263	264	265
L	(846,56)	232	236	237	235	237	236	238	237	236	236	237	237	237
M	(728,51)	152	160	158	158	156	159	157	160	159	160	160	162	160
N	(733,55)	125	129	128	129	133	132	131	128	128	129	132	131	130
O	(396,63)	90	93	93	93	93	93	93	93	93	93	93	93	92
P	(574,48)	80	83	81	82	83	82	82	83	83	83	83	83	83
Q	(242,64)	45	45	45	45	45	45	45	45	45	45	45	45	45
R	(158,55)	14	14	14	14	14	14	14	14	14	14	14	14	14
S	(89,42)	9	9	9	9	9	9	9	9	9	9	9	9	9
T	(82,51)	8	8	8	8	8	8	8	8	8	8	8	8	8
U	(90,48)	7	8	8	8	8	8	8	8	8	8	8	8	8
V	(80,56)	6	6	6	6	6	6	6	7	6	6	6	6	7
Total		11711	11902	11896	11887	11922	11869	11863	11856	11872	11859	11858	11864	11869

Table 4.19: *Results (in number of cycles traversed) for the extremal optimisation when the stopping criteria is varied at 100, 200 and a 300 iterations without an improvement in solution quality. The value of τ is varied at 1, 1.25, 1.5 and 1.75. The number of orders (O) and locations (L) are displayed for each data set.*

Data set	Size (O, L)	Stopping criteria 100				Stopping criteria 200				Stopping criteria 300			
		$\tau = 1$	$\tau = 1.25$	$\tau = 1.5$	$\tau = 1.75$	$\tau = 1$	$\tau = 1.25$	$\tau = 1.5$	$\tau = 1.75$	$\tau = 1$	$\tau = 1.25$	$\tau = 1.5$	$\tau = 1.75$
A	(1262,49)	1096	1001	1147	1358	1516	1541	1594	1542	2148	1999	2219	1987
B	(1264,54)	739	807	767	805	1218	1165	1232	1145	1587	1663	1715	1545
C	(1265,51)	806	764	791	858	1337	1240	1308	1308	1827	1891	1886	1803
D	(1263,56)	4424	3900	4215	5755	6418	5478	6178	5783	5434	5309	6293	6283
E	(1264,51)	1925	2112	2125	2332	2289	2300	2473	2606	3023	3049	2963	3455
F	(1258,55)	2546	2271	3394	4612	3860	4777	5075	5755	5338	7263	6700	6446
G	(1258,53)	5064	6534	5265	3534	10841	8946	7903	8932	13660	17536	12910	11530
H	(1244,54)	4781	2486	6043	4861	6809	8221	11486	9155	7778	11129	12036	11185
I	(1260,56)	4840	6416	5089	4882	6994	7723	9584	8799	9267	10576	11244	10660
J	(1264,56)	1950	2945	5082	3117	8136	7285	10943	5565	11872	10657	8256	12149
K	(943,63)	985	695	941	467	1383	1300	718	1117	1378	2099	2059	2601
L	(846,56)	456	316	451	444	636	504	581	819	826	685	909	870
M	(728,51)	150	158	166	158	222	238	220	256	330	327	340	332
N	(733,55)	160	144	157	115	211	199	213	305	313	306	318	308
O	(396,63)	57	66	48	49	82	77	80	82	105	102	108	113
P	(574,48)	79	84	77	73	144	127	130	125	178	180	189	181
Q	(242,64)	34	26	23	35	36	35	35	38	46	49	56	55
R	(158,55)	13	13	18	12	16	15	27	28	21	23	21	21
S	(89,42)	8	8	7	8	11	9	10	8	11	10	11	10
T	(82,51)	7	8	8	8	10	10	8	9	10	10	10	11
U	(90,48)	10	9	8	7	11	18	9	9	25	25	12	13
V	(80,56)	10	11	9	8	11	11	10	12	14	13	12	11
Total		30140	30774	35831	33498	52191	51219	59817	53398	65191	74901	70267	71569

Table 4.20: *Computational times, in milliseconds, for the extremal optimisation when the stopping criteria is varied at 100, 200 and a 300 iterations without an improvement in solution quality. The value of τ is varied at 1, 1.25, 1.5 and 1.75. The number of orders (O) and locations (L) are displayed for each data set.*

Data set	Size (O, L)	Maximal cut	NSO	Scope	RS	GAP	Pep
A	(1262,49)	1232	1253	1255	**1234**	1239	1301
B	(1264,54)	1226	1243	1242	**1232**	1255	1255
C	(1265,51)	1161	**1200**	**1200**	1206	1220	1254
D	(1263,56)	1072	**1120**	1130	1195	1205	1224
E	(1264,51)	1069	**1132**	1147	1148	1202	1234
F	(1258,55)	1025	**1072**	1109	1166	1193	1177
G	(1258,53)	1005	**1076**	1096	1093	1197	1222
H	(1244,54)	992	**1056**	1065	1066	1096	1242
I	(1260,56)	955	**1018**	1049	1037	1095	1227
J	(1264,56)	947	**999**	1007	1018	1070	1202
K	(943,63)	259	367	340	**285**	293	640
L	(846,56)	232	322	289	254	**248**	615
M	(728,51)	152	260	219	202	**182**	457
N	(733,55)	125	209	170	145	**135**	461
O	(396,63)	90	200	201	**110**	111	224
P	(574,48)	80	148	105	**102**	103	324
Q	(242,64)	45	110	107	**54**	57	142
R	(158,55)	14	28	21	**17**	18	82
S	(89,42)	9	12	**10**	11	13	40
T	(82,51)	8	15	11	11	**9**	36
U	(90,48)	7	14	**8**	**8**	11	40
V	(80,56)	6	10	8	**7**	8	38
Total		11711	12864	12789	12601	12960	15437

Table 4.21: *Summary of the number of cycles travelled when the OSP is solved with various heuristic algorithms. The next shortest order (NSO), the best solution obtained from the scope and ranking algorithms (Scope), the best solution obtained from the ratio spans (RS) and the best solution to the generalised assignment problem (GAP). The lower bound of the number of cycles travelled for each data set is indicated by the maximal cut approach. The number of orders (O) and locations (L) are displayed for each data set. The algorithm(s) that obtained the best results for each data set is indicated in boldface.*

Data set	Size (O, L)	Maximal cut	TS	SA	GA	GEO	Pep
A	(1262,49)	1232	**1233**	**1233**	**1233**	**1233**	1301
B	(1264,54)	1226	**1226**	**1226**	**1226**	**1226**	1255
C	(1265,51)	1161	1178	1168	**1163**	1167	1254
D	(1263,56)	1072	1114	1089	1098	**1080**	1224
E	(1264,51)	1069	1124	1083	1085	**1073**	1234
F	(1258,55)	1025	1062	**1038**	1053	1040	1177
G	(1258,53)	1005	1069	1042	1058	**1026**	1222
H	(1244,54)	992	1049	1021	1028	**1008**	1242
I	(1260,56)	955	1017	988	1002	**977**	1227
J	(1264,56)	947	1005	982	993	**971**	1202
K	(943,63)	259	293	276	269	**263**	640
L	(846,56)	232	247	241	236	**237**	615
M	(728,51)	152	171	155	**154**	160	457
N	(733,55)	125	134	132	**128**	**128**	461
O	(396,63)	90	101	**93**	**93**	**93**	224
P	(574,48)	80	84	**82**	**82**	83	324
Q	(242,64)	45	53	**45**	**45**	**45**	142
R	(158,55)	14	16	**14**	**14**	**14**	82
S	(89,42)	9	10	**9**	**9**	**9**	40
T	(82,51)	8	**8**	**8**	**8**	**8**	36
U	(90,48)	7	**8**	**8**	**8**	**8**	40
V	(80,56)	6	**6**	**6**	**6**	7	38
Total		11711	12208	11939	11991	11856	15437

Table 4.22: *Summary of the number of cycles travelled when the OSP is solved by means of metaheuristics. The best results of the tabu search (TS), simulated annealing (SA), genetic algorithm (GA) and the generalised extremal optimisation (GEO) are presented, together with the lower bound (LB) and Pep's current solution (Pep). The lower bound of the number of cycles travelled for each data set is indicated by the maximal cut approach. The number of orders (O) and locations (L) are displayed for each data set. The algorithm(s) that obtained the best results for each data set is indicated in boldface.*

to preference ratios are displayed in the column named RS. The best solution of the heuristic methods used to solve the generalised assignment problem (GAP) is displayed for each data set.

Overall, the best heuristic solutions are obtained by the RS, as this is the heuristic that results in the least number of cycles travelled over all the data sets. The NSO also present good solutions with respect to the maximal cut approach, and the method used by Pep performs the worst for all the data sets considered.

Table 4.22 displays the solutions obtained from the metaheuristic approaches used to solve the OSP. The TS, SA, GA and GEO are compared to the lower bound of the maximal cut approach and the solution quality obtained by Pep. The GA approach where chromosomes are modelled as starting locations in §4.8.5 are considered, since it outperforms the alternative approach. The best overall results are delivered by the GEO, followed by the SA and then the GA. The metaheuristic approaches outperform the heuristics, however, they require on average more computational time.

Figure 4.29 displays bar charts of all the results contained in Table 4.21. This presents a summary of the best results obtained from the algorithms used to solve the OSP. Similarly, Figure 4.30 displays bar charts of the results in Table 4.22 of the metaheuristic implementations for the OSP.

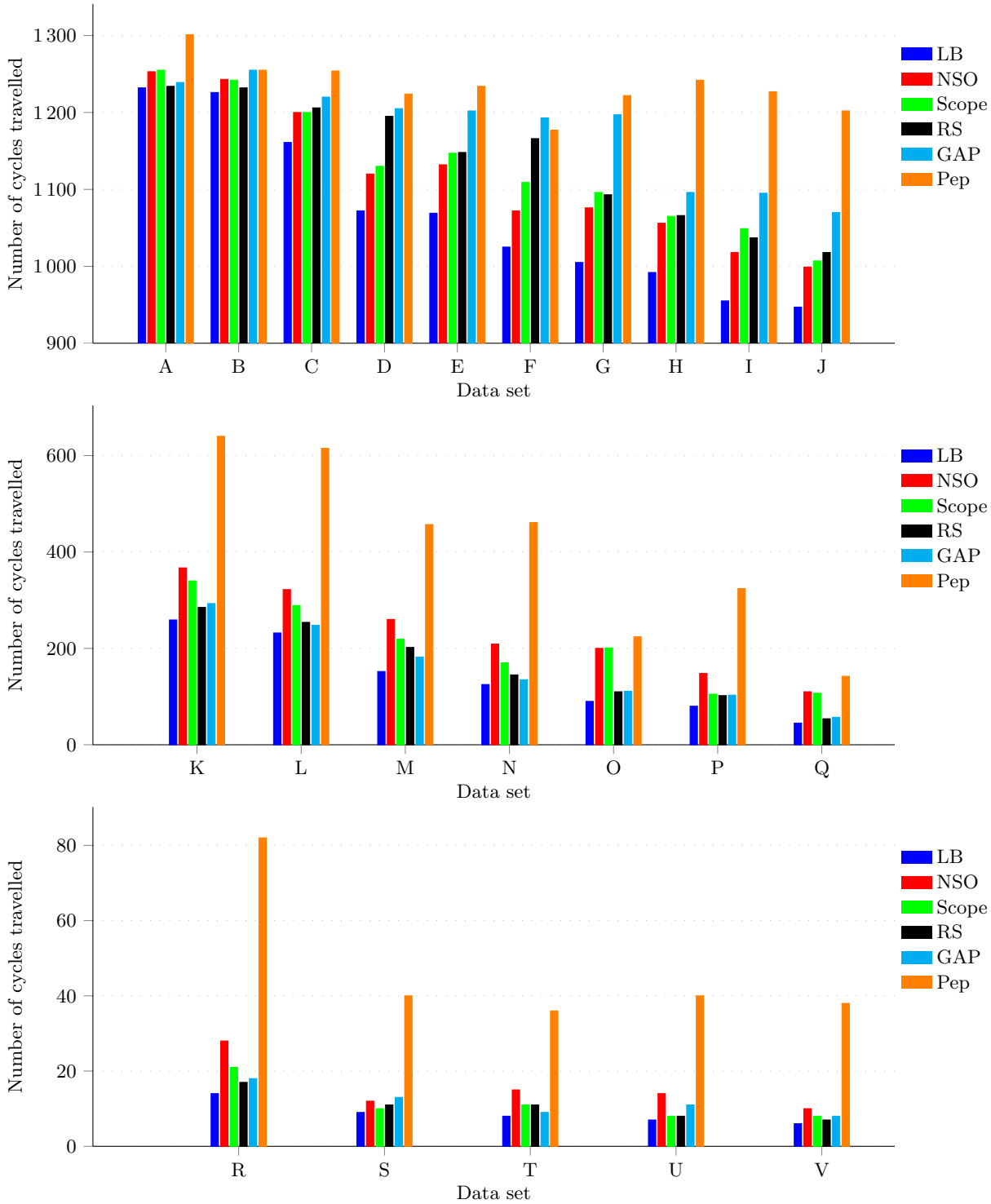


Figure 4.29: *A bar chart displaying the results (cycles travelled) obtained in Table 4.21 for each of the 22 data sets considered. These results obtain the best solutions found for all the algorithms used to solve the OSP. The lower bound (LB) is indicated by the results for the maximal cut formulation. The next shortest order (NSO), the best solution obtained from the ratio spans (RS), the best solution obtained from the scope and ranking algorithms (Scope) and the best solution to the generalised assignment problem (GAP). Finally, the method in which Pep would solve the OSP (Pep) is also indicated.*

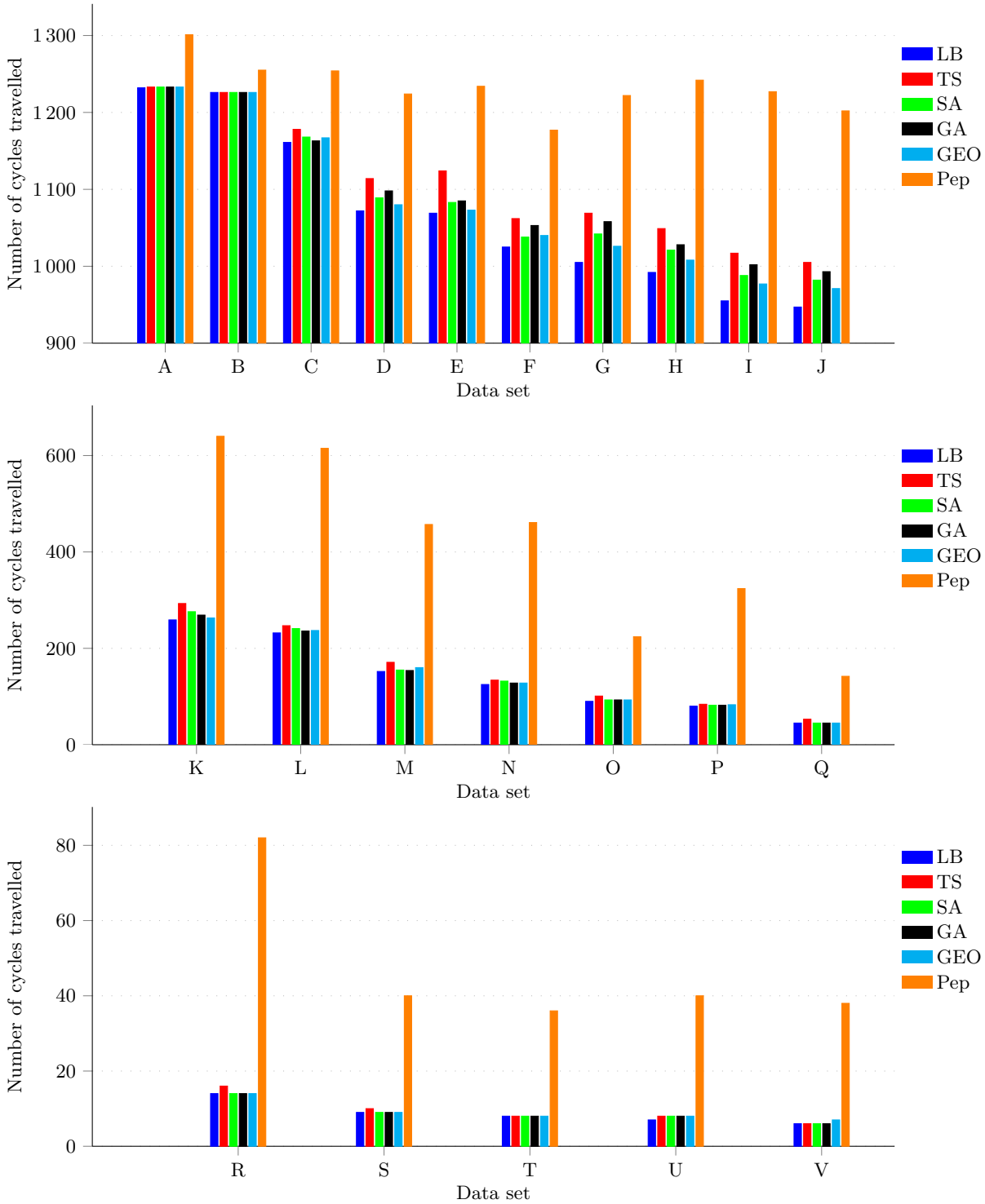


Figure 4.30: *A bar chart displaying the results (cycles travelled) obtained in Table 4.22 for each of the 22 data sets considered. These results obtain the best solutions found for all the algorithms used to solve the OSP. The lower bound (LB) is indicated by the results for the maximal cut formulation. The tabu search (TS), simulated annealing (SA), genetic algorithm (GA) and generalised extremal optimisation (GEO) approaches are compared with the results incurred by Pep are also indicated.*

4.10 Chapter summary

In this chapter a variety of methodologies are considered to solve the OSP. Exact solutions were considered in §4.2. Initailly the OSP is formulated as an equality generalised traveling salesman problem. Matthews & Visagie [77] formulated the maximal cut formulation which is used as a benchmark to evaluate solution quality.

Greedy heuristic algorithms of sequencing orders are presented in §4.4. The NSO, NSOP and NSOM were constructed and the NSO outperformed the other heuristics for all the data sets considered. From all the heuristics considered to solve the OSP, the NSO is the fastest algorithm to solve the OSP.

The greedy scope and ranking algorithms introduced in §4.5 performed better than the greedy heuristics in §4.4, delivering better results for medium and small data sets. Two variations of the greedy, scope and ranking algorithms were developed focussing on different important aspects, that address the shortcommings of the greedy heuristics.

A manner of allocating a number of spans of each order and then ranking each span according to a preference is described in §4.6. Two criteria for ranking the spans of each order is considered. However, the RS provided better solutions than the NSO for some of the data sets. The computational time was higher on average to that of the NSO, but is solvable in a reasonable amount of time.

A heuristic implementation of a generalised assignment problem is presented in §4.7. The generalised assignment problem is computationally more expensive, however, it is able to find better solutions for the medium and small data sets compared to the greedy heuristics.

This is followed by metaheuristic order sequencing in §4.8. Data structures used and a greedy starting heuristic are introduced, whereafter a tabu search, simulated annealing, genetic algorithm and generalised extremal optimisation approach are used to solve the OSP. A summary of the computational results are displayed in §4.9 where various algorithms were tested on a series of data sets. A GEO metaheuristic outperforms all other methods considered to solve the OSP.

CHAPTER 5

SKU location problem

Contents

5.1 A lower bound . . . 105
5.2 SKU location in literature . . . 107
5.2.1 Organ-pipe arrangement . . . 107
5.2.2 Greedy ranking and partitioning . . . 108
5.3 Ant colony algorithm for locating similar SKUs . . . 110
5.3.1 Behaviour of social insects . . . 110
5.3.2 Ant system . . . 111
5.3.3 Extensions of the ant system . . . 112
5.4 (Agglomerative) Clustering algorithms . . . 114
5.4.1 The single-link method . . . 115
5.4.2 The average-link method . . . 115
5.4.3 The centroid method . . . 117
5.4.4 Ward's algorithm . . . 117
5.4.5 When to stop the clustering process . . . 118
5.4.6 Implementation of clustering algorithms . . . 119
5.5 Heuristic clustering methods . . . 119
5.5.1 Clustering according to orders . . . 120
5.5.2 Clustering according to SKU frequency . . . 120
5.6 Computational results . . . 121
5.7 Random SLP configurations . . . 122
5.8 Chapter summary . . . 124

The SKU location problem (SLP) may be described as the allocation of SKUs within a picking line, once these SKUs have already been assigned to a particular picking line. Pep operates on a principle that SKUs may not be placed in more that one distinct location within a single picking line.

5.1 A lower bound

The SLP may be interpreted as a routing problem together with an assignment problem. SKUs assigned to a picking line, must be placed in various locations in such a way that when sequencing

the orders, minimal distance is travelled by the pickers. The placement of the SKUs have a direct impact on the sequence in which orders may be executed.

The following input data are required to formulate the SLP. Let

$\mathcal{O}_k$	be the set of SKUs that have to be collected in order k,
$\lvert\mathcal{O}_k\rvert$	be the number of SKUs that have to be collected in order k,
n	be the number of orders,
m	be the number of locations and
T	be the number of SKUs assigned to the picking line.

A number of variables required in modelling this problem are defined. Let

c_{ki}	be equal to 1 if order k passes location i and 0 otherwise,
x_i^t	be equal to 1 if SKU t is positioned at location i and 0 otherwise,
s_{ki}	be equal to 1 if order k starts at location i and 0 otherwise,
s_k	be the location at which order k starts order picking,
e_k	be the location at which order k ends order picking,
O_{ki}	be the i^{th} location that order k has to visit,
y_{ki}	be a set of auxilary binary variables and
C	be the maximal cut.

The problem may be formulated as a mixed integer programming (MIP) problem. The objective then is to minimise the maximal cut, *i.e.* to

$$\text{minimise } C \tag{5.1}$$

$$\text{subject to } \sum_{i=1}^{m} x_i^t = 1 \qquad t = 1, 2, \ldots, T, \tag{5.2}$$

$$\sum_{t=1}^{T} x_i^t = 1 \qquad i = 1, 2, \ldots, m, \tag{5.3}$$

$$\sum_{t \in \mathcal{O}_k} \sum_{i=1}^{m} i x_i^t = O_{ki} \qquad k = 1, 2, \ldots, n, \tag{5.4}$$

$$\sum_{i=1}^{m} s_{ki} = 1 \qquad k = 1, 2, \ldots, n, \tag{5.5}$$

$$\sum_{i=1}^{m} i s_{ki} = s_k \qquad k = 1, 2, \ldots, n, \tag{5.6}$$

$$O_{k1} - s_k \leq M y_{k|\mathcal{O}_k|} \qquad k = 1, 2, \ldots, n, \tag{5.7}$$

$$\max_{i=1,\ldots,|\mathcal{O}_k|} \{s_k - O_{ki}, 0\} \leq M y_{ki} \qquad k = 1, 2, \ldots, n, \tag{5.8}$$

$$\sum_{i=1}^{|\mathcal{O}_k|} O_{ki} y_{ki} = e_k \qquad k = 1, 2, \ldots, n, \tag{5.9}$$

$$\sum_{i=s_k}^{e_k} c_{ki} = 1 \qquad \text{if } s_k < e_k, \; k = 1, 2, \ldots, n, \tag{5.10}$$

$$\sum_{i=s_i}^{m} c_{ki} = 1 \qquad \text{if } s_k > e_k, \; k = 1, 2, \ldots, n, \tag{5.11}$$

$$\sum_{i=1}^{e_k} c_{ki} = 1 \qquad \text{if } s_k > e_k, \; k = 1, 2, \ldots, n, \tag{5.12}$$

$$\sum_{k=1}^{n} c_{ki} \leq C \qquad i = 1, 2, \ldots, m, \tag{5.13}$$

$$x_i^t \in \{0, 1\} \qquad t = 1, 2, \ldots, T, \; i = 1, 2, \ldots, m, \tag{5.14}$$

$$y_{ki} \in \{0, 1\} \qquad k = 1, 2, \ldots, n, \; i = 1, 2, \ldots, m, \tag{5.15}$$

$$s_{ki}, c_{ki} \in \{0, 1\} \qquad k = 1, 2, \ldots, n, \; i = 1, 2, \ldots, m. \tag{5.16}$$

The objective function (5.1) minimises the maximal cut. Constraint set (5.2) ensures that each SKU is assigned to one location, while constraint set (5.3) ensures that each location contains a single SKU. Constraint set (5.4) calculates all the locations each order needs to visit. Constraint set (5.5) ensures that each order has a unique starting location. Constraint set (5.6) assigns a starting location to each order. Constraint set (5.7) and (5.8) is used to aid in determining the ending location of each order. Constraint set (5.9) assigns an ending location to each order. Constraint set (5.10) – (5.12) calculates the cut at each location for each order. Constraint set (5.13) calculates the maximal cut.

The size of this formulation consists of $4nm+mT+2n+1$ variables of which $3nm+mT$ are binary variables and $T + 2m + 9n$ constrains. This model presents a significant reduction in number of variables and is used to solve the OSP with an exact formulation (see §4.3.2). However, it should be noted that this formulation only considers the maximal cut. The maximal cut formulation only considers locations contained in orders as possible starting locations, From these starting locations the sequence in which orders are picked must still be determined. However, it reduces the complexity of the formulation considerably. This model presents a lower bound for the SLP. A feasible solution may be obtained by using the subtour generating heuristic in §4.3.2.

5.2 SKU location in literature

Optimal SKU location in literature mainly focusses on the placement of SKUs within a bi-directional carousel system. Vickson and Fujimoto [110] considers a carousel system, containing a series of locations (referred to as *bins*). Each location contains several *shelves* (equal in number and equally spaced). Each shelf may contain a single SKU. Vickson & Lu [111] considers SKU location in one-dimensional storage racks. They consider a system where future orders are unknown in advance. The placement of the various SKUs are determined together with the location at which the server (a single picker) is to be situated. The following two manners of adressing the SLP are used most often in industry for uni- and bi-directional carousel systems.

5.2.1 Organ-pipe arrangement

An organ-pipe arrangement (OPA) is built by placing the most frequent SKU on the carousel, the next most frequent SKU immediately adjacent, the next most frequent SKU to the other

side, and so on, where the frequency of a SKU refers to the number of orders requiring that SKU [12]. The OPA was formally introduced by Lim *et al.* [72] and the optimality of OPA, for bi-directional carousel systems, for certain types of location problems involving linear or circular patterns of points was proved by Bergmans [18]. The location problem of these carousels may be divided into two subproblems. An inital *grouping problem* addresses how groups of SKUs are packed into one location[1]. The *location problem* is then used to address the problem of assigning groups to various bay locations. Figure 5.1 displays an example of how the OPA arranges SKUs in terms of their frequency, where the frequency refers to the number of orders requiring a specific SKU.

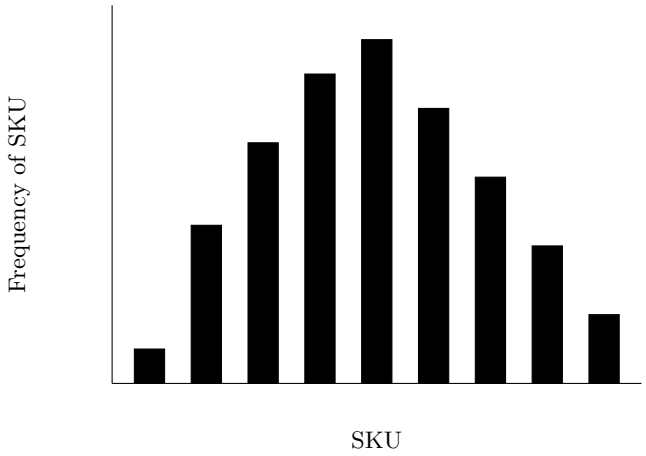


Figure 5.1: *An example of using OPA to solve the SLP. The frequency of a SKU refers to the number of orders requiring that SKU.*

Vickson and Fujimoto [110] proved the OPA to be an optimal location problem for the case in which a bi-directional carousel is used for orders of size one. Bartholdi *et al.* [12] states that the OPA is probably an effective strategy whenever typical orders visit few locations on any carousel. Pseudo code for the OPA is displayed in Algorithm 17.

5.2.2 Greedy ranking and partitioning

The greedy ranking and partitioning algorithm (GRP) was introduced by Lim *et al.* [72], Vickson and Fujimoto [110] and Vickson and Lu [111]. The GRP sorts the SKUs in decreasing order in terms of the demand frequencies. If the locations contain multiple shelves, the SKUs are partitioned into groups equal to the size of a single shelve. Once a group has been formed it is placed in the next location that has not received a group[2]. Figure 5.2 displays an example of how the GRP arranges SKUs.

Hassini [53] proved that the GRP may optimally partition n SKUs into m groups ($n > m$) when any *one-dimensional* carousel is used. A *two dimensional* carousel consists of a number of locations where each location contains a number of shelves [54]. The shelves are usually packed vertically and the locations are positioned in a horisontal fashion around the carousel. *Horisontal motion* refers to the situation in which the correct location is allocated and a *vertical*

[1] In the case of Pep, only one SKU may be allocated to a location. Conventional carousels often contain multiple *shelves* in one location. The grouping problem does not exist in the case of Pep, which simplifies the SLP in Pep's situation.

[2] In Pep's case a group will consist of a single SKU. Essentially SKUs are then ranked in decreasing order in terms of their demand frequencies and then packed onto the picking line in that sequence.

Algorithm 17: Organ pipe algorithm.

Data: A set of orders, $\mathcal{O}_k$ for each order k and a set of SKUs $\mathcal{U}$.
Result: An ordered set $\mathcal{U}'$ containing the sequence in which the SKUs should be placed into the picking line.
for $k \leftarrow 1$ **to** n **do**
  for $i \leftarrow 1$ **to** m **do**
    if $i \in \mathcal{O}_k$ **then**
      $U_i \leftarrow U_i + 1$;
    end
  end
end
sort the values of U_i in decreasing order for $i = 1, 2, \ldots, m$;
for $i \leftarrow 1$ **to** $|\mathcal{U}|$ **do**
  if $i \,(\mathrm{mod}\, 2) = 0$ **then**
    $U'_{\left\lceil \frac{|\mathcal{U}|+i}{2} \right\rceil} = U_i$;
  end
  else
    $U'_{\left\lceil \frac{|\mathcal{U}|-i}{2} \right\rceil} = U_i$;
  end
end

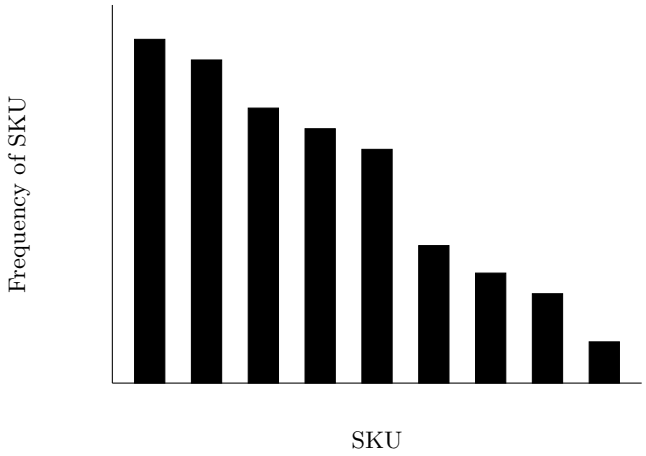


Figure 5.2: *An example of using greedy ranking and partitioning algorithm to solve the SLP. The frequency of a SKU refers to the number of orders requiring a SKU.*

motion denotes the situation in which the correct shelf is identified within a particular location. A one-dimensional carousel refers to the situation in which each location contains a single shelf[3]. Vertical motions are thus neglected. Hassini [53] also proved that the GRP optimally partitions n SKUs into m groups ($n > m$), for a one-dimensional carousel irrespective of its operations and direction of rotation. This, however, is for the case in which orders are not known in advance. Algorithm 18 displays the pseudo code for the GRP.

Vickson & Lu [111] used the GRP in an anticipatory operation of orders in a one-directional carousel system. They obtained simultaneous optima for the (picker's) server's home base and the SKUs locationed within a rack.

[3]All picking lines in Pep uses a one-dimensional system.

Algorithm 18: Greedy ranking and partitioning algorithm.

Data: A set of orders, $\mathcal{O}_k$ for each order k and a set of SKUs $\mathcal{U}$.
Result: An ordered set $\mathcal{U}$ containing the sequence in which the SKUs should be placed into the picking line.

```
// Arrange the set of SKUs in decreasing order in terms of their demand frequencies and place
   the arranged orders in an ordered set U.
```

for $k \leftarrow 0$ **to** n **do**
  for $i \leftarrow 0$ **to** m **do**
    if $i \in \mathcal{O}_k$ **then**
      $U_i \leftarrow U_i + 1$;
    end
  end
end
sort the values of U_i in decreasing order for $i = 1, 2, \ldots, m$;

5.3 Ant colony algorithm for locating similar SKUs

An alternative approach to the SLP is to place SKUs next to one another, on a picking line, that share a common pool of orders that have to visit these SKUs. The possibility exists that orders that must visit a number of SKUs will find these SKUs close to one another within a picking line, reducing distance travelled between orders.

This problem may be formulated as a TSP where each entry in the distance matrix d_{ij} denotes the number of orders that have to visit both SKU i and j. This problem may then be solved using an exact formulation. However, LINGO 11 [73] may require an excess of 1 hour to solve a TSP containing 50 SKUs (depending on the nature of the cost matrix) on an Intel(R) Core(TM)2 Duo 3 GHz with 3.7 GB ram. It is necessary to find an algorithm to solve this problem (for an average sized problem faced by Pep) within reasonable time.

Ant colony algorithms form a class of algorithms that are inspired from the collective behaviour of trail deposit and follow-up, observed in the ant colonies [34]. A number of agents (*ants*) communicate indirectly via modifications of their environment (*trails of pheromones*). The ants form a solution for a problem that is based upon the collective endeavours of all the ants.

5.3.1 Behaviour of social insects

The ants form a self-organisational system. Self-organisation is the process where a structure or pattern appears in a system without a central authority or external element imposing it through any interference. A formal definition has been proposed by Camazine *et al.* [24]:

> Self-organization is a process in which a *pattern* at the global level of a system *emerges* solely from numerous interactions amoung *lower-level* components of the system. Moreover, the rules specifying interactions amoung the system's components are executed using only local information, without reference to the *global* pattern.

A "pattern" indicates some form of organised positioning of objects, while an "emerging" property indicates an unforseen characteristic of the system that arises from interactions of components within the system. It is necessary to understand the interactions of the components in order to determine how patterns are produced. Two basic instruments for interpreting patterns consists of *feedback* and *information flow*. Feedback may either be of a positive or a negative nature. Positive feedback are processes which result in reinforcing certain actions. Positive

feedbacks may lead to fluctuations in the system. Negative feedback acts as stabilisers to the system and is used to offset fluctuations that may exist in the system.

Ant colony optimisation uses a self-organising system, where each ant only has a *local* vision of its environment. Decisions are made solely by the ants and their environment. Furthermore, ant colony algorithms makes use of a *dense heterarchy*. The notion of a dense heterarchy, initially introduced by Wilson and Hölldobler [119], used to describe the organisation of social insects (ant colonies in particular), where activities on all levels of operations influence all parties involved in the system. This manner of interacting is contrary to a hierarchy, where different levels of authority exist. Activities on lower levels of the hierarchy does not necessarily influence the higher levels. However, actions on higher levels have significant impact on lower levels.

Stigmergy froms the basis for communication within ant colony systems. It may be explained as modifying the environment of the system in order to communicate. Ant colony algorithms are based on the observation that social insects may, in general, solve complex problems in a natural way. Ants possess a characteristic that enables them to employ substances (called *pheromones*) to communicate. They are able to perceive these substances with their antennas. Ants deposit pheromones on the path they travel, by means of a gland, which forms an odorous trail. Ants use pheromones to lay a trail of their path. Ants may use these pheromones to find a trail from the nest to a source of food. A colony of ants may (under certain conditions) be able to find the shortest path to a source of food, without having knowledge of the *global* vision of the path [34]. If more ants follow the shortest path to the food source, a larger quantity of pheromones will be present on this path, increasing the probability of other ants also following this path with the aim of locating food. If paths become neglected that have been travelled previously by ants, the pheromones on these paths *evaporate*.

5.3.2 Ant system

An ant colony algorithm may be used to solve a TSP exactly. The goal is to generate a hamiltonian cycle that maximises the costs between any two SKUs. The ant system (AS) was one of the first implementations of ant colony algorithms and was used to solve TSPs. During each iteration t ($1 \leq t \leq t_{\max}$), each ant a ($k = 1, 2, \ldots, A$) builds a complete path of M stages. For every ant, during each iteration, the path between city i and j depends on

1. the list of SKUs that have already been visited, when ant a is currently at SKU i, J_i^a,
2. the reciprocal of the distance between cities i and j, $\eta_{ij} = \frac{1}{d_{ij}}$, which is called the *visibility* between two SKUs. This contributes to directing ants to SKUs containing similar orders, and to avoid SKUs having fewer orders in common,
3. the quantity of the pheromone on the edge connecting to SKUs i and j, τ_{ij} — this is called the *intensity of the trail*. This defines the attraction of the path and changes with each passage of an ant.

The rule of displacement (also called "random proportional transition rule" by Bonabeau *et al.* [20]) may be stated as

$$p_{ij}^k = \begin{cases} \frac{(\tau_{ij}(t))^\alpha \cdot (\eta_{ij}^\beta)}{\sum_{l \in J_i^a} (\tau_{il}(t))^\alpha \cdot (\eta_{il})^\beta} & \text{if } j \in J_i^a \\ 0 & \text{if } j \notin J_i^a, \end{cases} \tag{5.17}$$

where α and β are two parameters assigning the relative importance of the trail intensity, $\tau_{ij}(t)$, and visibility, η_{ij}. When $\alpha = 0$ only the visibility of a SKU is considered, and the SKU with with most orders in common with the current SKU will be selected. Similarly, if $\beta = 0$ only the pheromones are used to guide the ants. In order to balance visibilty and intensity a compromise between visibility and intensification must be reached. After an iteration, each ant leaves a quantity of pheromones

$$\Delta\tau_{ij}^{a}(t) = \begin{cases} \frac{W}{L^{a}(t)} & \text{if } (i,j) \in T^{a}(t) \\ 0 & \text{if } (i,j) \notin T^{a}(t) \end{cases} \tag{5.18}$$

on its course, where $T^a(t)$ is the path traversed by ant a during iteration t, $L^a(t)$ is the length of the path traversed by ant a and W is a fixed parameter.

Pheromones need to evaporate from undesirable edges for the algorithm to "forget" poor solutions. An update rule for trails may be

$$\tau_{ij} = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}(t), \tag{5.19}$$

where $\Delta\tau_{ij}(t) = \sum_{a=1}^{A}\tau_{ij}^{a}(t)$ and ρ is the evaporation rate. The initail quantity of pheromone on the edges is a uniform distribution. Algorithm 19 displays the pseudo code implementation for a basic AS metaheuristic when applied to the SLP.

Algorithm 19: Ant System.

Data: A distance matrix D.
Result: A sequence in which orders should be placed on a picking line.
for $t \leftarrow 1$ **to** $t_{\max}$ **do**
  for $a \leftarrow 1$ **to** A **do**
    Select a SKU at random;
    for *Each SKU that has not been visited* **do**
      Select a SKU j, from list J_i^a of the remaining SKUs from (5.17);
    end
    Calculate pheromones employed $\Delta\tau_{ij}^{a}(t)$ on the path $T^a(t)$ in accordance with (5.20);
  end
  Evaporate trails according to (5.19);
end

5.3.3 Extensions of the ant system

An ant colony optimisation (ACO) algorithm is presented that is inspired by AS, which achieves performance improvements through the introduction of new mechanisms based on ideas not included in the original AS.

The ant colony system (ACS) is used to improve the performance of the AS. ACS consists of modifications that may be made to the AS. ACS differs from AS in three different ways [33]. Firstly, the search space is explored more strongly by the ants than AS does. This is done by using a more aggressive action choice rule. Secondly, pheromone evaporation and deposit only take place on the arcs of the best tour. Finally, each time an ant makes use of arc (i,j) to move from SKU i to SKU j, some of the pheromone on that arc is removed to encourage exploration of alternative paths.

ACS introduces a rule of transition depending on a parameter q_0 ($0 \leq q_0 \leq 1$), which defines a balance between diversification and intensification. Tour construction is altered by an ant k

currently at SKU i choosing a SKU j according to a *pseudo random proportional* rule, given by

$$j = \begin{cases} \arg\max_{u \in J_i^k} \left[(\tau_{iu}(t)) \cdot (\eta_{iu})^{\beta} \right] & \text{if } r \leq q_0 \\ J & \text{if } r > q_0, \end{cases} \tag{5.20}$$

where r is a random variable randomly distributed in $[0, 1]$, q_0 $(0 \leq q_0 \leq 1)$ is a parameter, and $J \in J_i^a$ a SKU randomly selected according to the probability distribution given by

$$p_{iJ}^a = \begin{cases} \frac{(\tau_{iJ}(t)) \cdot (\eta_{iJ})^{\beta}}{\sum_{l \in J_i^a} (\tau_{il}(t)) \cdot (\eta_{il})^{\beta}} & \text{if } J \in J_i^a \\ 0 & \text{if } J \notin J_i^a. \end{cases} \tag{5.21}$$

If $r > q_0$, a SKU is chosen in the same manner as in the AS algorithm and the systems tends to diversify the search space. However, if $r \leq q_0$ the systems tends towards intensification of the search space.

The pheromones on the trails may be divided as a global update and a local update. Each ant deposits pheromones on the trail according to

$$\tau_{ij}(t+1) = (1 - \zeta) \cdot \tau_{ij}(t) + \zeta \cdot \tau_1,$$

where $0 < \zeta < 1$ and τ_1 is the initial pheromones on the trail. The pheromones are updated in this manner immediately after having crossed an arc during tour construction. This is known as the *local pheromone trail update* [33]. Through experimentation, a good value of ζ is considered to be 0.1, while a good avlue for τ_1 is considered to be $\frac{1}{C^{TT}}$, where T is the number of SKUs considered and C^{TT} is the length of a nearest neighbour tour. This local update rule decreases the pheromone τ_{ij} whenever an ant crosses an arc (i, j). The arc then becomes less desirable for the other ants. This increases the likelihood of ants exploring arcs that have not been visited yet.

During each iteration the visited edges see their quantity of pheromones decreasing in order to support diversification of unexplored paths. The *global pheromone trail update* may be defined as

$$\tau_{ij}(t+1) = (1 - \rho) \cdot \tau_{ij}(t) + \rho \cdot \Delta\tau_{ij}(t),$$

where the edges (i, j) belong to the best solution. The arcs of the best solution assigned to a set $\mathcal{T}^+$ with a length of L^+, and where $\Delta\tau_{ij}(t) = \frac{1}{L^+}$. This ensures that only the best trail during an iteration is updated. This is important, because in this manner the computational complexity of the pheromone update at each iteration is reduced from $\mathcal{O}(m^2)$ to $\mathcal{O}(m)$, where m is the size of the instance being solved [33]. The parameter ρ represents the pheromone evaporation, however, in equation (5.21) the deposited pheromone is discounted by a factor ρ. The new pheromone is a weighted average between the old pheromone value and the amount of pheromone deposited.

The system uses a list of candidates. This list stores closest neighbours for each SKU in decreasing order. An ant will consider an edge towards a SKU apart from this list, only if the edge has been visited once before. In this case the choice is made according to equation (5.21).

The ACS may easily be transformed in such a way as to maximise the number of orders in common between adjacent SKUs. The distances between two SKUs is equal to the number of orders that have to pick both these SKUs. The distance matrix has to be modified, since larger values in the distance matrix is associated with better similarities between SKUs. The largest single element in the distance matrix is calculated and denote this element $x_{\max}$. Each element

Parameter	Range
q_0	0.5
β	1, 2, 5, 10
ρ	0.01, 0.05, 0.10, 0.20
ζ	0.01, 0.05, 0.10, 0.20
Q	average arc weight $\times$ number of SKUs

Table 5.1: *Ant colony parameter ranges for the SLP data sets.*

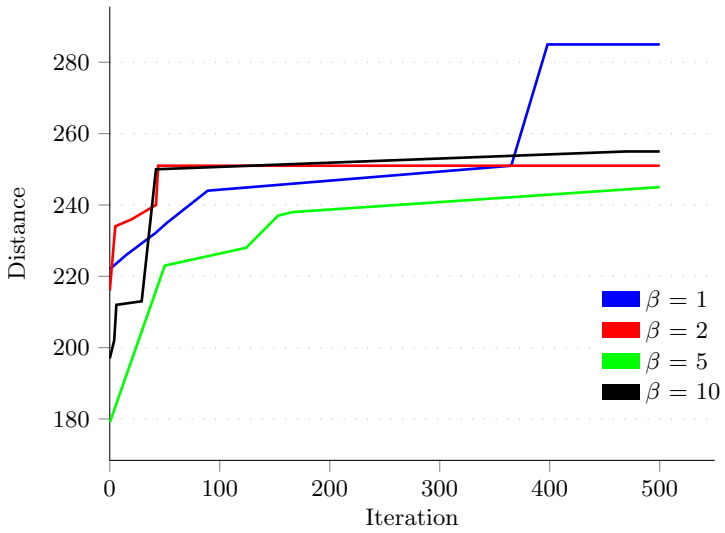


Figure 5.3: *A graphical representation of the solution quality of the ant colony system for data set K, when the value of β is varied at 1, 2, 5 and 10.*

in the distance matrix is then altered by replacing each element x_{ij} in the distance matrix by $x_{\max} - x_{ij}$, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, m$, where m is equal to the number of SKUs on a picking line.

The effectiveness of the ACS may be measured against the inverse ant colony system (IACS), where two SKUs with less orders in common are placed next to one another.

The ACS is tested for a series of parameters for each data set considered. Table 5.1 displays the ranges of the various parameters used to solve the ACS. The number of ants are fixed at 20, while 500 iterations are incurred in total for every set of parameters.

Figure 5.3 displays the solution quality of the ant colony system for data set K, when the value of β is varied at 1, 2, 5 and 10. The value of ζ is fixed at 0.10.

This ACS was used to determine which SKUs should be placed next to one another with the aim of maximising the similar orders amoung SKUs. To test the effect of this approach the opposite was also attempted where SKUs where placed in a cluster if they share as little orders as possible. This method was called the Inverse ACS (IACS).

5.4 (Agglomerative) Clustering algorithms

Clustering algorithms analyse the assignment of a set of observations (similarities) into subsets (called clusters) so that observations in the same cluster are similar in some sense. Clustering holds an advantage in that all data points within a cluster are represented by a single point

within the cluster. This may lead to computational efficiency when analysing data. Clustering depends on types of data used in the clustering analysis [66]. Clustering may be used to place SKUs with similar attributes on the same part of a picking line.

All the agglomerative clustering algorithms were computed using SAS [98], and MATLAB [78] was used to construct dendrograms[4] used to visually display the results obtained by the algorithms. A number of known clustering methods are explained, followed by new clustering algorithms created specifially for the SLP.

5.4.1 The single-link method

The single-link method (also referred to a the *nearest-neighbour method*) is one of the first and simplest clustering methods. Each data point is initially contained in a cluster of its own. The shortest distance between any two clusters merges these clusters during an iteration of the algorithm. The distance between two clusters, X and Y, is

$$d_{X,Y} = \min_{i \in X,\, j \in Y} \{d_{ij}\}, \tag{5.22}$$

where d_{ij} denotes the distance between two data points, $i \in X$ and $j \in Y$.

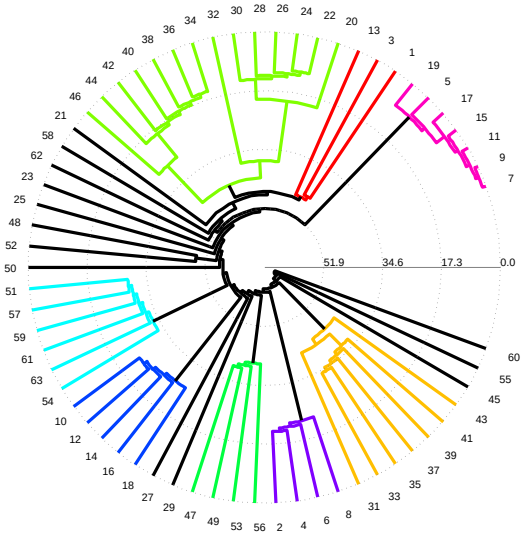


Figure 5.4: *Dendrogram of the clusters formed by the single-link method as a function of their threshold distances for data set K.*

The advantage of this clustering method lies in its simplicity of implementation. However, this clustering method contains a significant disadvantage in that two clusters, which are far apart, may be clustered together, merely because their nearest neighbours are close to one another. Figure 5.4 displays the dendrogram obtained from data set J used by Pep.

5.4.2 The average-link method

Similar to the single-link method, each data point is initially represented as a cluster. Two clusters closest to one another are fused together during each iteration of the algorithm. How-

[4] A dendrogram is a tree diagram that may be used to illustrate the arrangement of the clusters produced by hierarchical clustering.

ever, the distances between clusters are calculated as the average distance of the various points within each cluster. The distance between clusters X and Y is

$$d_{X,Y} = \frac{1}{n_X + n_Y} \sum_{\substack{i \in X \\ j \in Y}} d_{ij}, \tag{5.23}$$

where d_{ij} denotes the distance between two data points, $i \in X$ and $j \in Y$, and where n_X and n_Y denote the number of points in cluster X and Y, respectively. The distances in (5.23) may be recomputed between clusters after successive iterations of the algorithm, by altering the between-cluster average distances. When fusing clusters X and Y, distances are updated according to

$$d_{X \cup Y,Z} = \frac{n_X}{n_X + n_Y} d_{X,Z} + \frac{n_Y}{n_X + n_Y} d_{Y,Z}, \tag{5.24}$$

where $d_{X \cup Y,Z}$ denotes the distance between clusters $X \cup Y$ and Z, and where $d_{X,Z}$ and $d_{Y,Z}$ denote the distances between clusters X and Z, and between clusters Y and Z, respectively.

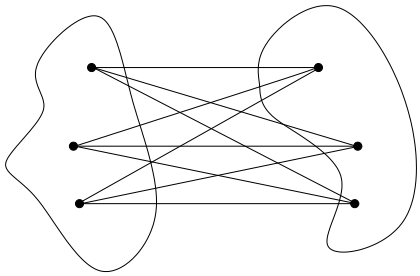


Figure 5.5: *Schematic illustration of the mechanism of the average-link method for data set K.*

Figure 5.5 represents a schemetic representation of the average-link method containing two clusters. Figure 5.6 displays the dendrogram obtained from data set K.

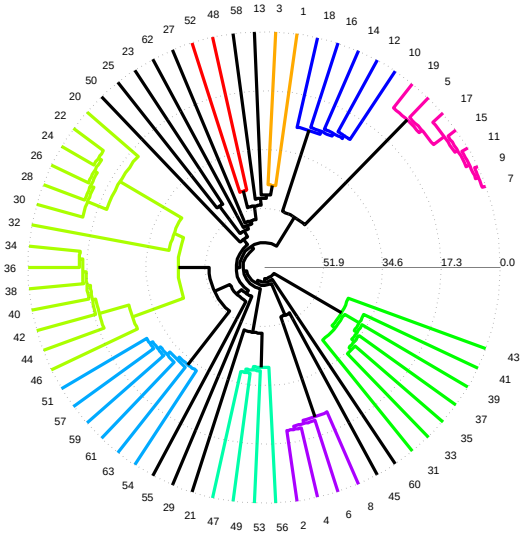


Figure 5.6: *Dendrogram of the clusters formed by the average-link method as a function of their threshold distances.*

5.4.3 The centroid method

In the centroid method, every point is once more represented as a cluster. Two clusters closest to one another are once more fused together during each iteration of the algorithm. The distance between two clusters, X and Y, are calculated as the centroids Z_X and Z_Y of the two clusters, respectively. If a cluster contains n distinct points $(x_1, y_1), \ldots, (x_n, y_n)$, then the x and y coordinates of the cluster centroid are

$$Z_X = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad Z_Y = \frac{1}{n}\sum_{i=1}^{n} y_i,$$

respectively.

The distances between clusters at successive iterations of the algorithm may be recomputed by changing the between-cluster centroid distances, when fusing clusters X and Y by

$$d_{X\cup Y,Z} = \frac{n_X}{n_X + n_Y} d_{X,Z} + \frac{n_Y}{n_X + n_Y} d_{Y,Z} - \frac{n_X n_Y}{(n_X + n_Y)^2} d_{X,Y}, \tag{5.25}$$

where $d_{X\cup Y,Z}$ is the distance between clusters $X \cup Y$ and Z, and where $d_{X,Z}$, $d_{Y,Z}$ and $d_{X,Y}$ denote the distances between clusters X and Z, between clusters Y and Z, and between clusters X and Y, respectively.

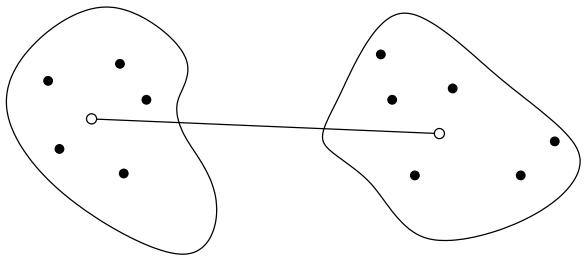


Figure 5.7: *Schematic illustration of the mechanism of the centriod method.*

Figure 5.7 represents a schemetic representation of the centriod method containing two clusters. Figure 5.8 displays the dendrogram obtained from data set K.

5.4.4 Ward's algorithm

Ward [114] developed a hierarchical aggregation procedure which is a routine for searching through groups of data to find which pair of basic data units shows the greatest mutual similarity with respect to specified characteristics [84]. The objective is to minimise some measure of the within-cluster variance when decisions are made regarding the clustering process. Every SKU initially represents a cluster and two clusters are then fused according to some criteria during each iteration. Distance between two clusters, X and Y, is the Euclidean distance between the centres of the clusters, respectively, raised to any power ϱ on the interval $(0, 2]$ [99]. The distance between clusters X and Y is

$$d_{X,Y} = \frac{1}{n_X + n_Y} \sum_{i=1}^{n_X+n_Y} d^{\varrho}(\boldsymbol{x}_i, \boldsymbol{c}_j), \tag{5.26}$$

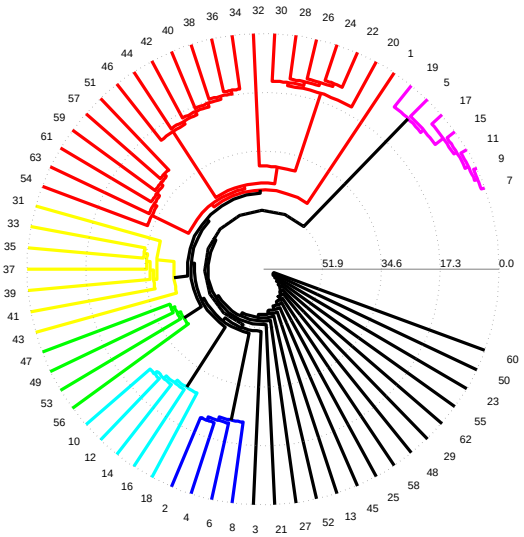


Figure 5.8: *Dendrogram of the clusters formed by the average-link method as a function of their threshold distances for data set K.*

where

$$\begin{aligned} \boldsymbol{c}_j &= \frac{1}{n_X + n_Y} \sum_{i=1}^{n_X+n_Y} \boldsymbol{x}_i, \\ X &= \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n_X}\}, \\ Y &= \{\boldsymbol{x}_{n_X+1}, \ldots, \boldsymbol{x}_{n_X+n_Y}\}, \end{aligned}$$

and where $d^{\varrho}(\cdot, \cdot)$ is an appropriate distance matrix.

Ward's algorithm simplifies the process of recomputing distances between clusters in successive iterations of the algorithm when fusing clusters X and Y. The fusion is updated by

$$d_{X\cup Y,Z} = \frac{n_X + n_Z}{n_X + n_Y + n_Z} d_{X,Z} + \frac{n_Y + n_Z}{n_X + n_Y + n_Z} d_{Y,Z} - \frac{n_Z}{n_X + n_Y + n_Z} d_{X,Y}, \tag{5.27}$$

where $d_{X\cup Y,Z}$ is the distance between clusters $X \cup Y$ and Z, and where $d_{X,Z}$, $d_{Y,Z}$ and $d_{X,Y}$ denote the distances between clusters X and Z, between clusters Y and Z, and between clusters X and Y, respectively [115]. Figure 5.9 displays the dendrogram obtained from data set K.

The *sum of squares* (ESS) method is a special case of Ward's Algorithm, when $\varrho = 2$. The *error sum of squares* of a cluster X is defined as the sum of squared Euclidean distances between the objects of the cluster and its centroid, it is

$$\mathrm{ESS}(X) = \sum_{i\in X} (x_i - \bar{x}(X))^2. \tag{5.28}$$

This is used to measure the tightness of a cluster [66]. Ward's algorithm is constructed by requiring that ΔESS be as small as possible during each iteration.

5.4.5 When to stop the clustering process

During implementation of various clustering algorithms, a result containing one cluster does not necessarily provide any benefit. The aim is to construct clusters of SKUs that should be

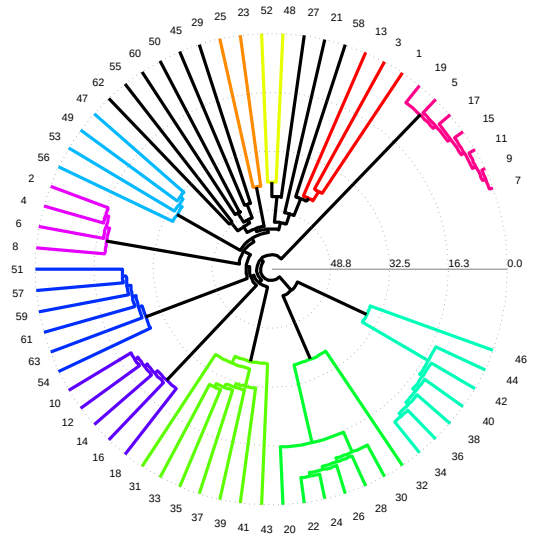


Figure 5.9: *Dendrogram of the clusters formed by Ward's algorithm to orders as a function of their threshold distances for data set K.*

placed adjacent on a picking line. A single cluster containing all the SKUs on a picking line does not specify which SKUs possess simlarities that other do not, nullifying the clustering process. In order to obtain meaningful results, some stopping criteria should be employed to prohibit hierarchical algorithms from continuing the clustering process untill all SKUs are fused into a single cluster. The following stopping criteria may be considered:

1. A predetermined number of iterations may be decided upon in advance.
2. Patterns after each iterations may be used as a guideline to determine the number of clusters. A dendrogram may be used to determine a desirable number of clusters.
3. Other methods exist where the ratio between the total cluster variance and the within-cluster variance are plotted. These considerations may be used in aiding the stopping criteria.

5.4.6 Implementation of clustering algorithms

In order to implement all the agglomerative hierarchical clustering methods, a distance matrix is used identical to the distance matrix in §5.3. The distances between two SKUs are equal to the number of orders that have to pick both these SKUs. The distance matrix has to be modified, since larger values in the distance matrix is associated with better similarities between SKUs. The largest single element in the distance matrix is calculated and denote this element x_{max}. Each element in the distance matrix is then altered by replacing each element x_{ij} in the distance matrix by $x_{\max} - x_{ij}$, for $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, m$, where m is equal to the number of SKUs on a picking line.

5.5 Heuristic clustering methods

Heuristic clustering methods use an initial partition of the data points as a basis for the clustering process. These clustering algorithms are created specifically to suite the SLP and characteristics

it possesses. The number of clusters are not specified and depends on the nature of the data. Contrary to the agglomerative clustering algorithms in §5.4, the aim of these clustering algorithms are to maximise the "distance" between clusters, where the distance refers to a measure of similarity between clusters.

5.5.1 Clustering according to orders

This heuristic is constructed in order to cluster SKUs that aim to contain all the SKUs within a cluster may form entire orders. Two clusters may only be fused if they share orders that have to visit all the SKUs in both clusters. The distance between clusters X and Y is

$$d_{X,Y} = \begin{cases} |\mathcal{P}_{X,Y}| & \text{if } X \cap Y = \emptyset \\ 0 & \text{otherwise,} \end{cases} \tag{5.29}$$

where $\mathcal{P}_{X,Y}$ is the set of orders that have to collect all the SKUs in cluster X and Y. Whenever all the distances between any two clusters are equal to 0, the algorithm is stopped. At this stage no order requires all the SKUs from any two clusters. Figure 5.10 displays the dendrogram obtained from a typical dataset used by Pep.

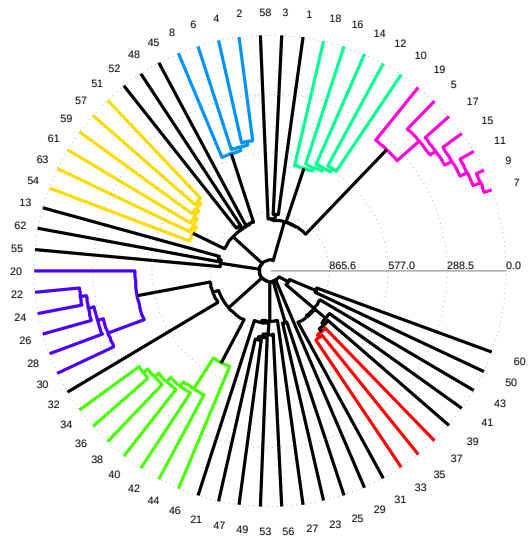


Figure 5.10: *Dendrogram of the clusters formed by heuristic manner of clustering according to orders as a function of their threshold distances for data set K.*

This clustering heuristic is used since it only fuses clusters if it may be beneficial to the orders that have to collect SKUs from both of those clusters.

5.5.2 Clustering according to SKU frequency

This clustering algorithm considers the number of orders that have to visit SKUs contained in a cluster. During an iteration, two clusters that have to be visited by the largest number of orders in common are fused. This distance is, however, normalised in order to enable clusters containing fewer SKUs to fuse with other clusters. The distance between clusters X and Y is

$$d_{X,Y} = \begin{cases} \frac{|\mathcal{P}_X \cap \mathcal{P}_Y|}{|\mathcal{P}_Y|} & \text{if } X \cap Y = \emptyset \\ 0 & \text{otherwise,} \end{cases} \tag{5.30}$$

where $\mathcal{P}_X$ is the set of orders that have to collect a SKU in cluster X. Figure 5.11 displays the dendrogram obtained from a typical dataset used by Pep.

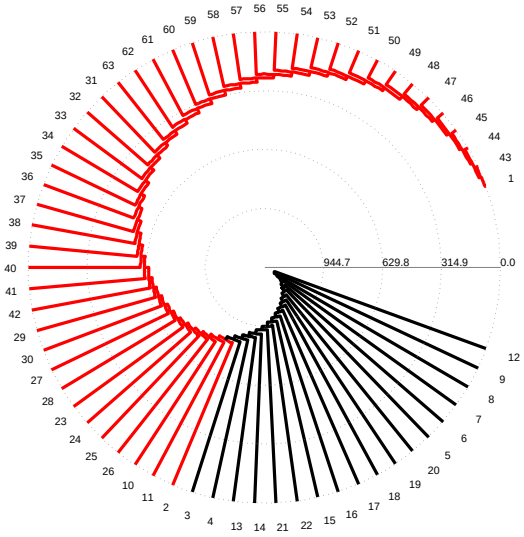


Figure 5.11: *Dendrogram of the clusters formed by heuristic manner of clustering according to SKU frequency as a function of their threshold distances for data set K.*

5.6 Computational results

A number of test cases were used to evaluate the effectiveness of the various algorithms used to address the SLP. Table 5.2 displays the number of cycles required to complete order picking on a picking line for each algorithm. The first column specifies the data set used, while the second column identifies the number of orders and the number of SKUs contained in the data set. The results of the OPA and GRP follows. Results from the ACS and the IACS is then displayed. The ACS and IACS algorithms were tested for a series of parameters and the best parameters were selected for implementation. The agglomerative hierarcical clustering algorithms, the average link (AL), centroid method (CM), single link (SL) and Ward's algorithm (W) follows. The two heuristic clustering algorithms, clustering according to orders (H_1) and clustering according to SKU frequency (H_2) is then displayed, followed by the number of cycles travelled by Pep's origional configuration (Pep). Once the SKUs were allocated according to the respective algorithms, the maximal cut algorithm (see §4.3.2) was used to test the quality of the solution.

All the agglomerative hierarchical clustering algorithms, as well as the heuristic clustering algorithms achieve the results indentical for each data set. The original configuration (Pep) is based on balancing higher frequency SKUs on the picking line with lower frequency SKUs. Often Pep employs a system where a high frequency SKU is placed in a location, followed by a number of SKUs with lower frequencies. In a typical picking line within Pep, about four lower frequency SKUs are seperated by a single higher frequency SKU [81]. This is done in order to reduce congestion of pickers.

From the results in Table 5.3 it is evident that certain data sets are not improved upon with any SLP algorithm and the solution quality also did not decrease. Table 5.4 displays the largest number of picks at a single location for each data set considered. In all the cases the maximal cut formulation matches these solutions. The SLP is unable to reduce the number of cycles travelled in these cases, since at least 1 226, 1 161 and 1 069 cycles have to be travelled for data

Data set	Size (O, L)	OPA	GRP	ACS	IACS	AL	CM	SL	W	H_1	H_2	Pep
A	(1262,49)	1232	1232	1232	1232	1232	1232	1232	1232	1232	1232	1232
B	(1264,54)	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226	1226
C	(1265,51)	1161	1161	1161	1179	1161	1161	1161	1161	1161	1161	1161
D	(1263,56)	1060	1011	1011	1079	1072	1072	1072	1072	1072	1072	1072
E	(1264,51)	1069	1069	1069	1069	1069	1069	1069	1069	1069	1069	1069
F	(1258,55)	941	872	863	863	1025	1025	1025	1025	1025	1025	1025
G	(1258,53)	1021	1002	997	1007	1005	1005	1005	1005	1005	1005	1005
H	(1244,54)	989	980	976	1012	992	992	992	992	992	992	992
I	(1260,56)	971	968	970	980	955	955	955	955	955	955	955
J	(1264,56)	967	957	949	959	947	947	947	947	947	947	947
K	(943,63)	277	284	274	287	259	259	259	259	259	259	259
L	(846,56)	224	226	233	233	232	232	232	232	232	232	232
M	(728,51)	148	149	152	154	152	152	152	152	152	152	152
N	(733,55)	117	121	117	123	125	125	125	125	125	125	125
O	(396,63)	121	93	116	146	90	90	90	90	90	90	90
P	(574,48)	74	80	81	79	80	80	80	80	80	80	80
Q	(242,64)	67	55	66	54	45	45	45	45	45	45	45
R	(158,55)	16	18	15	14	14	14	14	14	14	14	14
S	(89,42)	10	10	9	8	9	9	9	9	9	9	9
T	(82,51)	8	8	8	9	8	8	8	8	8	8	8
U	(90,48)	7	7	7	7	7	7	7	7	7	7	7
V	(80,56)	6	7	6	7	6	6	6	6	6	6	6
Total		11712	11536	11538	11727	11711	11711	11711	11711	11711	11711	11711

Table 5.2: *Results obtained when implementing the various SLP algorithms. Results from The OPA, GRP, ACS, IACS, AL, CM, SL, W, H_1 and H_2. The number of orders (O) and locations (L) are displayed for each data set.*

sets B, C and E, respectively.

Figure 5.12 displays the number of cycles travelled when using the maximal cut approach and the minimum number of cycles required, when considering the most frequent location(s), for each data set. Improvement in the SLP is only possible where the picks at the most frequent location is less than the solution obtained by the maximal cut approach.

The greatest saving was incurred in dataset F, where the number of cycles travelled may be reduced by as much as 162 cycles. However, any reduction in the number of cycles in comparison to Pep's configuration decreased the number of cycles by 2.50% on average.

5.7 Random SLP configurations

Due to the inconsistency of the results obtained using the various SLP algorithms, random SLP configurations are constructed. The minimum cut formulation is then used to determine the lowerbound of the number of cycles that have to be traversed. Then 200 random SLP configurations of each data set were created. The average number of cycles travelled, the minimum number of cycles travelled, the maximum number of cycles travelled and the frequency of the various spans used were retrieved to better understand the nature of each data set.

Table 5.5 displays the percentage of the shortest spans used when solving 200 instances of random SLP configurations for each of the 22 data sets considered. An average of 70.569% of all orders for large data sets should be picked on their minimum span according to the

Data set	Size (O, L)	Best performing SLP configuration SLP	Solution	Pep's Configuration	Percentage improvement
A	(1262,49)	All	1232	1232	0.00 %
B	(1264,54)	All	1226	1226	0.00 %
C	(1265,51)	All except IACS	1161	1161	0.00 %
D	(1263,56)	GRP and ACS	1011	1072	5.69 %
E	(1264,51)	All	1069	1069	0.00 %
F	(1258,55)	ACS and IACS	863	1025	15.80 %
G	(1258,53)	ACS	997	1005	0.80 %
H	(1244,54)	ACS	976	992	1.61 %
I	(1260,56)	AL, CM, SL, W, H_1 and H_2	955	955	0.00 %
J	(1264,56)	AL, CM, SL, W, H_1 and H_2	947	947	0.00 %
K	(943,63)	AL, CM, SL, W, H_1 and H_2	259	259	0.00 %
L	(846,56)	OPA	224	232	3.45 %
M	(728,51)	OPA	148	152	2.63 %
N	(733,55)	OPA and ACS	117	125	6.40 %
O	(396,63)	AL, CM, SL, W, H_1 and H_2	90	90	0.00 %
P	(574,48)	OPA	74	80	7.50 %
Q	(242,64)	AL, CM, SL, W, H_1 and H_2	45	45	0.00 %
R	(158,55)	ACS, IACS, AL, CM, SL, W, H_1 and H_2	14	14	0.00 %
S	(89,42)	IACS	8	9	11.11 %
T	(82,51)	All except IACS	8	8	0.00 %
U	(90,48)	All	7	7	0.00 %
V	(80,56)	All except GRP and IACS	6	6	0.00 %

Table 5.3: *Best performing configurations obtained when implementing the various SLP algorithms. Results from The OPA, GRP, ACS, IACS, AL, CM, SL, W, H_1 and H_2. The number of orders (O) and locations (L) are displayed for each data set.*

Data set	A	B	C	D	E	F	G	H	I	J	K
Most frequent	1232	1226	1161	1011	1069	835	959	817	855	729	95
Data set	L	M	N	O	P	Q	R	S	T	U	V
Most frequent	141	109	66	74	67	33	13	9	8	7	5

Table 5.4: *The number of picks at a location that is picked most often (most frequent location) for each data set.*

maximal cut formulation. Medium and small data sets require that 97.513% and 99.379% of orders should be picked on their minimum spans when solved by means of the maximal cut formulation. Medium and small data sets are relatively less complex to solve, since most spans should be minimum span. The problem is to determine which orders should be picked on their minimum spans. Another factor that may cause complications is to select the most desirable shortest span when multiple shortest spans exist.

The solution quality of these data sets were also calculated by means of the lowerbound in in §4.3.2. The solution qualities does not differ significantly. For data set A the average solution quality was 1 236.94 with a standard deviation of 2.63, while data set B had an average solution quality of 1 226 with a standard deviation of 0.

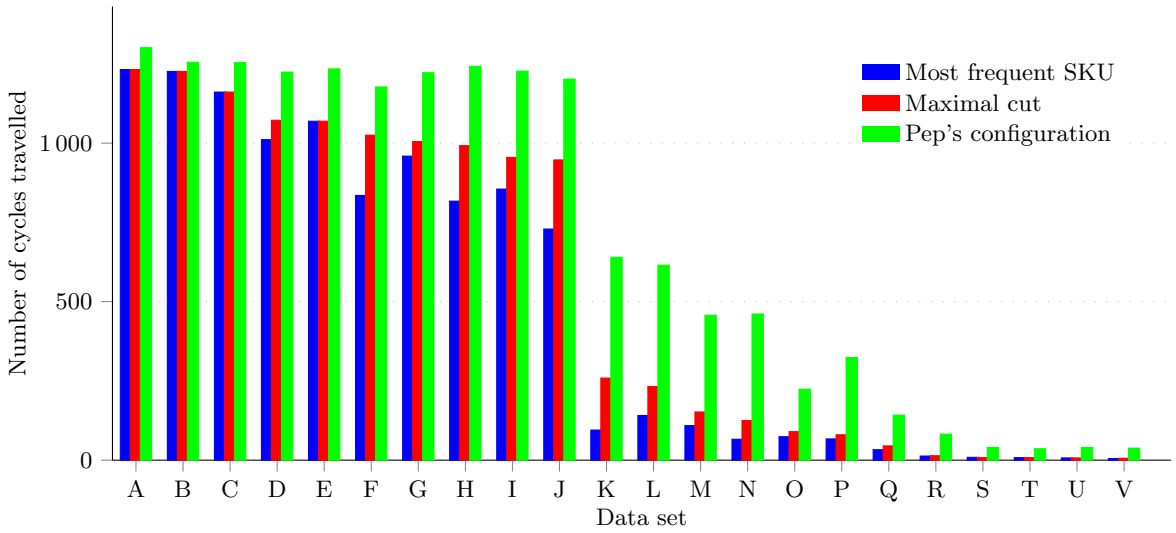


Figure 5.12: *The number of picks at a location that is picked most often for each data set, compared to the number of cycles travelled using Pep's configuration.*

5.8 Chapter summary

In this chapter a variety of methodologies are considered to solve the SLP. A model for the problem was presented in §5.1. Two well-known algorithms are presented — the OPA in §5.2.1 and the GRP in §5.2.2. An ant colony algorithm was used to place two products that share orders next to one another in §5.3. A number of agglomerative hierarchical clustering algorithms were used to cluster SKUs with similar charcteristic in §5.4. Two heuristic clustering algorithms are presented. The first heuristic clustering algorithm uses an alternative manner of clustering SKUs that have to be visited by the same orders. The second heuristic clusters SKUs based on the frequency of the SKUs. The chapter concludes with some computational results when using real life data sets supplied by Pep in §5.6 and results when considering random SLP configurations in §5.7. Results indicate that little saving may be incurred when adressing the SLP.

Spans	A	B	C	D	E	F	G	H	I	J	K
1	32.978 %	68.356 %	70.946 %	74.406 %	73.936 %	81.622 %	75.593 %	71.969 %	76.995 %	78.892 %	97.188 %
2	7.365 %	4.710 %	5.294 %	3.985 %	3.890 %	2.876 %	3.001 %	3.122 %	2.887 %	2.482 %	1.134 %
3	16.697 %	6.206 %	5.558 %	4.215 %	4.760 %	2.980 %	4.014 %	4.464 %	3.563 %	3.548 %	0.756 %
4	17.132 %	6.792 %	5.838 %	4.845 %	5.370 %	3.353 %	4.900 %	5.873 %	4.567 %	4.225 %	0.394 %
5	11.573 %	5.489 %	4.751 %	4.381 %	4.555 %	3.135 %	4.558 %	5.404 %	4.309 %	3.788 %	0.247 %
6	6.942 %	3.742 %	3.202 %	3.255 %	3.182 %	2.336 %	3.259 %	3.830 %	3.133 %	2.854 %	0.154 %
7	3.538 %	2.191 %	2.013 %	2.079 %	1.962 %	1.522 %	2.094 %	2.387 %	2.005 %	1.796 %	0.064 %
8	1.883 %	1.209 %	1.164 %	1.270 %	1.139 %	1.005 %	1.223 %	1.411 %	1.168 %	1.103 %	0.033 %
9	1.000 %	0.665 %	0.627 %	0.738 %	0.605 %	0.575 %	0.662 %	0.750 %	0.677 %	0.657 %	0.020 %
10	0.494 %	0.329 %	0.329 %	0.429 %	0.320 %	0.306 %	0.371 %	0.411 %	0.362 %	0.330 %	0.007 %
Rest	0.399 %	0.310 %	0.279 %	0.397 %	0.280 %	0.291 %	0.326 %	0.380 %	0.335 %	0.325 %	0.003 %

Spans	L	M	N	O	P	Q	R	S	T	U	V
1	96.927 %	97.781 %	99.029 %	95.856 %	99.331 %	96.481 %	99.516 %	99.680 %	99.250 %	99.889 %	98.563 %
2	0.880 %	0.968 %	0.668 %	0.902 %	0.540 %	1.167 %	0.275 %	0.208 %	0.146 %	0.089 %	0.150 %
3	0.919 %	0.474 %	0.229 %	0.845 %	0.107 %	0.740 %	0.076 %	0.112 %	0.165 %	0.022 %	0.200 %
4	0.460 %	0.306 %	0.050 %	0.822 %	0.022 %	0.574 %	0.060 %	—	0.146 %	—	0.344 %
5	0.315 %	0.199 %	0.016 %	0.573 %	—	0.407 %	0.035 %	—	0.110 %	—	0.250 %
6	0.210 %	0.128 %	0.004 %	0.362 %	—	0.267 %	0.006 %	—	0.037 %	—	0.225 %
7	0.144 %	0.078 %	0.004 %	0.331 %	—	0.130 %	0.019 %	—	0.055 %	—	0.125 %
8	0.092 %	0.031 %	0.001 %	0.135 %	—	0.120 %	0.003 %	—	0.030 %	—	0.075 %
9	0.039 %	0.018 %	—	0.086 %	—	0.060 %	0.009 %	—	0.043 %	—	0.038 %
10	0.000 %	0.012 %	—	0.039 %	—	0.033 %	—	—	0.018 %	—	0.019 %
Rest	0.013 %	0.006 %	—	0.049 %	—	0.021 %	—	—	—	—	0.013 %

Table 5.5: *Percentage of the shortest spans used when solving 200 instances of random SLP configurations for the 22 data sets considered.*

CHAPTER 6

Ideas and preliminary results on the SPLAP

Contents

6.1 Managerial considerations 127
6.2 Nearest neighbour search 128
6.3 A tabu search approach 130
6.4 Chapter summary 134

The SKU to picking line assignment problem (SPLAP) may be described as the allocation of SKUs to a picking line. Pep operates on a principle that a SKU may not be placed in more than one distinct picking line during any stage of order picking.

The SPLAP should aim to delegate the work amongst picking lines in such a way as to minimise the total number of cycles travelled over all picking lines, while balancing the workload over the picking lines.

6.1 Managerial considerations

Picking lines are constructed in a variety of manners. Depending on situations which the picking line scheduler may not have any control over, the best possible SPLAP configuration needs to be determined. Picking line schedulers are entrusted with the task of adhering to demands from the Pep Central Office regarding delivery of distributions that is sent from the Planning department within Pep, as well as adhering to requests from the pickers and balancing the workload evenly among picking lines.

Picking line managers have to construct picking lines to balance the "workload" over all the picking lines as well as ensuring that time required to pick the entire picking line is reduced as far as possible. Pep is unable to provide a clear definition on the balancing aspect of the picking lines. Picking line managers have various opinions on how to balance picking lines. These (sometimes conflicting) opinions are formulated from feedback from the pickers/managers working on the picking lines. Some picking line managers requests that pickers should not pick too frequently, varying time spent walking and picking regularly. Others may claim that minimising the total number of cycles is the only considerations that has to be taken into account.

Pickers are awarded incentive bonusses for the number of SKUs they pick in a given period. A picker may be pleased if the average orders he/she receives contains more picks than his/her

peers, if they receive the same number of orders within a picking line [80]. Pickers also prefer having an even spread of picks for the entire duration of order picking. Picking lines containing *too* many picks per cycle are also not favourable. Picking line managers construct a picking line, ensuring that pickers do not pick from most of the locations during any cycle. If picking lines containing too many picks per cycle relative to other picking lines, the pickers refrain from going to work on that picking line. Instead they may take a leave of absence [69]. The pickers consider these picking lines as containing a heavier workload, which is not percieved as just.

Picking line managers have also noted that if pickers pick orders that contain an excessive number of picks in succession, the pickers will complain that the workload is not balanced, resulting in certain pickers receiving a larger incentive bonus for incurring more picks [80]. Picking line managers are also entrusted with ensuring that pickers have an even spread of A- and B-type SKUs.

Picking line managers must also take note of any other complaints from the pickers. Typical examples include instances where pickers would complain that an A-type SKU requires a substancial amount of effort to pick. In this case the picking line manager would be required to alter the status of the SKU as well as balancing the different types of SKUs on picking lines accordingly in the future.

Some DBNs may be submitted after specified dates (*deadlines*). In this case picking line managers receive urgent requests that have to be delivered much earlier than other SKUs. Picking line managers must then build a new picking line in order to deliver SKUs to the destinations as soon as possible. Picking line managers may find it difficult to balance the workload evenly among these picking lines.

Identical SKUs may not be placed on more than one picking line. Furthermore, the warehouse management system used by Pep only allows a SKU to be placed in one location on that picking line. This implies that all orders that require that SKU should pick from that location. An example of an exact formulation for the SPLAP is presented in Appendix A.

6.2 Nearest neighbour search

The nearest neighbour search (NN) is an optimisation approach for finding the nearest neighbours in metric spaces [93]. The NN may be defined by a set $\mathcal{D}$ of n points in a d–dimensional metric space $M = (X, d)$ and a query point $q \in X$. The objective is to find the point $p \in \mathcal{D}$ closest to q.

The NN is adapted for the SPLAP to determine the construction of a "desirable" picking line, where managerial considerations mentioned in §6.1 are taken into account. As an initial attempt to solve the problem, only the average picks per cycle is considered as a measure of solution quality. The reasoning is that this may satisfy the requirements of the pickers and may cause a reduction in the number of cycles travelled.

The Pep DC uses an order picking system which is based on the concept of a wave. During a typical day of operations within the Pep DC, an average of two or three picking lines may be constructed. For this reason it is necessary to consider improvements between fewer picking lines. Also, since the definition of workbalance is not clear a number of various measures are proposed to address different aspects associated with order picking.

Initially all SKUs in two different picking lines are removed. Initially each picking line receives a SKU. One picking line receives the SKU that is requested by the most orders (highest frequency

SKU), while the second picking line receives the SKU requested by the fewest orders (lowest frequency SKU). All the remaining SKUs are then iteratively placed into the first picking line. The SKU that is best fit according to a measure is then selected and added to the first picking line. The same process is done for the second picking line.

A number of measures are formulated to construct picking lines with different objectives. Let c_h be the number of cycles travelled on a picking line h. The first criteria for adding SKUs to a picking line is by simply minimising the number of cycles travelled on each picking line, *i.e.*

$$\text{minimise} \sum_{h=1}^{H} c_h. \tag{6.1}$$

This criteria solely focusses on reducing the number of cycles travelled in each picking line. This criteria does not address the issue of work balance between picking lines.

Let the number of picks in a picking line h be p_h. Another measure may be to maximise the product of the picks per cycle for a number of picking lines, *i.e.*

$$\text{maximise} \prod_{h=1}^{H} \frac{p_h}{c_h}. \tag{6.2}$$

This criteria aims to reduce the number of cycles travelled with the aim of maintaining similar picks per cycle for each picking line considered. In this manner the work balance is addressed in such a way that all picking lines have the same picks per cycle.

The second criteria does not necessarily ensure that the number of picks in each picking line is in proportion to the number of SKUs placed on the picking line. For example, if two picking lines are constructed, both having an average of 20 picks per cycle, the first picking line contains 100 SKUs and the second picking line contains 20 SKUs, the work is not balanced. In this case the pickers in the second picking line will need to pick from each location during each cycle. The pickers in the first picking line will need to only pick approximately 1 in every 5 locations. For this reason it is necessary to take the number of SKUs that are placed into a picking line into account as well. Let the number of SKUs in picking line h be s_h. Another measure may be to maximise the product of the picks per cycle for a number of picking lines, by taking the number of SKUs of a picking line into consideration, *i.e.*

$$\text{maximise} \prod_{h=1}^{H} \frac{p_h}{c_h \times s_h}. \tag{6.3}$$

This measure is not considered for the nearest neighbour search, since the SKUs are divided approximately equally between multiple picking lines.

Let μ_c denote the average number of cycles travelled for all picking lines considered. In this case

$$\mu_c = \frac{1}{H} \sum_{h=1}^{H} (c_h), \tag{6.4}$$

and the standard deviation may be expressed as

$$\sigma_c = \sqrt{\frac{1}{H} \sum_{h=1}^{H} (c_h - \mu_c)^2}. \tag{6.5}$$

Simlarly, let μ_p denote the average picks per cycle for all picking lines considered. In this case

$$\mu_p = \frac{1}{H}\sum_{h=1}^{H}\left(\frac{p_h}{c_h}\right), \tag{6.6}$$

and the standard deviation may be expressed as

$$\sigma_p = \sqrt{\frac{1}{H}\sum_{h=1}^{H}\left(\frac{p_h}{c_h} - \mu_p\right)^2}. \tag{6.7}$$

The first manner of scoring considers the number of cycles travelled in each picking line. The objective is to

$$\text{minimise } [\alpha\mu_c + (1-\alpha)\sigma_c]. \tag{6.8}$$

The scalar, α, is used to shift the focus on either ensuring that the number of cycles travelled in each picking line is kept to a minimum (higher values of α), or ensuring that all picking lines considered travel approximately the same number of cycles (lower values of α).

A second manner may also be considered where the picks per cycle is considered. The objective is then to

$$\text{maximise } [\alpha\mu_p + (1-\alpha)\sigma_p]. \tag{6.9}$$

The scalar, α, is used to shift the focus on either ensuring that the picks per cycle in each picking line is kept to a minimum (larger values of α), or ensuring that all picking lines considered travel approximately the same number of picks per cycle (lower values of α).

This approach was implemeted for the criteria in (6.1) and (6.2), respectively. The SKUs from two picking lines were aggregated and two new picking lines were constructed. The first picking line will receive the SKU with the highest frequency and the second picking line will receive the SKU with the lowest frequency. The remainder of the SKUs are then iteratively added to each picking line at a time, where the SKU that best improves the criteria is added to the picking line. This approach does not yield significant savings. A tabu search implementation was able to achieve better results and follows in the following section.

6.3 A tabu search approach

A tabu search implementation may be performed once all the SKUs have been allocated to a respective picking line. Each SKU in each picking line is temporarily traded with each SKU from all the other picking lines. The trade with the largest *score* is selected, where the score may be defined in a variety of manners.

Initially the tabu search was implemented with the aim of increasing the product of the picks per cycle between two picking lines. The scoring method was thus to

$$\text{maximise } \prod_{h=1}^{H}\frac{p_h}{c_h}. \tag{6.10}$$

This manner of scoring aims to create the same work balance in all picking lines considered, while lowering the number of cycles travelled in each picking line. In this case the focus is

not shifted on either lowering the number of cycles travelled in each picking line or aiming to balance the workload over the picking lines. This manner of scoring may address both issues simultaneously. The data was seperated for all different groups of data sets. For instance, comparison between two large data sets form part of the group LL (large-large comparison), while a comparison between the medium and smaller data sets forms part of the group MS (medium-small comparison).

The NSO heuristic was used to solve the OSP[1]. The computational time of considering two data sets is on average 1 824 seconds (30 minutes and 24 seconds). Due to this long computational times to solve an instance by means of the tabu search, considering only two picking lines at a time as well as three large picking lines.

Table 6.1 displays the average number of cycles travelled when comparing the results of the original picking lines to the picking lines constructed by the tabu search. On average, the tabu search did not reduce the number of cycles considerably. Some cases do exist where two large pairs of data sets may incur significant savings. Table 6.2 displays the 10 pairs of picking lines that resulted in the largest savings in cycles travelled when using the tabu search when three picking lines are considered.

	Average number of cycles travelled			
Configuration	Original picking line	Tabu search	Percentage improvement	Average computational time
LL (45)	2136.80	2031.29	4.94 %	3788
LM (70)	1208.83	1201.33	0.62 %	2015
LS (50)	1077.20	1076.88	0.03 %	1851
MM (21)	280.86	273.33	2.68 %	401
MS (35)	149.23	147.91	0.88 %	238
SS (10)	17.60	17.50	0.57 %	63
Average	1064.48	1040.85		1824

Table 6.1: *The average number of cycles travelled for the data sets in each configuration, by considering the original picking lines and the picking lines modified using the tabu search when considering two picking lines. The computational times are indicated in seconds. The number in brackets next to each configuration indicates the number of the pairs of picking lines solved by the tabu search for each configuration.*

Figure 6.1 displays the frequency of each SKU before and after the tabu search is implemented on picking line A and B. All the SKUs are ranked in decreasing order in terms of their frequencies. Note that picking line B initially contained a single SKU that has a frequency in excess of 1 200. Except for this SKU, the SKU with the second largest frequency in picking line B has a frequency of less than 800. Picking line A, however, contain 7 SKUs with frequencies in excess of 1 200. By swapping the SKU with the high frequency in picking line B with a SKU in picking line A that has a frequency of less than 800, significant savings are achieved.

Figure 6.2 displays the frequency of each SKU before and after the tabu search is implemented on picking line B and C. All the SKUs are ranked in decreasing order in terms of their frequencies. Many SKUs present in picking line B are also present in picking line C. Before the tabu search implementation only a single SKU in picking line B had a frequency in excess of 1 200. However, after tabu search implementation, 4 SKUs have a frequency in excess of 1 200. Fewer SKUs are thus present on picking line C. The number of cycles travelled on picking line C increases, while the number of cycles travelled on picking line B decreases substancially.

[1]The GEO metaheuristic was also used. This metaheuristic delivered similar results, but requires far more computational time to execute a single instance of the OSP.

Data sets	Picking line 1 Original picking line	Picking line 1 Tabu search	Picking line 2 Original picking line	Picking line 2 Tabu search	Number of cycles saved	Total percentage improvement	Computational time
A & B	1232	1232	1226	909	317	12.90 %	8208
B & C	1226	863	1161	1226	298	12.48 %	7454
F & G	1025	1063	1005	726	241	11.87 %	4827
B & D	1226	853	1072	1226	219	9.53 %	9302
B & G	1226	809	1025	1226	216	9.60 %	4329
C & I	1161	1161	955	751	204	9.64 %	1596
B & E	1226	877	1069	1226	192	8.37 %	6740
A & C	1232	1232	1161	973	188	7.86 %	7333
C & G	1161	1161	1005	820	185	8.54 %	2587
C & E	1161	1161	1069	900	169	7.58 %	3368

Table 6.2: *The 10 best pairs of picking lines that resulted in the largest savings in cycles travelled when using the tabu search when considering two picking lines. All computational times are indicated in seconds.*

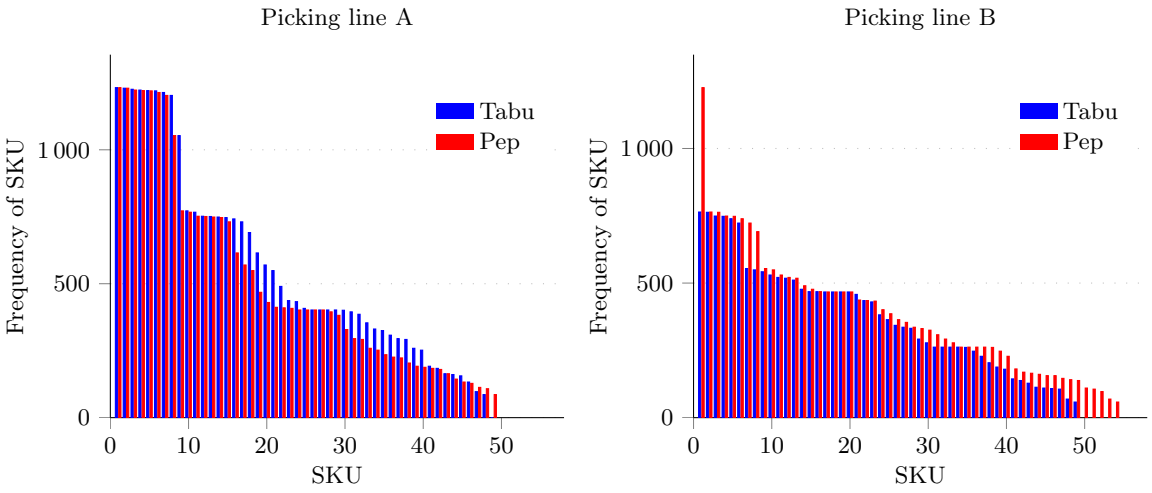


Figure 6.1: *Frequency of each SKU before and after the tabu search is implemented on picking line A and B.*

The best improvement outside the LL configuration is in the LM configuration for picking lines H and L. A number of SKUs with lower frequencies in picking line H are swapped with the highest frequency SKUs in picking line L. A total saving of 52 cycles travelled may be saved in this manner. Figure 6.3 displays the frequency of each SKU before and after the tabu search is implemented on picking line H and L. All the SKUs are ranked in decreasing order in terms of their frequencies.

Table 6.3 displays the 10 pairs of picking lines that resulted in the largest savings in cycles travelled when using the tabu search when three picking lines are considered. Only large data sets are considered, due to the large cost in computational time to solve these problems.

Figure 6.4 displays the frequency of each SKU before and after the tabu search is implemented on picking line A, B and E. All the SKUs are ranked in decreasing order in terms of their frequencies. The solutions obtained for data set A delivers the same result as the configuration proposed by Pep. Picking line B and E contain SKUs with lower frequencies than the configurations proposed by Pep. Picking line E contains a number of SKUs that have a frequency slightly less than 800. For data set B it would seem possible to swop the three SKUs with the largest

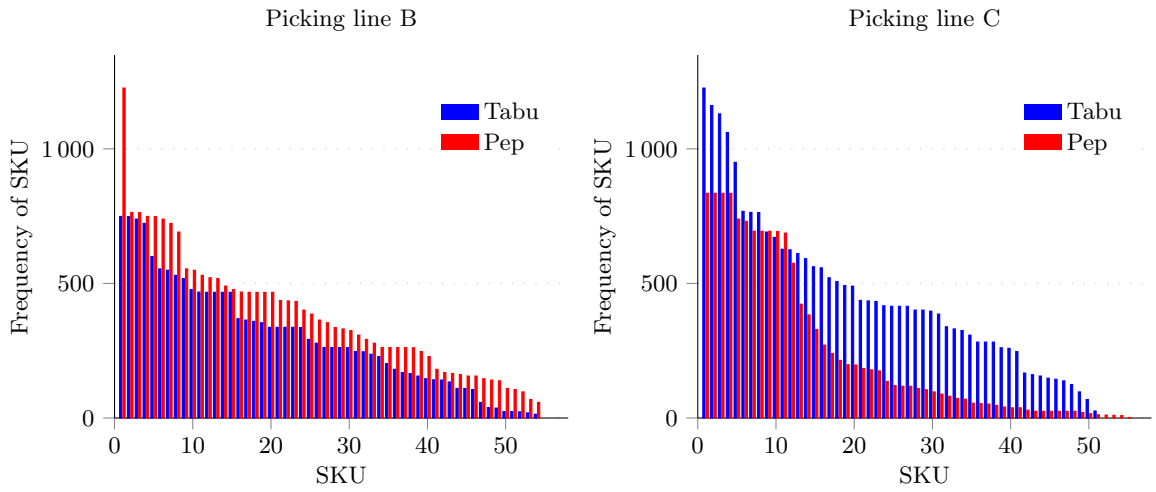


Figure 6.2: *Frequency of each SKU before and after the tabu search is implemented on picking line A and B.*

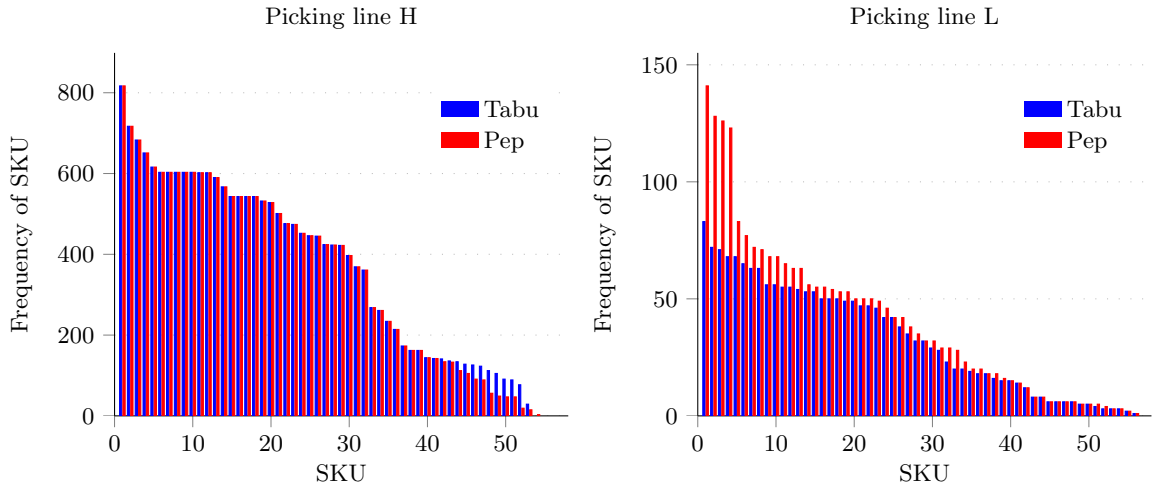


Figure 6.3: *Frequency of each SKU before and after the tabu search is implemented on picking line H and L.*

frequencies with SKUs of lower frequencies from data set A.

A possible explanation is that the greedy heuristic used to determine the solution quality did not assess swopping the higher frequency SKUs of data set B with lower frequency items from a different picking line as being beneficial. Another explanation may be that when swopping a single high frequency item from data set B does not improve the solution quality and that a number of SKUs must be swopped simultaneously to improve the solution quality.

The tabu search was also implemented using the scoring method

$$\text{minimise} \sum_{h=1}^{H} c_h.$$

This method displayed similar, and sometimes identical, results to the initial scoring method. For this reason the results are not included in this thesis.

Many open questions still have to be addressed with regards to SPLAP. This chapter merely considers initial attempts used for addressing the SPLAP. Questions relating the number of

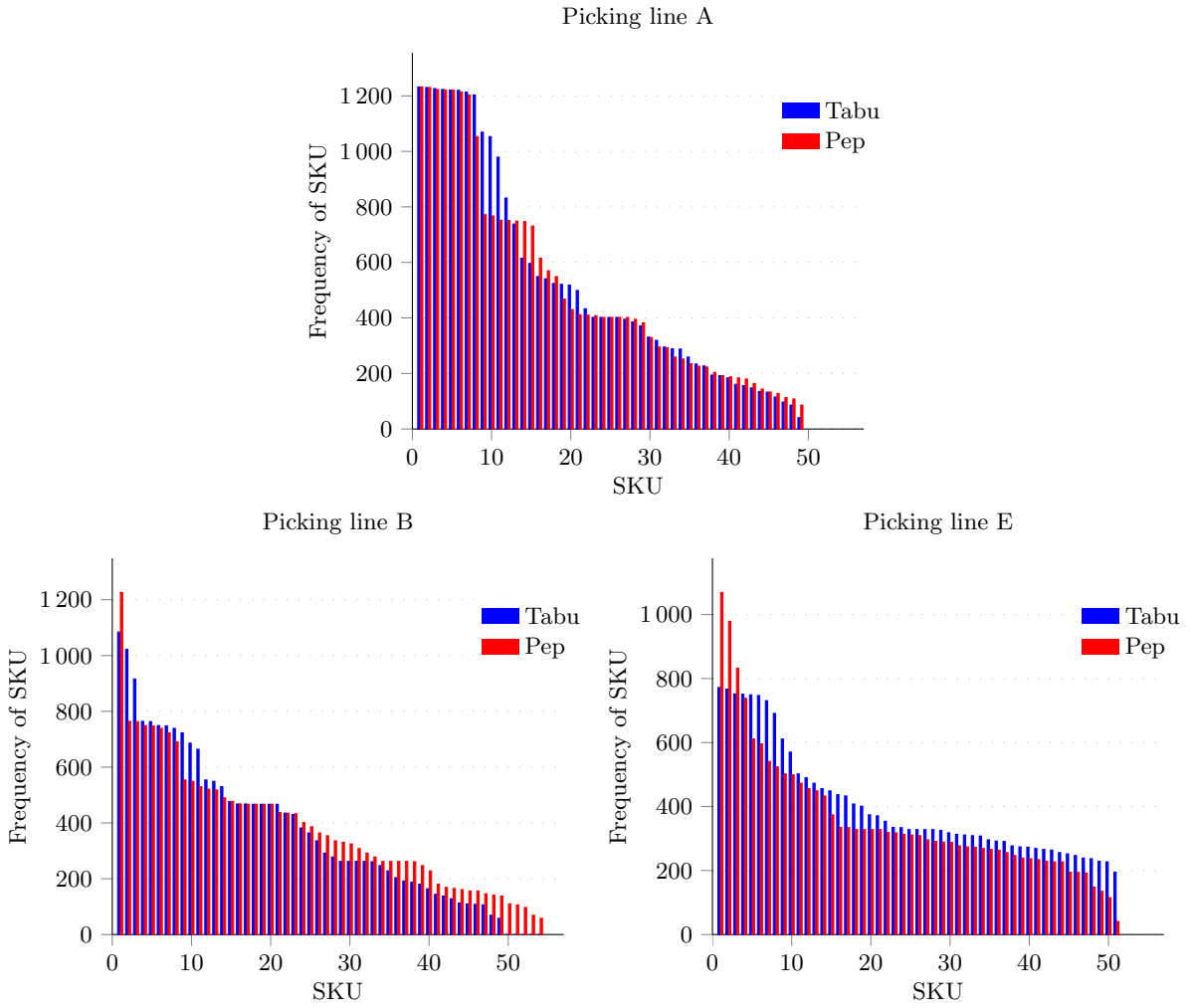


Figure 6.4: *Frequency of each SKU before and after the tabu search is implemented on picking line A, B and E.*

SKUs awarded to a picking line have not been considered in any form. Also, with the nearest neighbour and tabu search approaches emphasis was not placed on ensuring that the workbalance is equally divided amongst the picking lines. Instead only workbalance within each picking line is considered. More sophisticated approaches have to be developed which have to encompass these and other open questions to construct picking lines effectively according to Pep's guidelines.

6.4 Chapter summary

This chapter examines a number of approaches to address the SPLAP. Initailly some practical considerations are highlighted in §6.1. These considerations place context on the contradicting objectives that arises when the SPLAP is solved. A goal progamming formulation is then presented as an exact formulation to the problem. This formulation is, however, too large to solve in a reasonable time frame.

Consequently, two nearest neighbour search heuristics were then used in §6.2. Both these approaches aim to minimise the total picks per cycle for each picking line, minimising the

number of cycles travelled while balancing the workload amongst the various picking lines.

Initially all the SKUs in all of the 22 picking lines were consolidated and 20 new picking lines were constructed, of which 19 contained 56 SKUs and the final picking line containing 27 locations. This approach of considering all 22 picking lines simultaneously is unrealistic as usually no more than 3 or 4 picking lines receive a wave simulatneously.

A different approach in §6.3 was then undertaken in the form of a tabu search, where SKUs between two (or three) picking lines are swopped to effectively construct two (or three) new picking lines, such that each picking line is then awarded a more desirable wave. Significant savings may be incurred between large data sets.

Data sets	Picking line 1 Original picking line	Picking line 1 Tabu	Picking line 2 Original picking line	Picking line 2 Tabu	Picking line 3 Original picking line	Picking line 3 Tabu	Number of cycles saved	Total percentage improvement	Computational time
A & B & E	1232	1238	1226	890	1069	970	429	12.163%	6071
B & C & E	1226	859	1161	1226	1069	983	388	11.227%	13712
B & C & D	1226	863	1161	1226	1072	1003	367	10.610%	9310
A & B & G	1232	1232	1226	904	1005	1005	322	9.298%	4110
A & B & H	1232	1232	1226	904	992	992	322	9.333%	4208
A & B & J	1232	1232	1226	904	947	947	322	9.457%	4178
A & B & C	1232	1232	1226	912	1161	1161	314	8.676%	3928
A & B & D	1232	1232	1226	912	1072	1072	314	8.895%	4151
A & B & F	1232	1232	1226	912	1025	1025	314	9.015%	3954
A & B & I	1232	1232	1226	910	955	958	313	9.171%	11655

Table 6.3: *The 10 best groups of three picking lines that resulted in the largest savings in cycles travelled when using the tabu search when considering two picking lines. All computational times are indicated in seconds.*

CHAPTER 7

Conclusion

Contents

7.1 Thesis summary and achievement of goals . . . 137
7.2 Thesis contributions . . . 138
7.2.1 Quick heuristic and metaheuristic approaches for the OSP . . . 138
7.2.2 Novel clustering algorithms for the SLP . . . 139
7.2.3 Preliminary investigation into the SPLAP . . . 139
7.3 Possible future work . . . 139
7.3.1 Developing criteria for measuring performance of the SPLAP . . . 139
7.3.2 Finding bounds on improvements of the SLP . . . 139
7.3.3 Measuring the trade-offs for using non-exact approaches to solve the SPLAP . . . 139
7.3.4 Optimal number of pickers in a picking line . . . 140
7.3.5 Optimal mix of picking line sizes . . . 140
7.3.6 The effect of KPIs on the DC operations . . . 140
7.3.7 Permanent picking line . . . 140

This chapter contains a summary of the work contained in this thesis, followed by a description of ideas for future research.

7.1 Thesis summary and achievement of goals

This thesis starts by defining supply chains and the importance of picking lines and picking line operations within a broader logistics framework. An introduction of the current state of picking line activities with the Pep DC's are introduced. Three subproblems are identified that divides the picking line operations within the Pep DC into three problems that may be addressed individually, each subproblem relying on the results of its predecessor. Objectives set out for this thesis also forms part of the introductory chapter.

The purpose of Chapter 2 is to provide a basic understanding of all the components that influence the operations within a supply chain. A literature review is presented covering all these aspects and by putting special focus on picking line configurations and operations. Advances in picking line optimisation are included and the important contributions are mentioned. The operations

within the Pep DC is also discussed in detail in Chapter 3. All the operations and considerations before, during and after order picking are explained.

The three subproblems are modelled mathematically. An exact approach for the OSP is modelled in §4.2 as an integer progamming formulation, a lower bound is presented for the SLP in §5.1 in the form of a mixed integer programming formulation and a goal programming formulation is constructed for the SPLAP in Appendix A. This is done in fulfillment of Thesis Objective I as stated in §1.6. Each of these formulations cannot be solved in a reasonable time frame.

A number of definitions are then presented which lead to alternative methods of addressing the OSP. Instead of sequencing orders to be picked, each order receives a starting location. Once each order, in a wave, has received a starting location, a heuristic is used to deliver a sequence that do not add more than one pick cycle to the solution. Heuristic and metaheuristic approaches are constructed to address the OSP in this manner, which fulfills Thesis Objective III.

In fulfillment of Thesis Objective IV the SLP is solved in a variety of manners. Initially classical approaches from literature is used to formulate the SLP. Consequenctly, a metaheuristic approach, agglomerative clustering algorithms and novel clustering approaches specifically identified for the SLP are developed.

Managerial and practical implementations for SPLAP are examined. A number of measures for the SPLAP are presented that address various aspects of importance of the SPLAP. A nearest neighbour approach and a tabu search algorithm is presented to address the SPLAP. This is done in order to fulfull Thesis Objective V.

Computational results for each of the subproblems are presented as well as computational times where applicable. Conclusions are made in reference to the solution qualities (and computation times) of the various algorithms used to solve the subproblem. Thesis Objective II is addressed in this manner.

The following subsection contains open questions and new questions that have arised during the completion of this thesis, thus addressing Thesis Objective VI. Thesis Objective VII is presented in §7.3.

7.2 Thesis contributions

Some achievements of this thesis are presented in the following subsections.

7.2.1 Quick heuristic and metaheuristic approaches for the OSP

Matthews & Visagie [77] provided a lower bound to the OSP. The solution time to calculate the lower bound for an average data set may be excessive. Algorithms that find near-optimal solutions in a fraction of the time is needed when the SLP or the SPLAP is addressed.

The heuristic and metaheuristic implementations for solving the OSP may aid greatly in future work on the SLP and SPLAP. A wide range of various algorithms are presented with the solution qualities and computational times. Although the types of algorithms are known, all these approaches were adapted to solve a novel problem in literature.

7.2.2 Novel clustering algorithms for the SLP

Two clustering algorithms were presented for implementation of the SLP. These clustering algorithms were constructed specifically to suite the characteristics of the SLP. The first clustering heuristic was to group SKUs according to the number of orders that they have in common, while the second clustering heuristic grouped SKUs according to their respective frequencies.

7.2.3 Preliminary investigation into the SPLAP

Some initial attempts to solve the SPLAP are presented in this thesis and may be used as a guideline as to future work. Promising results were obtained from a tabu search implementation where products from two or three picking lines were reassigned to the picking lines to minimise the total distance travelled by the pickers. A number of criteria was discussed which may be used as an objective for the SPLAP.

7.3 Possible future work

Seven categories of suggestions are made with respect to possible future research emanating from the work presented in this thesis.

7.3.1 Developing criteria for measuring performance of the SPLAP

Conflicting objectives hinders a clear and direct goal when addressing the SPLAP. A number of possible measures for determining ideal waves assigned to picking lines are suggested. These measures do not, however, allow for much flexibility.

Alternative measures may be created to deliver specific results in the composition of possible waves. The measures used in this thesis do not specify the ideal number of SKUs in a wave, rather SKUs are sequentially placed in a wave. These and similar other aspects of constructing waves may be investigated for constructing more desirable waves.

As mentioned in §6.3, many issues concerning the SPLAP was not considered in this thesis. A manner of addressing all the components of SPLAP must be developed. Also, the term "workbalance" must remain flexible in such a way that alterations in the picking line manager's preferences may be incorporated in practical settings.

7.3.2 Finding bounds on improvements of the SLP

Little improvement were found in the number of cycles travelled when addressing the SLP. Due to the size of the SLP for an average sized problem, exact solutions could not be calculated. An attempt to find lower bounds of the SLP may aid in identifying areas of improvement.

7.3.3 Measuring the trade-offs for using non-exact approaches to solve the SPLAP

The results of the SPLAP displayed in this thesis used a greedy heuristic to solve the OSP for each SPLAP configuration. Relatively little computational time is required to retrieve the

results for using a greedy heuristic in comparison to using a metaheuristic to solve the OSP during the process of constructing a wave during the SPLAP process. However, the solution quality when moving from a greedy heuristic (NSO) to a metaheuristic (GEO) to solve the OSP does not improve the solution quality considerably.[1]

7.3.4 Optimal number of pickers in a picking line

In this thesis order picking is sequenced for a single picker whereby the work may be partitioned when multiple workers are present in the picking line. Orders may be allocated in a dynamic fashion to pickers in practise. However, no investigation was made into a desired number of workers within various picking line configurations. Cost-benefit analysis may be done to weigh additional savings when the number of pickers working in a picking line is increased or decreased.

7.3.5 Optimal mix of picking line sizes

The data presented in this thesis is based on past data gathered from operations with the Durban DC operated by Pep. Pep uses a system where a potential picking line containing 112 locations is partitioned into two picking lines each containing 56 locations. No investigation was done into the optimal size of a picking line. Smaller picking lines (in terms of the number of locations within the picking line) containing high frequency products and longer picking lines with lower frequency products may possibly increase productivity in the Pep DC.

7.3.6 The effect of KPIs on the DC operations

Currently Pep uses a FIFO-system, with a release period of 5 workings days. The release period takes into effect once the SKUs have been received by the DC and the distribution has been released from Central Office. The FIFO-system limits the scope of constructing waves, which may impact operations within the picking lines.

7.3.7 Permanent picking line

To improve efficiency within the picking line operations, Pep uses a permanent picking line. The permanent picking line contains SKUs that are fixed to their respective locations. These products often form part of the high frequency SKUs. Replenishment issues regularly occur in this picking line. This picking line makes use of flowracks which acts as locations, whith three levels of flowracks. The problem entails that the flowracks are unable to contain a large buffer of stock. Workers are often unable to consistently replenish these locations. Currently this picking line cannot operate on a permanent basis and consequently operated by means of wave picking.

[1] Experiments were conducted where the NSO was used to solve the OSP when the SPLAP was solved by means of the tabu search approach. Experiments were then also conducted when the OSP was solved by means of the GEO metaheuristic approach which was unable to deliver considerably better results.

References

[1] Aarts E & Korst J, 1990, *Simulated annealing and Boltzmann machines: A stochastic approach to combinatorial optimization and neural computing*, John Wiley & Sons, New York (NY).

[2] Alba E, Blum C & Roli A, 2005, *An introduction to metaheuristic techniques*, pp. 3–42 in *Parallel metaheuristics: A new class of algorithms*, John Wiley & Sons, Hoboken (NJ).

[3] Armbruster D, Gel ES & Murakami J, 2007, *Bucket brigades with worker learning*, European Journal of Operational Research, **176(1)**, pp. 264–274.

[4] Armstrong RD, Cook WD & Saipe AL, 1979, *Optimal batching in a semi-automated order picking system*, Journal of the Operational Research Society, **30(8)**, pp. 711–720.

[5] Ayers JB, 2006, *Handbook of supply chain management*, 2nd Edition, Auerbach Publications, Boca Raton (FL).

[6] Bäck T & Hoffmeister F, 1991, *Extended selection mechanisms in genetic algorithms*, Proceedings of the Fourth International Conference on Genetic Algorithms, San Mateo (CA), pp. 710–719.

[7] Ballou RH, 1999, *Business logistics management*, Prentice-Hall inc, Upper Saddle River (NJ).

[8] Bak P & Sneppen K, 1993, *Punctuated equilibrium and criticality in a simple model of evolution*, Physical Review Letters, **71(24)**, pp. 4083–4086.

[9] Bartholdi JJ & Eisenstein DD, 1996, *A Production line that balances itself*, Operations Research, **44(1)**, pp. 21–34.

[10] Bartholdi JJ, Eisenstein DD & Foley RD, 1999, *Performance of bucket brigades when work is stochastic*, Operations Research, **49(5)**, pp. 710–719.

[11] Bartholdi JJ, Eisenstein DD & Lim YF, 2006, *A chaos and convergence in bucket brigades with finite walk-back velocities*, [Online], [Cited September 29th, 2011], Available from `http://www2.isye.gatech.edu/jjb/bucket-brigades/papers/Bartholdi_Eisenstein_Lim_2005_v9.pdf`.

[12] Bartholdi JJ, Hackman ST, 2010, *Warehouse & distribution science*, Release 0.92, [Online], [Cited March 24th, 2010], Available from `http://www.tsp.gatech.edu/problem/index.html`.

[13] Bartholdi JJ & Platzman LK, 1982, *An $\mathcal{O}(N \log N)$ planar travelling salesman heuristic based on spacefilling curves*, Operations Research Letters, **1(4)**, pp. 121–125.

[14] Bartholdi JJ & Platzman LK, 1986, *Retrieval strategies for a carousel conveyor*, IIE Transactions, **18(2)**, pp. 166–173.

[15] Boettcher S & Percus AG, 1999, *Extremal optimization: Methods derived from co-evolution*, Proceedings of the Genetic and Evolutionary Computation Conference, San Francisco (CA).

[16] Boettcher S & Percus AG, 2001, *Extremal optimization for graph partitioning* , Physical Review E, **64(2)**, pp. 1–13.

[17] Bellmore M & Nemhauser GL, 1966, *The traveling salesman problem: A survey*, Operations Research, **16(3)**, pp. 538–558.

[18] Bergmans PP, 1972, *Minimizing expected travel time on geometrical patterns by optimal probability rearrangements*, Information and Control, **20**, pp. 331–350.

[19] Blum C & Roli A, 2003, *Metaheuristics in combinatorial optimization: Overview and conceptual comparison*, ACM Computing Surveys, **35(3)**, pp. 268–308.

[20] Bonabeau E, Dorigo M & Theraulaz G, 1999, *Swarm intelligence: From natural to artificial systems*, Oxford University Press, Oxford.

[21] Bowersox DJ, Closs DJ & Cooper MB, 2007, *Supply chain logistics management*, 2nd Edition, McGraw-Hill Education, Berkshire.

[22] Britannica: Academic edition, 2011, *Amorphous solid*, [Online], [Cited May 24th, 2011], Available from `http://www.britannica.com/EBchecked/topic/21328/amorphous-solid`.

[23] Carter R, Price PM & Emmett S, 2005, *Stores and distribution management*, 2nd Edition, McGraw-Hill Education, Wirral.

[24] Camazine S, Deneubourg J, Franks N, Sneyd J, Hermaulaz G & Bonabeau E, 2000, *Self-organistion in biological systems*, Liverpool Business Publishing, Princeton (NJ).

[25] Cook W, 2009, *Warehouse & distribution science*, [Online], [Cited March 24th, 2010], Available from `http://www.tsp.gatech.edu/problem/index.html`.

[26] Coyle JJ, Bardi EJ & Langley CJ, 2003, *The management of business logistics: A supply chain perspective*, Thomson Learning, Quebec.

[27] Cormier G & Gunn EA, 1992, *A review of warehouse models*, European Journal of Operational Research, **58(1)**, pp. 3–13.

[28] De Koster R, Le-Duc T & Roodbergen K J, 2007, *Design and control of warehouse order picking: A literature review*, European Journal of Operations Research, **182(2)**, pp. 481–501.

[29] De Koster R & Yu M, 2009, *The impact of order batching area zoning on order picking system performance*, European Journal of Operations Research, **198(2)**, pp. 480–490.

[30] de Koster R & Roodbergen K J, 2000, *Routing order pickers in a warehouse with a middle aisle*, European Journal of Operations Research, **133(1)**, pp. 32–43.

[31] De Sousa FL, Vlassov V & Ramos FM, 2004, *Generalized extremal optimization: An applicationin heat pipe design*, Applied Mathematical Modelling, **28(10)**, pp. 911–931.

[32] Domingo E, 2010, Distribution centre manager, *PepStores Ltd., South Africa*, [Personal Communication], Contactable at `dbn1@pepstores.com`.

[33] Dorigo M & Stützle T, 2004, *Ant colony optimization*, The MIT Press, Cambridge (MA).

[34] Dréo J, Pétrowski A, Siarry P & Taillard E, 2006, *Metaheuristics for hard optimisation: Methods and case studies*, Springer-Verlag, Berlin.

[35] Ellis Systems Corporation, 2011, *Businesses utilizing professionally designed industrial storage systems*, [Online], [Cited July 27th, 2011], Available from `http://www.ellismh.com/index.php?url=index`.

[36] Elsayed EA, 1981, *Algorithms for optemal material handling in automatic warehouse systems*, International Journal of Production Research, **19(5)**, pp. 525–535.

[37] Elsayed EA & Stern RG, 1983, *Computerized algorithms for order processing in automated warehousing systems*, International Journal of Production Research, **21(4)**, pp. 579–586.

[38] Elsayed EA & Unal OI, 1989, *Order batching algorithms and travel-time estimation for automated storage/retrieval systems*, International Journal of Production Research, **27(7)**, pp. 1097–1114.

[39] Feremans C, Labbé M & Laporte G, 2003, *Generalized network design problems*, European Journal of Operations Research, **148(1)**, pp. 1–13.

[40] Fogel BD, 2000, *What is evolutionary computation?*, IEEE Spectrum, February, pp. 26–32.

[41] Gendreau M, 2003, *An introduction to tabu search*, pp. 37–52 in *Handbook of metaheuristics*, Kluver Academic Publishers, Dordrecht.

[42] Ghosh JB & Wells CE, 1992, *Optimal retrieval strategies for carousel conveyors*, Mathematical and Computer Modelling, **16(10)**, pp. 59–70.

[43] Gibson DR & Sharp GP, 1992, *Order batching procedures*, European Journal of Operations Research, **58(1)**, pp. 57–67.

[44] Glover F, 1986, *Future paths for integer programming and links to artificial intelligence*, Computers and Operations Research, **13**, pp. 533–549.

[45] Glover F, 1990, *Tabu search — Part II*, Ooperations Research Society of America, **2(1)**, pp. 4–32.

[46] Goetschalckx M & Ashayeri J, 1989, *Classification and design of order picking*, Logistics Information Management, **2(2)**, pp. 99–106.

[47] Grant DB, Lambert DM, Stock TR & Ellram LM, 2006, *Fundamentals of logistics management: European edition*, McGraw-Hill Education, Berkshire.

[48] Graves SC, Hausman WH US & Schwarz LB, 1977, *Storage-retrieval interleaving in automatic warehousing systems*, Management Science, **23(9)**, pp. 935–945.

[49] Gray AE, Karmarkar US & Seidmann A, 1992, *Design and operation of an order-consolidation warehouse: Models and applications*, European Journal of Operations Research, **58(1)**, pp. 14–36.

[50] Gu J, Goetschalckx M & McGinnis LF, 2007, *Research on warehouse operations: A comprehensive review*, European Journal of Operational Research, **177(1)**, pp. 1–21.

[51] Gutin G & Punnun AP, 2007, *The travelling salesman problem and its variations*, Springer, Dordrecht.

[52] Gutin G & Yeo A, 2003, *Assignment problem based algorithms are impractical for the generalized TSP*, Australasian Journal of Combinatorics, **27**, pp. 149–153.

[53] Hassini E, 2009, *One-dimensional carousel storage problems: Applications, review and generalizations*, Information Systems and Operational Research, **47(2)**, pp. 81–92.

[54] Hassini E & Vickson R, 2003, *A two-carousel storage location problem*, Computers & Operations Research, **30(4)**, pp. 527–539.

[55] Helo PT, 2000, *Dynamic modelling of surge effect and capacity limitation in supply chains*, International Journal of Production Research, **38(17)**, pp. 4521–4533.

[56] Banzhaf W, Koza JR, Ryan C, Spector L & Jacob C, 2000, *Genetic programming*, IEEE Intelligent Systems and their Applications, **15(3)**, pp. 74–84.

[57] Hsieh L & Tsai L, 2006, *The optimum design of a warehouse system on order picking efficiency*, The International Journal of Advanced Manufacturing Technology, **28(5–6)**, pp. 626–637.

[58] Hsu CM, Chen KY & Chen CM, 2005, *Batching orders in warehouses by minimizing travel distance with genetic algorithms*, Computers in Industry, **56(2)**, pp. 169–178.

[59] Hwang H & Lee MK, 1988, *Order batching algorithms for a man-on-board automated storage and retrieval system*, Engineering Costs and Production Economics, **13(4)**, pp. 285–294.

[60] Hwang H, Baek W& Lee MK, 1988, *Order batching algorithms for a man-on-board automated storage and retrieval system*, International Journal of Production Research, **26(2)**, pp. 189–201.

[61] Ignizio JP, 1981, *Linear programming in single and multiple-objective systems*, 1st Edition, Princetn-Hall, London.

[62] Interactive Business Communications, 2011, *Manufacturing & logistics IT*, [Online], [Cited July 26th, 2011], Available from `http://www.logisticsit.com/absolutenm/templates/article-wms.aspx?articleid=4965&zoneid=4`.

[63] Jane CC & Laih YW, 2005, *A clustering algorithm for item assignment in a synchronised zone order picking system*, European Journal of Operations Research, **166(2)**, pp. 489–496.

[64] Kalczynski P, 1999, *A java implementation of the branch and bound algorithm: The asymmetric travelling salesman problem*, Journal of Object Technology, **4(1)**, pp. 156–163.

[65] Kardi Teknomo's Page, *Chebyshev Distance*, [Online], [Cited March 6th, 2011], Available from `http://people.revoledu.com/kardi/tutorial/Similarity/ChebyshevDistance.html`.

[66] Kaufman L & Rousseeuw PJ, 2005, *Finding groups in data: An introduction to cluster analysis*, John Wiley & Sons, Hoboken (NJ).

[67] Kim WB & Koenigsberg E, 1987, *The efficiency of two groups of N machines served by a single robot*, Journal of the Operational Research Society, **38(6)**, pp. 523–538.

[68] Koza J, 1992, *Genetic programming: On the programming of computers by means of natural selection*, MIT Press, Cambridge (MA).

[69] Klem C, 2010, Supply Chain Director, *PepStores Ltd., South Africa*, [Personal Communication], Contactable at `cornek@pepstores.com`.

[70] Koenigsberg E & Mamer J, 1982, *The analysis of production systems*, International Journal of Production Research, **20(1)**, pp. 1–16.

[71] Lam JKC & Postle R, 2006, *Textile and apparel supply chain management in Hong Kong*, International Journal of Clothing Science and Technology, **18(4)**, pp. 265–277.

[72] Lim WK, Bartholdi III JJ & Platzman LK, 1985, *Storage schemes for carousel corweyors under real time control*, Material Handling Research Center Tech, Technical Report, Georgia Institute of Technology, Atlanta (GA).

[73] Lindo Systems, *Lingo 11*, [Online], [Cited March 26th, 2010], Available from `http://www.lindo.com`.

[74] Litvak N & Vlasiou M, 2009, *A survey on performance analysis of warehouse carousel systems*, (Unpublished) Technical Report, Department of Applied Mathematics, University of Twente, Twente.

[75] Lötter T, 2010, DC operations analyst, *PepStores Ltd., South Africa*, [Personal Communication], Contactable at `tjaartl@pepstores.com`.

[76] Manzini R, Gamberi M, Persona A & Regattieri A, 2007, *Design of a class based storage picker to product order picking system*, The International Journal of Advanced Manufacturing Technology, **32(7–8)**, pp. 811–821.

[77] Matthews J & Visagie SE, 2011, *Order sequencing in a unidirectional picking line*, [Submitted].

[78] MathWorks, *Matlab — The language of technical computing*, [Online], [Cited November 18th, 2010], Available from `http://www.mathworks.com/products/matlab/`.

[79] Meller RD & Klote JF, 2004, *Throughput model for carousel/VLM pods*, IIE Transactions, **36(8)**, pp. 725–741.

[80] Moodley T, 2010, Picking line manager, *PepStores Ltd., South Africa*, [Personal Communication], Contactable at `dbn1@pepstores.com`.

[81] Mjokovana N, 2010, Picking line scheduler, *PepStores Ltd., South Africa*, [Personal Communication], Contactable at `dbn1@pepstores.com`.

[82] Mulcahy DE, 1994, *Warehouse distribution and operations handbook*, McGraw-Hill Education, Berkshire.

[83] Nance RE, 1971, *On time flow mechanisms for discrete system simulation*, Management Science, **18(1)**, pp. 59–73.

[84] Nel JH, Krygsman SC & de Jong T, 2008, *The identification of possible future provincial boundaries for South Africa based on an intramax analysis of journey-to-work data*, ORiON, **24(2)**, pp. 131–156.

[85] Pep Stores, 2010, *How we operate*, [Online], [Cited March 18th, 2010], Available from `http://www.pepstores.com`.

[86] Pan JHC & Wu M, 2009, *A study of storage assignment problem for an order picking line in a pick-and-pass warehousing system*, Computers & Industrial Engineering, **57**, pp. 261–268.

[87] Park BC, Park JY & Foley RD, 2003, *Carousel system performance*, Journal of Applied Probability, **40(3)**, pp. 602–612.

[88] Patnaik LM, 1994, *Genetic algorithms: A survey*, Computer, **27(6)**, pp. 17–26.

[89] Pentico DW, 2007, *Assignment problems: A golden anniversary survey*, European Journal of Operational Research, **176(2)**, pp. 774–793.

[90] Qi X, 1994, *Theoretical analysis of evolutionary algorithms with an infinite population size in continuous space part I: Basic properties of selection and mutation*, IEEE Transactions of Neural Networks, **5(1)**, pp. 102–119.

[91] Ratliff HD & Rosenthal AS, 1983, *Order-picking in a rectangular warehouse: A solvable case of the traveling salesman problem*, Operations Research, **31(3)**, pp. 507–521.

[92] Roli A, *An introduction to metaheuristics*, (Slide-show) Department of electronics, Computer science and systems, University of Bologna, Bologna.

[93] Riegger PM, 2010, *Literature survey on nearest neighbor search and search in graphs*, (Unpublished) Institute of Formal Methods in Computer Science, University of Stuttgart, Stuttgart.

[94] Roodberger KJ & de Koster R, 2001, *Routing methods for warehouses with multiple cross aisles*, International Journal of Production Research, **39(9)**, pp. 1865–1883.

[95] Rosenwein MB, 1996, *A comparison of heuristics for the problem of batching orders for warehouse selection*, International Journal of Production Research, **34(3)**, pp. 657–664.

[96] Rouwenhorst B, Reuter B, Stockrahm V, Van Houtum GJ, Mantel RJ & Zijm WHM, 2000, *Warehouse design and control: Framework and literature review*, European Journal of Operational Research, **122(3)**, pp. 515–533.

[97] Ruben RA & Jacobs FR, 1999, *Batch construction heuristics and storage assignment strategies for walk/ride and pick systems*, Management Science, **45(4)**, pp. 575–596.

[98] SAS, *The power to know*, [Online], [Cited November 18th, 2010], Available from `http://www.sas.com/ads/google_sasinfo/index_new2.html`.

[99] Szekely GJ & Rizzo ME, 1963, *Application of an hierarchical grouping procedure to a problem of grouping profiles*, Educational Psycology, **23(1)**, pp. 69–81.

[100] Scott M, 2009, *Optimisation of a single-aisle picking line*, HonsBSc seminar, University of Stellenbosch, Stellenbosch.

[101] Sun Mircosystems, *Java*, [Online], [Cited March 26th, 2010], Available from `http://java.sun.com`.

[102] Storage Systems, 2011, *Overview of small item storage*, [Online], [Cited July 27th, 2011], Available from `http://warehousestorage.blogspot.com/`.

[103] Theys C, Braysy O, Dullaert W & Simchi-Levi E, 2010, *Using a tsp-heuristic for routing order pickers in warehouses*, European Journal of Operations Research, **200(3)**, pp. 755–763.

[104] Thompkins JA, White JA, Bozer YA, & Tanchoco JMA, 2003, *Facilities planning*, 3rd Edition, John Wiley & Sons, Hoboken (NJ).

[105] Toth P & Vigo D, 2002, *The vehicle routing problem*, Society for Industrial and Applied Mathematics, Philadelphia (PA).

[106] Trifactor, 2011, *Distribution Solutions*, [Online], [Cited July 26th, 2011], Available from `http://www.trifactor.com/Material-Handling-Systems/Order-Picking-Systems`.

[107] Van Dieman G, 2006, *A comparison of exact and heuristic solution methodologies for the classical assignment problem and its variations*, (Unpublished), University of Stellenbosch, Stellenbosch.

[108] Van Den Berg JP, 1996, *Multiple order pick sequencing in a carousel system: A solvable case of the rural postman problem*, The Journal of the Operational Research Society, **47(12)**, pp. 1504–1515.

[109] Van Laarhoven PJM & EHL Aart, 1988, *Simulated annealing: Theory and applications*, D. Reidel Publishing Company, Dordrecht.

[110] Vickson RG & Fujimoto A, 1996, *Optimal storage locations in a carousel storage and retrieval system*, Location Science, **4(4)**, pp. 237–245.

[111] Vickson RG & Lu X, 1998, *Optimal product and server locations in one-dimensional storage racks*, European Journal of Operational Research, **105(1)**, pp. 18–28.

[112] Vlasiou M & Adan X, 2004, *A lindley-type equation arising from a carousel problem*, Journal of Applied Probability, **41(4)**, pp. 1171–1181.

[113] Vogt JJ, Pienaar WJ & De Wit PWC, 2002, *Business logistics management: Theory and practise*, 1st Edition,Oxford University Press, Cape Town.

[114] Ward JH, 1963, *Hierarchical grouping to optimize an objective function*, Journal of the American Statistical Association, **58(301)**, pp. 236–244.

[115] Ward JH & Hook ME, 1963, *Hierarchical clustering via joint between-within distances: Extending Ward's minimum variance method*, Educational Psycology, **22**, pp. 151–183.

[116] Warehouse & Logistics News, 2011, *Drax harvests benefits from demag crane*, [Online], [Cited July 26th, 2011], Available from `http://www.warehousenews.co.uk/2009/11/drax-harvests-benefits-from-demag-crane/`.

[117] Wen UP & Chang DT, 1988, *Picking rules for a carousel conveyor in an automated warehouse*, International Journal of Management Science, **16(2)**, pp. 145–151.

[118] Wiese T, 2011, *Global optimization algorithms: Theory and application*, 2nd Edition, [Online], [Cited May 31st, 2011], Available from `http://www.it-weise.de/`.

[119] Wilson E & Hölldobler, 1988, *Dense heterarchy and mass communication as the basis of organisation in ant colonies*, Trend in Ecology and Evolution, **3**, pp. 65–68.

[120] Winston WL, 2004, *Operations research: Applications and algorithms*, 4th Edition, Duxbury Press, Toronto.

[121] Yoon CS & Sharp GP, 1996, *A structured procedure of analysis and design of order picks systems*, IIE Transactions, **28(5)**, pp. 379–389.

[122] Yu M & de Koster RBM, 2009, *The impact of order batching and picking area zoning on order picking system performance*, European Journal of Operational Research, **198(2)**, pp. 480–490.

APPENDIX A

An exact formulation for the SPLAP

An exact formulation by means of using a mixed integer programming approach is presented. This formulation is an extension of the formulation presented by Scott [100], where the objective is to minimise the total picking distance of a single picker on a single picking line. This formulation considers all the pickers each picking line and all the picking lines within a DC.

The formulation is based on a two travelling salesman problem where each node is a location. The pickers assigned to the various picking lines may visit the locations that contain SKUs necessary for the order — these locations are called *active* locations for a customer order. All the pickers are therefore not required to visit all the locations, *i.e.* all the nodes. To avoid creating subtours, an *inactive* picker is created for each picking line to visit all the locations that do not contain SKUs necessary for the order that is being picked.

The problem is formulated as a mixed integer goal programming problem. Multiple-objectivity is a highly flexible and practical methodology for modelling, solving and analysing problems for which it is necessary to consider the impact of multiple, conflicting objectives [61]. The first objective is used to ensure that the total distance travelled by all the pickers are minimised, while the second goal ensures that each picker in each picking line travels approximately the same distance. A picking line finishes its operations when the last picker finishes the orders assigned to him/her, therefore, the workload among the pickers have to be balanced.

The input data for this formulation consists of

$D_{k\ell h}^{ij}$	as the number of locations that must be picked from, when starting at location i in order k and travelling to location j in order ℓ in picking line h,
P_{kh}^{t}	which is equal to 1 if SKU t is in order k in picking line h and
M_h	be the expected distance travelled in picking line h.

A number of variables required in modelling this problem are defined. Let

$z^{ij}_{k\ell rh}$ be equal to 1 if a picker r in picking line h travels from location i in order k to location j in order ℓ,
$y^{ij}_{k\ell h}$ be equal to 1 if the inactive picker in picking line h travels from location i in order k to location j in order ℓ,
x^t_{ih} be equal to 1 if SKU t in picking line h is in location i and
w^k_{ih} be equal to 1 if location i in picking line h must be visited in order k,
η_h be the negative deviation of the total distance travelled in picking line h,
ρ_h be the positive deviation of the total distance travelled in picking line h,
η_{rh} be the negative deviation of the total distance travelled by picker r in picking line h and
ρ_{rh} be the positive deviation of the total distance travelled by picker r in picking line h.

A set of constants are defined. Let

n be the number of orders,
m be the number of locations,
T be the number of SKUs,
T_h be the number of SKUs assigned to picking line h,
R be the number of pickers and
H be the number of picking lines in the DC.

The objective is to firstly minimise the total number of cycles travelled by all pickers and then to balance the number of cycles travelled over all the pickers in a picking line, *i.e.* to

$$\text{lexicographically minimise } \boldsymbol{\alpha} = \left\{ \left(\sum_{h=1}^{H} \eta_h \right), \left(\sum_{h=1}^{H} \sum_{r=1}^{R} \eta_{rh} \right) \right\} \tag{A.1}$$

$$\text{subject to } \sum_{t=1}^{T} x^t_{ih} = 1 \qquad \begin{matrix} i = 1, 2, \ldots, m, \\ h = 1, 2, \ldots, H, \end{matrix} \tag{A.2}$$

$$\sum_{i=1}^{m} x^t_{ih} = 1 \qquad \begin{matrix} t = 1, 2, \ldots, T, \\ h = 1, 2, \ldots, H, \end{matrix} \tag{A.3}$$

$$\sum_{k=1}^{n} \sum_{\ell=1}^{n} \sum_{i=1}^{m} \sum_{j=1}^{m} D^{ij}_{k\ell h} z^{ij}_{k\ell rh} + \eta_h - \rho_h = M_h \qquad \begin{matrix} h = 1, 2, \ldots, H, \\ r = 1, 2, \ldots, R, \end{matrix} \tag{A.4}$$

$$\sum_{r=1}^{R} \sum_{k=1}^{n} \sum_{\ell=1}^{n} \sum_{i=1}^{m} \sum_{j=1}^{m} D^{ij}_{k\ell h} z^{ij}_{k\ell rh} + \eta_{rh} - \rho_{rh} = \frac{M_h}{T_h} \qquad h = 1, 2, \ldots, H, \tag{A.5}$$

$$P^t_{kh} x^t_{kh} \leq w^k_{ih} \qquad \begin{matrix} t = 1, 2, \ldots, T, \\ k = 1, 2, \ldots, n, \\ i = 1, 2, \ldots, m, \\ h = 1, 2 \ldots, H, \end{matrix} \tag{A.6}$$

$$1 + P^t_{kh} - x^t_{kh} \geq w^k_{ih} \qquad \begin{matrix} t = 1, 2, \ldots, T, \\ k = 1, 2, \ldots, n, \\ i = 1, 2, \ldots, m, \\ h = 1, 2, \ldots, H, \end{matrix} \tag{A.7}$$

$$\sum_{r=1}^{R} \sum_{k=1}^{n} \sum_{i=1}^{m} z^{ij}_{k\ell rh} + \sum_{k=1}^{n} \sum_{i=1}^{m} y^{ij}_{k\ell h} = 1 \qquad \begin{matrix} j = 1, 2, \ldots, m, \\ \ell = 1, 2, \ldots, n, \\ h = 1, 2, \ldots, H, \end{matrix} \tag{A.8}$$

$$\sum_{\ell=1}^{n}\sum_{j=1}^{m}(y_{k\ell h}^{ij} - y_{\ell k h}^{ji}) = 0 \qquad \begin{array}{l} i = 1, 2, \ldots, m, \\ k = 1, 2, \ldots, n, \\ h = 1, 2, \ldots, H, \end{array} \tag{A.9}$$

$$\sum_{r=1}^{R}\sum_{\ell=1}^{n}\sum_{j=1}^{m}(z_{k\ell r h}^{ij} - z_{\ell k r h}^{ji}) = 0 \qquad \begin{array}{l} i = 1, 2, \ldots, m, \\ k = 1, 2, \ldots, n, \\ h = 1, 2, \ldots, H, \end{array} \tag{A.10}$$

$$\sum_{j=1}^{m}\sum_{\ell=1}^{n} y_{0\ell h}^{0j} = 1 \qquad h = 1, 2, \ldots, H, \tag{A.11}$$

$$\sum_{r=1}^{R}\sum_{\ell=1}^{n}\sum_{j=1}^{m} z_{0\ell r h}^{0j} = 1 \qquad h = 1, 2, \ldots, H, \tag{A.12}$$

$$\sum_{k=1}^{n}\sum_{i=1}^{m} y_{k0h}^{i0} = 1 \qquad h = 1, 2, \ldots, H, \tag{A.13}$$

$$\sum_{r=1}^{R}\sum_{k=1}^{n}\sum_{i=1}^{m} z_{k0rh}^{i0} = 1 \qquad h = 1, 2, \ldots, H, \tag{A.14}$$

$$u_{kh}^{i} - u_{\ell h}^{j} + mnH\left(y_{k\ell h}^{ij} + \sum_{r=1}^{R} z_{k\ell r h}^{ij}\right) \leq mnH - 1 \qquad \begin{array}{l} 1 \leq i, j \leq m, \\ 1 \leq k, \ell \leq n, \\ \text{if } i = j,\ k \neq \ell, \\ \text{if } k = \ell,\ i \neq j, \\ h = 1, 2, \ldots, H \end{array} \tag{A.15}$$

$$\sum_{i=1}^{m}\sum_{k=1}^{n}(y_{k\ell h}^{ij} + w_{kh}^{i}) = 1 \qquad \begin{array}{l} j = 1, 2, \ldots, m, \\ k = 1, 2, \ldots, n \\ h = 1, 2, \ldots, H, \end{array} \tag{A.16}$$

$$z_{k\ell r h}^{ij} \in \{0, 1\} \qquad \begin{array}{l} i = 1, 2, \ldots, m, \\ j = 1, 2, \ldots, m, \\ k = 1, 2, \ldots, n, \\ \ell = 1, 2, \ldots, n, \\ r = 1, 2, \ldots, R, \\ h = 1, 2, \ldots, H, \end{array} \tag{A.17}$$

$$y_{k\ell h}^{ij} \in \{0, 1\} \qquad \begin{array}{l} i = 1, 2, \ldots, m, \\ j = 1, 2, \ldots, m, \\ k = 1, 2, \ldots, n, \\ \ell = 1, 2, \ldots, n, \\ h = 1, 2, \ldots, H, \end{array} \tag{A.18}$$

$$x_{ih}^{t} \in \{0, 1\} \qquad \begin{array}{l} i = 1, 2, \ldots, m, \\ t = 1, 2, \ldots, T, \\ h = 1, 2, \ldots, H, \end{array} \tag{A.19}$$

$$w_{ih}^{k} \in \{0, 1\} \qquad \begin{array}{l} i = 1, 2, \ldots, m, \\ k = 1, 2, \ldots, n, \\ h = 1, 2, \ldots, H, \end{array} \tag{A.20}$$

$$u_{kh}^{i} \in \{0, 1\} \qquad \begin{array}{l} i = 1, 2, \ldots, m, \\ k = 1, 2, \ldots, n, \\ h = 1, 2, \ldots, H. \end{array} \tag{A.21}$$

Constraint sets (A.2) and (A.3) assign SKUs to locations. Constraint sets (A.4) reflect the

goal of minimising the total distance travelled by all the pickers in the DC, while constraint sets (A.5) ensures that the distance travelled by each picker is minimised. Constraint sets (A.6) and (A.7) states that a SKU must be in an order and be assigned to a location in order for the location to be considered active. It is ensured in constraint set (A.8) that each location is visited by exactly one of the pickers in that picking line. From constraint sets (A.9) and (A.10) it is ensured that the picker that visits a location should be the picker to leave the location. Constraint sets (A.11) to (A.12) ensure that both all pickers have to start and end at location 0, *i.e.* at the depot (or the first location). The subtour breaking constraints are defined in (A.13) and constraint set (A.14) states that the pickers must only visit active locations and the inactive salesmen visit the inactive locations.

Values for M_h in constraint sets (A.4) and (A.5) may be considered as an upper bound for the sum of the distance travelled by all the pickers in a picking line n. The size of this formulation consists of $m^2n^2H(1+R)+nH(T+m)+H(4+R+mn)$ variables of which $m^2n^2H(1+R)+nH(T+m)$ are binary variables and $H(m+T+R+2Tmn+5mn+6)$ constraints.