

An Ant Colony Optimisation Approach to Scheduling Truck and Drone Delivery Systems

by

Tsietsi John Moremi



*Dissertation presented for the degree of Doctor of Philosophy
in the Faculty of Engineering at Stellenbosch University*

Supervisor: Prof. Jacomine Grobler

Co-supervisor: Prof. Phil M. Kaminsky

April 2022

Declaration

By submitting this dissertation 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.

April 2022

Copyright © 2022 Stellenbosch University

All rights reserved

Abstract

'Last mile' logistic scheduling is a complex problem businesses are facing today. Competitive pressure has increased with technological growth. The speed of delivering parcels to customers can be an excellent source of competitive advantage, since businesses are facing the challenge of efficiently delivering parcels to customers on a daily basis. The use of delivery drones in conjunction with traditional delivery vehicles is a new highly promising research direction explored in this thesis.

This dissertation proposes various truck and drone delivery system optimisation problems where a delivery drone is launched from a purpose-built truck, completes additional deliveries while the truck is en route between two customer locations, and intercepts the truck after completing the additional delivery. The dissertation describes the development of an ant colony optimisation algorithm used to solve the problem. More specifically, an ant colony system with k-means clustering was used in this research. Adaptive algorithm control parameters were also used to ensure an acceptable balance between exploration and exploitation throughout the search process.

The algorithm was tested on drone scheduling benchmark problems, optimal solution and other population based metaheuristics and compared against a truck-only delivery system. It was shown that the truck and drone delivery system has a significant positive impact on delivery time performance.

Opsomming

'Last mile' skedulering is 'n ingewikkelde probleem wat ondernemings vandag moet kan hanteer. Mededingende druk het toegeneem met tegnologiese groei. Die spoed van die aflewering van pakkies aan kliënte kan 'n uitstekende bron van mededingende voordeel wees, aangesien ondernemings daaglik die uitdaging ondervind om pakkies doeltreffend aan kliënte te lewer. Die gebruik van onbemande lugvoertuie saam met tradisionele afleweringvoertuie is 'n nuwe, baie belowende navorsingsrigting wat in hierdie tesis ondersoek word.

Hierdie verhandeling beskryf 'n vragmotor-lugvoertuig afleweringstelsel waar 'n onbemande lugvoertuig vanaf 'n doelgemaakte vragmotor gelanseer word, addisionele aflewering voltooi terwyl die vragmotor tussen twee kliënte beweeg, en die vragmotor onderskep nadat die addisionele aflewering voltooi is. Die verhandeling beskryf die ontwikkeling van 'n mierkolonieoptimeringsalgoritme wat gebruik word om die probleem op te los. Meer spesifiek, is 'n mierkoloniestelsel in hierdie navorsing gebruik. Aanpasbare algoritme beheerparameters is ook gebruik om 'n aanvaarbare balans tussen eksplorasië en ontginning gedurende die soekproses te verseker.

Die algoritme is getoets op standaard probleme vir aflewering lugvoertuig skedulering en vergelyk met 'n afleweringstelsel wat slegs uit tradisionele voertuie bestaan. Daar word gewys dat die vragmotor-lugvoertuig afleweringstelsel 'n beduidende positiewe invloed op die afleweringstydprestasie het.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations. To my supervisor Prof. Jacomine Grobler for her patience, dedication and for believing in me. Even though at some points the road seemed impossible, she never gave up on me.

To Prof. P. Kaminsky for offering guidance and shining a light on to the path when we were lost. I would further extend my gratitude to my wife Mbokeleng Khanye and my daughter Tshepiso Moremi for tolerating my frustration, and for their support and encouragement.

Lastly, I would like to thank Almighty God for health, wisdom and protection.

Contents

Declaration	i
Abstract	ii
Opsomming	iii
Acknowledgements	iv
Contents	v
List of Figures	viii
List of Tables	xv
List of Abbreviation	xvii
Nomenclature	xviii
1 Introduction	1
1.1 Problem statement	3
1.2 Objectives	4
1.3 Dissertation structure	5
2 Literature review	6
2.1 Last mile challenges	6
2.2 Optimisation	7
2.3 Travelling salesman problem	7
2.4 Vehicle routing problem	9
2.5 Vehicle routing problem with time windows	11
2.6 Truck and drone delivery systems	12
2.7 Summary	19

3	Solution Strategy Selection	21
3.1	Exact solution algorithms	21
3.2	Heuristics	25
3.3	Metaheuristics	26
3.4	K-means	42
3.5	Existing truck and drone optimisation strategies	42
3.6	Summary	46
4	Solving the single truck and drone scheduling problem with interception	48
4.1	Problem description	48
4.2	Mathematical formulations	51
4.3	The ACS-based scheduling algorithm for the single truck and drone scheduling problem	57
4.4	Empirical evaluation of the ACS single truck and drone scheduling algorithms	61
4.5	Summary	69
5	Solving the multiple truck and drone scheduling problem with interception	74
5.1	Problem description	74
5.2	The ACS-based scheduling algorithm for the multiple truck and drone scheduling problem	75
5.3	Empirical evaluation of the ACS multiple drone-truck scheduling algorithm	76
5.4	Summary	83
6	Solving the multiple trucks and drones scheduling problem with time window constraints	86
6.1	Problem description	86
6.2	Mathematical model	87
6.3	The ACS-based scheduling algorithm for the multiple trucks and drones scheduling problem with time windows	88
6.4	Empirical evaluation of the ACS multiple trucks and drones scheduling algorithm with time windows	89
6.5	Summary	92
7	Conclusion	98
7.1	Summary	98
7.2	Appraisal of dissertation contributions	100
7.3	Future Research Opportunities	101

<i>CONTENTS</i>	vii
List of References	103
Appendices	111
Additional drone and truck scheduling problem with interception examples	112
Additional multiple truck and drone scheduling problem with interception examples	144
Additional drones and trucks scheduling problem with interception and time window examples	172

List of Figures

1.1	Drone (Source: Amazon.com)	3
2.1	Example of a last mile delivery system	7
2.2	Typical travelling salesman tour	8
2.3	Example of routing solution for VRP	10
2.4	Parallel drone scheduling travelling salesman problem (PDSTSP) on the left and flying sidekick with a drone (FSTSP) on the right (Murray and Chu (2015)).	13
2.5	Arc-based truck-drone operations (Marinelli <i>et al.</i> (2018))	14
2.6	Multiple trucks and drones delivery system (Kitjacharoenchai <i>et al.</i> (2019))	16
2.7	Drones launched from the centroid location (Liu <i>et al.</i> (2018))	17
2.8	Example of vehicle routing problem with a drone (Wang and Sheu (2019))	17
2.9	Vehicle routing problem with a drone where drone is allowed to return to the same vertex (Schermer (2019)).	20
3.1	Different size TSPs solved on Concorde (a TSP solver) (Applegate <i>et al.</i> , 1998)	22
3.2	Common metaheuristic algorithms	27
4.1	Truck and drone system with interception points (black circles), truck route (solid lines) and drone route (dotted lines)	49
4.2	Schematic A: Launching of drone in the arc and Schematic B: Interception after truck leaves customer	50
4.3	Truck and drone subtour	52
4.4	Truck and drone assignment and scheduling method	59
4.5	ACO application process flowchart	60
4.6	The drone and truck solution for 500 nodes (uniform-5-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	65
4.7	The drone and truck solution for 500 nodes (uniform-6-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	66

5.1	A truck-drone system with interception points (black circles), truck routes (solid lines), and drone routes (dotted lines).	75
5.2	ACS algorithm process flowchart	77
5.3	500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-6-n500)	80
5.4	Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-6-n500)	81
5.5	Solutions search diversity for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-6-n500)	82
6.1	A truck and drone system with interception points (black circles), truck routes (solid lines), drone routes (dotted lines), and time window constraints in square brackets	87
6.2	ACS algorithm process flowchart for optimising the multiple trucks and drones scheduling problem with time window constraints	90
6.3	500 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time-window constraints (Uniform-6-n500)	93
6.4	Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time windows (Uniform-6-n500)	94
6.5	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time windows (Uniform-6-n500)	95
1	The drone and truck solution for 10 nodes (uniform-52-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10	113
2	The drone and truck solution for 10 nodes (uniform-53-n10) with drone speed (v_d) of 20 and truck speed v_t of 10	114
3	The drone and truck solution for 20 nodes (uniform-62-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10	115
4	The drone and truck solution for 20 nodes (uniform-63-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10	116
5	The drone and truck solution for 50 nodes (uniform-71-n50) with drone speed (v_d) of 20 and truck speed v_t of 10	117
6	The drone and truck solution for 50 nodes (uniform-73-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10	118
7	The drone and truck solution for 100 nodes (uniform-92-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10	119

LIST OF FIGURES

x

8	The drone and truck solution for 100 nodes (uniform-92-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10	120
9	The drone and truck solution for 250 nodes (uniform-2-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10	121
10	The drone and truck solution for 500 nodes (uniform-6-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10	122
11	The drone and truck solution for 10 nodes (uniform-52-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	123
12	The drone and truck solution for 10 nodes (uniform-53-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	124
13	The drone and truck solution for 20 nodes (uniform-62-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	125
14	The drone and truck solution for 20 nodes (uniform-63-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	126
15	The drone and truck solution for 50 nodes (uniform-72-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	127
16	The drone and truck solution for 50 nodes (uniform-73-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	128
17	The drone and truck solution for 100 nodes (uniform-92-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	129
18	The drone and truck solution for 100 nodes (uniform-93-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	130
19	The drone and truck solution for 250 nodes (uniform-2-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	131
20	The drone and truck solution for 500 nodes (uniform-6-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	132
21	The drone and truck solution for 10 nodes (uniform-51-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1	133
22	The drone and truck solution for 20 nodes (uniform-61-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1	134
23	The drone and truck solution for 50 nodes (uniform-72-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1	135
24	The drone and truck solution for 100 nodes (uniform-91-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1	136
25	The drone and truck solution for 250 nodes (uniform-1-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1	137
26	The drone and truck solution for 10 nodes (uniform-51-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	138
27	The drone and truck solution for 20 nodes (uniform-61-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	139

28	The drone and truck solution for 50 nodes (uniform-71-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	140
29	The drone and truck solution for 100 nodes (uniform-91-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	141
30	The drone and truck solution for 250 nodes (uniform-1-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	142
31	The drone and truck solution for 250 nodes (uniform-1-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2	143
32	50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-71-n50)	145
33	Best solutions for 50 customers receiving deliveries from either a truck only system or a multiple drone-truck delivery system (Uniform-71-n50)	146
34	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-71-n50)	147
35	50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-72-n50)	148
36	Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-72-n50)	149
37	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-72-n50)	150
38	50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-73-n50)	151
39	Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-73-n50)	152
40	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-73-n50)	153
41	100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-91-n100)	154
42	Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-91-n100)	155
43	Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-91-n100)	156
44	100 customers receiving deliveries from either a truck only system or a multiple drone-truck delivery system (Uniform-92-n100)	157
45	Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-92-n100)	158

LIST OF FIGURES

xii

46	Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-92-n100)	159
47	100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-93-n100)	160
48	Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-93-n100)	161
49	Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-93-n100)	162
50	250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-1-n250)	163
51	Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-1-n250)	164
52	Solutions search diversity for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-1-n250)	165
53	250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-2-n250)	166
54	Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-2-n250)	167
55	Solutions search diversity for 250y customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-2-n250)	168
56	500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-5-n500)	169
57	Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-5-n500)	170
58	Solutions search diversity for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-5-n500)	171
59	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time windows (Uniform-71-n50)	173
60	Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-71-n50)	174
61	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-71-n50)	175

LIST OF FIGURES

xiii

62	50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-72-n50)	176
63	Best solutions for 50 receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-72-n50)	177
64	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-72-n50)	178
65	50 customers receiving deliveries from either a truck only system or a multiple drone-truck delivery system,all with time windows (Uniform-73-n50)	179
66	Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-73-n50)	180
67	Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-73-n50)	181
68	100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-91-n100)	182
69	Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-91-n100)	183
70	Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-91-n100)	184
71	100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-92-n100)	185
72	Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-92-n100)	186
73	Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-92-n100)	187
74	100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)	188

LIST OF FIGURES

xiv

75	Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)	189
76	Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)	190
77	250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-1-n250)	191
78	Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-1-n250)	192
79	Solutions search diversity for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-1-n250)	193
80	250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)	194
81	Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)	195
82	Solutions search diversity for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-2-n250)	196
83	500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-5-n500)	197
84	Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-5-n500)	198
85	Solutions search diversity for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-5-n500)	199

List of Tables

3.1	Existing literature on truck-drone delivery systems	45
3.2	Existing literature on truck-drone delivery systems	46
4.1	TSP data sets (Bouman <i>et al.</i> (2015))	61
4.2	TSP data sets used for tuning ACS parameters (Bouman <i>et al.</i> , 2015) .	62
4.3	ACS parameters	63
4.4	Problem parameters	63
4.5	F-Race results	64
4.6	Delivery time results of the ACS-DT1 and ACS-DT2 algorithms	68
4.7	Delivery distance results of the ACS-DT1 and ACS-DT2 algorithms . .	70
4.8	ACS-DT1 and ACS-DT2 benchmarked against optimal delivery solu- tions for the two different models	71
4.9	Benchmarking delivery completion time ACS-DT2 against GCPSO and SaNSDE (Ernst, 2021)	72
4.10	Benchmarking delivery distance of ACS-DT2 against GCPSO and SaNSDE (Ernst, 2021)	73
5.1	Algorithm and problem parameters	78
5.2	Results of the VRP and VRPD system performance	84
5.3	Hypothesis tests comparing the VRP and VRPD system with regard to distance and time	85
6.1	Algorithm and problem parameters	91
6.2	Results of the VRPTW and VRPDTW system performance	96
6.3	Hypothesis tests comparing the VRPTW and VRPDTW system with regard to distance and time	97

List of Abbreviation

ACO	-	Ant colony Optimisation
ACS	-	Ant colony system
ADI	-	Adaptive insertion heuristic
CMAES	-	Covariance matrix adapting evolutionary strategy algorithm
CO	-	Combinatorial optimisation
CVRP	-	Capacitated VRP
DE	-	Differential evolution
EC	-	Evolutionary computation
FIT	-	Field Innovation Team
FSTSP	-	Flying sidekick with a drone'
GCPSO	-	Guaranteed convergence particle swarm optimisation
GLS	-	Guided local search
GRASP	-	Greedy randomised adaptive search procedure
HACO	-	Hybrid ant colony optimisation
ILS	-	Iterated local search
M-BBX	-	Modular Bento box systems
MILP	-	Mixed integer programming
mTSPD	-	Multi-travelling salesman problem with drones
NSDE	-	Neighbourhood search differential evolution
PDSTSP	-	Parallel drone scheduling travelling salesman problem'
PSO	-	Particle swarm optimisation
RCL	-	Restricted candidate list
SA	-	Simulated annealing
SaDE	-	Self-adaptive neighbourhood search differential evolution
SaNSDE	-	Self-adaptive neighbourhood search DE
TS	-	Tabu search
TSP	-	Travelling salesman problem
TSP-D	-	Travelling salesman problem with a drone
VND	-	Variable neighbourhood descent
VNS	-	Variable neighbourhood search
VRP	-	Vehicle routing problem
VRPD	-	Vehicle routing problem with a drone
VRPPD	-	Vehicle routing problem with pick-up and delivery
VRPTW	-	Vehicle routing problem with time windows

Nomenclature

$U(0, 1)$ and w	The inertia weight
α	An ACS positive constant
\bar{w}	Branching node of \mathbf{P} in the branch-and-bound algorithm
β	An ACS parameter
δ	The time it takes a drone to offload the parcel
$\Delta\tau_{ij}^k(t)$	The ACS pheromone deposited on each edge at iteration t by ant k
λ	The regularisation parameter in GLS
\mathbf{A}	An arc set
\mathbf{E}	An edge set
\mathbf{F}_0 and \mathbf{F}_1	Ordered pairs in \mathbf{S}
\mathbf{N}_t	The neighbourhood structure at iteration t
\mathbf{N}_i^k	The set of feasible nodes connected to node i with respect to ant k
\mathbf{P}	The active set in the branch-and-bound algorithm
\mathbf{Q}	A subset of \mathbf{E}
\mathbf{S}	The family of ordered pairs of disjoint edge sets
\mathbf{V}	The vertex set
\mathbf{v}_t^{ik}	The velocity of the truck at time t while travelling between nodes i and k
$\rho(t)$	The PSO time-dependent scaling factor at time t
ρ_1	The ACS pheromone evaporation rate

τI^j	Time at interception point
τ^j	The time the drone arrives at customer j
τ_0	The positive constant produced by the nearest neighbour heuristic (part of the ACS)
τ_{ij}	The pheromone concentration of the arc between node i and j
c_x^i and c_y^i	The x and y coordinates of the node i
c_1 and c_2	The cognitive and social acceleration constants in the PSO algorithm
C_k	The capacity of vehicle k
c_{ij}	the weights of the edge from node i to j
D	Diversity of the ant colony over time
$d(\mathbf{c}^i, \mathbf{c}^j)$	Distance travelled between nodes i and j
d_j	The customer demand at node j
D_k	The vehicle k 's maximum allowed distance
e_j	The earliest delivery time at node j
$f(S^*)$	The fitness function values of population S^*
$f(S^*)$	The new population fitness function values in an EA
$h((x))$	New objective function after penalty factor addition in GLS
i^*	Intermediate location where the truck stops and launches a drone
$I_i(x)$	The indicator function to show i is present in x in GLS
L	The length of the tour
l_j	The delivery deadline at node j
M	A very large value
m	The number of vehicles
n	The number of nodes
n_s	Number of individual members in the population

NOMENCLATURE

xx

n_w	Total number of offspring in a branched node w in the branch-and-bound algorithm
$P(t)$	The probability of accepting the solution at iteration t in the SA algorithm
$P(t)_{ij}^k$	The probability of selecting the next node j if an ant k is currently located at i
p^j	The position of the truck at the start of the drone delivery at customer j
P_i	Penalty parameter i in GLS
$P_{id}(t)$	The personal best (pbest) position in dimension d of the i^{th} particle at time t
pI^j	Position at interception point
q	The number of penalty features in GLS
$rand()$	Uniform random distribution
S^{**}	A mutated population
s_i	The time when the truck arrives at customer i for delivery
s_j^k	The vehicle k arrival time at node j
SSD	The sum of square distances
$T(t)$	The temperature that is updated per iteration t in the SA algorithm
t_{ij}	The travelling time from i to j
t_{max}	The maximum number of iterations
TI^j	The time it takes a drone to travel from j to intercept with the truck
u^*	Intermediate location where the truck stops to retrieve a drone
v_d	The speed of the drone
$v_{id}(t)$	The velocity of particle i in dimension d at time t
v_t	The speed of the truck
W	The waiting time
w	The time it takes a truck to offload and send off a drone

NOMENCLATURE

xxi

$x_d^*(t)$ The global best (gbest) position in dimension d at time t

$x_{id}(t)$ The position of particle i in dimension d at time t

Z The overall measure of performance that a model seeks to optimise

Chapter 1

Introduction

Online shopping has proliferated over the past few years. Currently, almost everyone has access to online shopping through their smartphone or home internet. Consumers can buy products from anywhere in the world using their smart devices and these will be delivered to their doorstep. Online shopping has placed last mile deliveries under enormous pressure due to more parcels being delivered more frequently than before. Companies have increased the number of their last mile vehicles to cope with the challenges related to last mile deliveries. However, the addition of more vehicles causes issues with cost as well as environmental problems. Companies are now confronted with the problem of making the last mile delivery system more efficient as this can give them a competitive advantage (Lim *et al.*, 2018).

A large amount of research has already been done into optimising the last mile leg. Popular routing problems often investigated include the travelling salesman problem (TSP), which was introduced in 1934, by Hassler Whitney in a seminar talk at Princeton University (Flood, 1956), vehicle routing problem (VRP) (Dantzig and Ramser, 1959), and vehicle routing problem with time windows (VRPTW) (Baker and Schaffer, 1986; Solomon, 1987).

Recently, companies and institutions have been searching for more innovative solutions that would improve the last mile delivery system. Some of the areas being researched are (Ranieri *et al.*, 2018):

1. Inclusion of ‘innovative vehicles’ on the last mile leg;
2. Proximity stations;
3. Collaborative and cooperative logistics;
4. Optimisation of transport management and routing; and
5. Innovations in public policies and infrastructure.

This research focuses only on the inclusion of innovative vehicles on the last mile leg. There are numerous types of innovative vehicles that are currently being researched. Examples of innovative vehicles include:

1. Modular Bento box systems (M-BBX) (Dell'Amico and Hadjidimitriou, 2012). M-BBX delivers parcels to bento boxes for customers to pick up.
2. Self-driving parcels (Slabinac, 2015). A self-driving parcel carries parcels to customers.
3. Drones (Slabinac, 2015). Drones can assist a truck to make deliveries or deliver goods straight from the depot to the customer.

Presently, incorporating drones in the last mile delivery system is the most popular solution being investigated. Drones, also known as unmanned aerial vehicles (UAVs), are remotely controlled aircraft. There are different types of drones, namely fixed-wing and multirotor (Brooke-Holland, 2013). Fixed-wing drones have static fixed wings to generate lift similar to aircraft and are mainly used in the aviation industry (Brooke-Holland, 2013). Multirotor drones are preferred for last mile delivery systems.

Furthermore, drones are being investigated for completing humanitarian assistance delivery missions. They can be used to deliver medication (Rabta *et al.*, 2018; Thiels *et al.*, 2015), vaccines (Haidari *et al.*, 2016), or disaster relief items (Scott and Scott, 2017) to areas with poor infrastructure or areas which have experienced a natural disaster.

Commercially, Amazon Prime Air is developing autonomous drones as shown in Figure 1.1. The drone will weigh 25 kg and will be able to carry a payload of 2.26 kg at a speed of 80 kph and it should deliver packages to customers safely within 30 minutes flying time or less (Spires, 2019). Meanwhile, Domino's Pizza and Flirtey drone delivery services have already conducted trials with autonomous drones controlled using GPS to deliver their first order (Ryan, 2016). Additionally, Flirtey, John Hopkins University School of Medicine, and the non-profit Field Innovation Team (FIT) were involved in a demo mission of transporting medicine, medical samples and water using drones from an onshore medical relief camp and test facility (Knight, 2016).

Drones provide a better solution regarding emissions compared with trucks. They utilise batteries and not fuel, but careful consideration must be taken when comparing the carbon footprint of drones and trucks. Emissions must be compared on a broader scale considering total carbon footprint contribution from the batteries' manufacturing process and drone operation and comparing that figure with total truck fuel and fuel manufacturing emissions. Vehicle travel distance, speed, and weight can directly relate to emissions when energy is provided by fuel



Figure 1.1: Drone (Source: Amazon.com)

(Eun *et al.*, 2019). It is thought that drones can reduce operational carbon dioxide emissions compared with other ground vehicles, even when working under strict regulations (Elsayed and Mohamed, 2020).

Several barriers still need to be addressed when incorporating drones into a truck delivery system. The current governmental regulations are making drone delivery implementation impossible. Most countries find drones invasive and disruptive in airspace territory (Raj and Sah, 2019; Elsayed and Mohamed, 2020; Perera *et al.*, 2020).

However, in the meantime, researchers have started developing routing and scheduling algorithms for drone delivery systems. One of these studies is by Agatz *et al.* (2018) which involved a truck and drone waiting for each other at a customer location on their way to completing deliveries. This dissertation expands on this work and allows a drone to intercept a truck in motion. This interception minimises the waiting time and improves the total delivery time.

The rest of this chapter describes the problem statement in Section 1.1, research objectives in Section 1.2, and the dissertation structure in Section 1.3.

1.1 Problem statement

The aim of this dissertation is to investigate a truck and drone delivery system where a drone can be launched from a delivery truck to perform additional deliveries and intercept the truck again at any point on the arc between the truck's departing and delivery nodes. The vehicle interception allows the drone to fly directly from its delivery node to the truck instead of flying the longer distance to the next truck delivery node, decreasing the distance travelled by the drone versus

a system which does not allow for interception. This optimisation problem involves determining which deliveries should be conducted by truck and which by drone and the ideal sequence of deliveries to minimise total travelling time. The problem is considered NP-hard, thus the use of metaheuristics is appropriate. Additional complexities such as the simultaneous scheduling of multiple truck and drone delivery systems and the inclusion of time windows also need to be addressed.

1.2 Objectives

The following objectives are pursued in this dissertation:

- I To provide a rationale for the inclusion of a drone in a truck delivery system through the review of the logistics of last mile challenges and innovative solutions applied to optimise delivery systems.
- II To review the literature on the application and limitations of drones as a means of goods transportation.
- III To review existing literature on truck and drone delivery systems.
- IV To develop and test a metaheuristic-based truck and drone scheduling algorithm for solving the truck and drone scheduling problem with interception.
- V To tune the truck and drone scheduling algorithm control parameters for best performance.
- VI To benchmark the developed truck and drone scheduling algorithm against exact solutions obtained by CPLEX[®] in Knoetze (2021).
- VII To benchmark the developed truck and drone scheduling algorithm against other metaheuristics.
- VIII To develop and test a metaheuristic-based truck and drone scheduling algorithm for solving the **multiple** truck and drone scheduling problem with interception.
- IX To develop and test a metaheuristic-based truck and drone scheduling algorithm for solving the **multiple** truck and drone scheduling problem with interception and **time windows**.
- X To compare the performance of the above-mentioned truck and drone delivery systems to a truck-only delivery system on the same benchmarked problems.

- XI To solve truck and drone scheduling problem instances of significantly larger size than currently reported in literature.
- XII To recommend appropriate future work relevant to the contributions of this dissertation.

1.3 Dissertation structure

Following this introductory chapter, the remainder of this dissertation is composed of six more chapters, three appendices, and a bibliography. The next chapter, Chapter 2, provides a review of the relevant literature in drone scheduling. More specifically, in support of Objectives I to III, existing models solving truck and drone delivery systems are investigated.

Chapter 3 summarises the algorithms commonly used to solve routing problems. First, the general concept of optimisation is introduced and then exact algorithms, heuristics, and metaheuristics are discussed. A population-based metaheuristic algorithm is chosen to solve the truck-drone problem. The ant colony system (ACS) was selected because it is popularly used to solve graph-based problems.

Chapter 4 serves the purpose of fulfilling Objectives III to VII, X, and XI. The truck and drone scheduling problem with interception is introduced. An ACS-based drone-truck scheduling algorithm is proposed and extensive algorithm control parameter tuning and benchmarking are performed against optimal solutions and other metaheuristic algorithms.

Chapter 5 serves the purpose of fulfilling Objectives VIII, X, and XI. The **multiple** truck and drone scheduling problem with interception is introduced. A cluster first-route second algorithm utilising k-means and ACS is proposed for this problem variation and compared against a truck-only delivery system.

Chapter 6 serves the purpose of fulfilling Objectives IX, X, and XI. The **multiple** truck and drone scheduling problem with interception and **time windows** is introduced. Another ACS-based drone-truck scheduling algorithm is proposed for this problem variation and compared against a truck-only delivery system.

Finally, Chapter 7 concludes the dissertation. More specifically, Chapter 7 provides a summary and an appraisal of the contributions of the dissertation. In fulfilment of Objective XII, appropriate future work is also recommended.

Chapter 2

Literature review

This chapter gives essential background to the problem this research addresses. First, last-mile challenges are discussed in Section 2.1. A generic optimisation model is explained, and a formulation is given in Section 2.2. Thereafter, the travelling salesperson problem is introduced in Section 2.3. Section 2.4 introduces a generic vehicle routing problem and Section 2.5 expands the vehicle routing problem with the inclusion of time windows. Existing work with regard to drone scheduling is presented in Section 2.6, before Section 2.7 concludes the chapter.

2.1 Last mile challenges

A logistic system is the movement of goods from the manufacturer to customers. It involves the delivery of goods from a hub to the customers. The last mile is the last leg of the logistic system. Figure 2.1 illustrates a simple example of this scenario. It shows goods moving from a manufacturer (denoted by a star) through the distribution hub (denoted by a square) to the customers (denoted by circles). Previous research has shown that finding the best solution to make these deliveries is very complex. There are various factors to be considered when optimising this kind of problem and some include customers' density (dispersed or clustered), the number of vehicles in the delivery system, the capacity of the vehicles, and time windows (Bräysy and Gendreau, 2005*b,a*; Bektas, 2006; Eksioglu *et al.*, 2009).

Adding to the last mile complexity is the growth of online shopping. Customers, especially in urban areas, have access to the internet through smartphones and wireless networks. They find it easier to buy goods globally, from anywhere, and have their purchases delivered directly to them. Online shopping has additionally pressured the last mile delivery system since more goods must be delivered more frequently and efficiently. Businesses view an optimised logistic system as a competitive advantage and are now searching for efficient delivery methods.

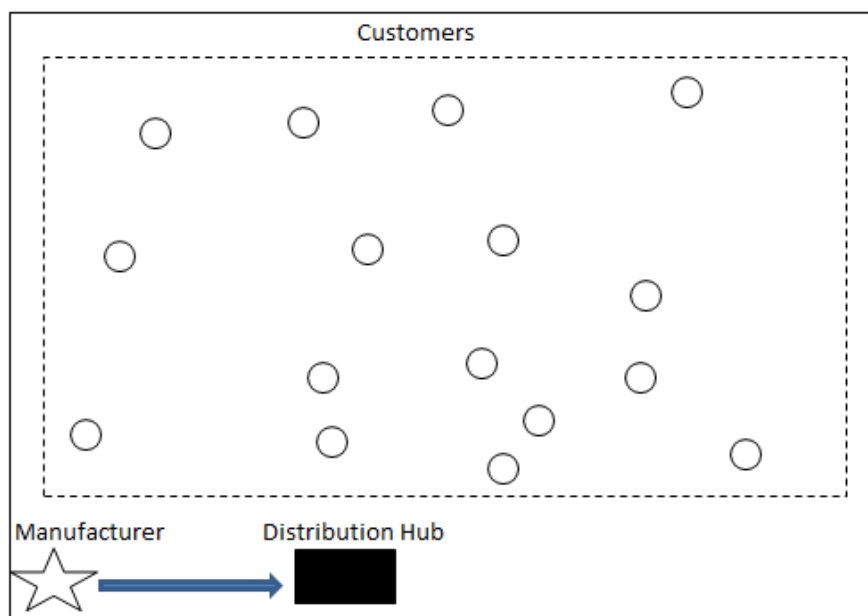


Figure 2.1: Example of a last mile delivery system

2.2 Optimisation

The Oxford dictionary defines *optimisation* as “the action of making the best or most effective use of a situation or resource”. The mathematical definition of optimisation is searching for the best solution from a combination of parameters with given constraints (Lieberman and Hillier, 2005). Optimisation models comprise decision variables, variable domains, goals, objective functions, and constraints. The optimisation model seeks to optimise its objective function by either maximising or minimising it depending on the objective goal.

2.3 Travelling salesman problem

The *travelling salesman problem* (TSP) is a historical combinatorial optimisation problem of a salesman seeking to find the shortest route to visit a given set of cities, with known intermediate distances, exactly once. It has various applications such as logistic scheduling, job sequencing, and drilling sequencing. Figure 2.2 shows an example of a TSP tour. The salesman travels from his home node depicted by a black block, travels through a path depicted by arrows, and visits cities denoted by nodes with numbered circles. The salesman starts and ends at the home node and the other nodes are visited only once.

The distance, $d(\mathbf{c}^i, \mathbf{c}^j)$, between nodes i and j is calculated using a Euclidean

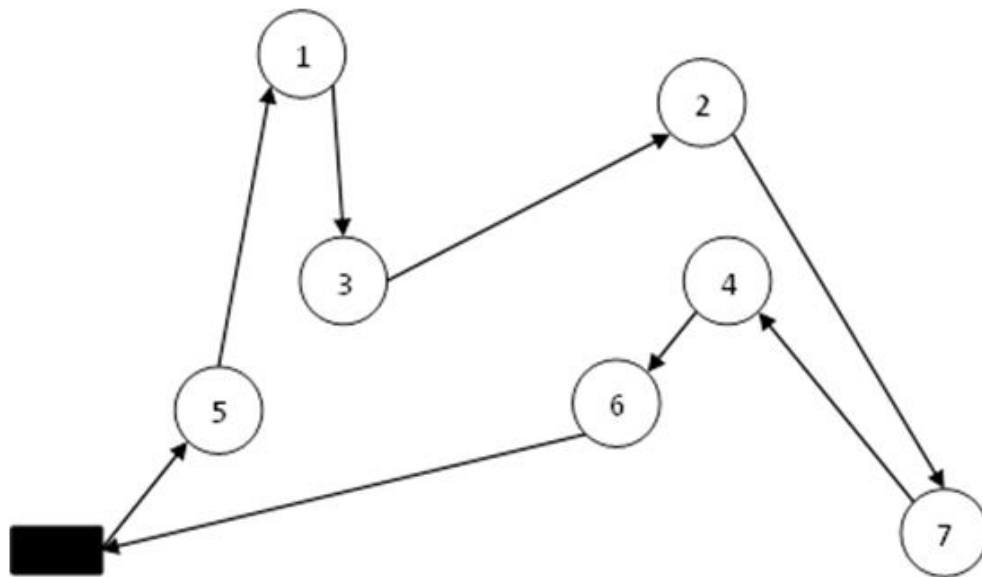


Figure 2.2: Typical travelling salesman tour

distance formulation given in (2.1)

$$d(\mathbf{c}^i, \mathbf{c}^j) = \sqrt{(c_x^i - c_x^j)^2 + (c_y^i - c_y^j)^2} \quad (2.1)$$

where c_x^i and c_y^i are the x and y coordinates of the node i .

TSPs are classified into two types called symmetric and asymmetric. In symmetric TSPs, a distance from node i to node j is equivalent to the distance from node j to node i , that is $d(\mathbf{c}^i, \mathbf{c}^j) = d(\mathbf{c}^j, \mathbf{c}^i)$. An asymmetric TSP distance from node i to node j is generally not equivalent to the distance from node j to node i , $d(\mathbf{c}^i, \mathbf{c}^j) \neq d(\mathbf{c}^j, \mathbf{c}^i)$. Symmetric TSPs are defined on an undirected graph as $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and asymmetric TSPs are defined on a directed graph as $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ where $\mathbf{V} = \{1, \dots, n\}$ is the vertex set, $\mathbf{E} = \{(i, j) : i, j \in \mathbf{V}, i < j\}$ is the edge set, $\mathbf{A} = \{(i, j) : i, j \in \mathbf{V}, i \neq j\}$ is the arc set, and n is the number of nodes.

Dantzig *et al.* (1954) developed an asymmetric integer linear programming formulation for TSP shown in Equations (2.2) to (2.6). In Equation (2.2), Z denotes the minimum distance the travelling salesman must travel to visit all the cities, c_{ij} is the distance of the arc from node i to j , x_{ij} is one if the arc from node i to j is included in the route and zero otherwise. Equations (2.3) and (2.4) ensure that once the travelling salesman has visited a node, he must leave it, and also that each node must be visited at least once. Equations (2.5) ensure that sub tours are eliminated during solution construction with $|\mathbf{Q}|$ being the number

of nodes and \mathbf{Q} a subset of \mathbf{V} . Lastly, Equations (2.6) pose binary conditions on variables.

$$\text{Minimise } Z = \sum_{i=1}^n \sum_{j \neq i, j=1}^n c_{ij} x_{ij} \quad (2.2)$$

Subject to:

$$\sum_{i \in \mathbf{V} \setminus \{j\}} x_{ij} = 1 \quad \forall j \in \mathbf{V} \quad (2.3)$$

$$\sum_{j \in \mathbf{V} \setminus \{i\}} x_{ij} = 1 \quad \forall i \in \mathbf{V} \quad (2.4)$$

$$\sum_{i \in \mathbf{Q}} \sum_{j \in \mathbf{Q}, j \neq i} x_{ij} \leq |\mathbf{Q}| - 1 \quad \forall \mathbf{Q} \subset \mathbf{V}, |\mathbf{Q}| \geq 2 \quad (2.5)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j, i \neq j \in \mathbf{V} \quad (2.6)$$

Several distinct methods are available to solve the (a)symmetric TSP. These are classified into exact methods, where an optimal solution is found, and approximation methods, where a ‘good enough’ solution is found. Details of these algorithms are discussed in Chapter 3.

2.4 Vehicle routing problem

The vehicle routing problem (VRP) is a combinatorial optimisation problem that seeks to find the optimal delivery or collection routes from a depot(s) to geographically dispersed customers, subject to constraints. Dantzig and Ramser (1959) introduced the VRP called the “Truck Dispatching Problem”. They modelled a delivery system with a homogenous fleet of trucks servicing customers from the central depot. The goal of the model was to minimise distance.

Exact methods are used to solve the VRP optimally, but VRPs with larger numbers of nodes require a significant computational time to solve optimally. In such cases, heuristic methods are applied to find a near optimal (good enough) solution in reasonable time. One of the most widely known heuristics applied to solve large VRPs is the Clark and Wright savings algorithm (Clarke and Wright, 1964).

Over time, constraints were added to the VRP which made it more complex and resulted in various VRP variants; for example, the vehicle routing problem with time windows (VRPTW) (Desrochers *et al.*, 1992), vehicle routing problem with pick-up and delivery (VRPPD) (Min, 1989), and the multi-depot vehicle routing problem (Wren and Holliday, 1972).

The traditional VRP is known as the capacitated VRP (CVRP). It has vehicles with homogenous capacities (C) servicing geographically dispersed customers from a central depot. Figure 2.3 depicts an example of the VRP. The black square symbol represents the depot, the circle symbols are customers geographically dispersed in the x-y plane, the dashed line is the route of one vehicle, and the solid line is the route of another vehicle. The vehicle must visit each customer once, and the deliveries of each route must not exceed the vehicle's capacity. The goal of the VRP is to determine the minimum distance or time to service all the customers. When the capacities of the vehicles are varied, the problem becomes a heterogeneous fleet VRP.

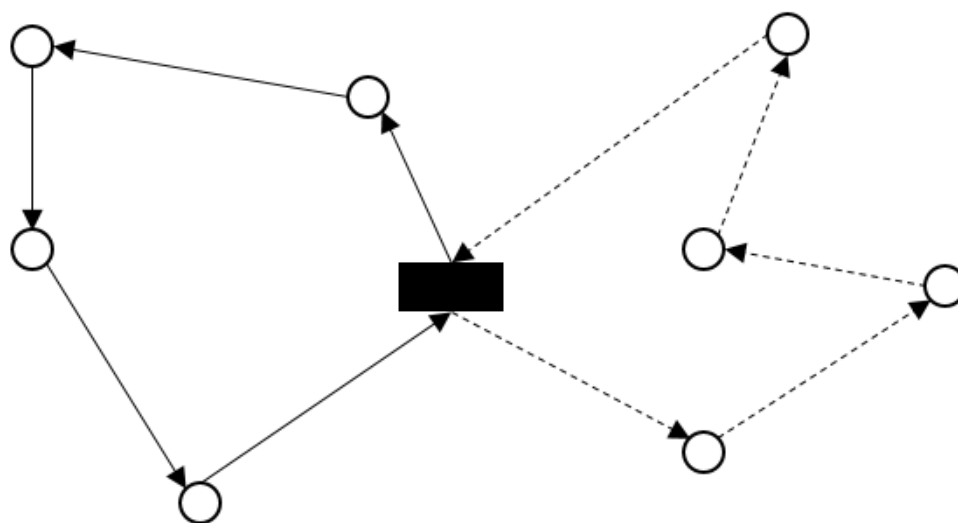


Figure 2.3: Example of routing solution for VRP

The VRP is defined on a graph as $\mathbf{G} = (\mathbf{V}, \mathbf{A})$ where $\mathbf{V} = \{0, \dots, n\}$ and node 0 corresponds to the depot and each geographically located customer $j \in \mathbf{V} \setminus \{0\}$ has a known demand, d_j . $\mathbf{A} = \{(i, j) : i, j \in \mathbf{V}\}$ is a set of edges where $i \neq j$. Each edge $(i, j) \in \mathbf{A}$ has an associated nonnegative cost, c_{ij} , which is the cost associated with travelling from node i to node j . The asymmetric VRP has a set of directed arcs, \mathbf{A} , and the symmetric VRP has a set of undirected arcs denoted by \mathbf{E} .

Given a set of customers $\mathbf{S} \subseteq \mathbf{V}$, the total demand of the set is denoted by $d(\mathbf{S}) = \sum_{j \in \mathbf{S}} d_j$, and each of the K vehicles has a capacity, C . The variable x_{ij} ($i \neq j$) is defined as a binary variable and is equal to 1 if in the solution edge (i, j) is serviced by a vehicle otherwise x_{ij} is equal to 0.

Various formulations have been developed over the years. For illustration purposes a two-index vehicle flow formulation of an asymmetrical VRP with the following assumptions is described in Equations (2.7) to Equation (2.13) (Laporte, 1992):

1. Each vehicle route starts and ends at the depot.
2. Each customer $j \in \mathbf{V} \setminus \{0\}$ is visited exactly once.
3. The sum of the customers' demand visited by each vehicle does not exceed the vehicle capacity, C .

$$\text{Minimise } Z = \sum_{i < j}^n c_{ij} x_{ij} \quad (2.7)$$

Subject to:

$$\sum_{i \in \mathbf{V} \setminus \{j\}} x_{ij} = 1 \quad \forall j \in \mathbf{V} \setminus \{0\} \quad (2.8)$$

$$\sum_{j \in \mathbf{V} \setminus \{i\}} x_{ij} = 1 \quad \forall i \in \mathbf{V} \setminus \{0\} \quad (2.9)$$

$$\sum_{i \in \mathbf{V} \setminus \{0\}} x_{i0} = K \quad (2.10)$$

$$\sum_{j \in \mathbf{V} \setminus \{0\}} x_{0j} = K \quad (2.11)$$

$$\sum_{i \notin \mathbf{S}} \sum_{j \in \mathbf{S}} x_{ij} \geq r(\mathbf{S}) \quad \forall \mathbf{S} \subseteq \mathbf{V} \setminus \{0\}, \mathbf{S} \neq \emptyset \quad (2.12)$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in \mathbf{V} \quad (2.13)$$

The objective function (2.7) minimises the distance travelled (or travel time) by each vehicle. Constraints (2.8) and (2.9) are *indegree* and *outdegree* constraints that ensure that one arc enters and leaves each vertex. Constraints (2.10) and (2.11) are the degree requirements for the depot vertex. Constraints (2.12) are subtour elimination constraints with $r(\mathbf{S})$ the lower bound of the number of vehicles required to visit all vertices of \mathbf{S} .

2.5 Vehicle routing problem with time windows

A vehicle routing problem with time windows (VRPTW) is a vehicle routing problem with the added complexity of time windows. The customers in this kind of

delivery system require deliveries to occur within a certain time interval. Each customer j will stipulate the earliest delivery e_j and delivery deadline l_j . The time window can either be classified as hard or soft. The hard window does not allow deliveries to occur outside of time intervals, while the soft window allows for such deliveries. In a hard system, the customer, j can only be serviced at interval $[e_j, l_j]$. If the vehicle k arrives at customer j at s_j^k , before the earliest delivery time, the vehicle will wait for $\max(e_j - s_j^k, 0)$. However in the case where the vehicle arrives at $s_j^k > l_j$, the customer will not be served.

The time window constraints (2.14) and (2.15), adopted from Desrochers *et al.* (1992), can be added to the three-index VRP formulation to give a VRPTW formulation. Constraints (2.15) ensure that whenever a vehicle, k , travels from i to j it will service j after $s_i^k + t_{ij}$, where t_{ij} is the travelling time between i and j , s_i^k is the arrival time of truck k at node i and x_{ijk} is 1 if truck k travels from node i to node j and 0 otherwise. Constraints (2.15) ensure time windows are observed. M is a very large value.

$$s_i^k + t_{ij} - M(1 - x_{ijk}) \leq s_j^k \quad \forall k \in \{1, \dots, m\}, i, j \in \{1, \dots, n\} \quad (2.14)$$

$$e_i \leq s_i^k \leq l_i \quad \forall k \in \{1, \dots, m\}, i \in \{1, \dots, n\} \quad (2.15)$$

2.6 Truck and drone delivery systems

Technology has driven businesses to search for new solutions to improve logistics efficiencies. One solution considered is including a drone in the delivery system. The truck and drone are integrated to complete parcel deliveries. The drone is launched off of a truck to make deliveries and meet up with the truck at the following location. This development has resulted in a new field of research in drone TSPs. Recent survey papers have been compiled by Chung *et al.* (2020), Macrina *et al.* (2020), and Poikonen and Campbell (2021).

One of the first articles was by Murray and Chu (2015). They modelled a truck and drone travelling together to make deliveries. The model was called a ‘flying sidekick with a drone’ (FSTSP). The truck and drone departed from the depot either together or separately to make deliveries. A drone flew to the customer’s location while the truck travelled to other locations. The two vehicles met up at the next customer location. The vehicle that arrived first waited for the other at the customer’s location. Once the truck arrived at this location, the drone was retrieved, loaded and sent off to deliver other parcels and then the truck continued to make deliveries. Figure 2.4 shows an example of the FSTSP. The green circle around the depot denotes the drone’s flying range from the depot. A drone could fulfil customers’ orders within this range. In the first case, a drone could fly from the depot to customers within this green circle or fly back to the depot. The

second option is that a drone could travel with a truck to make deliveries. Dashed arrows denote a drone route, and solid arrows denote the truck route. The green square nodes denote the eligible customers to receive deliveries by either a truck or drone, while the red circle nodes can receive deliveries only by a truck.

Murray and Chu (2015) called the model that uses a drone to make deliveries to customers within the drone range while a truck is also making deliveries the ‘parallel drone scheduling travelling salesman problem’ (PDSTSP). Figure 2.4 shows an example of the PDSTSP model.

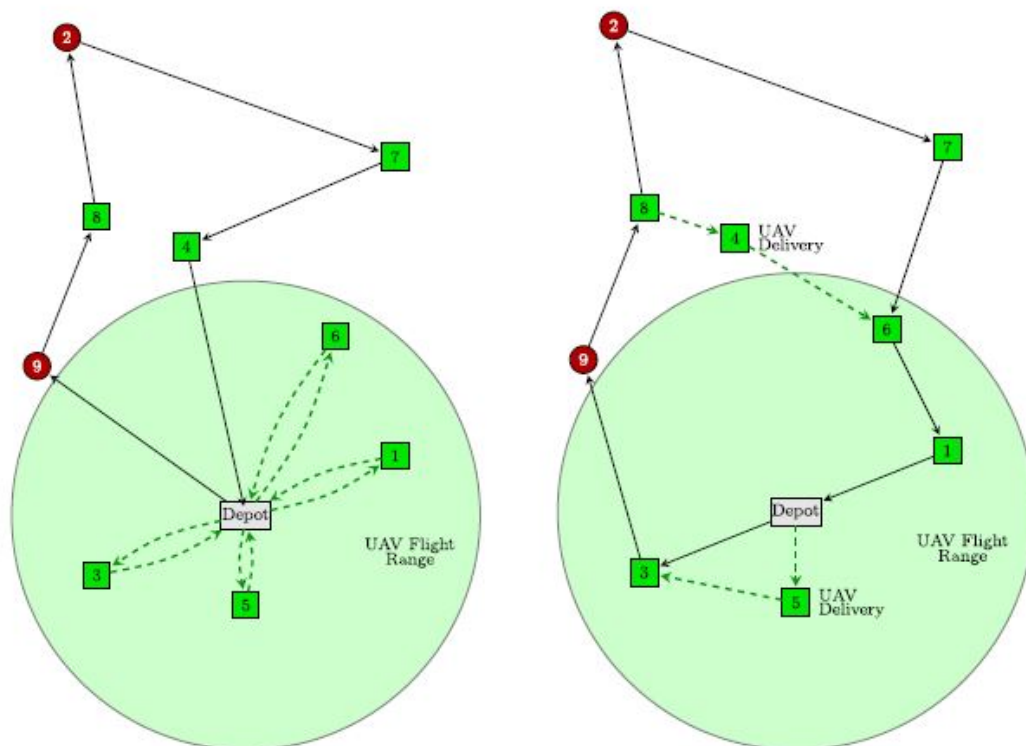


Figure 2.4: Parallel drone scheduling travelling salesman problem (PDSTSP) on the left and flying sidekick with a drone (FSTSP) on the right (Murray and Chu (2015)).

Agatz *et al.* (2018) adopted the FSTSP model and called it the travelling salesman problem with a drone (TSP-D). In this model, the truck also travelled with a drone to make deliveries. A drone followed the truck route network. A drone was launched off of a truck to the customer’s location to deliver a parcel. It later met up with a truck at the next customer’s location. After meeting up, the parcel and batteries were loaded on the drone and then it was sent off to the next

customer's location. Agatz *et al.* (2018) further showed the theoretical benefits of incorporating a drone with a truck to make deliveries. However, Ha *et al.* (2018) developed a novel approach. They minimised the operation cost of TSP-D. Their model considered the cost per unit distance for both vehicles and the waiting cost.

In both the FSTSP and TSP-D models, a truck and drone met up at the customer's location. Whichever vehicle arrived first at the meeting location waited for the other. This scenario resulted in waiting time and higher travelling costs. To reduce travelling cost, increase drone travelling time and endurance, Marinelli *et al.* (2018) introduced an intermediate drone launching point and an intermediate truck-drone meet-up point (see Figure 2.5). The truck left a depot or customer's location i to go to customer b . A solid line shows the truck route, and a dotted line shows the drone route. At the intermediate point i^* , the truck stops and launches a drone, which flies to the customer's location j to make deliveries. In the meantime, the truck travels towards customers b and u . At an intermediate location u^* , the truck waits for the drone to meet up with it. The total drone travel time between the launch and intermediate point had to be less than or equal to the drone endurance time. Carlsson and Song (2018) used the same approach as Marinelli *et al.* (2018) where a truck launched a drone at an intermediate location and rejoined it at the following intermediate location.

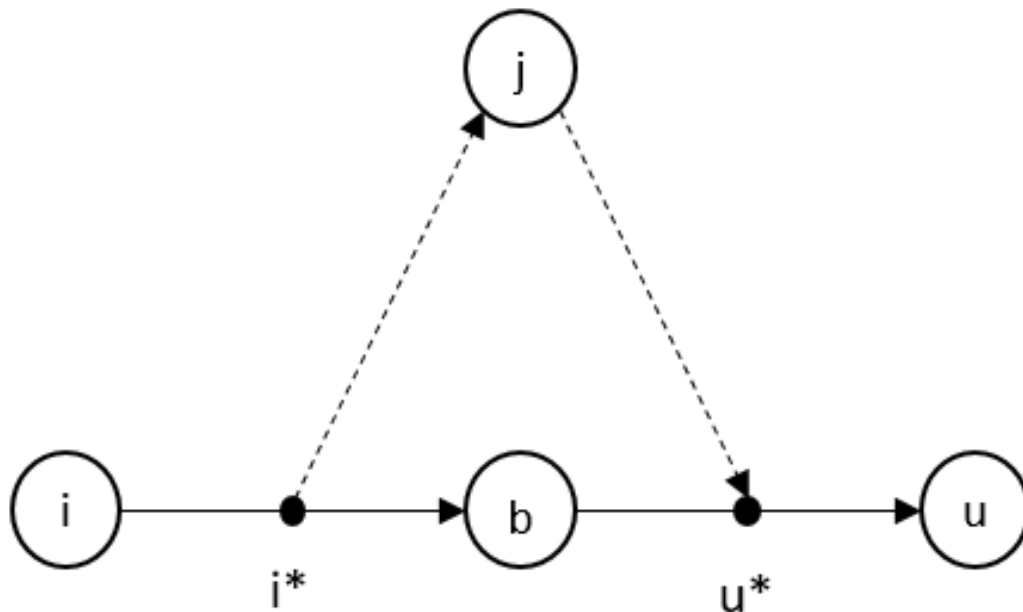


Figure 2.5: Arc-based truck-drone operations (Marinelli *et al.* (2018))

Wang *et al.* (2017) studied a vehicle routing problem with a drone (VRPD) using multiple trucks and drones to deliver parcels to customers. A customer demanded one parcel at a time and could only be visited once. The trucks and drones started the journey at a depot and ended the journey at the same depot. A drone could either be launched from a depot or a truck. A truck and drone were only allowed to meet at the customer's location. The same truck had to collect the drone that it had launched. Whichever vehicle arrived first at the customer's location waited for the other.

Kitjacharoenchai *et al.* (2019) also expanded Murray and Chu (2015)'s model with the addition of trucks and drones in the system. This model was called the multi-travelling salesman problem with drones (mTSPD). Trucks and drones travelled together or separately to make deliveries. Once a drone had made deliveries, it could fly to the customer's location or fly back to the depot. Drones were not assigned to a specific truck. A drone that flew to the retrieval node (customer's location) was collected by any truck. Drones were not allowed to land on a moving truck. Figure 2.6 shows an example of the mTSPD, where two trucks (one of which carries two drones), leave a depot denoted by a square. The trucks follow the paths denoted by solid arrows to make deliveries. When the first truck reaches the first customer (denoted by a circle), the first drone is sent to the customer's location. After launching a drone, a truck continues with its deliveries. Once the drone has made its deliveries, it flies to the next customer's location and follows the dashed arrows. A second truck picks up the drone on its way to deliveries and travels with it while making deliveries. The first truck launches the second drone before it returns to the depot. The drone flies to make deliveries, and once it has completed the deliveries, it flies to the following customer location, where the second truck picks it up. Once the second truck has picked up the second drone, it returns to the depot. Bakir and Tiniç (2020) further added capacity to Kitjacharoenchai *et al.* (2019)'s model resulting in a VRP with a flexible drone. Upon investigation, they concluded that including drone flexibility in VRPD reduced the total travelling time by a considerable amount.

The PDSTSP model allowed drones to travel from the depot to the customers within the drone range and fly back to the depot. Ham (2018) added to the mTSPD model the possibility of using drones to pick up and drop off parcels. The drone could leave the depot to make deliveries and collect items before it returned to the depot, or it could drop off a parcel from the truck and collect a parcel from the customer on its way back to the truck. Ham (2018) added further constraints to the mTSPD. He added time windows, number of customer visits, and pick-ups to the model. The model created by Dorling *et al.* (2016) had one truck and a drone performing multiple deliveries. The model considered the drone multi-rotor energy consumption and estimated the energy consumption of the parcel it carried.

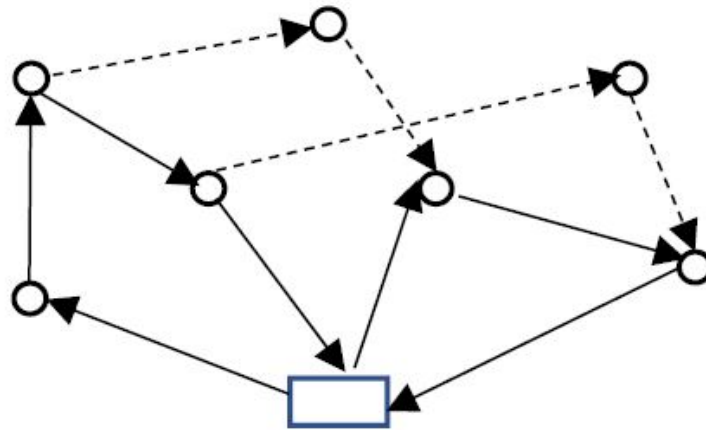


Figure 2.6: Multiple trucks and drones delivery system (Kitjacharoenchai *et al.* (2019))

Liu *et al.* (2018) followed the same approach as Ferrandez *et al.* (2016). This model used a truck and multiple drones to make deliveries to customers. It searched for the customers that are clustered together and found centroids of the customers. Centroids were used as hubs which a truck had to travel to. The best path to visit the hubs was determined. Once the truck had reached the hub, multiple drones were launched to deliver parcels, and the drones returned to the truck on completion of their deliveries. Figure 2.7 shows an example of this model.

Wang and Sheu (2019) added drone docking hubs to the trucks and drones delivery system. The docking hub allowed drones to land safely during deliveries. Figure 2.8 shows a VRPD model with hubs where a shaded square denotes a depot, a circular shape denotes a customer, triangular shapes denote hubs, truck routes are solid arrows and drone routes are dashed arrows. At the start, two trucks leave the depot, with one of them (Truck 2), carrying a drone. The other drone flies straight from the depot to the customer's location to make deliveries. The second drone is launched from the truck to the customer's location. After the first drone has made its deliveries, it flies to the docking hub and is picked up by the second truck. The drone that was launched from the second truck makes two deliveries before flying to the docking hub. Once at the hub, it waits for the first truck to arrive. Once the truck arrives, it will load the batteries and parcels onto the drone then it is sent to another customer's location. After completion of deliveries, all vehicles return to the depot either together or individually.

Raj and Murray (2020) also changed the FSTSP model developed by Murray

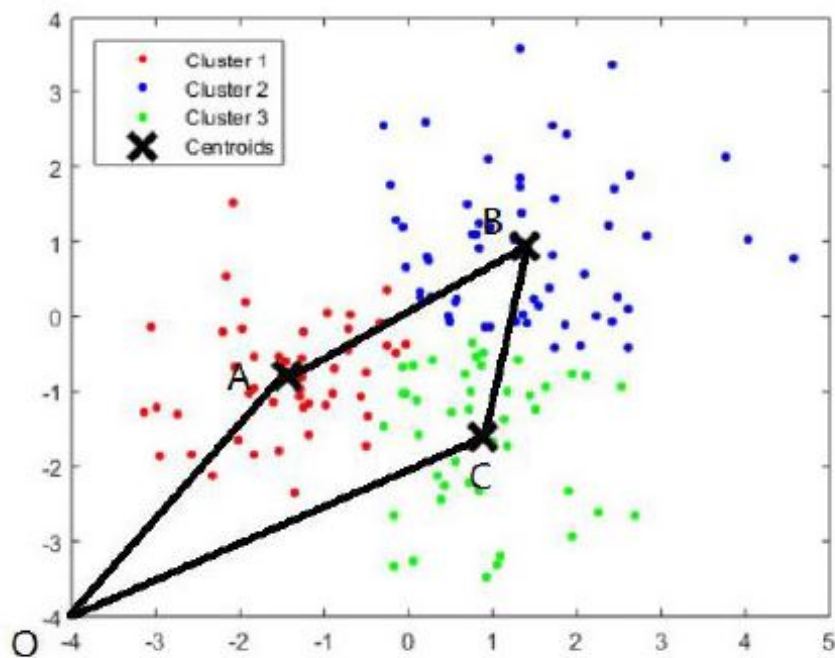


Figure 2.7: Drones launched from the centroid location (Liu *et al.* (2018))

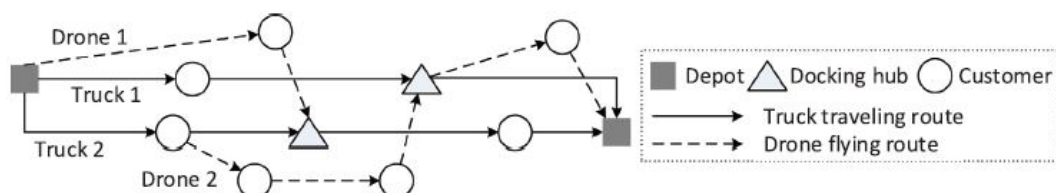


Figure 2.8: Example of vehicle routing problem with a drone (Wang and Sheu (2019))

and Chu (2015) to accommodate multiple drones. The truck travelled with multiple drones to deliver parcels. Once the truck was at the customer location, it launched drones to make deliveries. Drones were retrieved again at the customer location. Drones had limited carrying capacity, thus not all customers were eligible for drone deliveries. The drones had take-off, cruising, and landing speeds. The model assumed that the drones could not return to the same location they were launched from. The truck could continue to serve customers between the drone launch and retrieval point. Delivery of the trucks was scheduled, along with launching and retrieving the drones. Different drones also had different endurance. The model's objective was to minimise the makespan (time required to deliver all

parcels and return to the depot). The problem was tested with 100 customers. Up to 4 drones were used to solve the problem. The following parameter combinations were considered: low speed and low range, high speed and low range, low speed and high range, and high speed and high range. The model considered the following four types of endurance: first, non-linear endurance, which included both parcel weight and drone energy; secondly, consumption depended on parcel weight; thirdly, linear endurance, in which drones had the same maximal flight duration regardless of speed, distance, or parcel weight; and fourthly, unlimited endurance was considered.

The model developed by Dayarian *et al.* (2020) used multiple drones to supply parcels to trucks already making deliveries. The model was designed to improve quick turnaround time for online orders. The truck and drones met at some intermediate location, and the truck received the parcels, after which the drone returned to the depot. The truck then continued with its deliveries.

Schermer *et al.* (2018) modelled a VRPD where trucks with drones were assigned to specific trucks to deliver parcels. The same truck, at the next customer's location, retrieved a drone launched at a customer's location. However, a drone could not return to the same customer it was launched from. An endurance constraint was added to limit the drone flying time. Waiting time between the truck and drone was allowed, and it could occur if the truck arrived at the retrieval location first and waited for a drone or if the drone arrived first and waited for a truck. Drones could carry a single item, while trucks could carry large numbers of items. The truck speed was set to 1, and drone speed was set to α times the truck's speed. The VRPD was benchmarked against a VRP (no drones) with five different endurance settings (percentages). For the experiment, three trucks were used, each carrying a drone. The VRPD model was tested using six TSP instances from TSPLib. The results showed improvement when drones were added to a truck delivery system.

Schermer (2019) added complexity to the model with the addition of intermediate launches and retrievals for drones. A vehicle travelled with multiple drones that were launched and retrieved at different locations. Drones were either launched or retrieved at the customer's location, the depot, or at intermediate locations along the truck route. The drone retrieval point was the only place where there was a waiting time. If the drones arrived first at the retrieval point, the drones waited for the truck, or if the truck arrived first, the trucks waited for the drone. The truck had an endurance limit ϵ . The model objective was to minimise makespan and waiting time.

From their previous work Schermer *et al.* (2018), Schermer *et al.* (2020) added further complexity to the model by allowing drones to return to the exact location they were launched from after making a delivery. The following assumptions were

made; drones could only carry one unit, and trucks had a limited carrying capacity, while drones and trucks had limited endurance, i.e. batteries powered both vehicles. Customers were scheduled according to the truck's carrying capacity. All customers were eligible for drone deliveries. While the drone was on its way to make deliveries, the truck could continue making deliveries. Figure 2.9 shows a typical example of that VRPD solution. Further modelling was conducted using MILP and several variations that could be adapted to the current model were presented. The variants included limited flight distance, limited flight time without a recharge, limited flight time with non-instantaneous recharge, and heterogeneous drone fleets.

2.7 Summary

The last mile delivery is a complex problem to solve. Historical research has shown TSP to be a complex problem, especially when dealing with many nodes. This chapter also discussed the different existing truck and drone models.

There are different methods of solving TSP and VRP, which are exact algorithms, heuristics, and metaheuristics. These methods are discussed in the next chapter. It is clearly shown that solving the model proposed in this dissertation is more complex than solving a TSP or VRP. The reason for this additional complexity is the drone which is added to assist the truck to make deliveries. All trucks and drones must be optimised. In Chapter 3, the different approaches researchers have taken to solve truck-drone systems are discussed.

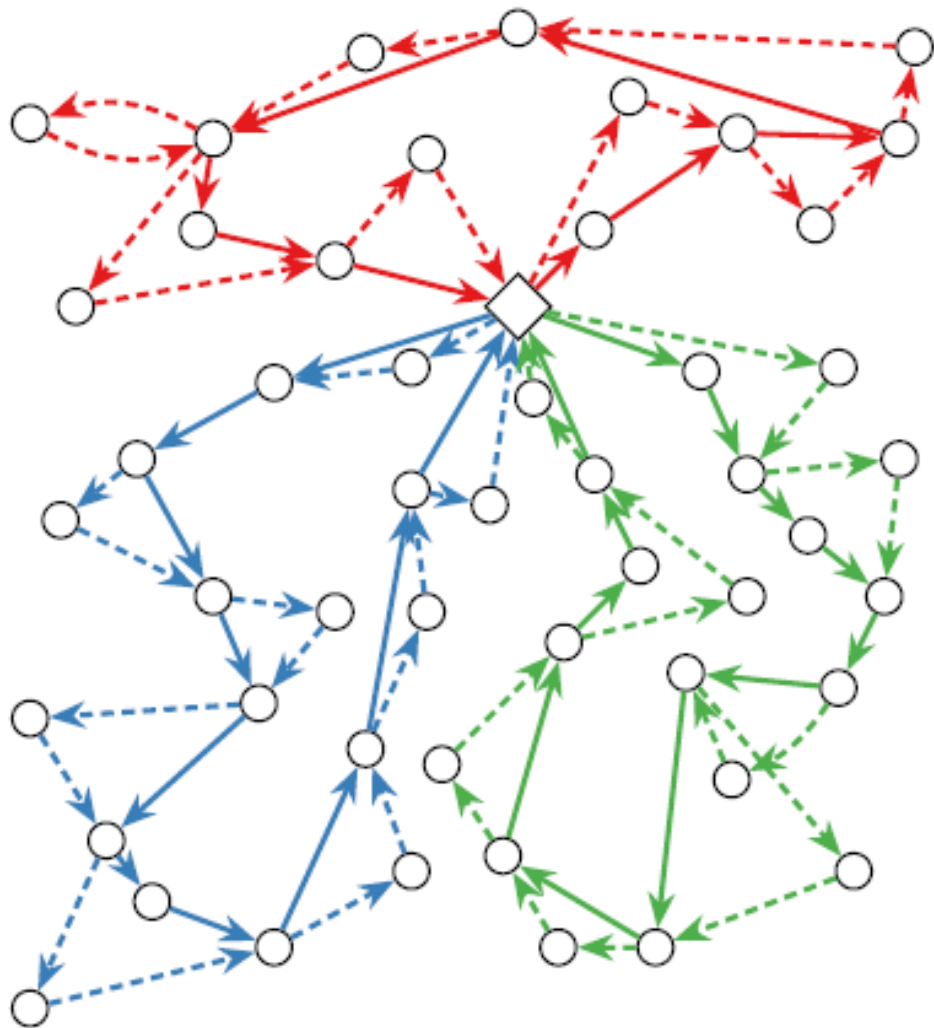


Figure 2.9: Vehicle routing problem with a drone where drone is allowed to return to the same vertex (Schermer (2019)).

Chapter 3

Solution Strategy Selection

This chapter explores existing solution strategies used to determine good solutions for routing problems. The solution strategies are categorised into exact and approximate strategies. The exact methods are used to find optimal solutions, and approximation algorithms are used to find ‘good enough’ solutions. These strategies are discussed in Section 3.1 (exact algorithms), Section 3.2 (heuristics), and Section 3.3 (metaheuristics). Section 3.4 introduces the k-means algorithm which is later used in one of the proposed truck and drone scheduling algorithms. Section 3.5 describes existing applications of exact and approximation algorithms to scheduling truck and drone delivery systems. Finally, the chapter is concluded in Section 3.6.

3.1 Exact solution algorithms

The TSP is one of the oldest routing problems. This section first describes the historical background of the TSP solution strategies in Section 3.1.1. Next, the branch-and-bound, and branch-and-cut algorithms are discussed in Sections 3.1.2 and 3.1.3.

3.1.1 Historical background

Exact solutions are algorithms that guarantee the optimal solution of an optimisation problem. Dantzig *et al.* (1954) introduced the first exact solution algorithm to solve the TSP. The formulation was based on the model defined in Section 2.3 with the relaxation of Equation (2.5). Dantzig *et al.* (1954) solved the TSP optimally by relaxing the integrality conditions and regaining feasibility with the application of a subtour elimination constraint, connectivity constraint, and other constraints. Martin (1966) applied a subtour elimination constraint after visually

n	Nodes in search tree	Computation time
120	1	3.3 seconds
318	1	24.6 seconds
1002	1	94.7 seconds
666	1	260.0 seconds
532	3	294.3 seconds
2392	1	342.2 seconds
225	1	438.9 seconds
3038	193	1.5 days
4461	159	1.7 days
7397	129	49.5 days
13509	9539	~ 10 years

Figure 3.1: Different size TSPs solved on Concorde (a TSP solver) (Applegate *et al.*, 1998)

examining the results from a relaxed Dantzig *et al.* (1954) model. An extension of the ‘Method of Integer Form’ of Gomory (1963) called the ‘Accelerated Euclidian algorithm’ was applied to obtain integrality (Martin, 1966). A fully automated algorithm based on branch-and-bound and cutting-planes (known as Gomory cuts) was later introduced by Miliotis (1976, 1978) for integrality.

The cut-and-price algorithm composed of Gomory cuts, subtour elimination, and column generation was applied optimally to solve 100 node TSP instances. Twelve out of thirteen 100 node TSP problems were solved optimally. The results were achieved without branching. The algorithm had a dynamic generation of subtour elimination constraints, 2-matching constraints and Gomory cuts. A pricing part of the algorithm was used to determine the option whether or not to introduce new variables (Land, 2021). Solving large TSPs with branch-and-cut was made more effective by incorporating methods such as clique trees and path inequalities (other inequalities are outlined in (Naddef and Thienel, 2002)).

Polytopes associated with TSP were also studied incorporating facet identification procedures. The incorporation resulted in the branch-and-cut procedure that was capable of solving more than 100 node TSPs. Applegate *et al.* (1998) used branch-and-cut-and-price, 2-matching and comb inequalities and certain path inequalities to solve the TSPs with thousands of nodes. The example of various size TSP optimal solutions using 48 workstations was reported by Applegate *et al.* (1998) and is shown in Figure 3.1.

3.1.2 Branch-and-bound

The branch-and-bound algorithm solves discrete optimisation problems by breaking up the feasible sets into small subsets, then calculating the bounds of each subset's objective function. Bounds are obtained by relaxing the problem. Other subsets with poor objective functions are discarded (fathomed) from further consideration. The search procedure stops when feasible solutions are produced by each subset or when solutions from the subsets contain no better solution than the existing best solution (Balas and Toth, 1985). Algorithm 1 shows an eager branch-and-bound algorithm as described by Clausen (1999). In Algorithm 1, **Activeset** is the active set, P_k is the branching node of the active set, k is the total number of offspring in a branched node P_k , LB is the lower bound associated with the k^{th} offspring and the *Incumbent* is the value of the current best solution.

Algorithm 1 Branch-and-bound (Clausen, 1999)

```

Initialise:  $Incumbent := \infty$ ;  $LB(P_0) = g(P_0)$ ; Activeset :=  $\{(P_0, LB(P_0))\}$ 
while Activeset  $\neq \emptyset$  do
  Select a branching node  $P \in$  Activeset; Activeset := Activeset  $\setminus \{P\}$ 
  Branch on  $P$  generating  $P_1, \dots, P_k$ 
  for  $i = 1$  to  $k$  do
    Bound  $P_i$ :  $LB(P_i) := g(P_i)$ 
    if  $LB(P_i) = f(\mathbf{X})$  for all solution of  $\mathbf{X}$  then
      if  $f(\mathbf{X}) < Incumbent$  then
         $Incumbent := f(\mathbf{X})$ 
         $Solution = \mathbf{X}$ 
        go to EndBound
      end if
    if  $LB \geq Incumbent$  then
      fathom  $P_i$ 
    else
      Activeset := Activeset  $\cup \{(P_i; LB\{P_i\})\}$ 
      Endbound
    end if
  end if
end for
end while
 $OptimalSolution := Solution$ 
 $Optimalvalue := Incumbent$ 

```

The branch-and-bound algorithm minimises the travel distance between nodes

in the TSP by assigning a binary value ($x_{ij} = 0$ or 1) to the edge (i, j) (branching) and computes the objective function value of each feasible solution (bounding). Whenever a better solution is obtained in the search space, the solution is set as the lowest bound (LB). The search space is further divided into subspaces, and the search for better solutions continues. The search stops when there are no better solutions than LB.

3.1.3 Branch-and-cut

The branch-and-cut algorithm was developed to improve the branch-and-bound algorithm that could efficiently solve less than 100 nodes but struggled to solve larger problems. Padberg and Rinaldi (1987) reported the results of solving 532 city TSPs using branch-and-cut.

Branch-and-cut is an approach that integrates cutting planes with the branch-and-bound enumeration phase (Fischetti *et al.*, 1997; Padberg and Rinaldi, 1991). Given a set of inequalities, $\mathbf{L} \subset \mathbf{L}_n$, that are facet inducing on \mathbf{Q}^n , \mathbf{S} is a family of ordered sets, $(\mathbf{F}_0, \mathbf{F}_1) \subset \mathbf{E}$ are two disjoint edge sets in an ordered pair in \mathbf{S} , and $P(\mathbf{L}, \mathbf{F}_0, \mathbf{F}_1)$ is the linear program denoted by the following notation.

$$\min \quad cx \tag{3.1}$$

$$\text{subject to } Ax = 2 \tag{3.2}$$

$$l_x \geq l_e \quad \forall (l, l_e) \in \mathbf{L} \tag{3.3}$$

$$x_e = 0 \quad \forall e \in \mathbf{F}_0 \tag{3.4}$$

$$x_e = 1 \quad \forall e \in \mathbf{F}_1 \tag{3.5}$$

$$0 \leq x \leq 1 \tag{3.6}$$

Algorithm 2 shows the branch-and-cut algorithm, where \mathbf{x}^* is the incidence vector of a certain tour of K_n , c is the cost, and $\bar{\mathbf{x}}$ is the optimal solution of $P(\mathbf{L}, \mathbf{F}_0, \mathbf{F}_1)$.

Algorithm 2 Branch-and-cut

Input $n, c, \mathbf{x}^*, \mathbf{S}$
Step 0
Set $\mathbf{L} \leftarrow \emptyset$
Step 1
if $\mathbf{S} = \emptyset$ **then**
 Stop
else
 Otherwise pick an ordered pair $(\mathbf{F}_0, \mathbf{F}_1)$ from \mathbf{S} and replace \mathbf{S} by $\mathbf{S} - (\mathbf{F}_0, \mathbf{F}_1)$
 Step 2
 Solve the linear program $P(\mathbf{L}, \mathbf{F}_0, \mathbf{F}_1)$
 if the program is inconsistent **then**
 Go to Step 1
 else
 Let $\bar{\mathbf{x}}$ be its optimal solution
 end if
 if $c\bar{\mathbf{x}} \geq c\mathbf{x}^*$ **then**
 Go to Step 1.
 end if
 Find one or more inequalities of \mathbf{L}_n that are violated by $\bar{\mathbf{x}}$
 if Violation **then**
 Add the violated inequalities to \mathbf{L} and go to Step 2.
 else
 if $\bar{\mathbf{x}} = \text{integer}$ **then**
 Replace \mathbf{x}^* by $\bar{\mathbf{x}}$ and go to Step 1
 end if
 end if
 Pick an edge $e \in \mathbf{E}$ such that $0 < \bar{x}_e < 1$
 Replace \mathbf{S} by $\mathbf{S} + (\mathbf{F}_0 + \{e\}, \mathbf{F}_1) + (\mathbf{F}_0, \mathbf{F}_1 + \{e\})$ and go to Step 1, once an ordered pair is removed from \mathbf{S}
end if

3.2 Heuristics

Heuristic algorithms are an alternative way to solve combinatorial problems such as travelling salesman problems. They often find “good” solutions in a reasonable time even though they do not guarantee an optimal solution (Lin and Kernighan, 1973; Rosenkrantz *et al.*, 1977). They search for the best solution in a search space, stop when found, and discontinue searching in other areas. The search often results

in a local minimum solution. The heuristics are divided into three categories: tour construction, tour improvement, and the combination approach. The tour construction approach follows a three-step procedure when applied. These steps are *initialisation, selection, and insertion*. Examples of this type of heuristics are nearest neighbour, greedy, and insertion heuristics. The tour improvement approach starts from a feasible solution and applies improving moves. The widespread improvements that are applied are 2-opt and 3-opt. They are applied by removing two or three edges and randomly reconnecting them to other nodes. The procedure is applied until no improvements are noticed (Lin, 1965). Metaheuristics can also be applied in a tour improvement approach, to improve the tour. Lastly, compound improvement is the combination of the tour construction and tour improvement approach. It first applies the construction procedure and then improves the tour solution by using the improvement approach. This strategy allows the search to escape local minima the same way as other metaheuristic algorithms.

3.3 Metaheuristics

Metaheuristics are used to guide the heuristic search process to escape from local minima to find a better near-optimal solution (Blum and Roli, 2003). A balance between exploration and exploitation is needed to find this good solution (Blum and Roli, 2003). Metaheuristics are classified into trajectory methods and population-based methods (Blum and Roli, 2003). Figure 3.2 summarises a number of common metaheuristic strategies.

3.3.1 Trajectory methods

Trajectory methods are local search-based metaheuristics that start from an initial state and move to a new state through an attractor. Its search space depends on problem representation, neighbourhood structure, and problem instances (Blum and Roli, 2003).

3.3.1.1 Simulated annealing

Simulated annealing (SA) is applied to solve an extensive combinatorial optimisation (CO) problem similar to the annealing simulation of solids. It was introduced by Kirkpatrick *et al.* (1983). In the physical annealing process, solids are melted in a heat bath at a high temperature. The solids arrange themselves at maximum temperature when solids change to the liquid state. Then, as the temperature slowly decreases, the solids arrange themselves in a highly liquid lattice structure, and the system's energy is minimised. Each physical state represents solutions

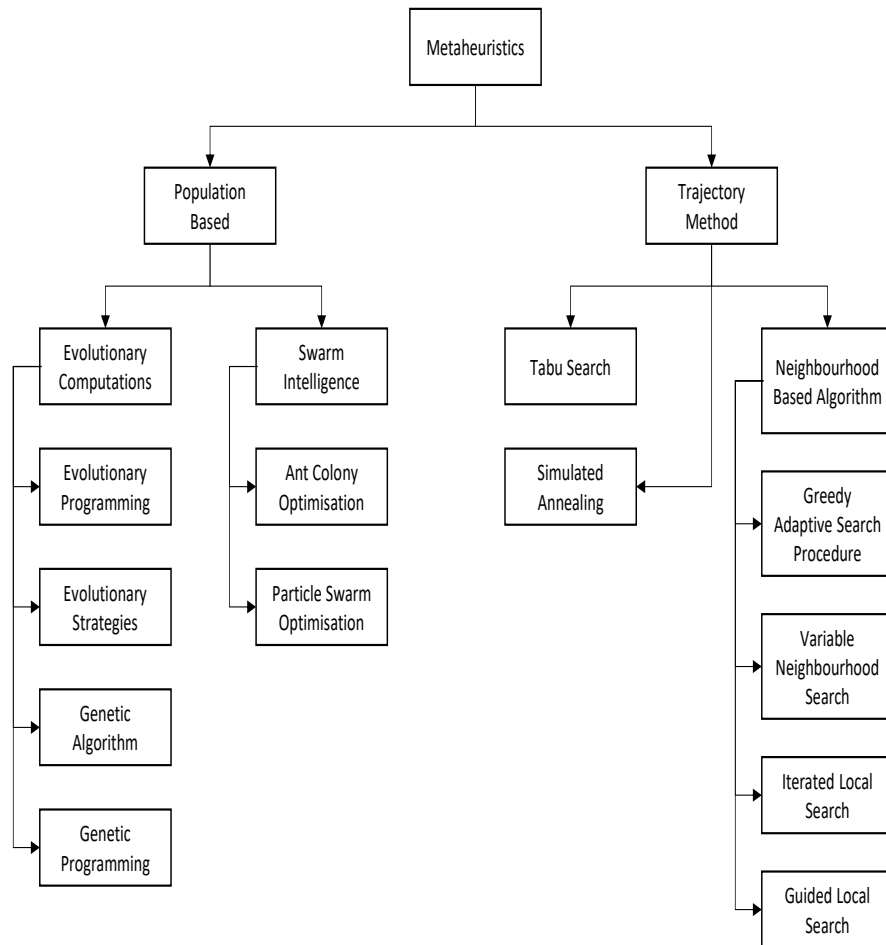


Figure 3.2: Common metaheuristic algorithms

of a CO problem, and the cost of a solution is the same as the energy of a state (Černý, 1985).

The temperature is decreased throughout the search process. At the higher temperatures the algorithm explores more of the solution space by accepting worse quality solutions. This strategy allows the search to escape local minima. As the temperature slowly decreases the algorithm focuses more on converging to a good solution. During the search, if the nearest neighbourhood solution is better or worse than the current solution, the solution is accepted using the Boltzmann distribution in Equation (3.7),

$$P(t) = \begin{cases} 1 & f(\mathbf{x}) < f(\mathbf{x}^*) \\ e^{-\frac{-(f(\mathbf{x})-f(\mathbf{x}^*))}{T}} & \text{Otherwise} \end{cases} \quad (3.7)$$

where $P(t)$ denotes the probability of accepting the solution at iteration t , $f(\mathbf{x})$ is the objective function of the current solution, $f(\mathbf{x}^*)$ is the objective function of the best solution found so far, and T is the temperature that is updated per iteration t (Černý, 1985). Algorithm 3 shows the basic steps of the simulated annealing algorithm.

Algorithm 3 Simulated annealing

```

 $T \leftarrow T_0$  (Initial temperature)
while Stopping condition not met do
   $\mathbf{x} \leftarrow \text{PickRandomNeighbour } N(\mathbf{x}^*)$ 
  if  $f(\mathbf{x}) < f(\mathbf{x}^*)$  then
     $\mathbf{x}^* \leftarrow \mathbf{x}$ 
  else
    Accept  $\mathbf{x}$  using (3.7)
  end if
  Update( $T$ )
end while

```

3.3.1.2 Tabu search

Glover (1986) proposed the tabu search (TS) algorithm in 1986. The algorithm applies the best improvements in local search and short-term memory (through a tabu list) to find the best solution. The tabu list (TL) is used to store solutions to avoid cycling through the search space and to keep track of recently visited solutions (Glover and Laguna, 1997). Movements toward solutions in the list are forbidden. The solutions outside the TL are chosen at each iteration. The TL is updated using the best solution per iteration.

The length of the TL determines the memory of the search process. The length is set long for a large search space and short for a small search space. The length can also be set dynamically, and it is increased to improve exploration when stagnation is experienced or decreased to exploit the current best solution when improvement is not obtained (Glover and Laguna, 1997).

In other instances, the solution attributes are stored in the TL; however, a good solution can be missed. To ensure that suitable solutions are not missed, aspiration criteria are included to ensure that even forbidden solutions can be visited. Algorithm 4 shows a summary of the TS algorithm.

Algorithm 4 Tabu search

```

x ← Initial Solution
TabuList ← ∅
while Stopping Condition Not Met do
  x ← SelectTheBestNeighbour( $N(\mathbf{x}^*)$ ) not on the TabuList
  Update(TabuList)
end while

```

3.3.1.3 Greedy randomised adaptive search procedure

The *greedy randomised adaptive search procedure* (GRASP) is an iterative procedure that combines a construction heuristic and a local search (Feo and Resende, 1995). At each iteration of the construction phase, a feasible solution is constructed one element at a time. The next element is chosen randomly from the candidate list. Then, the elements are ranked and given scores. These scores are updated at each iteration. The best candidates are stored in a list called a restricted candidate list (RCL). Different solutions are obtained per iteration using the probabilistic random part of GRASP to choose a candidate from the RCL.

Algorithm 5 GRASP

```

 $\mathbf{x}_0$  ← Initial Instances()
while Stopping Condition Not Met do
   $\mathbf{x}_1$  ← ConstructGreedyRandomisedSolution( $x_o$ )
   $\mathbf{x}_2$  ← LocalSearch( $\mathbf{x}_1$ )
   $\mathbf{x}_3$  ← UpdateSolution( $\mathbf{x}_0$ ,  $\mathbf{x}_3$ )
end while
BestSolutionFound ←  $\mathbf{x}_3$ 

```

3.3.1.4 Guided local search

The *guided local search* (GLS) is a metaheuristic that explores a search space by guiding the local search out of the local minima to a better solution. The local search method starts from a random solution and moves to the next neighbouring solution in the search space to find the best solution that minimises the objective function. The local searches are performed either to find the best improvement (greedy) or first improvement. The greedy improvement replaces the current solution with the solution that primarily improves the objective function.

The shortcoming of local search heuristics is that they typically become stuck in local minima. The GLS assists the local search to escape from local minima (Voudouris and Tsang, 1999). The local search is guided out of the local minima by augmenting the cost function. When the penalty is applied, the search escapes a current search space to explore another area. The process continues until the best overall solution is found (Voudouris and Tsang, 1999). Equation (3.8) is used by GLS to escape local minima

$$h(\mathbf{x}) = g(\mathbf{x}) + \lambda \cdot \sum_{i=1}^m P_i \cdot I_i(\mathbf{x}) \quad (3.8)$$

where $g(\mathbf{x})$ denotes the current objective function, $h(\mathbf{x})$ denotes a new objective function after the penalty factor is added, λ denotes the regularisation parameter, $I_i(\mathbf{x})$ is an indicator function to show i is present in \mathbf{x} , P_i is a penalty parameter and q is the number of penalty features. The modification of the penalty is performed using equation (3.9):

$$util(i, i) = I_i(\mathbf{x}) \cdot \left(\frac{c_i}{1 + P_i} \right) \quad (3.9)$$

where c_i is the cost assigned to every feature in the local minima. Algorithm 3.8 shows the algorithm of applying GLS.

Algorithm 6 Guided local search

```

GuidedLocalSearch ( $p, g, \lambda, [I_1, \dots, I_m], [c_1, \dots, c_m], m$ )
 $t \leftarrow 0$ 
 $\mathbf{x}(0) \leftarrow$  Random or heuristic solution
while  $i \leftarrow 1$  until  $m$  do
   $P_i \leftarrow 0$  set all penalties to zero
  while Stopping criteria not met do
    Apply Equation (3.8)
     $\mathbf{x}(t + 1) \leftarrow$  LocalSearch( $\mathbf{x}(t), h$ )
    while  $i \leftarrow 1$  Until  $m$  do
      Apply Equation (3.9)
    end while
  end while
   $P_i \leftarrow P_i + 1$ 
   $t \leftarrow t + 1$ 
end while
 $\mathbf{x}^* \leftarrow$  Best solution (minimum  $g$ )

```

To summarise about GLS, a heuristic solution is constructed, and a local search method is applied as the improvement method. The method will generate a local minimum that is not necessarily of high quality. After each improvement, the penalty feature is incremented from zero to the value that maximises the utility formula.

3.3.1.5 Variable neighbourhood search

The variable neighbourhood search (VNS) is a method that searches for the best solution by dynamically changing the neighbourhood structure. It was proposed by Hansen and Mladenović (1999) and the principle and application were introduced by Hansen and Mladenović (2001). The algorithm contains shaking, local search, and move operators. Algorithm 7 shows the basic VNS algorithm, where t is the number of iterations, t_{max} is the maximum number of iterations, and \mathbf{N}_t is the neighbourhood structure at each iteration.

Algorithm 7 Variable neighbourhood search

```

SelectNeighbourhoodStructure  $\mathbf{N}_t(t = 1, \dots, t_{max})$ 
 $\mathbf{x} \leftarrow$  GenerateInitialSolution
while Stopping Condition Not Met do
   $t \leftarrow 1$ 
  while  $t \leq t_{max}$  do
     $\mathbf{x} \leftarrow$  Shake( $\mathbf{N}_t(\mathbf{x})$ )
     $\mathbf{x}' \leftarrow$  LocalSearch( $\mathbf{x}$ )
     $(\mathbf{x}, t) \leftarrow$  NeighbourhoodChange( $\mathbf{x}, \mathbf{x}', t$ )
  end while
end while

```

The initial solution is generated with a set neighbourhood structure. Then, the solution within the t^{th} neighbourhood structure is randomly selected. The solution becomes a starting point of the local search (LS). The LS is applied to obtain the best minimum solution. The LS solution is compared with the best solution found so far, and if the LS solution is better than the best solution, it replaces the best solution. If better solutions are not found, a new shaking operator is applied. In the shaking phase, two or more edges are exchanged. The shaking diversifies the search to aid the algorithm to escape the local minima. This property is better exploited by the variable neighbourhood descent (VND). The disadvantage of using VNS is that the algorithm becomes stuck when local minima are clustered. It requires a more flexible acceptance criterion to escape from the cluster and search for more solutions in other areas (Hansen and Mladenović, 2001).

3.3.1.6 Iterated local search

Whenever an iterated local search (ILS) is used to find the best solution, the following steps outlined in Algorithm 8 are applied. The initial solution is first generated randomly or by using a greedy construction heuristic. Secondly, a local search is applied to the solution. The greedy initial solution and local search result in a good quality initial solution and minimal CPU time usage (Blum and Roli, 2003).

The local descent can become trapped in local minima, and perturbation is used to guide the search out of the local minima. A small perturbation explores fewer solutions and often returns to the same solution, while a higher perturbation results in more randomness. Perturbation can either be deterministic, randomised, or adaptive. A local search is applied to the perturbation solution, and the result is evaluated using the acceptance criteria. The solution is accepted on the condition

that it is better than the current solution. The cost function compares these two solutions and accepts the one with the better solution.

Algorithm 8 Iterated local search

```

 $\mathbf{x} \leftarrow$  Initial Solution
 $\mathbf{x} \leftarrow$  LocalSolution( $\mathbf{x}$ )
while Stopping Condition Not Met do
   $\mathbf{x}' \leftarrow$  Perturbation ( $\mathbf{x}, history$ )
   $\mathbf{x}'' \leftarrow$  LocalSearch( $\mathbf{x}'$ )
   $\mathbf{x} \leftarrow$  AcceptanceCriteria ( $\mathbf{x}, \mathbf{x}'', history$ )
end while

```

3.3.2 Population-based methods

Population-based methods produce a set of solutions per iteration instead of a single solution. This section discusses three types of population-based algorithms. First, evolutionary computation is discussed, secondly, particle swarm optimisation, and finally, ant colony optimisation.

3.3.2.1 Evolutionary computation

Evolutionary computation (EC) is a set of algorithms inspired by how living beings evolve and adapt to their environment. There are different kinds of evolutionary computation. These are evolutionary programming (EP), evolutionary strategies (ES) and genetic algorithms (GA). EP was developed for machine intelligence with discrete representation (Fogel, 1962; Fogel *et al.*, 1966). Rechenberg (1973) proposed ES that solves continuous optimisation problems. Holland (1975) and Mitchell (1998) introduced the GA that solves the combinatorial optimisation problem.

EC has operators applied to the individual in a population to generate individuals of the following population. Operators are either self-adaptive or combinations of different individuals in a population. The EC algorithms apply the steps outlined in Algorithm 9. The well-known operators of EC are recombination or crossover between individuals and mutation or modification of individuals. In Algorithm 9, n_s denotes the number of individual members in the population, $\mathbf{f}(\mathbf{S})$, are the fitness function values of the population, \mathbf{S}^* is the recombined population, \mathbf{S}^{**} , is the mutated population, and $\mathbf{f}(\mathbf{S}^*)$ are the new population fitness function values.

Algorithm 9 Evolutionary computation

```

 $\mathbf{S} \leftarrow \text{GeneratePopulation}(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{n_s})$ 
Evaluate  $\mathbf{f}(\mathbf{S})$ 
while stopping condition not met do
   $\mathbf{S}^* \leftarrow \text{Recombine}(\mathbf{S})$ 
   $\mathbf{S}^{**} \leftarrow \text{Mutate}(\mathbf{S}^*)$ 
  Evaluate  $\mathbf{f}(\mathbf{S}^{**})$ 
  SelectBestIndividual from  $\mathbf{S}$  and  $\mathbf{S}^{**}$ 
end while

```

Two specific ECs deserve further mention; the differential evolution algorithm and the covariance matrix adapting evolutionary strategy.

Differential evolution: Differential evolution (DE) is also a population-based stochastic optimisation algorithm. The differential mutation operator allows the step sizes of the algorithm to adapt automatically to the objective function landscape (Qin and Suganthan, 2005). Whenever the population converges around a local optimum while the algorithm is still in the exploration phase (randomly sampled individuals are still scattered in the search space), larger step sizes are applied to escape the local optimum. Later in the iteration, the individual converges around a single minimum, searching for the best solutions. Algorithm 10 shows the pseudocode of the DE algorithm.

Neighbourhood search differential evolution (NSDE) and self-adaptive neighbourhood search differential evolution (SaDE) are improvements on the basic DE algorithm. NSDE has no self-adaption but mixes search bias through different neighbourhood search operators, whereas SaDE has a self-adaptive mutation strategy and crossover rate (Grobler, 2015). The combination of these two algorithms is called the self-adaptive neighbourhood search DE (SaNSDE). SaNSDE also applies alternate sampling values from the Gaussian and Cauchy distribution, where the Gaussian distribution uses small steps and the Cauchy distribution uses bigger steps.

The covariance matrix adapting evolutionary strategy: Evolution strategies were inspired by biological evolution. They are commonly applied to black-box optimisation problems in continuous search spaces. They sample new candidate solutions using a multivariate normal probability distribution (Hansen *et al.*, 2015). One of the most successful evolutionary strategy algorithms is the covariance matrix adapting evolutionary strategy algorithm (CMAES) (Auger and Hansen, 2005).

CMAES consists of four main phases: solution generation, selection and recombination, updating the covariance matrix, and updating the step size. Strategy parameters are self-adapted to determine the best search direction and maximum

Algorithm 10 Differential Evolution (Grobler, 2015)

```

Initialise population of  $n_s$  individuals
T=1;
while condition not met do
  for All individual  $i$  do
    From population randomly select individual  $i_1$ ,  $i_2$  and  $i_3$ 
    for  $i_1 = i_2$  or  $i_2 = i_3$  do
      Randomly select new individual,  $i_1$ ,  $i_2$  and  $i_3$ 
    end for
    Calculate trial vector
    Calculate the offspring solutions
  end for
  for All individual  $i$  do
    if new solution is better than the currently stored solution then
      Store the new solution
    end if
  end for
end while

```

step size (Engelbrecht, 2007). The algorithm adopts the strategy parameters in parallel with the population and optimises the optimisation process. A population of solutions is generated at each iteration according to a multivariate normal distribution. The pseudocode of CMAES is shown in Algorithm 11.

Algorithm 11 CMAES Algorithm (Grobler, 2015)

```

Initialise all algorithm parameters
while Stopping criteria not met do
  Generate offspring by mutating the mean
  Evaluate offspring
  Sort the offspring by fitness
  Update the mean of the population
  Update the step-size cumulation path
  Update the covariance-matrix cumulation path
  Update the step size
  Update the covariance matrix
end while

```

3.3.2.2 Particle swarm optimisation

The particle swarm optimisation (PSO) is inspired by the social behaviour of bird flocking (Kennedy and Eberhart, 1995). In PSO, a group of particles are randomly placed in a search space. The position of each particle represents a solution. The position of each particle is improved at each iteration using the current particle position, \mathbf{x}_i , and velocity, \mathbf{v}_i , of particle i . Every time the particle moves, it searches for a better solution. The PSO searches for solutions in this multidimensional space using the procedure shown in Algorithm 12. During each iteration particle positions are updated using Equations (3.10) and (3.11):

$$v_{id}(t+1) = wv_{id}(t) + c_1 \text{rand}() (P_{id}(t) - x_{id}(t)) + c_2 \text{rand}() (x_d^*(t) - x_{id}(t)) \quad (3.10)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (3.11)$$

where $v_{id}(t)$ is the velocity of particle i in dimension j at time t , c_1 and c_2 are the cognitive and social acceleration constants, $P_{id}(t)$ is the personal best (pbest) position in dimension d of the i^{th} particle at time t , $x_{id}(t)$ is the position of particle i in dimension d at time t , and $x_d^*(t)$ denotes the global best (gbest) position in dimension d at time t . Finally $\text{rand}()$ are samples from a uniform random distribution, $U(0, 1)$ and w is the inertia weight.

Algorithm 12 Particle swarm optimisation

Initialise Population of particles with random positions and velocities in d -dimensions

for ($i = 0; i < n_s, i++$) **do**

 Evaluate $f(\mathbf{x}_i)$

while Termination criterion is not met **do**

for ($i = 0; i < n_s, i++$) **do**

if $f(\mathbf{x}_i) > f(\mathbf{P}_i)$ **then**

$\mathbf{P}_i = \mathbf{x}_i$

end if

if $f(\mathbf{x}_i) > f(\mathbf{x}^*)$ **then**

$\mathbf{x}^* = \mathbf{x}_i$

end if

 Update the velocity of the particle using Equation (3.10)

 Update the position of the particle using Equation (3.11)

 Evaluate $f(\mathbf{x}_i)$

end for

end while

end for

Guaranteed Convergence Particle Swarm Optimisation

In the basic PSO, the particles move through the decision space and are attracted toward the swarm *pbest* and *gbest* positions. If the particles' *pbest* and *gbest* positions are equal in the search space, $\mathbf{x}_i(t) = \mathbf{x}^*(t) = \mathbf{P}_i(t)$, then the momentum term $w_i v_{id}(t)$, which remains the driving force for a specific particle to continue exploring the rest of the search space, tends to zero. This condition can lead to premature algorithm convergence (Van den Bergh and Engelbrecht, 2002). The algorithm might converge to the current best position discovered and convergence is not even guaranteed to a local optimum.

A new parameter is introduced to the PSO to resolve the issue. Let an index, ω , represent the global best particle. Then the velocity and displacement of particle ω is updated using Equations (3.12) and (3.13). The update forces the *gbest* particle into a random search around the global best position. The search space size is adjusted based on the number of consecutive particles' successes or failures, where *success* is defined as an improvement in the objective function value. All other particles are adjusted using Equations (3.10) and (3.11):

$$v_{\omega d}(t+1) = -x_{\omega d}(t) + x_d^*(t) + wv_{\omega d}(t) + \rho(t)(1 - 2rand()) \quad (3.12)$$

and

$$x_{\omega d}(t+1) = x_d^*(t) + wv_{\omega d}(t) + \rho(t)(1 - 2rand()) \quad (3.13)$$

where $\rho(t)$ is the time-dependent scaling factor at time t .

3.3.2.3 Ant colony optimisation

Dorigo (1992) proposed the ant colony optimisation (ACO) algorithm. Real ants foraging for food inspired the ACO algorithm. The ants leave their nest and scatter, searching for food, equivalent to an algorithm exploration process. During their search, they deposit pheromone on their chosen route. A route used the most frequently between the nest (their starting point) and food will have the highest pheromone deposit. After some time, evaporation will occur on all the edges, and an edge with less pheromone concentration becomes unattractive to the ants. The best path would have the highest pheromone deposit concentration on the edges.

In ACO, the ant constructs a path on the graph, $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ where \mathbf{V} denotes the set of vertices (nodes) and \mathbf{E} denotes the set of edges. The costs associated with the edges \mathbf{E} is denoted as C_{ij} for the edge between nodes i and j , where $C_{ij} = C_{ji}$ for all $(i, j) \in \mathbf{E}$ for an undirected graph, and $C_{ij} \neq C_{ji}$ for all $(i, j) \in \mathbf{E}$ in a directed graph. The pheromone deposited by ant k between node i and node j is denoted by τ_{ij} .

At the beginning of route construction, ants are placed randomly on nodes. As each ant k starts to move to the next node j , the ant k will deposit pheromone of

τ_{ij} on the associated edge. Each ant will visit all the nodes once and then return to the starting node. The total cost to move through this Hamiltonian circuit is the sum of the costs associated with each edge in the route. For an ant k to move from node i to node j , it has to choose the best node j . This decision depends on heuristic information (Equation (3.14)),

$$\eta_{ij} = \frac{1}{c_{ij}} \quad (3.14)$$

which is calculated as the inverse of the cost (distance) between node i and node j , and the pheromone τ_{ij} . The probability of selecting the next node j if an ant k is currently located at i is computed as

$$P(t)_{ij}^k = \begin{cases} \frac{\tau_{ij}^\alpha(t) \cdot \eta_{ij}^\beta(t)}{\sum_{j \in \mathbf{N}_i^k(t)} \tau_{ij}^\alpha(t) \cdot \eta_{ij}^\beta(t)} & \text{if } j \in \mathbf{N}_i^k(t) \\ 0 & \text{if } j \notin \mathbf{N}_i^k(t) \end{cases} \quad (3.15)$$

where $\mathbf{N}_i^k(t)$ is the set of feasible nodes connected to node i with respect to ant k at iteration t , $\tau_{ij}(t)$ is the pheromone concentration on the edge between node i and j at iteration t , α and β are parameters that control the relative importance of pheromone τ_{ij} and heuristic information η_{ij} , and t is the iteration number.

The pheromone is evaporated using using Equation (3.16):

$$\tau_{ij}(t) = (1 - \rho) \cdot \tau_{ij}(t) \quad (3.16)$$

where ρ is the evaporation rate.

The pheromone amount to be deposited on each edge (i, j) is calculated using Equation (3.17)

$$\Delta\tau_{ij}^k(t) = \begin{cases} Q/L_k(t) & \text{if ant } k \text{ used edge } (i, j) \text{ on its tour} \\ 0 & \text{Otherwise} \end{cases} \quad (3.17)$$

and lastly pheromone is updated on each edge (i, j) using Equation (3.18):

$$\tau_{ij}(t+1) = \tau_{ij}(t) + \sum_{k=1}^{n_s} \Delta\tau_{ij}^k(t) \quad (3.18)$$

where L_k is the total distance travelled by ant k from the start node to all nodes and back to the start node (the Hamiltonian cycle), Q is a constant, n_s is the total number of ants, $\Delta\tau_{ij}^k(t)$ is the amount of pheromone deposited on each edge (i, j) at iteration t by ant k .

Algorithm 13 shows a simple ACO algorithm. There are other types of ACO algorithms, such as the max-min ant system, which controls the maximum and

minimum value of the pheromone to avoid stagnation when dealing with complex problems, and the ant colony system (ACS) algorithm discussed in the following section.

Algorithm 13 Ant colony optimisation (Engelbrecht, 2007)

```

 $t \leftarrow 0$ 
Input datasets  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_j\}$ ,
Initialise  $n_s, \alpha, \beta$ , maximum number of iterations ( $t_{max}$ ),  $Q, \rho$ 
Initialise the pheromone  $\tau_{ij}(0)$  for all edges
Compute heuristic information using Equation (3.14)
while ( $t \leq t_{max}$ ) do
  Place  $n_s$  ants at random nodes
  for each ant  $k = 1, \dots, n_s$  do
     $\mathbf{x}^k(t) = \emptyset$ , where  $\mathbf{x}^k(t)$  is the route of ant  $k$  (permutation of nodes) at
    iteration  $t$ 
    while 'Hamiltonian Circuit'=False do
      From node  $i$ , select next node,  $j$ , using Equation (3.15)
       $\mathbf{x}^k(t) \leftarrow$  (Add edge  $(i, j)$  to the tour)
    end while
    Compute the Hamiltonian tour's fitness function value,  $f(\mathbf{x}^k(t)) = \sum c_{ij}$ 
    for all edge  $(i, j)$  in the tour
  end for
  for Each edge  $(i, j)$  of the graph do
    Evaporate pheromone using Equation (3.16)
    Calculate the amount of pheromone to be deposited on edge  $(i, j)$  using
    Equation (3.17)
    Update pheromone using Equation (3.18)
  end for
   $t \leftarrow t + 1$ 
end while
Output:  $\mathbf{x}^k(t), f(\mathbf{x}^k(t)) = \min_{k'=1, \dots, n_s} \{f(\mathbf{x}^{k'}(t))\}$ 

```

The ant colony system

ACS uses a different state transition rule compared to a typical ant colony optimisation algorithm to balance exploration and exploitation. The transition rule applies both a local and global (or iteration best) pheromone update. The method first encourages exploration at the beginning of a search and later intensifies the search around a single solution (Dorigo and Di Caro, 1999). Algorithm 14 shows the pseudocode of the ACS algorithm.

An ant k located at node i selects the next node j using equation (3.19), where $r_0 \in [0, 1]$ is set by the user to control exploitation and exploration and $r \sim U(0, 1)$.

$$j = \begin{cases} \operatorname{argmax}_{u \in \mathbf{N}_i^k(t)} \{ \tau_{iu}(t) \eta_{iu}^\beta(t) \} & \text{if } r \leq r_0 \\ J & \text{if } r > r_0 \end{cases} \quad (3.19)$$

where $\tau_{iu}(t)$ is the pheromone concentration between node i and node u at iteration t , η_{iu} is the attractiveness, or desirability, between node i and node u , β is the parameter that control the relative importance of heuristic information and $\mathbf{N}_i^k(t)$ denotes the set of feasible nodes for ant k when located on node i . Finally, $J \in \mathbf{N}_i^k(t)$ is a node randomly selected according to the probability

$$P_{iJ}^k(t) = \frac{\eta_{iJ}^\beta(t) \tau_{iJ}(t)}{\sum_{u \in \mathbf{N}_i^k(t)} \eta_{iu}^\beta(t) \tau_{iu}(t)}. \quad (3.20)$$

The edges that correspond to the best route, \mathbf{x} , use the global best ant to update its pheromone concentration. The pheromone is updated using Equations (3.21) and (3.22),

$$\tau_{ij}(t+1) = (1 - \rho_1) \tau_{ij}(t) + \rho_1 \Delta \tau_{ij}(t) \quad (3.21)$$

$$\Delta \tau_{ij}(t) = \begin{cases} \frac{1}{f(\mathbf{x})} & \text{if } (i, j) \in \mathbf{x} \\ 0 & \text{otherwise} \end{cases} \quad (3.22)$$

where ρ_1 is the pheromone evaporation rate and $f(\mathbf{x})$ is the global best fitness function value of a tour.

Local search is also used to update all the edges visited by the ant using Equation (3.23),

$$\tau_{ij}(t) = (1 - \rho_2) \tau_{ij}(t) + \rho_2 \tau_0 \quad (3.23)$$

where τ_0 is a positive constant calculated using the number of nodes, n , and the length of the nearest neighbour heuristic for TSP is L , giving $\tau_0 = (nL^{-1})$ or another estimate of the path length. The pheromone evaporation rate is denoted by $\rho_2 \in [0, 1]$.

Algorithm 14 Ant colony system (Engelbrecht, 2007)

```

 $t \leftarrow 0$ 
Input datasets  $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_j\}$ ,
Initialise  $n_s, \rho_1, \rho_2, \tau_0, r_0, \beta, t_{max}, Q$ 
Initialise the pheromone  $\tau_{ij}(0) \sim U(0, \tau_0)$  for all edges
Compute heuristics information using Equation (3.14)
Create solution from nearest neighbour heuristic,  $\mathbf{x}_{nn}$ 
 $\mathbf{x}(t) = \mathbf{x}_{nn}$ 
Compute cost,  $f(\mathbf{x}(t))$ , of solution  $\mathbf{x}(t)$ 
while ( $t \leq t_{max}$ ) do
  Place  $n_s$  ants at random nodes
  for Each ant  $k = 1, \dots, n_s$  do
     $\mathbf{x}^k(t) = \emptyset$ 
     $f(\mathbf{x}^k(t)) = 0$ 
    while 'Hamiltonian Circuit'=False do
      From node  $i$ , select next node,  $j$ , using Equation (3.19)
       $\mathbf{x}^k(t) \leftarrow$  (Add edge  $(i, j)$  to the tour)
      Evaporate pheromone using Equation (3.23)
    end while
    Compute the Hamiltonian tour's fitness  $f(\mathbf{x}^k(t)) = \sum c_{ij}$  for all edge  $(i, j)$ 
    in the tour
  end for
   $\mathbf{x}' = \mathbf{x}^k(t), f(\mathbf{x}^k(t)) = \min_{k'=1, \dots, n_s} \{f(\mathbf{x}^{k'}(t))\}$ 
  if  $f(\mathbf{x}'(t)) < f(\mathbf{x}(t))$  then
     $\mathbf{x}(t) \leftarrow (\mathbf{x}'(t))$ 
     $f(\mathbf{x}(t)) \leftarrow f(\mathbf{x}'(t))$ 
  end if
  for Each edge  $(i, j) \in \mathbf{x}(t)$  of the graph do
    Update pheromone on edge  $(i, j)$  using Equation (3.21)
  end for
   $t \leftarrow t + 1$ 
   $\mathbf{x}(t+1) \leftarrow \mathbf{x}(t)$ 
   $f(\mathbf{x}(t+1)) \leftarrow f(\mathbf{x}(t))$ 
end while
Output:  $\mathbf{x}(t), f(\mathbf{x}(t))$ 

```

3.4 K-means

MacQueen (1967) proposed a clustering algorithm called k-means. Given a dataset $\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_s\}$ with s instances, the algorithm partitions the data into distinct groups, \mathbf{C}_K , where K is the number of clusters and \mathbf{c}_k is the centroid of cluster k . Each data point, \mathbf{x}_j , is allocated to the nearest centroid obtained by calculating the Euclidian distance from every data point to every centroid.

The k-means pseudocode is shown in Algorithm 15. Here K is randomly initialised as the number of centroids. Each \mathbf{x}_j point is assigned to the nearest cluster with centroid \mathbf{c}_k . New centroids are determined using the cluster means. The sum of squares distance is calculated at each iteration. During clustering, if there are no more changes in the members of the clusters the algorithm is said to have converged and the algorithm outputs the best centroids $(\mathbf{c}_1, \dots, \mathbf{c}_K)$, otherwise it will continue searching for a better solution.

Algorithm 15 The k-means algorithm

```

Input: Dataset  $\mathbf{X}$ , Number of clusters  $K$ 
Randomly initialise  $k$  centroids
while Criteria met (Convergence) do
  for Each data point  $\mathbf{x}_i$  do
    Find the nearest centroid
    Assign point  $\mathbf{x}_i$  to the nearest cluster
  end for
  for Each cluster,  $\mathbf{C}_k$  do
    new centroid = mean of all points assigned to that cluster
  end for
end while

```

K-means has several shortcomings. It is sensitive to the selection of the initial cluster centres, different initial values of k lead to different results, and it is easily affected by outliers (Yadav and Sharma, 2013). It is, however, computationally simple and a good first choice clustering algorithm.

3.5 Existing truck and drone optimisation strategies

Different types of heuristics have already been applied to optimise truck and drone systems. Murray and Chu (2015) introduced a new integer programming (IP) model to solve the FSTSP with reasonable size instances and applied heuristics to

solve large instances. The heuristics solved the large instances faster and obtained good quality solutions. Then Agatz *et al.* (2018) solved their TSP-D using the cluster first, route second procedure developed by Beasley (1983). That is, the truck route is solved first as a TSP. Drone customers are then added second, using greedy and exact partition heuristics.

Ha *et al.* (2018) introduced GRASP to solve large instances of the TSPD and benchmarked the results with those of the FSTSP. Raj and Murray (2021) modelled their single truck and multiple drone system using a mixed-integer linear programming formulation for small-scale problems. An ant-pair colony system heuristic is proposed for problems size of up to 100.

Kitjacharoenchai *et al.* (2019) applied an adaptive insertion heuristic (ADI) to solve the mTSPD. First, three heuristics were applied individually, to solve the multi-travelling salesman problem (mTSP), namely a genetic algorithm, k-means, and random clustering. Secondly, the drone nodes were then inserted into the mTSP using ADI. It gradually switched nodes and evaluated the objective function for each configuration. The results of the heuristics were benchmarked against mixed-integer programming using a small-scale problem.

Yurek and Ozmutlu (2018) used a decomposition-based iterative optimisation algorithm to solve a TSPD. The time to solve small instances of TSPD using a decomposition-based iterative were compared with MILP results obtained from Murray and Chu (2015), and Agatz *et al.* (2018). The model was run for an hour to find the exact solution. If the solution was not found, the model was stopped. The results of the algorithm were compared and showed that as the problem size increased, Murray and Chu (2015) and Agatz *et al.* (2018) could not find the optimum solution in an hour. The decomposition-based iterative method outperformed the exact method of Murray and Chu (2015) and Agatz *et al.* (2018) by finding the optimum solution quicker. The performance was due to decomposing the exact algorithm in two stages. First, it solved the TSPD using the MILP, and secondly, IP was applied (Yurek and Ozmutlu, 2018). TSP-D was solved with a branch-and-bound method. Each node of the branch-and-bound tree corresponds to a potential order to deliver a subset of packages. Solution quality and computational times of the heuristic approach with additional variants were also evaluated (Poikonen *et al.*, 2019).

Dinh *et al.* (2021) applied hybrid ant colony optimisation (HACO) to minimise the truck and drone delivery completion time of the PDSTSP. They combined other problem-tailored components with dynamic programming to efficiently solve the PDSTSP. The HACO was compared with other state-of-the-art algorithms, and it was found that it performed better on solution quality and running time. Out of 90 instances considered, they obtained 23 new best-known solutions. Finally, Ponza (2016) followed a different approach and solved the FSTSP using a metaheuristic

algorithm known as simulation annealing (SA) . The results were also compared with those of MILP.

The solution strategies commonly used in literature are summarised in Tables 3.1 and 3.2. The literature is compared with regard to the type of problem, solution scalability (according to number of drones and trucks), the problem sizes solved and whether drone and truck interception were allowed at any point on the map.

Table 3.1: Existing literature on truck-drone delivery systems

Author	Type	Problem scalability	Solution strategy	Nodes	Interception
Agatz <i>et al.</i> (2018)	FSTSP	m-Drones 1-Truck	Heuristics	10 20 100	No
Murray and Chu (2015)	FSTSP	1-Drone 1-Truck	Heuristics	10 20	No
Ulmer and Thomas (2018)	FSTSP	1-Drone 1-Truck	GVNS	10 50 100 150 200	No
Ha <i>et al.</i> (2018)	FSTSP	1-Drone 1-Truck	Heuristics	10 50 100	No
Ponza (2016)	FSTSP	1-Drone 1-Truck	SA	50 100 150 250	No
Ham (2018)	FSTSP	m-Drones n-Trucks	Constraint programming	20 50 100	No
Bouman <i>et al.</i> (2018)	VRPD	n-Drones m-Trucks	Dynamic programming	Small problem size	No
Daknama and Kraus (2017)	FSTSP	1-Drone 1-Truck	Dynamic programming	None	No
bin Othman <i>et al.</i> (2017)	FSTSP	1-Drone 1-Truck	2 Approximation min-cost bi-portal	None	No
Di Puglia Pugliese and Guerriero (2017)	FSTSP	n-Drones m-Trucks	IP formulation	None	No
Kitjachoenchai <i>et al.</i> (2019)	FSTSP	m-Drones n-Trucks	ADI GA	Variety Max (100)	No
Yurek and Ozmutlu (2018)	FSTSP	1-Drone 1-Truck	Heuristics	10 11 12 20	No
Marinelli <i>et al.</i> (2018)	FSTSP	1-Drone 1-Truck	Heuristics	10 20 50	No
Ferrandez <i>et al.</i> (2016)	FSTSP	m-Drones 1-Truck	GA K-means	50 250	No
Wang and Sheu (2019)	FSTSP	m-Drones n-Trucks	Heuristics	20	No

Table 3.2: Existing literature on truck-drone delivery systems

Author	Type	Problem scalability	Solution strategy	Nodes	Interception
Wang <i>et al.</i> (2017)	VRPD	m-Drones n-Trucks	None	10 20 100	No
Dorling <i>et al.</i> (2016)	VRPD	m-Drones 1-Truck	SA	6 7 8	No
Carlsson and Song (2018)	VRPD	1-Drone 1-Truck	None	None	No
Raj and Murray (2020, 2021)	VRPD	m-Drones n-Trucks	Heuristics	100	No
Schermer <i>et al.</i> (2018); Schermer (2019); Schermer <i>et al.</i> (2020)	VRPD	m-Drones n-Trucks	Heuristics	100	No
This dissertation	FSTSP	m-Drones n-Trucks	Metaheuristics	10 20 50 100 250 500	Yes

From the summary of the literature it is clear that no other authors have investigated truck and drone interception at any point on the map. Another interesting finding was the size of problems typically solved in literature. Only small problems could be solved optimally, highlighting the value of using a metaheuristic approach. For example, Ham (2018) applied an exact method to solve the multiple truck and multiple drone problem of 100 nodes. None of the instances solved in the experiment could arrive at an optimum solution within a set time.

3.6 Summary

In this chapter different solution strategies were explored to solve a TSP, TSP-D and VRP-D. Tables 3.1 and 3.2 summarised the existing research in the field. It can be seen that the research proposed in this dissertation differs from the existing literature. The proposed algorithm allows a drone and truck interception while the truck is in motion, which is intended to minimise the waiting time.

Exact algorithms guarantee the optimum solution; however, they require extensive computational time to solve a large TSP-D. Heuristics and metaheuristics do not guarantee finding the optimum solution, but they require less computational time. The shortcoming of heuristics is that they often become stuck in local min-

ima, whereas metaheuristics will have a mechanism to move the algorithm out of the local minima and explore other parts of the search space. Trajectory methods and population-based algorithms were investigated and ACO was selected as the basis for the algorithm development in this dissertation due to it being the most popular method for solving graph-based problems such as machine scheduling and VRPs.

In the next chapter, the problem formulation is provided for the single truck and drone scheduling problem with interception. An ACS-based algorithm is proposed for solving the defined problem.

Chapter 4

Solving the single truck and drone scheduling problem with interception

This chapter presents the **single** truck and drone scheduling problem with interception. First, Section 4.1 describes the problem in detail; the model assumptions that simplify the problem are stated, and truck and drone interception is explained. The mathematical formulation for the problem is presented in Section 4.2. Thirdly, Section 4.3 presents the ACS-based truck and drone scheduling algorithms for scheduling the deliveries. The experimental setup and results are described in 4.4. Finally, Section 4.5 summarises Chapter 4.

4.1 Problem description

The problem is based on a truck and drone making deliveries to various customers. The objective is to minimise the total delivery time of the system. A drone is launched from the customer's location. The truck and drone leave the depot together to make deliveries. Once the drone has made a delivery, it flies back to intercept the truck while it is in motion. The vehicle interception allows the drone to fly directly from its delivery node to the truck instead of flying the longer distance to the next truck delivery node, decreasing the distance travelled by the drone versus a system which does not allow for interception. The drone is then carried on the truck to the next customer's location. While the truck is offloading parcels, a drone is launched to the next customer or remains on the truck. All vehicles return to the depot together or independently once all customer orders are fulfilled. Figure 4.1 shows the problem just described. It shows a truck and drone following the paths denoted by solid and dashed arrows respectively. The

CHAPTER 4. SOLVING THE SINGLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION

49

truck and drone leave the depot (denoted by a grey rectangle), to make deliveries to customers (denoted by numbered circles). First, the drone leaves the depot and visits customer node 3 while the truck visits node 1. Before the truck reaches customer node 1, the drone flies from customer node 3 to intercept the truck. Solid black circles indicate the interception points. The process ends at the depot after the truck and drone have visited nodes 1, 4, 7, and 8; and 3, 2, 6, and 5, respectively.

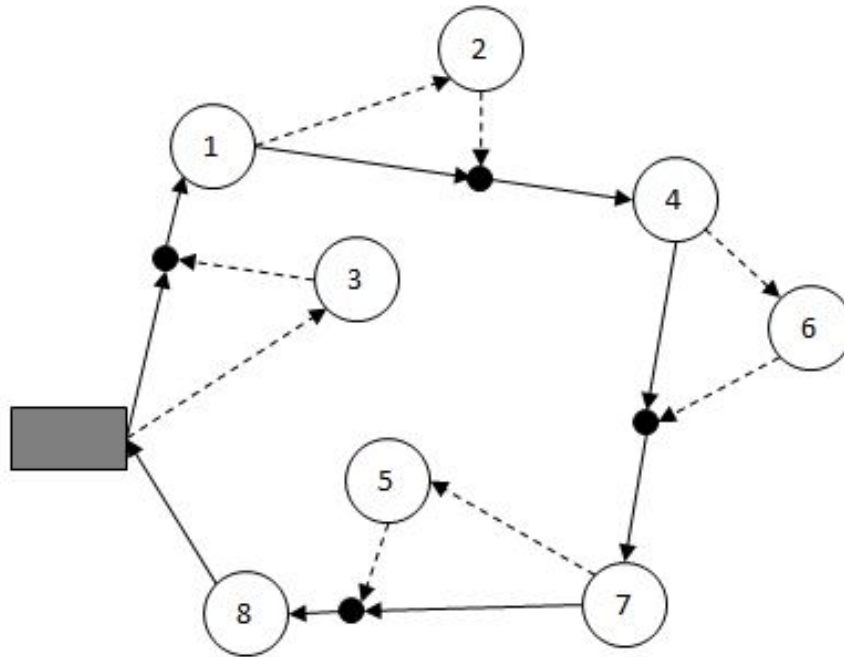


Figure 4.1: Truck and drone system with interception points (black circles), truck route (solid lines) and drone route (dotted lines)

Several forbidden moves are defined for the system. One disallowed move is launching a drone at an intermediate location or while the truck is in motion. From a practical perspective, launching a drone while the truck is in motion is difficult. Typically, deliveries would be conducted by a two person delivery team. While the truck driver is completing a delivery at a customer location, the drone operator has the ideal opportunity to replace batteries and launch the drone off of the stationary truck. Schematic A in Figure 4.2 marks this move with a red X and the correct interception move with a green tick.

The second move that is not allowed is the interception at the intermediate location after the truck has left the first customer's location to deliver to the next

CHAPTER 4. SOLVING THE SINGLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION 50

customer. When this scenario occurs, the truck changes direction, resulting in more mathematical complexity, which can be avoided by disallowing the move. A red X illustrates this move in Schematic B in Figure 4.2.

The following assumptions are also made to simplify the truck and drone scheduling model:

- A drone can carry one parcel at a time for deliveries.
- A drone has unlimited battery power.
- All nodes are eligible to receive packages from all vehicles.
- Drones can only be launched at the customer's location or depot, and other launching positions are not allowed.
- The truck may not visit multiple customers while a drone is delivering a parcel.
- A drone can visit non-customer nodes (i.e. interception locations).
- The drone and truck travel at a constant speed (v_d for the drone, and v_t for the truck).
- The delay during truck and drone interception is negligible.
- The drone can find the truck's location at all times.
- When the interception cannot happen at the intermediate location, the truck waits for the drone at the next customer's location.

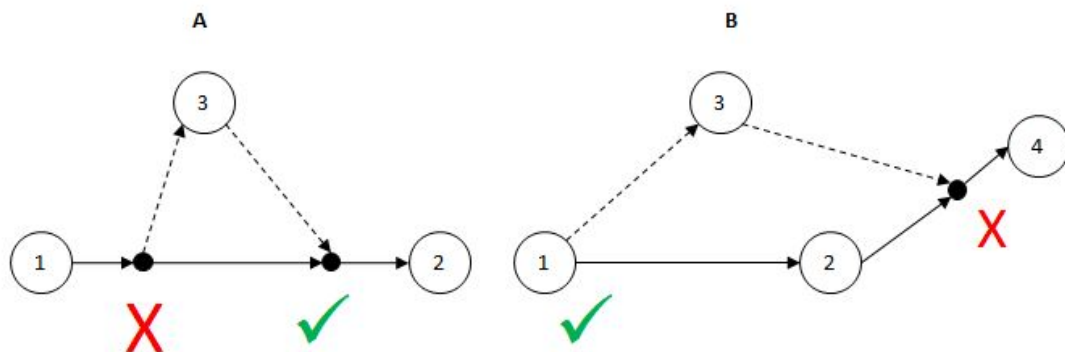


Figure 4.2: Schematic A: Launching of drone in the arc and Schematic B: Interception after truck leaves customer

The objective of the truck and drone model with interception is to minimise the truck travelling time and waiting time in the ‘last mile’ delivery system. The main benefit of a drone returning to the truck (at the interception point) after deliveries is that the drone traveling time will be reduced and the truck does not have to wait for the drone at the customers’ location. The time savings will reduce the total delivery time.

4.2 Mathematical formulations

Figure 4.3 illustrates the movement of the truck and drone over one subtour. A truck and drone start at node i , and a truck travels to customer k , while a drone is launched from a truck position \mathbf{c}^i , at the customer’s location to deliver parcels to customer j . The drone takes $(\tau^j - s_i)$ time to travel to the customer, where s_i is the time when the truck arrives at customer i for delivery and τ^j is the time the drone arrives at customer j . When a drone arrives at customer j , a truck would have travelled to p^j , marked with a star on the schematic. A drone takes δ time to offload the parcel, whereafter, the drone takes TI^j time to intercept with the truck. The time and position at interception are τI^j and pI^j , respectively. The time it takes the truck to travel from customer i to customer k is $(s_k - s_i)$ or T_{ik} . When a truck reaches customer i , it spends w time at node i to complete the delivery and launch the drone. The velocity of the truck travelling from customer i to customer k is denoted by \mathbf{v}_t^{ik} and the speed is denoted by $|\mathbf{v}_t^{ik}|$ or v_t . The drone speed travelling from customer i to j is denoted by v_d .

A standard mathematical equation for determining two object interceptions was applied to calculate the truck and drone interception time and position. The equation is shown in Equation (4.2) and is used to calculate the interception time of a drone travelling from customer location j to the truck’s location pI^j , where \mathbf{c}^j is a 2-dimensional vector of the x and y coordinates of customer j , $d(\mathbf{c}^i, \mathbf{c}^j)$ is the distance between node j and the position of a truck at the beginning of a drone launch at node i , respectively.

The time from node j to the interception point pI^j , TI^j , is calculated as:

$$TI^j = \frac{-2(\mathbf{c}^j - \mathbf{c}^i) \cdot (\mathbf{v}_t^{ik}) \pm \sqrt{(2((\mathbf{c}^j - \mathbf{c}^i) \cdot \mathbf{v}_t^{ik})^2 - (4(v_d^2 - v_t^2)(-d(\mathbf{c}^i, \mathbf{c}^j))^2))}}{2(v_d^2 - v_t^2)} \quad (4.1)$$

where $\mathbf{c}^j - \mathbf{c}^i$ is a 2-dimensional vector of the difference between the x coordinates of nodes j and i and the difference between the y coordinates of nodes j and i .

The interception position can then be calculated as:

$$\mathbf{pI}^j = \mathbf{c}^i + (\mathbf{v}_t^{ik}) \cdot (TI^j + \delta + (\tau^j - s_i)) \quad (4.2)$$

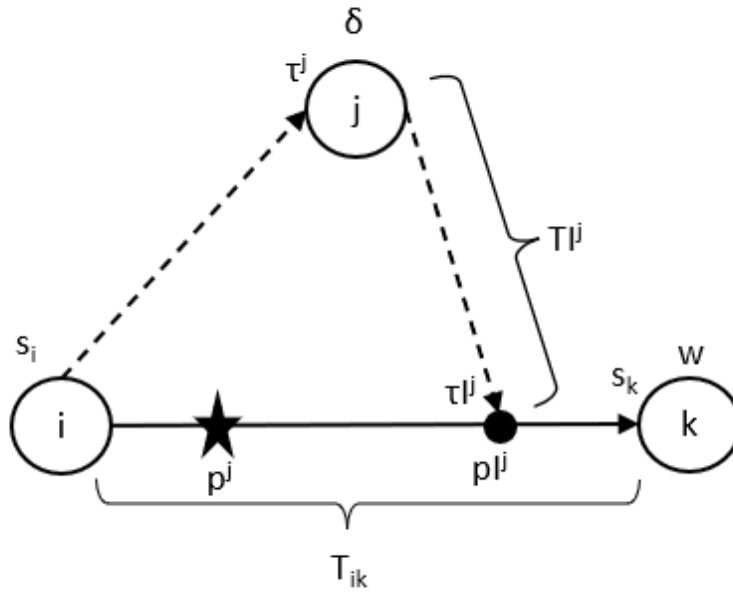


Figure 4.3: Truck and drone subtour

Two scenarios can occur. The drone intercepts a truck before it reaches customer k or the drone flies straight to customer k 's location. A drone only flies straight to the customer's location when it cannot intercept the truck before the truck reaches node k . In this situation, a truck arrives at the customer's location and waits for the drone. The total truck travelling time from customer location i to k is T_{ik} , the total truck travelling time from customer i to the interception point, p^{I^j} , is $(\tau^{I^j} - s_i)$, where τ^{I^j} is the time at interception of the truck and drone after the drone delivery at customer j . This travelling time is equivalent to the drone travelling time from i to interception point, p^{I^j} .

The waiting time only occurs if the truck arrives at the customer location k without intercepting the drone, and it has to wait for the drone's arrival. Waiting time, W is calculated as total drone travelling time from customer i to k , τ_{ijk} , minus truck travelling time from customer i to k , T_{ik} and is represented by the equation $W = \tau_{ijk} - T_{ik}$, which is valid only when $\tau_{ijk} > T_{ik}$. The total time to complete the deliveries of nodes i and j , and travel to k is $T_{ijk} = \max(T_{ik}, \tau_{ijk})$. The last vehicle to arrive at customer k determines T_{ijk} .

Given the above information, two mathematical formulations of the truck-drone system is developed. The first model assumes that a drone delivery has to be scheduled in between each two truck deliveries and is presented below, where the input parameters are:

CHAPTER 4. SOLVING THE SINGLE TRUCK AND DRONE SCHEDULING
PROBLEM WITH INTERCEPTION 53

- n is the number of nodes to be serviced ($n - 1$ customers plus a depot: node 1).
- T_{ijk} denotes the time required for servicing nodes i and j and traveling to k if the truck travels between nodes i and k and the drone services node j in between.

The decision variables are:

- u_i which is used to eliminate subtours.

$$x_{ijk} = \begin{cases} 1 & \text{if the truck services node } i \text{ before } k \text{ while the drone services } j \text{ in between} \\ 0 & \text{Otherwise} \end{cases}$$

$$b_j = \begin{cases} 1 & \text{if the customer } j \text{ is serviced by the drone} \\ 0 & \text{if customer } j \text{ is serviced by the truck} \end{cases}$$

$$z_{ik} = \begin{cases} 1 & \text{if the truck travels between nodes } i \text{ and } k \\ 0 & \text{Otherwise} \end{cases}$$

$$\min \sum_{i=1}^n \sum_{k=1}^n \sum_{j=1}^n x_{ijk} T_{ijk} \quad (4.3)$$

Subject to:

$$z_{ii} = 0 \quad \forall i \in \{1, \dots, n\} \quad (4.4)$$

$$x_{iii} = 0 \quad \forall i \in \{1, \dots, n\} \quad (4.5)$$

$$z_{ik} = \sum_{j=1}^n x_{ijk} \quad \forall i, k \in \{1, \dots, n\} \quad (4.6)$$

$$b_j = \sum_{i=1}^n \sum_{k=1}^n x_{ijk} \quad \forall j \in \{1, \dots, n\} \quad (4.7)$$

$$\sum_{i=1}^n z_{iq} - \sum_{k=1}^n z_{qk} = 0 \quad \forall q \in \{1, \dots, n\} \quad (4.8)$$

$$\sum_{k=1}^n z_{ik} = 1 - b_i \quad \forall i \in \{1, \dots, n\} \quad (4.9)$$

$$\sum_{i=1}^n z_{ik} = 1 - b_k \quad \forall k \in \{1, \dots, n\} \quad (4.10)$$

$$\sum_{j=1}^n x_{ijk} \leq 1 \quad \forall i, k \in \{1, \dots, n\} \quad (4.11)$$

$$u_i + z_{ij} \leq u_j + (n-1)(1 - z_{ij}) \quad \forall i, j, j \neq 1 \in \{1, \dots, n\} \quad (4.12)$$

$$u_1 = 0 \quad (4.13)$$

$$x_{ijk}, b_i, z_{ik} \in \{0, 1\} \quad \forall i, j, k \in \{1, \dots, n\} \quad (4.14)$$

- The objective function (4.3) minimises the total time that the drone and truck travel.
- Constraints (4.4) and (4.5) simplify the model by ensuring that the truck and drone cannot travel from the current node back to a previously visited node.
- Constraints (4.6) links z_{ik} and x_{ijk} .
- Constraints (4.7) links b_j and x_{ijk} .
- Constraints (4.8) ensure that if a delivery is conducted at a customer location, the truck or drone leaves that same customer.
- Constraints (4.9) and (4.10) ensure that each customer is arrived at from exactly one other customer (by truck), and that from each customer there is a departure to exactly one other customer (by truck). If node j is serviced by a drone, then no trucks enter or exit node j .

- Constraints (4.11) ensures that only one drone delivery can occur between two truck deliveries.
- Constraints (4.12) and (4.13) eliminate any subtours.
- Constraints (4.14) restricts the decision variables to binary values.

The second model removes the assumption that a drone needs to be scheduled between each two truck deliveries and is presented below, where the input parameters are:

- n is the number of nodes to be serviced ($n - 1$ customers plus a depot: node 1).
- T_{ijk} denotes the time required for servicing nodes i and j and traveling to k if the truck travels between nodes i and k and the drone services node j in between.
- T_{ik} denotes the time required for servicing nodes i and travelling to k if the truck travels between nodes i and k and there is no drone delivery in between.

The decision variables are:

- u_i which is used to eliminate subtours.

$$x_{ijk} = \begin{cases} 1 & \text{if the truck services node } i \text{ before } k \text{ while the drone services } j \text{ in between} \\ 0 & \text{Otherwise} \end{cases}$$

$$b_j = \begin{cases} 1 & \text{if the customer } j \text{ is serviced by the drone} \\ 0 & \text{if customer } j \text{ is serviced by the truck} \end{cases}$$

$$z_{ik} = \begin{cases} 1 & \text{if the truck travels between nodes } i \text{ and } k \\ 0 & \text{Otherwise} \end{cases}$$

$$\min \sum_{i=1}^n \sum_{k=1}^n \sum_{j=1}^n x_{ijk} T_{ijk} + \sum_{i=1}^n \sum_{k=1}^n z_{ik} T_{ik} (1 - \sum_{j=1}^n x_{ijk}) - w \quad (4.15)$$

Subject to:

$$z_{ii} = 0 \quad \forall i \in \{1, \dots, n\} \quad (4.16)$$

$$x_{iii}, x_{iij}, x_{iji}, x_{ijj} = 0 \quad \forall i, j \in \{1, \dots, n\} \quad (4.17)$$

$$\left(\sum_{j=1}^n x_{ijk} = 1 \right) \Rightarrow (z_{ik} = 1) \quad \forall i, k \in \{1, \dots, n\} \quad (4.18)$$

$$b_j = \sum_{i=1}^n \sum_{k=1}^n x_{ijk} \quad \forall j \in \{1, \dots, n\} \quad (4.19)$$

$$\sum_{i=1}^n z_{iq} - \sum_{k=1}^n z_{qk} = 0 \quad \forall q \in \{1, \dots, n\} \quad (4.20)$$

$$(b_i = 0) \Rightarrow \left(\sum_{k=1}^n z_{ik} = 1 \right) \quad \forall i \in \{1, \dots, n\} \quad (4.21)$$

$$(b_k = 0) \Rightarrow \left(\sum_{i=1}^n z_{ik} = 1 \right) \quad \forall k \in \{1, \dots, n\} \quad (4.22)$$

$$\sum_{j=1}^n x_{ijk} \leq 1 \quad \forall i, k \in \{1, \dots, n\} \quad (4.23)$$

$$u_i + z_{ij} \leq u_j + (n-1)(1 - z_{ij}) \quad \forall i, j, j \neq 1 \in \{1, \dots, n\} \quad (4.24)$$

$$u_1 = 0 \quad (4.25)$$

$$x_{ijk}, b_i, z_{ik} \in \{0, 1\} \quad \forall i, j, k \in \{1, \dots, n\} \quad (4.26)$$

- The objective function (4.15) minimises the total time that the drone and truck travel, which now includes the travel time of all truck deliveries which do not include a drone release and intercept.
- Constraints (4.16) and (4.17) again simplify the model by ensuring that the truck and drone cannot travel from the current node back to a previously visited node.
- Constraints (4.18) links z_{ik} and x_{ijk} . The notation $(\sum_{j=1}^n x_{ijk} = 1) \Rightarrow (z_{ik} = 1)$ indicates that $z_{ik} = 1$ if $\sum_{j=1}^n x_{ijk} = 1$.
- Constraints (4.19) links b_j and x_{ijk} .
- Constraints (4.20) ensures that if a delivery is conducted at a customer location, the truck or drone leaves that same customer.
- Constraints (4.21) and (4.22) ensure that each customer is arrived at from exactly one other customer (by truck), and that from each customer there is

a departure to exactly one other customer (by truck). If node j is serviced by a drone, then no trucks enter or exit node j .

- Constraints (4.23) ensures that only one drone delivery can occur between two truck deliveries.
- Constraints (4.24) and (4.25) eliminate any subtours.
- Constraints (4.26) restricts the decision variables to binary values.

4.3 The ACS-based scheduling algorithm for the single truck and drone scheduling problem

Two versions of the ACS-based scheduling algorithm are developed, each using a different route construction type to either require drone deliveries between each two truck deliveries or not. The first algorithm is proposed in Section 4.3.1 and the second in Section 4.3.2.

4.3.1 Route construction type 1

In the first method of construction (referred to as ACS-DT1), all vehicles are set to start at the depot. The truck and drone are assigned to nodes one after the other. The truck is assigned first to the nodes using the ACS transition rule, Equation (3.19), then the drone is assigned to available nodes. Every time a vehicle is assigned a node, the node becomes unavailable to another vehicle. The truck and drone return to the depot once all customers are serviced. Figure 4.4 illustrates an example of the procedure of assigning vehicles to nodes. A shaded rectangle denotes the depot. Numbered circles denote available customer nodes. Dashed circles denote serviced customer nodes. Dotted arrows are the edges available from a node. **Thick black arrows** are the paths chosen from one node to another. The figure is divided into twelve sub-figures labelled A to L .

- Sub-figure A shows the search for a suitable node for the truck. There are five nodes available from the depot, and they are numbered 1 to 5.
- Sub-figure B shows node 3 is chosen using the ACS transition rule (Equation (3.19)) and the node becomes $3T$ (indicating that node 3 will be serviced by the truck).
- Sub-figure C shows a search for the drone node from the available nodes, 1, 2, 4 and 5.

- Sub-figure D shows that node 4 is chosen using an ACS transition rule and becomes $4D$ (indicating that node 4 will be serviced by the drone).
- Sub-figure E shows 3 options for truck nodes available from node 4: nodes 1, 2, and 5.
- Sub-figure F shows that node 2 is chosen using an ACS transition rule and becomes $2T$.
- Sub-figure G shows 2 drone nodes available, nodes 1 and 5.
- Sub-figure H shows node 1 is chosen as a drone node, which becomes $1D$.
- Sub-figure I shows that the only available node for the truck is node 5.
- Sub-figure J shows that node 5 is assigned as a truck node and becomes $5T$.
- Sub-figure K shows that the only option left is the depot node.
- Sub-figure L shows node 5 connected with a solid line to the depot. All vehicles' journeys end at the depot.

The final assigned and scheduled path for the truck and drone is ($Depot-3T-4D-2T-1D-5T-depot$), where the truck nodes are 3, 2, and 5, and the drone nodes are 4 and 1. The truck and drone interception would be between nodes $4D$ and $2T$; and nodes $1D$ and $5T$. Figure 4.5 explains the ACS-based truck and drone scheduling algorithm by means of a flowchart.

All ants (n_s) start at the depot node. Nodes are assigned as truck and drone nodes using equation (3.19) and equation (3.20). First, an ant representing a truck visits the first node; once it arrives at the node, the node becomes unavailable. Secondly, an ant representing a drone travels to the next node of available candidate nodes, then this node is similarly taken out of the candidate list. The ants continue to visit nodes until all the nodes have been visited alternately. As mentioned above, Figure 4.4 shows an example of the assignment of vehicles to nodes. The pheromone is updated on all the edges visited by the ants using the local pheromone equation (3.23). After nodes are visited and the ants, one representing a truck and the other a drone, have returned to the depot, a drone path with interception is constructed into the network using equation (4.1) and equation (4.2). The total delivery and waiting times are computed per completed journey. The entire process repeats itself until all ants in the system have travelled the network and generated solutions. After all ants have produced solutions, the best performing solution is used to update the pheromone of the best path using equation (3.21). The overall process is repeated several times until the maximum number of iterations has been reached. The best solution composed of total delivery time, waiting time, truck travel distance and drone travel distance is then computed.

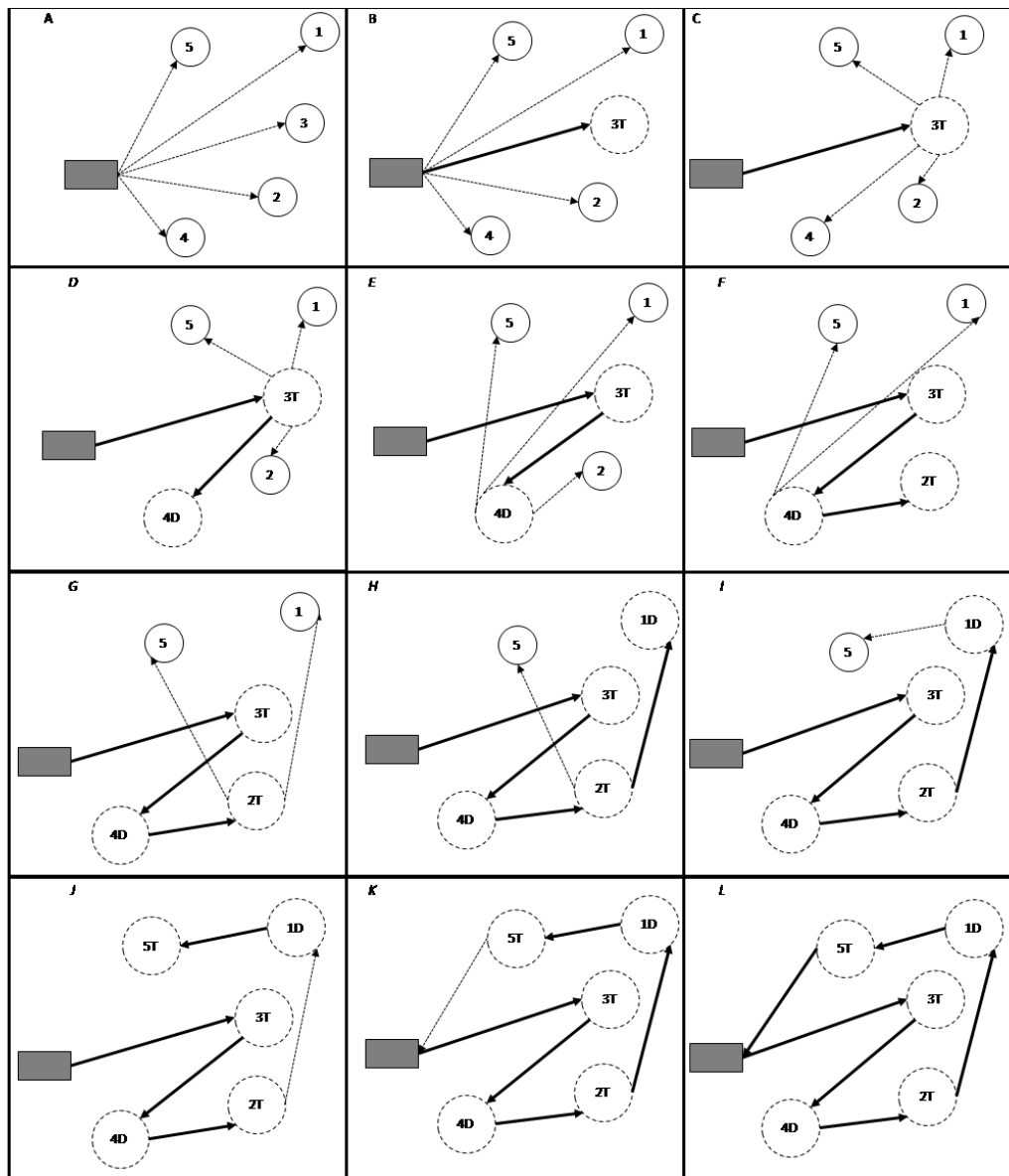


Figure 4.4: Truck and drone assignment and scheduling method

4.3.2 Route construction type 2

This section presents the second method of truck and drone route construction. In this dissertation, the method is referred to as ACS-DT2. The ants perform the assignment and sequencing of vehicles to nodes. All vehicles start at the depot, and ants decide using Equation (3.19) which nodes are assigned to a truck or drone. A constraint is set so that drone nodes must not exceed truck nodes, and a

CHAPTER 4. SOLVING THE SINGLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION

60

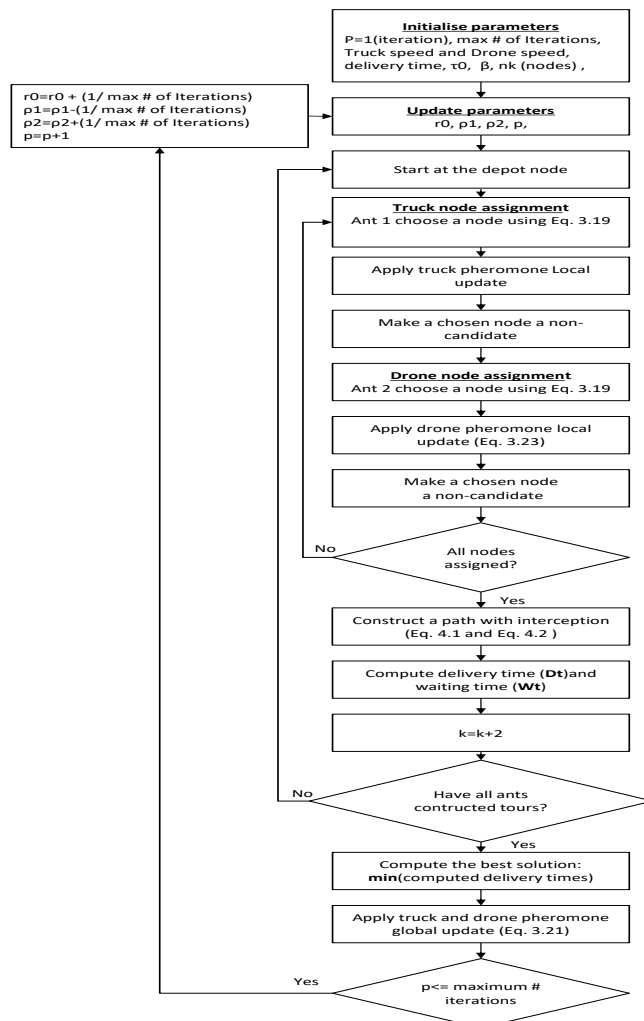


Figure 4.5: ACO application process flowchart

mechanism is implemented to avoid infeasible solutions. Infeasibility occurs when there are successive drone nodes assigned between two truck nodes. A mechanism is implemented to check if the current assignment is a truck or drone node. If it is a truck node, then the next node can either be assigned to a truck or drone, but if it is a drone node, then the next assignment must be a truck.

4.4 Empirical evaluation of the ACS single truck and drone scheduling algorithms

The ACS single truck and drone scheduling algorithm was developed in MATLAB[®]. The experiments were run on a Dell computer with the following hardware configuration: Processor with Intel(R) Core i5-8365U CPU and 8.00GB of RAM.

The ACS algorithms are evaluated on 10 data sets adopted from Bouman *et al.* (2015). Small, middle and large data sizes were chosen arbitrarily from all available datasets in Bouman *et al.* (2015). The data sets chosen have customers located uniformly on the x-y Cartesian plane. The data sets are shown in Table 4.1.

Table 4.1: TSP data sets (Bouman *et al.* (2015))

TSP data sets	Nodes
Uniform-51-n10	10
Uniform-52-n10	10
Uniform-53-n10	10
Uniform-61-n20	20
Uniform-62-n20	20
Uniform-63-n20	20
Uniform-71-n50	50
Uniform-72-n50	50
Uniform-73-n50	50
Uniform-91-n100	100
Uniform-92-n100	100
Uniform-93-n100	100
Uniform-1-n250	250
Uniform-2-n250	250
Uniform-5-n500	500
Uniform-6-n500	500

The parameters are initialised as follows:

- The truck and drone are treated as different ants walking through the same network searching for the best path and they cannot sense each other's pheromone. In mathematical terms, the pheromone matrix of a truck and drone are separated but initiated with the same values and are represented by $\tau_{ik}^{truck}(t)$ and $\tau_{ij}^{drone}(t)$ where $\tau_{ik}^{truck}(t)$ is the pheromone between customer i and k observed by the truck at iteration t , $\tau_{ij}^{drone}(t)$ is the pheromone be-

tween customer i and j observed by the drone at iteration t , and t is the iteration number.

- Evaporation rate ρ_1 of Equation (3.21) is linearly decreased from 1 to 0. The evaporation rate ρ_2 of Equation (3.23) is increased from 0 to 1 over the maximum number of iterations to promote greater exploration at the start of the search and more exploitation towards the end of the search.
- A nearest neighbour heuristic is used to set τ_0 . The nearest neighbour heuristic searches for the node closest to the current node and selects it as the next node. The process continues until all nodes are visited and the full path (L) is determined. A small positive value constant is calculated as $\tau_0 = (nL^{-1})$.
- The user-specified parameter, r_0 , is used to control exploration and exploitation. It was linearly increased from 0 to 1 over the maximum number of iterations.
- Other parameters initialised were number of ants (n_k), maximum iterations, truck speed (v_t), drone speed (v_d), truck delivery time (w) and drone delivery time (δ).
- The best β of Equation (3.20) is obtained through an extensive tuning process. The tuning process of β is described below.

F-Race is used to compare the results of different β values. F-Race is a “racing algorithm for the task of automatic algorithm configuration. The strategy is based on a statistical approach for selecting the best configuration from a set of candidate configurations under stochastic evaluations” (Birattari *et al.*, 2009). It is the fastest way to fine-tune the control parameters of the ACS algorithm. The F-Race uses statistics to evaluate the difference in performance between different control parameter configurations. The parameter tuning was done using the data sets in Table 4.2 to determine the optimum β of the ACS algorithm where β s were tested in the range: $\beta = \{1, 2, 3, 4, 5\}$.

Table 4.2: TSP data sets used for tuning ACS parameters (Bouman *et al.*, 2015)

TSP data sets	Nodes
Uniform-51-n10	10
Uniform-52-n10	10
Uniform-61-n20	20
Uniform-62-n20	20
Uniform-71-n50	50

In summary, the other algorithm control and problem parameters are set as indicated in Table 4.3 and Table 4.4. The notation $a \rightarrow b$ indicates that the parameter increased or decreased linearly between a and b over the maximum iteration number. These adaptive parameter settings were selected to obtain the desired search balance between more exploration at the start of the search process and more exploitation at the end of the search process.

Table 4.3: ACS parameters

ACS Parameters	Values
ρ_1	$1 \rightarrow 0$
ρ_2	$0 \rightarrow 1$
r_0	$0 \rightarrow 1$
Max. no. of iterations	3000

Table 4.4: Problem parameters

Problem Parameters	Values
v_d	20
v_t	10
No. of Trucks	1
No. of Drones	1
w (Truck delivery time)	0.1
δ (Drone delivery time)	0.1

Different β s are tested over 30 independent simulation runs and Mann-Whitney U-tests at 5% significance are used to compare the performance of different β s. A win is recorded if the first setting statistically significantly outperforms the second setting; if no statistical difference is observed, a draw is recorded. If the second parameter setting outperforms the first setting, a loss is recorded for the first parameter setting. The total number of wins, draws and losses are recorded for all comparisons of the parameter combination under evaluation. The results of the β s performance are shown in Table 4.5. It is evident that β equal to 2 outperformed other β settings.

The ACS algorithms (both ACS-DT1 and ACS-DT2) are evaluated over 30 independent runs, and the results are summarised in Tables 4.6 and 4.7. The mean and standard deviation of the total delivery time (completion time), the mean and standard deviation of total truck distance, the mean of total waiting time, and algorithm running time (CPU time) are presented for each problem for

Table 4.5: F-Race results

β	Win	Draw	Lose
1	4	8	3
2	5	8	2
3	1	7	4
4	0	5	3
5	0	5	3

the ACS-DT1 and ACS-DT2 algorithms. The results are benchmarked against the TSP solutions of Bouman *et al.* (2015), showing that data with fewer nodes produced more consistent results than those with more nodes.

The ACS-DT1 results are also represented graphically per data set. The graphs show the truck and drone making deliveries, the search for the best solution over time and diversity graphs showing different routes ants searched for solutions in the TSP network. Sub-figure a of Figure 4.6 and Figure 4.7 show a truck and drone route.

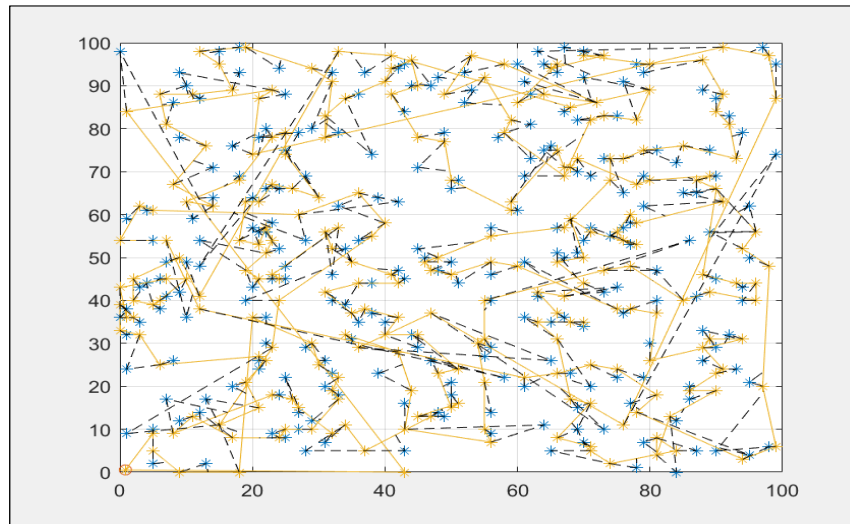
Sub-figure b of Figure 4.6 and Figure 4.7 show the best solution found per iteration. The figures further show that as the ants explore for solutions, they find better solutions with each iteration.

Sub-figure c Figure 4.6 and Figure 4.7 show the diversity of the ant colony over time. Diversity, D , is calculated as:

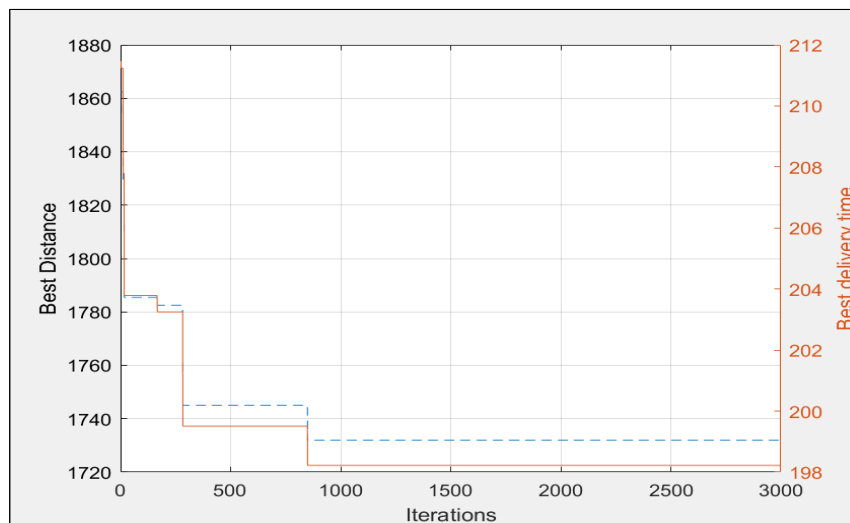
$$D = \frac{1}{n_s} \sum_{i=1}^{n_s} \sqrt{\sum_{j=1}^{n_x} (x_{ij}(t) - \bar{x}_j(t))^2} \quad (4.27)$$

where n_s is the number of ants, n_x is the number of variables/dimensions of each ant, $x_{ij}(t)$ is the position of the j^{th} dimension of the i^{th} ant at a time t and $\bar{x}(t)$ is the mean of the j^{th} dimension of all ants in the colony at time t .

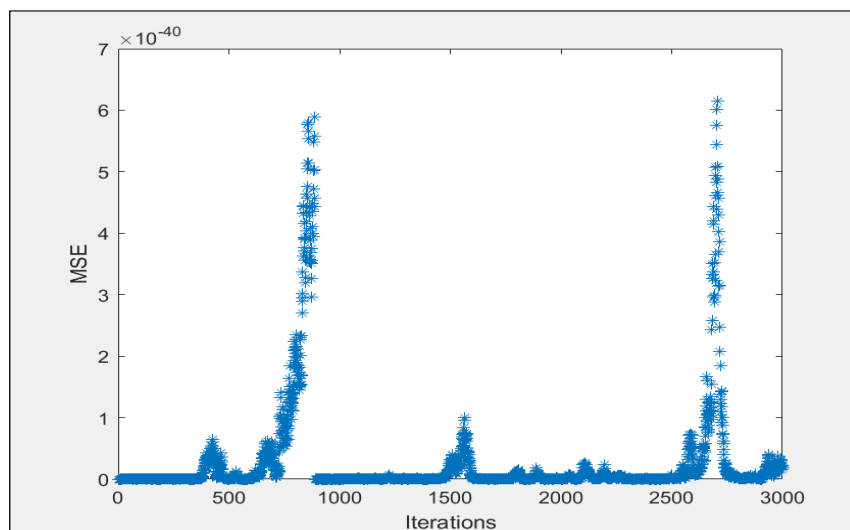
It can be seen that the diversity of the colony is high at the start of the optimisation process, indicating a greater focus on exploration of the search space and that it decreases toward the end of the search process when exploitation of existing solutions is more critical. On larger data sets, the convergence happens later on since the ants have more nodes to explore and this scenario results in more exploration even after 1000 iterations. These results are clearly seen in sub-figures b and c of Figure 4.6 and Figure 4.7. The remaining graphs for the other data sets can be found in Appendix 7.3.



(a) Best path

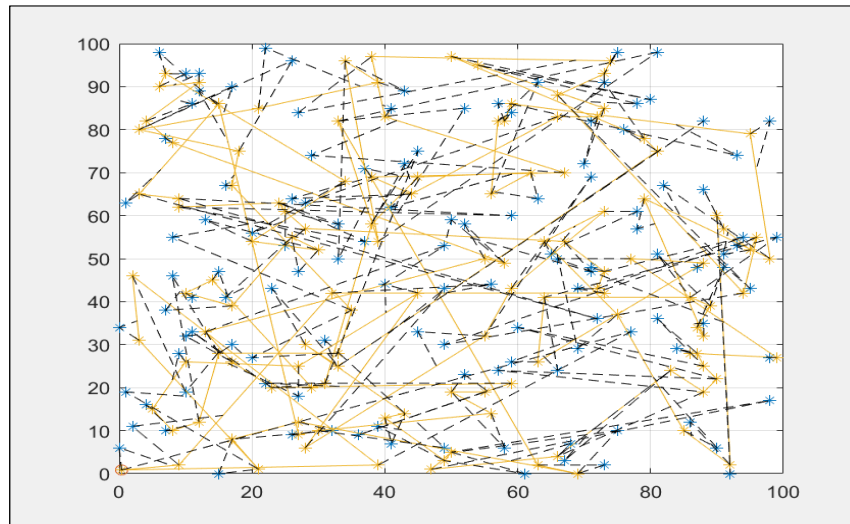


(b) Best cost per iteration

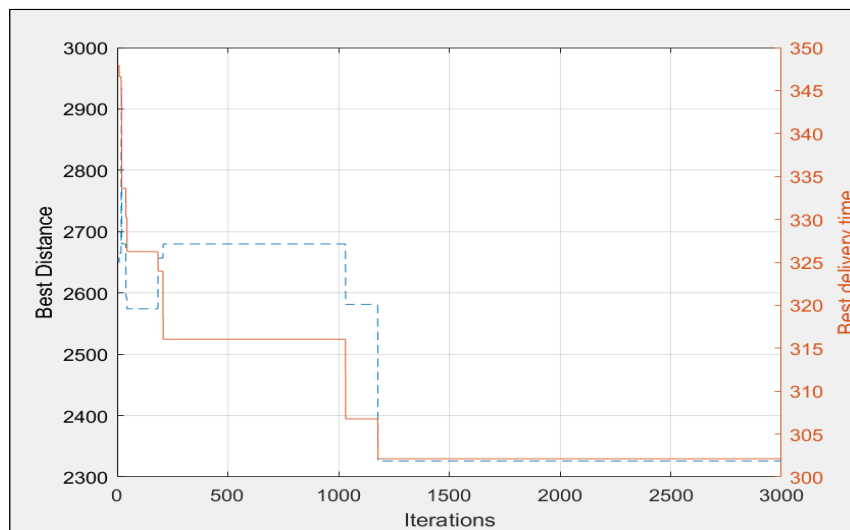


(c) Ant diversity per iteration

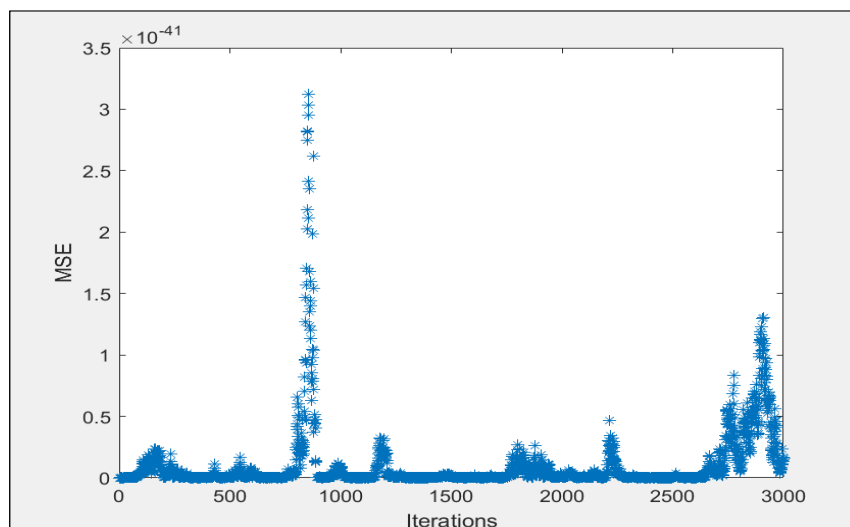
Figure 4.6: The drone and truck solution for 500 nodes (uniform-5-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2



(a) Best path



(b) Best cost per iteration



(c) Ant diversity per iteration

Figure 4.7: The drone and truck solution for 500 nodes (uniform-6-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

The results of the ACS-DT1 and ACS-DT2 algorithms, are also compared with the optimal solution values that Knoetze (2021) obtained for the first model with CPLEX, DTSP1 and the optimal solution values obtained for the second model, DTSP2. The results are shown in Table 4.8. The results show that ACS-DT1 solved the truck and drone scheduling problem optimally for the problem sizes of 10 nodes. With regard to the 20 node problems, the ACS-DT1 algorithm was able to obtain solution values between -4.2% and -13.4% of the optimal solution values. The results show that ACS-DT2 struggled to solve the truck and drone scheduling problem optimally.

It should be noted that the optimal solution values, DTSP2, are sometimes lower than the DTSP1 solution values. This finding makes sense since DTSP2 is a relaxation of DTSP1. This finding can, however, not be observed when studying the ACS-DT1 and ACS-DT2 computational results. The ACS-DT2 is solving a harder problem than the ACS-DT1 algorithm because at each node selection, ACS-DT2 needs to consider all remaining truck and drone delivery nodes versus ACS-DT1 which will either consider the remaining truck OR drone nodes, but not both. Further performance enhancements such as the inclusion of local search is required for ACS-DT2 to perform better against the optimal solution values.

The algorithms were further compared with Ernst (2021)'s SaNSDE and GCPSO based truck and drone scheduling algorithms. Ernst (2021)'s algorithm, SaNSDE and GCPSO, assign truck and drones to nodes similar to ACS-DT2. The delivery completion time and distance of TSP solutions, ACS-DT2, SaNSDE and GCPSO are shown in Tables 4.9 and 4.10. The results show that for a problem size of fewer than 20 customers, the SaNSDE algorithm performed better than ACS-DT1, and GCPSO. For the problem size of 50, the ACS-DT1 outperformed other algorithms, including the truck only solution. The other algorithms performed worse than the truck-only solution (TSP). For a problem size of more than 100 nodes, all algorithms performed worse against the TSP. The result for ACS-DT1 showed that it performed better compared with SaNSDE, and GCPSO.

CHAPTER 4. SOLVING THE SINGLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION

Table 4.6: Delivery time results of the ACS-DT1 and ACS-DT2 algorithms

Data	TSP	ACS-DT1					ACS-DT2				
	Time	Time (Mean)	Time (Standard deviation)	Waiting time (Mean)	Performance (%)	Running Time (Minutes)	Time (Mean)	Time (Standard deviation)	Waiting time (Mean)	Performance (%)	Running time (Minutes)
Uniform-51-n10	31.02	25.47	0.00	0.24	18%	4	25.87	0.25	1.78	17%	4
Uniform-52-n10	31.29	21.83	0.00	0.15	30%	4	20.71	0.20	1.31	34%	4
Uniform-53-n10	29.37	20.20	0.00	0.87	31%	4	20.98	0.10	2.07	29%	4
Uniform-61-n20	37.52	28.80	0.33	5.21	23%	5	32.67	0.63	3.72	13%	5
Uniform-62-n20	39.47	31.07	0.34	4.75	21%	5	33.87	0.33	4.25	14%	5
Uniform-63-n20	41.37	29.90	0.48	5.78	28%	5	33.95	0.94	3.67	18%	5
Uniform-71-n50	63.47	56.27	1.24	15.66	11%	8	83.34	3.17	16.35	-31%	8
Uniform-72-n50	66.60	59.19	1.54	16.22	11%	8	85.56	1.49	16.95	-28%	8
Uniform-73-n50	65.78	57.48	1.46	14.03	13%	8	83.58	1.22	13.16	-27%	8
Uniform-91-n100	90.42	91.18	2.11	45.38	-1%	19	161.09	3.08	36.20	-78%	18
Uniform-92-n100	84.71	88.09	1.26	23.53	-4%	17	149.18	2.30	27.98	-76%	17
Uniform-93-n100	86.52	90.93	1.47	26.82	-5%	19	150.93	2.85	36.74	-74%	17
Uniform-1-n250	142.00	183.20	4.01	83.19	-29%	100	382.98	5.30	91.87	-164%	80
Uniform-2-n250	145.00	179.90	6.76	65.97	-24%	123	385.04	7.62	90.15	-79%	83
Uniform-5-n500	215.60	308.60	11.87	144.50	-43%	359	744.17	16.74	187.77	-249%	364
Uniform-6-n500	213.00	307.60	9.04	134.67	-44%	349	748.07	10.19	186.97	-251%	335

4.5 Summary

The single truck and drone delivery system studied in this dissertation was explained in this chapter. Two mathematical models of the problem was presented and the ACS-based algorithms developed to solve this model were described. The algorithm control parameters were tuned and the algorithm was benchmarked against optimal solutions, two other metaheuristic algorithms, and a truck-only delivery system.

..

CHAPTER 4. SOLVING THE SINGLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION

70

Table 4.7: Delivery distance results of the ACS-DT1 and ACS-DT2 algorithms

Data	TSP		ACS-DT1				ACS-DT2			
	Distance		Distance (Mean)	Distance (Standard deviation)	Performance (%)	Running time (Minutes)	Distance (Mean)	Distance (Standard deviation)	Performance (%)	Running time (Minutes)
Uniform-51-n10	301.2		248.31	0.00	18%	4	250.98	2.18	17%	4
Uniform-52-n10	303.9		179.38	0.00	41%	4	195.22	3.31	36%	4
Uniform-53-n10	284.7		190.48	0.00	33%	4	182.92	6.85	36%	4
Uniform-61-n20	356.2		257.23	14.89	28%	5	287.03	14.98	19%	5
Uniform-62-n20	375.7		292.97	9.01	22%	5	315.17	7.83	16%	5
Uniform-63-n20	394.7		280.84	5.41	29%	5	311.44	13.09	21%	5
Uniform-71-n50	585.7		502.68	43.36	14%	8	732.46	59.76	-25%	8
Uniform-72-n50	617.0		555.63	40.61	10%	8	741.82	47.97	-20%	8
Uniform-73-n50	608.8		529.35	35.23	13%	8	730.77	25.68	-20%	8
Uniform-91-n100	805.2		815.33	47.43	-1%	19	1400.48	34.08	-74%	18
Uniform-92-n100	748.1		768.23	59.05	-3%	17	1294.68	68.68	-73%	17
Uniform-93-n100	766.2		821.44	38.69	-7%	19	1275.24	63.22	-66%	17
Uniform-1-n250	1171.0		1585.10	57.56	-35%	100	3093.95	153.67	-164%	80
Uniform-2-n250	1201.0		1533.00	140.33	-28%	123	3234.35	74.98	-169%	83
Uniform-5-n500	1657.4		2582.60	228.43	-56%	359	5830.13	270.42	-252%	364
Uniform-6-n500	1631.0		2608.20	68.50	-60%	349	6156.83	248.79	-277%	335

Table 4.8: ACS-DT1 and ACS-DT2 benchmarked against optimal delivery solutions for the two different models

Data	DTSP1		ACS-DT1			DTSP2			ACS-DT2			ACS-DT1 Performance	ACS-DT2 Performance
	Time	Time (Mean)	Time (Standard deviation)	Waiting time (Mean)	Time	Time (Mean)	Time (Standard deviation)	Waiting time (Mean)	Time (%)	Time (%)	Time (%)	Time (%)	
Uniform-51-n10	25.47	21.83	0.00	0.24	25.4	25.87	0.25	1.78	0.0%	0.0%	-1.6%		
Uniform-52-n10	20.20	21.83	0.00	0.15	20.3	20.71	0.20	1.31	0.0%	0.0%	-2%		
Uniform-53-n10	25.49	20.20	0.00	0.87	20.20	20.98	0.10	2.07	-13%	0.0%	-3.9%		
Uniform-61-n20	29.07	28.80	5.21	5.21	25.49	32.67	0.63	3.72	-6.9%	-13%	-28.2%		
Uniform-62-n20	28.63	31.07	0.34	4.75	28.87	33.87	0.33	4.25	-4.4%	-6.9%	-17.3%		
Uniform-63-n20	28.63	29.90	0.48	5.78	28.62	33.95	0.94	3.67	-4.4%	-4.4%	-18.6%		

Table 4.9: Benchmarking delivery completion time ACS-DT2 against GCPSO and SaNSDE (Ernst, 2021)

Data	ACS-DT2		SaNSDE		GCPSO	
	Time (Mean)	Time (Stan- dard Devia- tion)	Time (Mean)	Time (Stan- dard Devia- tion)	Time (Mean)	Time (Stan- dard Devia- tion)
Uniform-51-n10	25.87	0.25	25.47	0.00	25.61	0.24
Uniform-52-n10	20.71	0.20	20.36	0.08	20.63	0.29
Uniform-53-n10	20.98	0.10	20.20	0.00	20.59	0.49
Uniform-61-n20	32.67	0.63	27.57	1.28	29.98	2.43
Uniform-62-n20	33.87	0.33	30.24	0.62	32.67	1.59
Uniform-63-n20	33.95	0.94	28.83	0.34	33.38	3.28
Uniform-71-n50	83.34	3.17	85.54	4.25	89.75	8.34
Uniform-72-n50	85.56	1.49	92.79	4.29	94.44	9.13
Uniform-73-n50	83.58	1.22	88.51	3.60	90.15	9.56
Uniform-91-n100	161.09	3.08	202.62	6.60	209.22	10.35
Uniform-92-n100	149.18	2.30	201.59	5.41	211.25	17.00
Uniform-93-n100	150.93	2.85	N/A	N/A	N/A	N/A
Uniform-1-n250	382.98	5.30	640.00	9.90	673.62	32.00
Uniform-2-n250	385.04	7.62	633.17	11.15	659.68	30.99
Uniform-5-n500	744.17	16.74	N/A	N/A	N/A	N/A
Uniform-6-n500	748.07	10.19	N/A	N/A	N/A	N/A

Table 4.10: Benchmarking delivery distance of ACS-DT2 against GCPSO and SaNSDE (Ernst, 2021)

Data	ACS-DT2		SaNSDE		GCPSO	
	Distance (Mean)	Distance (Standard Deviation)	Distance (Mean)	Distance (Standard Deviation)	Distance (Mean)	Distance (Standard Deviation)
Uniform-51-n10	250.98	2.18	248.31	0.00	248.42	0.68
Uniform-52-n10	195.22	3.31	191.93	1.39	194.88	4.14
Uniform-53-n10	182.92	6.85	190.48	0.00	192.11	3.19
Uniform-61-n20	287.03	14.98	235.85	17.78	258.85	27.14
Uniform-62-n20	315.17	7.83	277.56	6.42	301.20	15.42
Uniform-63-n20	311.44	13.09	274.43	4.10	307.23	31.60
Uniform-71-n50	732.46	59.76	701.47	55.72	765.80	80.07
Uniform-72-n50	741.82	47.97	782.66	54.01	809.91	82.78
Uniform-73-n50	730.77	25.68	747.67	53.90	781.56	89.60
Uniform-91-n100	1400.48	34.08	1642.26	99.42	1751.24	128.38
Uniform-92-n100	1294.68	68.68	1663.78	59.89	1774.03	160.17
Uniform-93-n100	1275.24	83.22	N/A	N/A	N/A	N/A
Uniform-1-n250	3093.95	153.67	5250.31	164.57	5610.62	343.99
Uniform-2-n250	3234.35	74.98	5209.33	147.41	5523.64	336.98
Uniform-5-n500	5830.13	270.42	N/A	N/A	N/A	N/A
Uniform-6-n500	6156.83	248.79	N/A	N/A	N/A	N/A

Chapter 5

Solving the multiple truck and drone scheduling problem with interception

This chapter describes a **multiple** truck and drone delivery system where the drone can intercept its truck anywhere on the map. The problem is explained in detail in Section 5.1. Section 5.2 describes the ACS-based algorithm for scheduling of vehicles to customers, and Section 5.3 describes the experimental setup and results of the ACS-based multiple truck and drone scheduling algorithm. Section 5.4 concludes the chapter.

5.1 Problem description

This problem is the “multiple truck and drone” version of the “single truck and drone” problem considered in Section 4.1. The problem is based on **multiple** trucks and drones making deliveries to various customers. Various truck and drone systems leave the depot together. Drones are launched from each truck at a customer’s location and fly to other customers to deliver packages. After a drone has completed a delivery it flies back to the truck it was launched from. The drone intercepts the truck while the truck is still in transit, and the truck then carries the drone to the next customer’s location. While trucks are in transit, packages are loaded onto the drone and fresh batteries are installed. When a truck reaches the following customer location, it again launches the drone to the next customer and continues its deliveries. The delivery process repeats itself until all customers’ orders are fulfilled. All trucks and drones then return to the depot, together or independently. Figure 5.1 illustrates the model just described. The first, second and third trucks visit nodes 2, 3; 4, 6, 7; and 11 and 9, respectively. Meanwhile,

the drones visit nodes 1 and 5; 8; and 10.

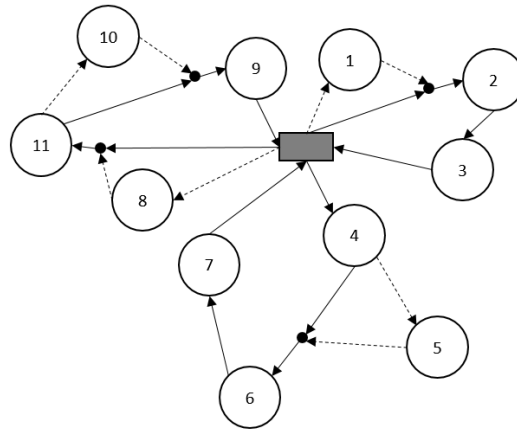


Figure 5.1: A truck-drone system with interception points (black circles), truck routes (solid lines), and drone routes (dotted lines).

The following forbidden moves are applicable. Launching a drone at an intermediate location or while the truck is in motion is not allowed and only one drone delivery between two truck deliveries is allowed. It should also be noted that a drone can only return to the same truck that launched it.

5.2 The ACS-based scheduling algorithm for the multiple truck and drone scheduling problem

The ACS-based scheduling algorithm for the multiple truck and drone scheduling problem requires the following input parameters: the number of customer nodes, n , the demand of each customer j , d_j , the vehicle capacity, Q , and T_{ijk} , which denotes the time required for servicing nodes i and j and traveling to k if the truck travels between nodes i and k and the drone services node j in between. The calculation of T_{ijk} is explained in Section 4.2.

The ACS-based algorithm is based on a cluster first, route second approach. The customers are first grouped using a k-means clustering algorithm. The number of clusters is determined using customer demand versus vehicle capacity. The algorithm considers the following inputs: Truck speed v_t , drone speed v_d , data set $X = \{x_1, \dots, x_n\}$, node demand d_j and vehicle capacity Q

The total demand, D_{tot} for all customers is $D_{tot} = \sum d_j$. To determine the number of clusters, n_c , required, total demand, D_{tot} , is divided by vehicle capacity, Q , that is $n_c = D_{tot}/Q$. If the computed value is not an integer, it is rounded

up to the integer value, n_C . After clustering is completed and it happens that a cluster's demand, d_c where $c \in \{1, \dots, n_C\}$, exceeds the vehicle capacity, Q , then customers on the periphery of the current cluster are moved to the next nearest cluster, and the centroids of each cluster are recalculated. The process continues until cluster demand is either on par or below vehicle capacity.

As soon as all nodes are allocated to a cluster, each cluster of nodes is scheduled using the single truck and drone scheduling algorithm described in Chapter 4 (ACS-DT1). The truck and drone are represented as different ants walking through the same network searching for the best path. They cannot sense each other's pheromone. All ants (n_k) start at the depot node. Nodes are assigned to either a truck or a drone using Equation (3.19) and Equation (3.20). The pheromone is then updated on all the paths visited by the ants using local pheromone according to Equation (3.23). After nodes have been visited by the ants representing the truck and drone, a drone path with interception is constructed on the truck route and the system delivery and waiting time is computed. This construction of truck and drone routes is repeated for all ants. Every time the routes have been constructed, the delivery and waiting time of the system is calculated. The best cost (maximum time) of the iteration is then used to update the ant pheromone using Equation (3.21). The overall process is repeated for several iterations until the maximum number of iterations has been reached. A vehicle assignment rule was adopted to eliminate infeasible solutions and avoid repairing solutions after the route construction. The rule states that if a node is assigned to a drone, the next node cannot be a drone node, but must be a truck node. However, if a node is assigned to a truck, the next node can be either a truck or a drone node. Figure 5.2 provides a flowchart of the ACS truck-drone delivery system.

In mathematical terms, the pheromone matrix of a truck and drone are separate but they are initialised with the same values and are represented by $\tau_{ik}^{truck}(t)$ and $\tau_{ij}^{drone}(t)$ where $\tau_{ik}^{truck}(t)$ is the pheromone between customer i and k observed by the truck at iteration t , $\tau_{ij}^{drone}(t)$ is the pheromone between customer i and j observed by the drone at iteration t , and t is the iteration number.

5.3 Empirical evaluation of the ACS multiple drone-truck scheduling algorithm

The ACS algorithm is evaluated on 10 data sets of different sizes adapted from Bouman *et al.* (2015). The number of clusters was calculated from the total demand divided by the vehicle capacity. The vehicle capacity was set to $C_k = 40$ for data sets of size 50 and 100, and $C_k = 100$ for data sets of size 250 and 500. The built-in k-means clustering algorithm in MATLAB was used. K-means was

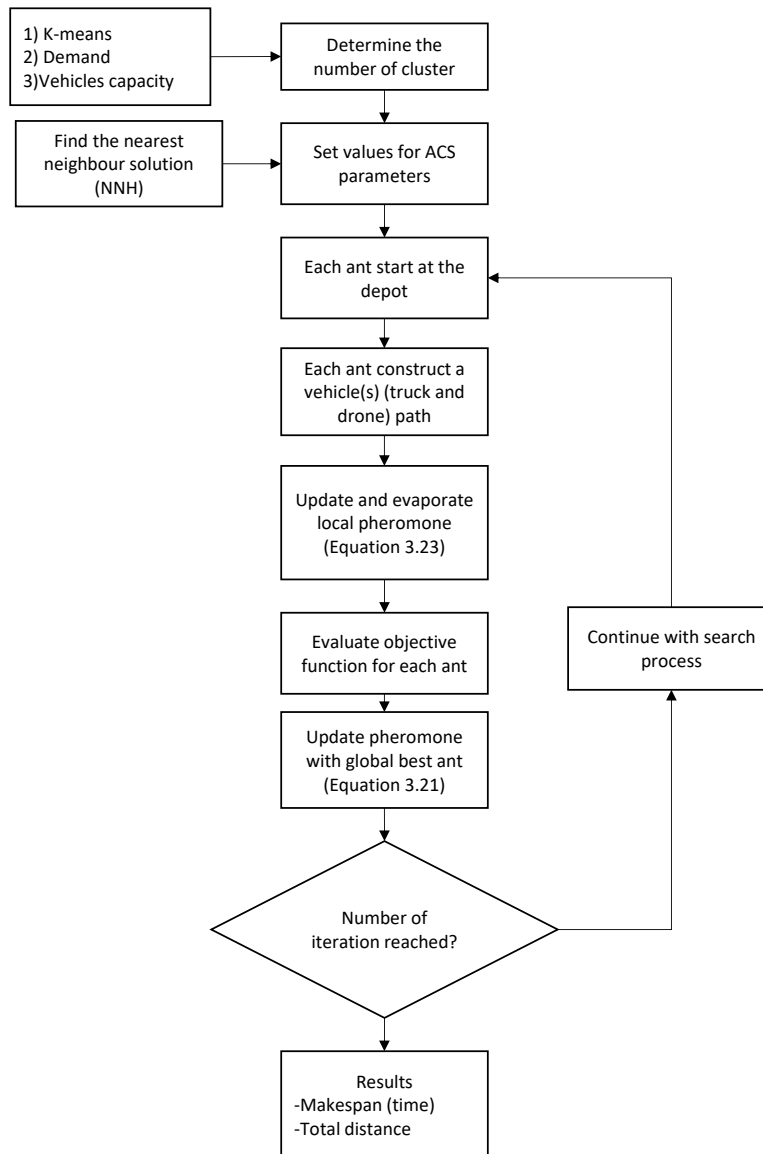


Figure 5.2: ACS algorithm process flowchart

set to run for 100 iterations, and the best clustering with a minimal sum of square distance was selected as the best solution.

The ACS algorithm control parameters were set as follows:

- τ_0 is set by determining the nearest neighbour of the problem set and counting the number of nodes (n). Heuristically, the nearest neighbour searches for the node closest to the current node and selects it as the next node. The process continues until all nodes are visited and the full path (L) is determined. A small positive value constant is calculated using ($\tau_0 = (nL^{-1})$).
- Other control parameters are number of ants (n_k), maximum number of iterations, truck speed (v_t), drone speed (v_d), truck delivery time (ω), and drone delivery time (δ).
- The ACS algorithm parameter, β , was set to 2 as determined by the F-Race procedure and the same data sets as described in Section 4.4.

The ACS parameter settings are set to obtain the desired search balance between more exploration at the start of the search process and more exploitation at the end of the search process as indicated in Table 5.1.

Table 5.1: Algorithm and problem parameters

ACS Parameters	Values
ρ_1	$1 \rightarrow 0$
ρ_2	$0 \rightarrow 1$
r_0	$0 \rightarrow 1$
Max. no. of iterations	3000
Problem Parameters	Values
v_d	20
v_t	10
Truck Capacity (Q)	40
w (Truck delivery time)	0.1
δ (Drone delivery time)	0.1

The ACS algorithm is evaluated over 30 independent algorithm runs on each data set. The delivery completion time and distance means, and average waiting time of the algorithm, are shown in Table 5.2. The delivery completion time is the time it takes to complete all deliveries, and the distance is the total distance travelled by trucks.

The multiple trucks and drone delivery system (VRPD) is compared with the truck-only (VRP) system solved with an ACS algorithm (Algorithm (14)). The

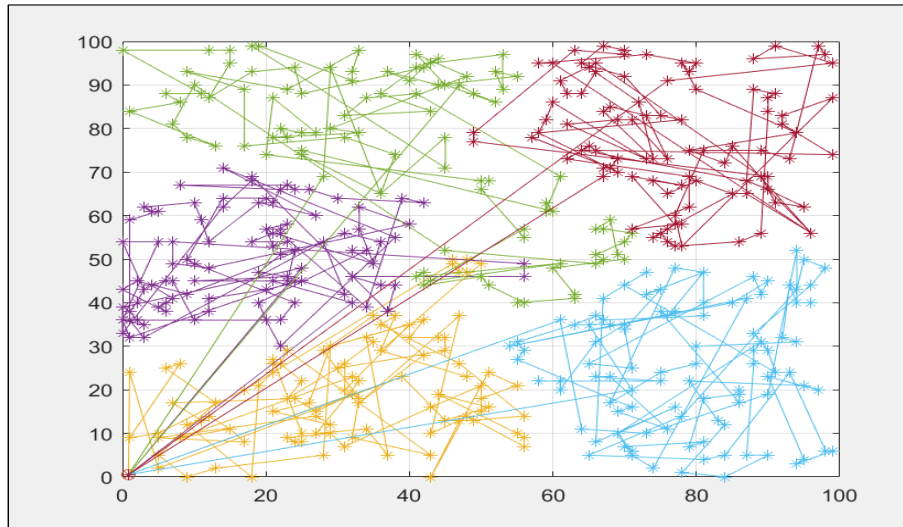
deliveries to nodes of the cluster determined from k-means are scheduled with the truck only. After that, the deliveries of the same cluster are scheduled with trucks and drones. The truck and drones are scheduled similarly to ACS-DT1 in Section 4.3.1.

The system yielded positive results compared with the truck-only system for all data sets investigated. The improvement in distance ranged from 5% to 26%. Meanwhile, the total delivery time improvement ranged from 7% to 22%, and the maximum system waiting time was 33%. The CPU time spent (running time) solving the VRP and VRPD with ACS algorithm are shown in Table 5.2.

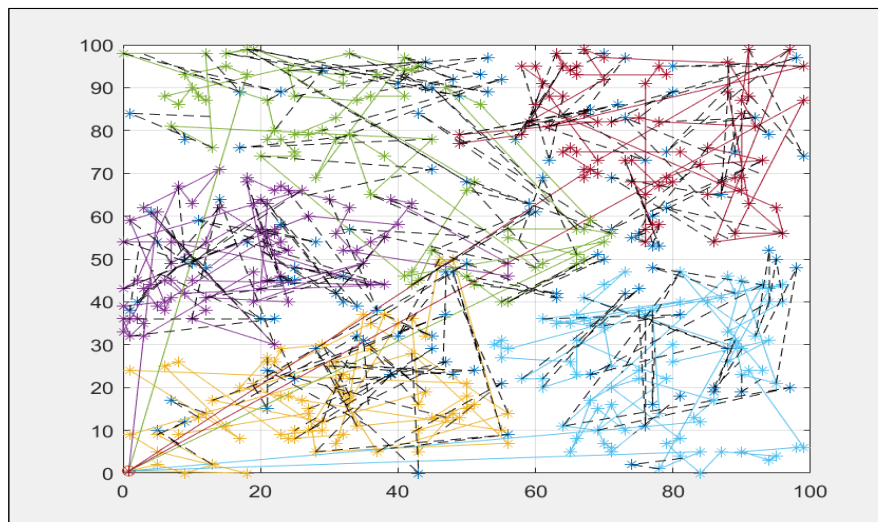
Graphical representations of a selected sample of the truck-only solutions and multiple truck and drone solutions are shown in Figures 5.3. From these figures it is clear that the solutions obtained by the ACS-based algorithm are feasible and comply with the problem constraints. The best cost per iteration for the data sets over time are shown in Figures 5.4. These figures show the rate of decrease of the objective function over time. It can be seen that the ACS-based algorithm continues to improve the best solution found over a large percentage of the iterations and does not quickly converge to a potentially suboptimal solution. The diversity of the ants' search per iteration is shown in Figures 5.5. These graphs again confirm that a sufficient level of diversity is maintained throughout the search process and that the algorithm does not converge to a single solution early on during the search. Due to space constraints only the analyses of a selected number of data sets of different sizes are shown here. The remaining graphs for the other data sets can be found in Appendix 7.3.

Non-parametric statistical tests were also used to evaluate the statistical significance of the results obtained. The results in Table 5.3 were obtained by comparing the results for each problem-system combination to the results of each one of the other problem-system combinations. A Mann-Whitney U test at 5% significance was performed (using the two sets of 30 performance metric values of the two problem-system combinations under comparison). If the first combination statistically significantly outperformed the second combination, a win was recorded. If no statistical difference could be observed, a draw was recorded. If the second problem-algorithm combination outperformed the first problem-system combination, a loss was recorded for the first problem-system combination. The total number of wins, draws, and losses were then recorded for all comparisons of the problem-algorithm combinations under evaluation. As an example, (10 – 0 – 0) in row 1 of Table 5.3 indicates that the VRPD statistically significantly outperformed the VRP algorithms 10 times when the distance and time were used as a performance metric. No draws were recorded, and 10 VRP losses were recorded.

From the statistical analysis, it can be seen that VRPD is the best performing algorithm when distance and time are used as a performance metric. On closer

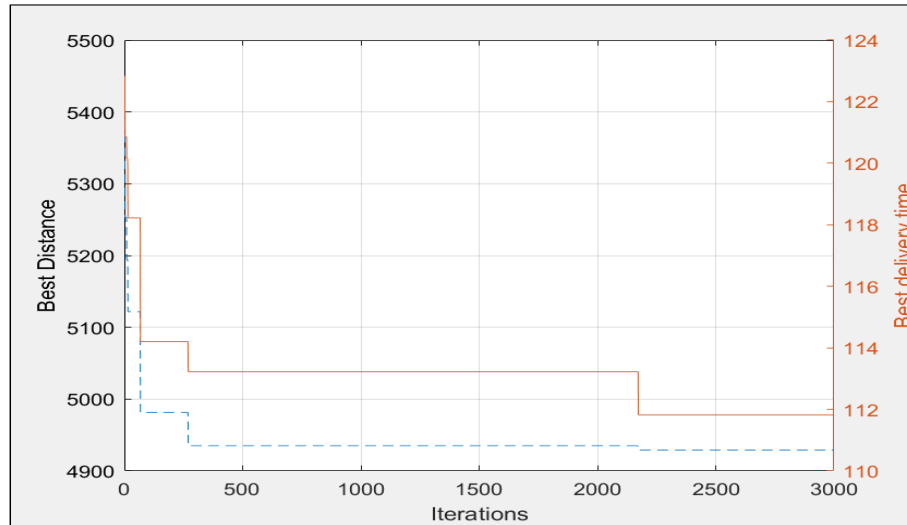


(a) Truck-only

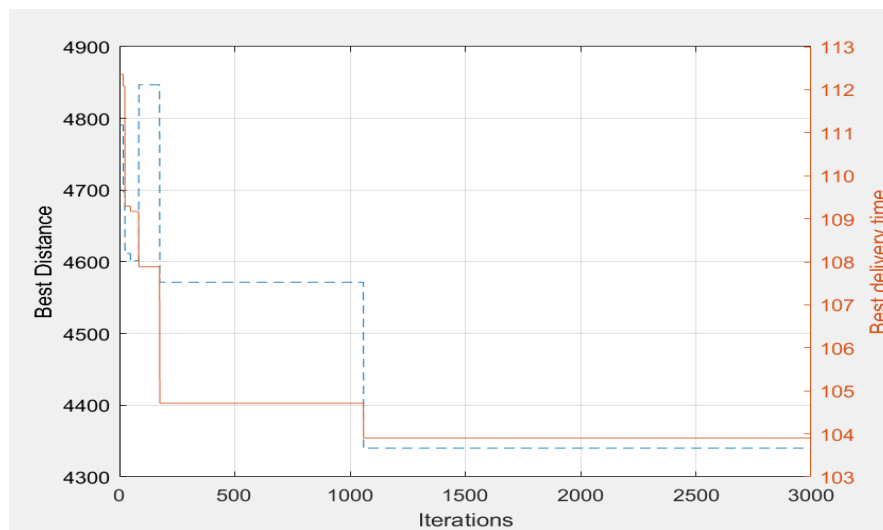


(b) Drones and trucks

Figure 5.3: 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-6-n500)

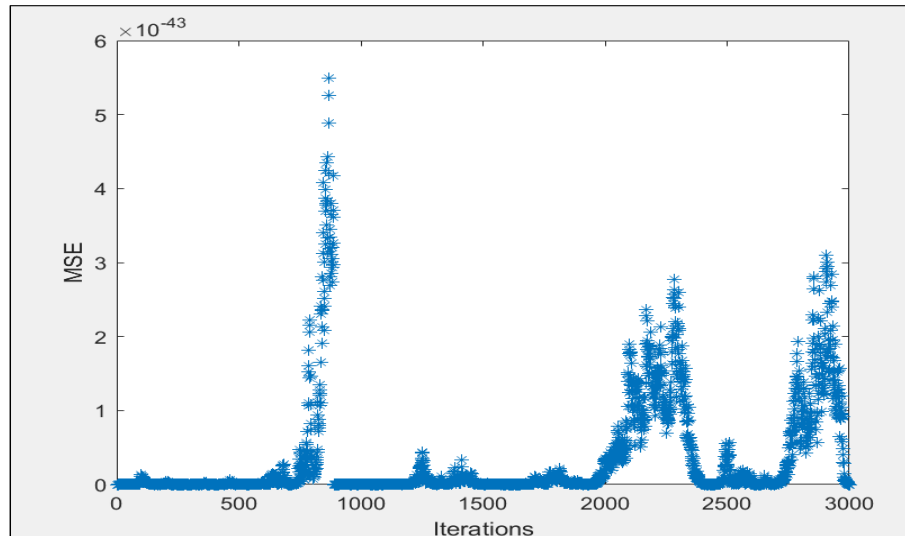


(a) Truck-only

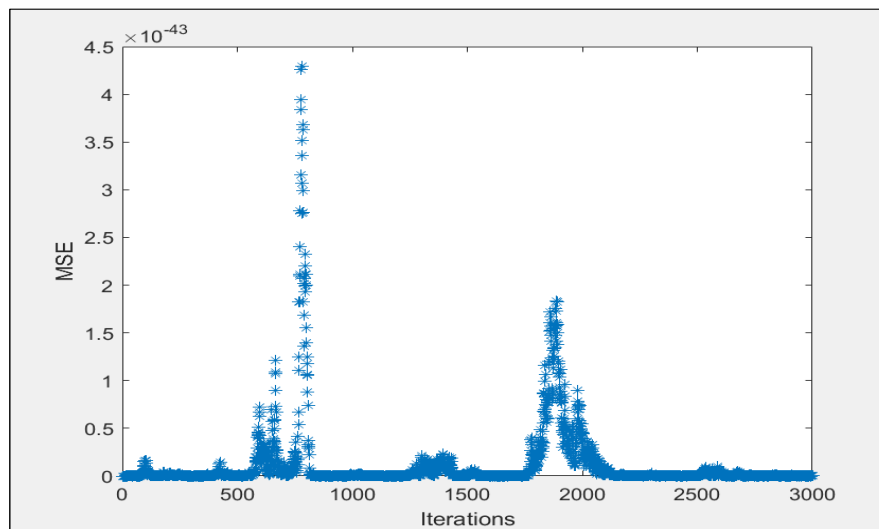


(b) Drones and trucks

Figure 5.4: Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-6-n500)



(a) Truck-only



(b) Drones and trucks

Figure 5.5: Solutions search diversity for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-6-n500)

inspection of the results, it became clear that all problem sizes of the multiple truck and drone system outperformed a truck-only system.

5.4 Summary

This chapter introduced the multiple truck and drone scheduling problem with interception. The problem was solved by means of a k-means/ACS-based truck and drone scheduling algorithm. This algorithm was shown to outperform a truck-only delivery system over a range of benchmark problems of different sizes.

CHAPTER 5. SOLVING THE MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION

Table 5.2: Results of the VRP and VRPD system performance

Data	VRP						VRPD						Performance	
	Distance (Mean)	Distance (Standard deviation)	Time (Mean)	Time (Standard deviation)	Running time (Minutes)		Distance (Mean)	Distance (Standard deviation)	Time (Mean)	Time (Standard deviation)	Waiting time (Mean)	Running time (Minutes)	Time (%)	Distance (%)
Uniform-71-n50	905.9	17.4	49.8	0.4	18		752.4	18.2	42.3	0.4	0.0	2	15%	17%
Uniform-72-n50	879.1	20.4	54.4	0.9	20		834.6	54.5	45.2	1.0	0.6	2	17%	5%
Uniform-73-n50	961.0	13.7	51.3	0.4	11		710.7	44.6	40.0	0.5	2.2	2	22%	26%
Uniform-91-n100	1615.6	12.1	58.2	0.9	24		1348.1	85.8	49.3	0.7	5.2	4	15%	17%
Uniform-92-n100	1587.6	36.3	56.5	0.4	35		1219.5	29.8	45.6	0.5	0.8	2	19%	23%
Uniform-93-n100	1518.7	11.9	59.8	0.5	36		1313.3	17.2	50.9	1.9	1.7	2	15%	14%
Uniform-1-n250	3028.5	72.3	111.0	2.5	55		2692.2	21.7	99.6	3.7	7.9	10	10%	11%
Uniform-2-n250	3176.7	58.9	120.3	2.0	78		2745.0	170.3	106.0	3.1	12.9	10	12%	14%
Uniform-5-n500	4929.0	111.0	111.8	7.9	94		4340.1	101.1	103.9	4.0	33.0	27	7%	12%
Uniform-6-n500	5001.5	46.45418	114.0	2.5	141		4406.6	23.3	103.6	2.0	32.8	24	9%	12%

Table 5.3: Hypothesis tests comparing the VRP and VRPD system with regard to distance and time

	Distance			Time		
Type	Wins	Draws	Losses	Wins	Draws	Losses
VRP	0	0	10	0	0	10
VRPD	10	0	0	10	0	0

Chapter 6

Solving the multiple trucks and drones scheduling problem with time window constraints

This chapter describes a **multiple** trucks and drones delivery system where the drone can intercept its truck anywhere on the map, and where customers are only available in certain time intervals to receive their deliveries. The problem is explained in detail in Section 6.1. The mathematical formulation of the problem is presented in Section 6.2. Section 6.3 describes the ACS-based algorithm for scheduling of vehicles to customers and Section 6.4 describes the experimental setup and results of the ACS-based multiple trucks and drones with time windows scheduling algorithm. Finally, Section 6.5 concludes the chapter.

6.1 Problem description

The problem is based on **multiple** trucks and drones making deliveries to various customers within specific time window constraints. Various truck and drone systems leave the depot together, and drones are launched from each truck at a customer's location and fly to other customers to deliver packages. After a drone has completed a delivery, it flies back to the truck it was launched from. The drone intercepts the truck while the truck is still in transit, and the truck then carries the drone to the next customer's location. While trucks are in transit, packages are loaded onto the drone, and batteries are swapped for fresh ones. When a truck reaches the following customer location, it again launches the drone to its next customer and then continues with its deliveries. The delivery process repeats itself until all customers' orders are fulfilled. All vehicles return to the depot together or independently. Figure 6.1 illustrates the model just described. The first and

second trucks visit nodes 1 and 7; and 3, 5, and 6; respectively. Meanwhile, the drones visit nodes 2 and 9; and 10, 4, and 8.

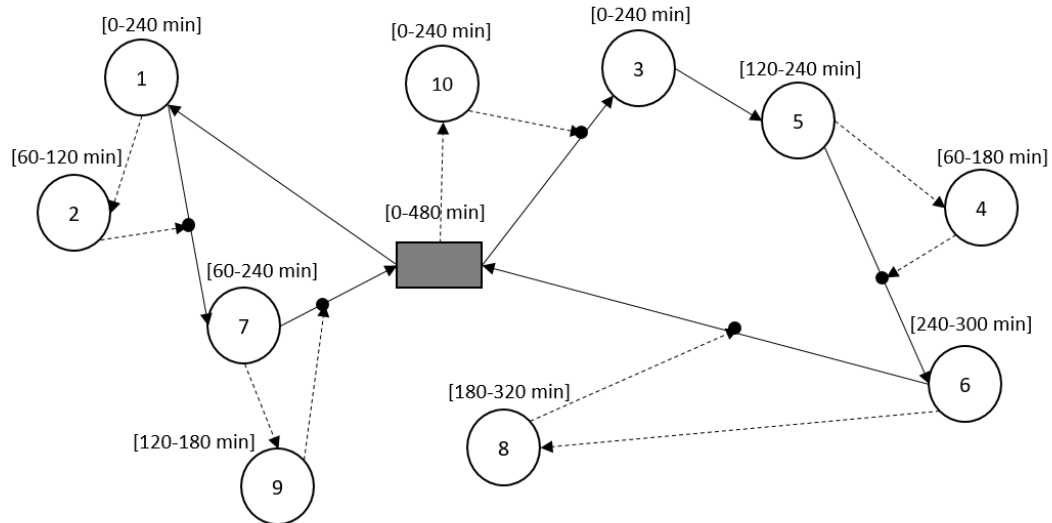


Figure 6.1: A truck and drone system with interception points (black circles), truck routes (solid lines), drone routes (dotted lines), and time window constraints in square brackets

The same forbidden moves defined in Section 4.1 are applicable. Launching a drone at an intermediate location or while the truck is in motion is not allowed, and only one drone delivery between two truck deliveries is allowed. Again, it should also be noted that a drone can only return to the same truck that launched it.

6.2 Mathematical model

The same model described in 4.2 is used to route each cluster of customers, with the addition of a time window constraint:

$$e_i \leq s_i^k \leq l_i \quad \forall k \in \{1, \dots, m\}, i \in \{1, \dots, n\} \quad (6.1)$$

where e_i is the earliest delivery time and l_i is the latest allowable delivery time for customer i .

6.3 The ACS-based scheduling algorithm for the multiple trucks and drones scheduling problem with time windows

The ACS-based scheduling algorithm for the multiple truck and drone scheduling problem with time windows requires the following input parameters: number of customer nodes, n , the demand of each customer j , d_j , the vehicle capacity, Q , and T_{ijk} , which denotes the time required for servicing nodes i and j and traveling to k if the truck travels between nodes i and k and the drone services node j in between. The calculation of T_{ijk} is explained in Section 4.2. In addition, the time window (in terms of e_i and l_i) is required for each customer.

The ACS-based scheduling algorithm for the multiple truck and drone scheduling problem with time windows is also based on a cluster first, route second approach. The customers are first grouped using a k-means clustering algorithm. The number of clusters is determined using customer demand versus vehicle capacity. For example, the total demand, $D_{tot} = \sum d_j$ where d_j is the customer demand, dividing the total customer demand, D_{tot} by vehicle capacity, Q gives the number of clusters $n_C = D_{tot}/Q$. If the computed value is not an integer, it is rounded up to the next integer value, n_C . After clustering is completed and it happens that a cluster's demand, d_c where $c \in \{1, \dots, n_C\}$, exceeds the vehicle capacity, Q , then the customers on the periphery of the current cluster are moved to the nearest next cluster, and the centroids of each cluster are recalculated. The process continues until cluster demand is either on par or below vehicle capacity.

As soon as all nodes have been allocated to a cluster, each cluster of nodes is scheduled using the single truck and drone scheduling algorithm described in Chapter 4. All ants, n_k , start at the depot node. Nodes are assigned to either a truck or a drone using Equation (3.19) and Equation (3.20). An ant making deliveries stops when it returns to the depot, and another ant starts to make deliveries. The pheromone is then updated on all the paths visited by the ants using local pheromone according to Equation (3.23). The ants first prioritise nodes with the smallest value of e_i . Once the deliveries have been made to a customer, i , the ant will check for the next customer with the smallest earliest delivery time, e_i .

If the vehicle arrives early at a customer location, i , that is $\tau^j < e_i$ it will wait, $e_i - \tau^j$ for the customer to be available. However, the vehicle is not allowed to be late, $\tau^j > l_i$. After nodes have been visited by the ants representing the truck and drone, a drone path with interception is constructed on the truck route and the system delivery and waiting time is computed. This construction of truck and drone routes is repeated for all ants. Each time the routes have been constructed, the delivery and waiting time of the system is calculated. The best cost (maximum

time) of the iteration is then used to update the ant pheromone using Equation (3.21). The overall process is repeated for several iterations until the maximum number of iterations has been reached. As described in Chapter 5, a vehicle assignment rule was also adopted to eliminate infeasible solutions and avoid the need for repairing solutions after the route construction. Figure 6.2 provides a flowchart of the ACS truck-drone delivery system.

The truck and drone are treated as different ants walking through the same network, searching for the best path. They cannot sense each other's pheromone. In mathematical terms, the pheromone matrix of a truck and drone are separated but they are initiated with the same values and are represented by $\tau_{ik}^{truck}(t)$ and $\tau_{ij}^{drone}(t)$ where $\tau_{ik}^{truck}(t)$ is the pheromone between customer i and k observed by the truck at iteration t , $\tau_{ij}^{drone}(t)$ is the pheromone between customer i and j observed by the drone at iteration t , and t is the iteration number.

6.4 Empirical evaluation of the ACS multiple trucks and drones scheduling algorithm with time windows

The ACS algorithm is evaluated on 10 data sets of different sizes adapted from Bouman *et al.* (2015). The number of clusters is again calculated from the total demand divided by the vehicle capacity. The vehicle capacity is set to $Q = 40$ for data sets of size 50 and 100, and $Q = 100$ for data sets of size 250 and 500. The first 25% of the coordinate pairs, excluding the depot, are assigned time windows of 0 to 240 min, 50% of the coordinate pairs are given a time window of 240 minutes to 480 minutes. These time windows are in line with existing South African retailer policies. The remaining data points are not assigned time windows. The built-in k-means clustering algorithm in MATLAB[®] is used. The k-means is set to run for 100 iterations, and the best clustering with a minimal sum of square distance is selected as the best solution.

The ACS algorithm control parameters are set as follows:

- A nearest neighbour heuristic searches for the node closest to the current node that does not violate the time window constraint and selects it as the next node. In cases where a node cannot be added due to time window constraint violation, the nearest neighbour heuristic restarts the search at a new random point. The process continues until all nodes are visited and the full path (L) is determined. The initial pheromone concentration, τ_0 , is then calculated as: $\tau_0 = (nL^{-1})$.

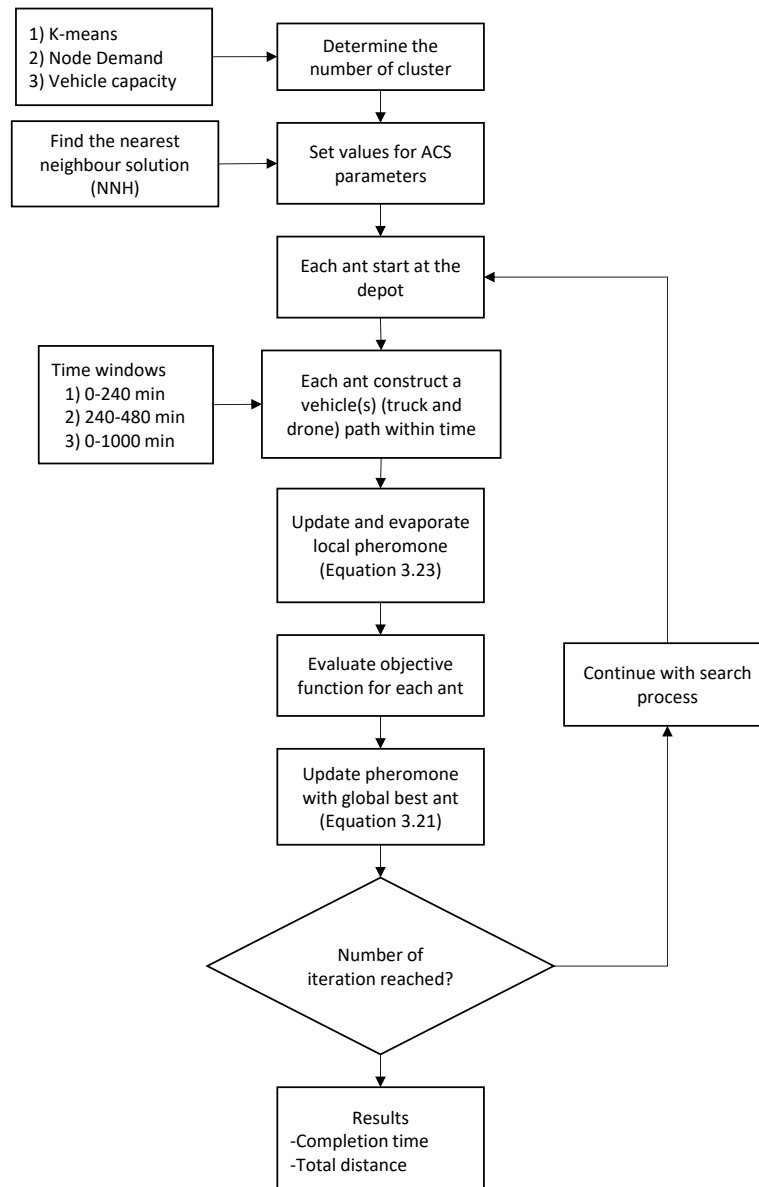


Figure 6.2: ACS algorithm process flowchart for optimising the multiple trucks and drones scheduling problem with time window constraints

- Other control parameters are the number of ants (n_k), maximum iterations, truck speed (v_t), drone speed (v_d), truck delivery time (w_i), and drone delivery time (δ_i).
- The ACS algorithm parameter, $\beta = 2$, was set in Section 4.4.

The ACS parameter settings are set to obtain the desired search balance between more exploration at the start of the search process and more exploitation at the end of the search process as indicated in Table 6.1.

Table 6.1: Algorithm and problem parameters

ACS Parameters	Values
ρ_1	$1 \rightarrow 0$
ρ_2	$0 \rightarrow 1$
r_0	$0 \rightarrow 1$
Max. no. of iterations	3000
Problem Parameters	Values
v_d	20
v_t	10
Truck Capacity (Q)	40
w_i (Truck delivery time)	0.1
δ_i (Drone delivery time)	0.1

The ACS algorithm is evaluated over 30 independent algorithm runs on each data set. The delivery time and distance means and standard deviations of the algorithms are shown in Table 6.2. The delivery completion time is the time it takes to complete all deliveries, and the distance is the total distance travelled by trucks. The multiple trucks and drone delivery system with time windows (VRPDTW) is compared with the truck-only (VRPTW) system with time windows solved with an ACS algorithm (Algorithm (14)). The deliveries to nodes of the cluster determined from k-means are scheduled with the truck only (VRPTW). After that, the deliveries of the same cluster are scheduled with trucks and drones (VRPDTW). The truck and drones are scheduled similarly to ACS-DT1 in Section 4.3.1.

The system yielded positive results compared with the truck-only system for all data sets investigated. The improvement in distance ranged from 0% to 32%. Meanwhile, the total delivery time improvement ranged from 3% to 12%. The CPU time spent by both models are also shown in the table.

A graphical representation of a selected sample of the truck-only solution and multiple truck and drone solutions are shown in the two graphs in Figure 6.3.

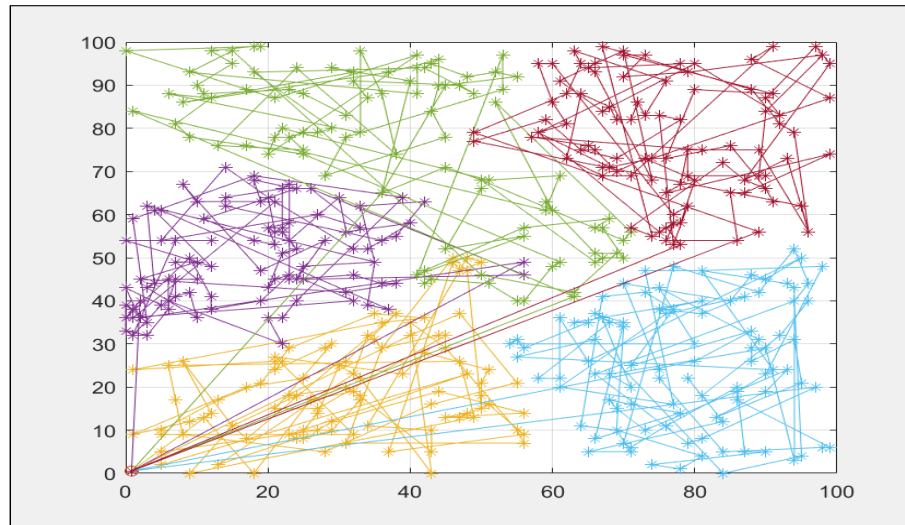
These figures show that the ACS-based algorithm's solutions are feasible and comply with the problem constraints. The best cost per iteration for the data sets over time are shown in the two graphs in Figure 6.4. These figures show the rate of decrease of the objective function over time. It can be seen that the ACS-based algorithm continues to improve; They further show the best solutions found over a large percentage of the iterations and does not quickly converge to a potentially suboptimal solution. The diversity of the ants' search per iteration is shown in Figure 6.5. These graphs again confirm that a sufficient level of diversity is maintained throughout the search process. The algorithm does not converge to a single solution early on during the search. Due to space constraints, only the analyses of a selected number of data sets of different sizes are shown here. The remaining graphs for the other data sets can be found in Appendix 7.3.

Non-parametric statistical tests were also used to evaluate the statistical significance of the results obtained. The results in Table 6.3 were obtained by comparing the results for each problem-system combination to the results of each of the other problem-system combinations. A Mann-Whitney U test at 5% significance was performed (using the two sets of 30 performance metric values of the two problem-system combinations under comparison). If the first combination statistically significantly outperformed the second combination, a win was recorded. If no statistical difference could be observed, a draw was recorded. If the second problem-algorithm combination outperformed the first problem-system combination, a loss was recorded for the first problem-system combination. The total number of wins, draws, and losses were then recorded for all comparisons of the problem-algorithm combinations under evaluation. As an example, (10 – 0 – 0) in row 2 of Table 6.3 indicates that the VRPDTW statistically significantly outperformed the VRPTW scenario 10 times when time was used as a performance metric; 0 draws were recorded, and 0 VRPDTW losses were recorded.

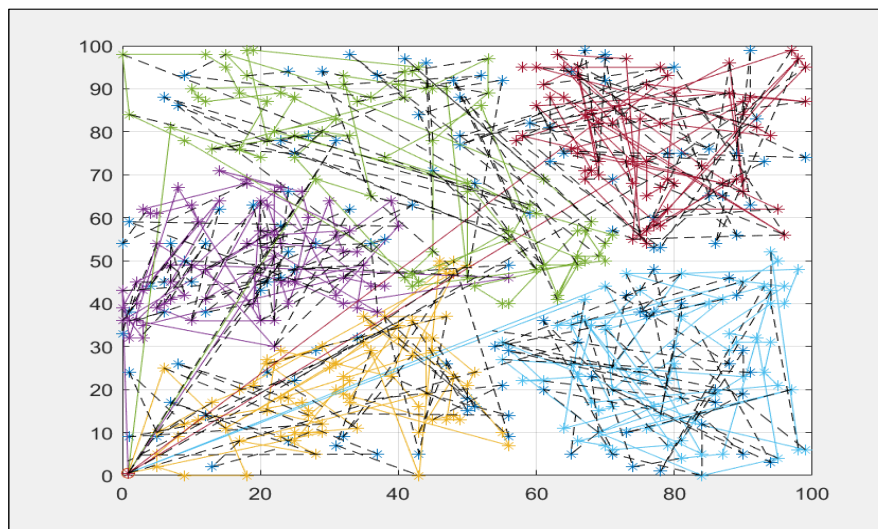
From the statistical analysis, it can be seen that VRPDTW is the best performing system when distance and time are used as a performance metric. On closer inspection of the results, it became clear that all problem sizes of the multiple trucks and drones system with time windows outperformed the truck-only system.

6.5 Summary

This chapter introduced the multiple truck and drone scheduling problem with interception and time windows. The problem was formulated mathematically and solved utilising a k-means/ACS-based truck and drone scheduling algorithm. This algorithm was shown to outperform a truck-only delivery system over a range of benchmark problems of different sizes.

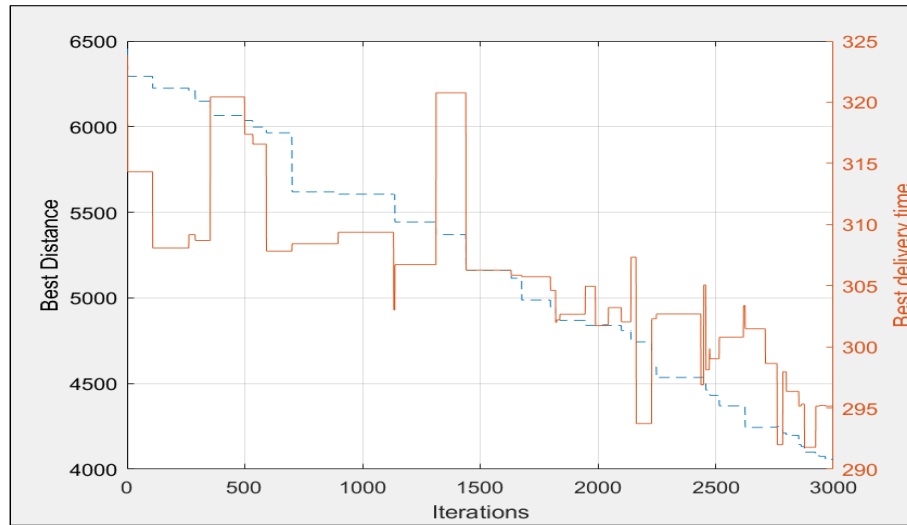


(a) Truck-only

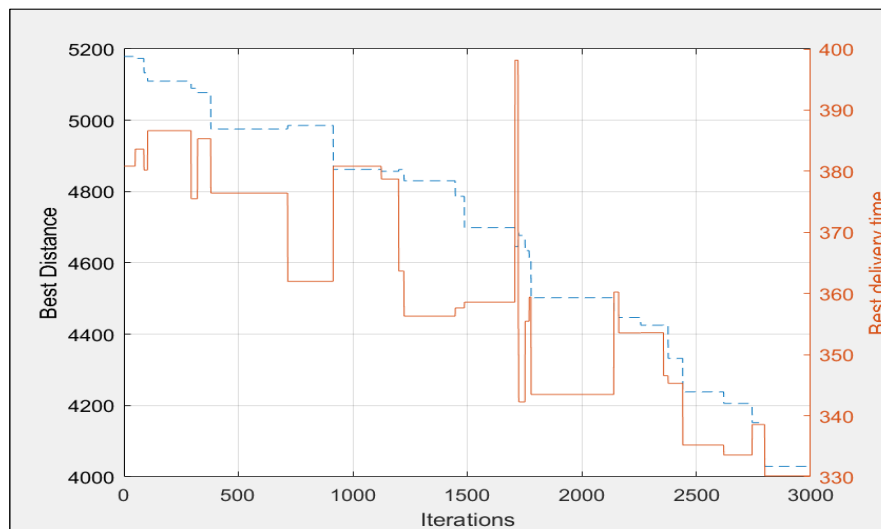


(b) Drones and trucks

Figure 6.3: 500 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time-window constraints (Uniform-6-n500)

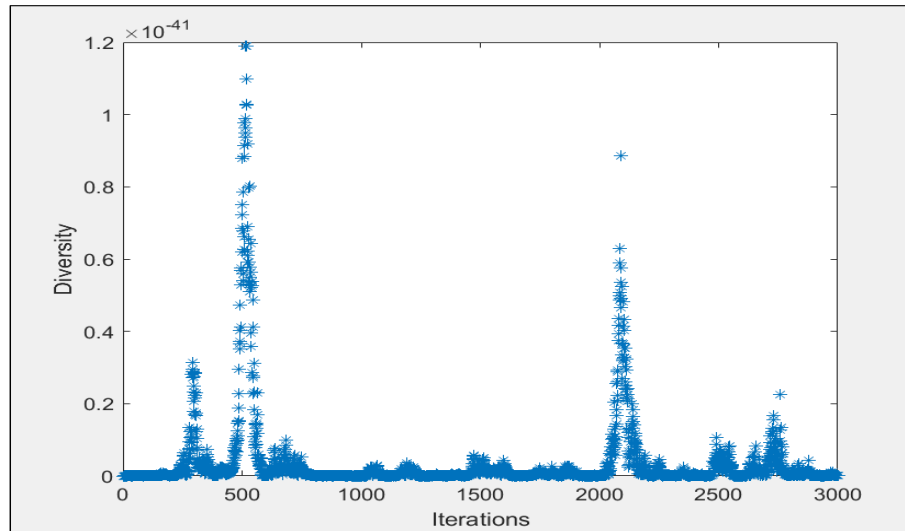


(a) Truck-only

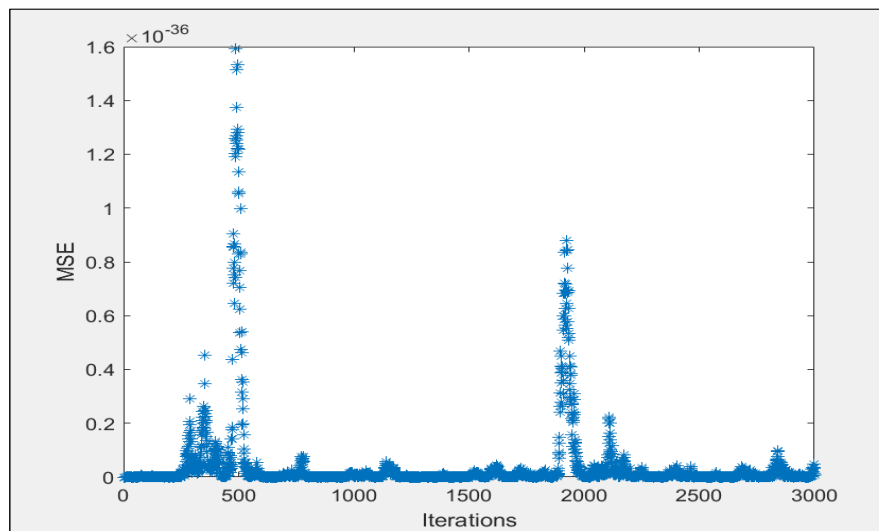


(b) Drones and trucks

Figure 6.4: Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time windows (Uniform-6-n500)



(a) Truck-only



(b) Drones and trucks

Figure 6.5: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time windows (Uniform-6-n500)

Table 6.2: Results of the VRPTW and VRPDTW system performance

Data	VRPTW						VRPDTW						Performance	
	Distance (Mean)	Distance (Standard deviation)	Time (Mean)	Time (Standard deviation)	Running time (Minutes)	Distance (Mean)	Distance (Standard deviation)	Time (Mean)	Time (Standard deviation)	Running time (Minutes)	Time (%)	Distance (%)		
Uniform-71-n50	1145	22.2	418	0.27	7	773	31.2	380	1.16	14	32%	9%		
Uniform-n72-50	1072	18.3	321	1.56	7	810	27.0	311	1.38	14	24%	3%		
Uniform-73-n50	1200	34.6	407	1.85	7	883	14.6	358	1.35	13	26%	12%		
Uniform-91-n100	1800	36.9	403	4.11	10	1448	15.6	365	2.35	28	20%	9%		
Uniform-92-n100	1659	22.9	389	1.84	13	1356	38.8	347	1.47	27	18%	11%		
Uniform-93-100	1622	13.9	302	1.23	8	1405	39.2	273	1.82	31	13%	10%		
Uniform-1-250	2593	63.8	343	4.43	98	2586	57.4	329	1.41	165	0%	4%		
Uniform-2-250	2620	73.0	336	3.20	65	2605	49.8	313	3.14	256	1%	7%		
Uniform-5-500	4186	98.4	327	4.99	191	4061	77.4	304	4.68	816	3%	7%		
Uniform-6-500	4286	77.5	327	3.13	232	4034	93.3	313	2.47	649	6%	4%		

Table 6.3: Hypothesis tests comparing the VRPTW and VRPDTW system with regard to distance and time

	Time			Distance		
Type	Win	Draw	Lose	Win	Draw	Lose
VRPTW	0	0	10	0	2	8
VRPDTW	10	0	0	8	2	0

Chapter 7

Conclusion

Section 7.1 of this final chapter summarises the dissertation. An appraisal of the dissertation's contributions is provided in Section 7.2. Finally, opportunities for future work are described in Section 7.3.

7.1 Summary

In Chapter 1, 'last mile' challenges were presented with emphasis on optimum deliveries of packages. The increased use of technology and access to the internet has resulted in a rise in e-commerce, which has placed significant pressure on last mile deliveries. Innovative ideas of adding different vehicles to assist in deliveries were also highlighted, with drone deliveries being the most popular. The addition of a drone into the delivery system gave rise to the new research area of optimising a truck and drone delivery system. The problem statement was defined, research objectives developed, and a dissertation summary was provided showing where in the dissertation each research objective was fulfilled.

In Chapter 2, popular vehicle routing models were discussed, including the travelling salesman problem (TSP), the vehicle routing problem (VRP), and the vehicle routing problem with time windows (VRPTW). Existing models of truck and drone delivery systems were also reviewed. These models included, amongst others, a truck and drone delivery system where a truck waits for a drone at a customer location, the truck recovers the drone and relaunches it. Other models involving multiple trucks and drones were discussed, where a drone lands at a customer's location. It can be recovered by any truck making deliveries; otherwise, a drone can land and be launched from a hub. Another model which was discussed allowed a truck to stop on its way to making a delivery, launch a drone, continue to make a delivery, and lastly stop at an intermediate location to collect a drone.

Chapter 3 summarised the algorithms such as the TSP, VRP and VRPTW

which are typically used to solve popular routing problems. First, optimisation in general was discussed, then exact algorithms, heuristics, and metaheuristics were described. Solution strategies for solving truck and drone scheduling problems commonly used in literature were also reviewed. A population-based metaheuristic algorithm was chosen to solve the truck and drone scheduling algorithm. The Ant Colony System (ACS) was selected because it is popularly used to solve graph-based problems.

The single truck and drone scheduling problem with interception was introduced in Chapter 4. A mathematical formulation for the problem was provided which allows the drone to intercept the truck at any point on the map. The proposed ACS-based truck and drone scheduling algorithm was also described in Chapter 4. The algorithm was tested on a sample of different sized problems from the TSP-drone data sets of Bouman *et al.* (2015). For the algorithm to output good results, its parameters had to be optimised and tuned to the data. ACS had one parameter that needed tuning, for which a F-race approach was used. Finally, the algorithm was benchmarked against the truck-only (TSP) results provided by Bouman *et al.* (2015) as well as optimal solutions from Knoetze (2021) and solutions obtained by other metaheuristic algorithms in Ernst (2021). Good performance was obtained against a truck-only delivery system for problems of 20 nodes and less. The best performing ACS algorithm obtained the optimal solutions for all 10 node problems but performed between 4.2 and 13.4% worse on the 20-node problems. CPLEX was, however, not able to solve problems larger than 20 nodes within a 24-hour period, whereas the ACS-based algorithm solved problems of up to 500 nodes. The ACS algorithm also performed well against the other metaheuristics.

The **multiple** truck and drone scheduling problem with interception was introduced in Chapter 5. The problems allowed for multiple truck and drone systems delivering goods to customers. An ACS-based truck and drone scheduling algorithm was proposed. The algorithm is based on a route-first, cluster-second approach and utilised a k-means clustering algorithm to cluster customers into different groups. Each group of customers was then scheduled by means of the single truck and drone scheduling algorithm introduced in Chapter 4. The ACS truck and drone scheduling algorithm was benchmarked against a truck-only system which was a standard VRP solved by an ACS algorithm. It was shown that incorporating a drone with a truck for deliveries improves the VRP solution by at least 7% and at most 22% with respect to time and at least 5% and at most 26% with respect to distance.

Chapter 6 introduced the **multiple** truck and drone scheduling problem with interception and **time windows**. A k-means and ACS-based algorithm was proposed to optimise the problem. The developed ACS truck and drone scheduling

algorithm was also benchmarked against a truck-only system which was a standard VRPTW solved by an ACS algorithm. It was shown that incorporating a drone with a truck for deliveries improves the VRPTW solution by at least 3% and at most 12% with respect to time and obtains at least similar performance and at most improves distance by 32%.

7.2 Appraisal of dissertation contributions

The contributions of this dissertation are threefold. This section gives an overview and appraisal of these contributions.

- **Contribution 1** *The development of a truck and drone scheduling algorithm for the **single** truck and drone scheduling problem with interception.*

The algorithm developed in this dissertation was the first approximation algorithm developed to solve the single truck and drone scheduling problem with interception. All other truck and drone scheduling algorithms allow trucks and drones to intercept each other at predetermined points, whereas the scenario considered in this dissertation allows for truck and drone interception at **any** point on the map. It is also notable that the developed algorithm is able to solve problems of significantly larger sizes than previously documented in the truck and drone scheduling literature.

- **Contribution 2** *The development of a truck and drone scheduling algorithm for the **multiple** truck and drone scheduling problem with interception.*

This dissertation is the first publication to consider the **multiple** truck and drone scheduling problem with interception at any point on the map. Benchmarking against a classical VRP showed the performance benefits achievable by incorporating drones into a multi-truck delivery system.

- **Contribution 3** *The development of a truck and drone scheduling algorithm for the **multiple** truck and drone scheduling problem with interception and **time windows**.*

This dissertation is also the first publication to consider the **multiple** truck and drone scheduling problem with **time windows** and interception at any point on the map. Benchmarking against a classical VRPTW showed the performance benefits achievable by incorporating drones into a multi-truck delivery system where time window constraints need to be met.

7.3 Future Research Opportunities

This dissertation investigated the optimisation of a delivery system comprised of trucks and drones. Various ACS and k-means based optimisation algorithms were developed to solve a number of truck and drone scheduling problem variations. This research led to a number of areas for future research being identified. The following opportunities are either aimed at improving the realism of the problem or improving and investigating the algorithm efficiency:

- Develop and solve a model that will allow a drone to launch anywhere on the arc between two truck deliveries and intercept the truck anywhere on the road network. This modification will allow the truck to continue making deliveries and not waste time waiting for the drone and this modification will therefore improve delivery time.
- Develop and solve a model that will allow a truck to travel with more than one drone to complete deliveries.
- Develop and solve a model that will allow a drone to be launched and picked up by any truck during deliveries.
- Develop and solve a model that will allow a drone to complete multiple deliveries in between truck deliveries.
- Develop and solve a model that will allow for unique customer demands and time windows.
- Incorporate a local search algorithm into the ACS-based truck and drone scheduling algorithms to improve the efficiency of the ACS-based algorithms.
- Determine the largest problem size for each problem variation that can be solved optimally and validate ACS solutions of large data sets against the optimal solutions.
- Conduct further benchmarking of all problem variations against other meta-heuristic algorithms to obtain a better idea of algorithm performance.
- Conduct parameter tuning of the ACS algorithm specifically for each problem variation, or adopt self-adaptive parameters for the ACS algorithm so that the best algorithm control parameters are used for each problem variation.
- Incorporate the use of multi-objective optimisation to optimise both distance and time, simultaneously.

- Solve a dynamic version of the truck and drone scheduling problem to incorporate changing travel times or new deliveries.
- Investigate the use of alternative clustering methods in the multiple drone-truck scheduling algorithms.
- Investigate the performance of the ACS algorithm on more benchmark problem instances of different types.

List of References

- Agatz, N., Bouman, P. and Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, vol. 52, no. 4, pp. 965–981.
- Applegate, D., Bixby, R., Cook, W. and Chvatal, V. (1998). On the solution of travelling salesman problems. *Documenta Mathematica*, pp. 645–656.
- Auger, A. and Hansen, N. (2005). A restart cma evolution strategy with increasing population size. In: *2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1769–1776. IEEE.
- Baker, E.K. and Schaffer, J.R. (1986). Solution improvement heuristics for the vehicle routing and scheduling problem with time window constraints. *American Journal of Mathematical and Management Sciences*, vol. 6, no. 3-4, pp. 261–300.
- Bakir, I. and Tiniç, G.Ö. (2020). Optimizing drone-assisted last-mile deliveries: the vehicle routing problem with flexible drones. *Optimization-online.org*, pp. 1–28.
- Balas, E. and Toth, P. (1985). Branch and bound methods for the traveling salesman problem. In: Lawler, E.L., Lenstra, J., Rinnooy-Kan, A. and Shmoys, D. (eds.), *The traveling salesman problem. A guided tour of combinatorial optimization*, pp. 1–63. J. Wiley.
- Beasley, J.E. (1983). Route first-cluster second methods for vehicle routing. *Omega*, vol. 11, no. 4, pp. 403–408.
- Bektas, T. (2006). The multiple traveling salesman problem: an overview of formulations and solution procedures. *Omega*, vol. 34, no. 3, pp. 209–219.
- bin Othman, M.S., Shurbevski, A., Karuno, Y. and Nagamochi, H. (2017). Routing of carrier-vehicle systems with dedicated last-stretch delivery vehicle and fixed carrier route. *Journal of Information Processing*, vol. 25, pp. 655–666.
- Birattari, M., Yuan, Z., Balaprakash, P. and Stützle, T. (2009). Automated algorithm tuning using f-races: Recent developments. In: *Proceedings of MIC*, vol. 9, pp. 1–10.

- Blum, C. and Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys (CSUR)*, vol. 35, no. 3, pp. 268–308.
- Bouman, P., Agatz, N. and Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, vol. 72, no. 4, pp. 528–542.
- Bouman, P.C., Schmidt, M. and Agatz, N.A.H. (2015). Instances for the TSP with drone. Available at: <http://dx.doi.org/10.5281/zenodo.22245>.
- Bräysy, O. and Gendreau, M. (2005a). Vehicle routing problem with time windows, part i: Route construction and local search algorithms. *Transportation science*, vol. 39, no. 1, pp. 104–118.
- Bräysy, O. and Gendreau, M. (2005b). Vehicle routing problem with time windows, part ii: Metaheuristics. *Transportation science*, vol. 39, no. 1, pp. 119–139.
- Brooke-Holland, L. (2013). Unmanned Aerial Vehicles (drones): An Introduction. Available at: <https://fas.org/irp/world/uk/drones.pdf>
- Carlsson, J.G. and Song, S. (2018). Coordinated logistics with a truck and a drone. *Management Science*, vol. 64, no. 9, pp. 4052–4069.
- Černý, V. (1985). Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of Optimization Theory and Applications*, vol. 45, no. 1, pp. 41–51.
- Chung, S.H., Sah, B. and Lee, J. (2020). Optimization for drone and drone-truck combined operations: A review of the state of the art and future directions. *Computers & Operations Research*, vol. 123, p. 105004.
- Clarke, G. and Wright, J.W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, vol. 12, no. 4, pp. 568–581.
- Clausen, J. (1999). *Branch and bound algorithms-principles and examples*. Ph.D. thesis, University of Copenhagen.
- Daknama, R. and Kraus, E. (2017). Vehicle routing with drones. *arXiv preprint arXiv:1705.06431*.
- Dantzig, G., Fulkerson, R. and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, vol. 2, no. 4, pp. 393–410.
- Dantzig, G.B. and Ramser, J.H. (1959). The truck dispatching problem. *Management Science*, vol. 6, pp. 80–91.

- Dayarian, I., Savelsbergh, M. and Clarke, J. (2020). Same-day delivery with drone resupply. *Transportation Science*, vol. 54, no. 1, pp. 229–249.
- Dell’Amico, M. and Hadjidimitriou, S. (2012). Innovative logistics model and containers solution for efficient last mile delivery. *Procedia-Social and Behavioral Sciences*, vol. 48, pp. 1505–1514.
- Desrochers, M., Desrosiers, J. and Solomon, M. (1992). A new optimization algorithm for the vehicle routing problem with time windows. *Operations Research*, vol. 40, no. 2, pp. 342–354.
- Di Puglia Pugliese, L. and Guerriero, F. (2017). Last-mile deliveries by using drones and classical vehicles. In: *International Conference on Optimization and Decision Science*, pp. 557–565. Springer.
- Dinh, Q.T., Do, D.D. and Hà, M.H. (2021). Ants can solve the parallel drone scheduling traveling salesman problem. In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 14–21.
- Dorigo, M. (1992). Optimization, learning and natural algorithms. *Ph.D. Thesis, Politecnico di Milano*.
- Dorigo, M. and Di Caro, G. (1999). Ant colony optimization: a new meta-heuristic. In: *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, vol. 2, pp. 1470–1477. IEEE.
- Dorling, K., Heinrichs, J., Messier, G.G. and Magierowski, S. (2016). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 1, pp. 70–85.
- Eksioglu, B., Vural, A.V. and Reisman, A. (2009). The vehicle routing problem: A taxonomic review. *Computers & Industrial Engineering*, vol. 57, no. 4, pp. 1472–1483.
- Elsayed, M. and Mohamed, M. (2020). The impact of airspace regulations on unmanned aerial vehicles in last-mile operation. *Transportation Research Part D: Transport and Environment*, vol. 87, p. 102480.
- Engelbrecht, A.P. (2007). *Computational intelligence: an introduction*. John Wiley & Sons.
- Ernst, R. (2021). *Efficiency study of various algorithms on the travelling salesperson problem with drone and with interceptions*. Master’s thesis, University of Stellenbosch.
- Eun, J., Song, B.D., Lee, S. and Lim, D. (2019). Mathematical investigation on the sustainability of UAV logistics. *Sustainability*, vol. 11, no. 21, p. 5932.
- Feo, T.A. and Resende, M.G. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, vol. 6, no. 2, pp. 109–133.

- Ferrandez, S.M., Harbison, T., Weber, T., Sturges, R. and Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management (JIEM)*, vol. 9, no. 2, pp. 374–388.
- Fischetti, M., Salazar González, J.J. and Toth, P. (1997). A branch-and-cut algorithm for the symmetric generalized traveling salesman problem. *Operations Research*, vol. 45, no. 3, pp. 378–394.
- Flood, M.M. (1956). The traveling-salesman problem. *Operations Research*, vol. 4, no. 1, pp. 61–75.
- Fogel, L.J. (1962). Toward inductive inference automata. In: *IFIP Congress*, vol. 62, pp. 395–400.
- Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966). Artificial intelligence through simulated evolution. *Wiley New York*.
- Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research*, vol. 13, no. 5, pp. 533–549.
- Glover, F. and Laguna, M. (1997). Tabu search. *Kluwer Academic Publishers*.
- Gomory, R.E. (1963). An algorithm for integer solutions to linear programs. *Recent Advances in Mathematical Programming*, vol. 64, no. 260-302, p. 14.
- Grobler, J. (2015). *The heterogeneous meta-hyper-heuristic: from low level heuristics to low level meta-heuristics*. Ph.D. thesis, University of Pretoria.
- Ha, Q.M., Deville, Y., Pham, Q.D. and Hà, M.H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, vol. 86, pp. 597–621.
- Haidari, L.A., Brown, S.T., Ferguson, M., Bancroft, E., Spiker, M., Wilcox, A., Ambikapathi, R., Sampath, V., Connor, D.L. and Lee, B.Y. (2016). The economic and operational value of using drones to transport vaccines. *Vaccine*, vol. 34, no. 34, pp. 4062–4067.
- Ham, A.M. (2018). Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 1–14.
- Hansen, N., Arnold, D.V. and Auger, A. (2015). Evolution strategies. In: *Springer handbook of computational intelligence*, pp. 871–898. Springer.
- Hansen, P. and Mladenović, N. (1999). An introduction to variable neighborhood search. in metaheuristics: Advances and trends in local search paradigms for optimization. *Kluwer Academic Publishers*, pp. 433–458.

- Hansen, P. and Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European Journal of Operational Research*, vol. 130, no. 3, pp. 449–467.
- Holland, J. (1975). *Adaptation in natural and artificial systems*. University of Michigan Press.
- Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In: *Proceedings of ICNN'95-International Conference on Neural Networks*, vol. 4, pp. 1942–1948. IEEE.
- Kirkpatrick, S., Gelatt, C.D. and Vecchi, M.P. (1983). Optimization by simulated annealing. *Science*, vol. 220, no. 4598, pp. 671–680.
- Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J.M. and Brunese, P.A. (2019). Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, vol. 129, pp. 14–30.
- Knight, R. (2016). Drones In Health Care. <https://www.dronesinhealthcare.com/>, Last accessed on 2 Feb 2019.
- Knoetze, F.E. (2021). Solving a last mile truck and drone delivery schedule to optimality. Final year project, University of Stellenbosch.
- Land, A. (2021). The solution of some 100-city travelling salesman problems. *EURO Journal on Computational Optimization*, pp. 1–8.
- Laporte, G. (1992). The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, vol. 59, no. 3, pp. 345–358.
- Lieberman, G.J. and Hillier, F.S.H. (2005). *Operations Research*. 8th edn. McGraw Hil.
- Lim, S.F.W., Jin, X. and Srari, J.S. (2018). Consumer-driven e-commerce: A literature review, design framework, and research agenda on last-mile logistics models. *International Journal of Physical Distribution & Logistics Management*, vol. 48, no. 3, pp. 308–332.
- Lin, S. (1965). Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, vol. 44, no. 10, pp. 2245–2269.
- Lin, S. and Kernighan, B.W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, vol. 21, no. 2, pp. 498–516.
- Liu, J., Guan, Z., Shang, J. and Xie, X. (2018). Application of drone in solving last mile parcel delivery. *Journal of Systems Science and Information*, vol. 6, no. 4, pp. 302–319.

- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, vol. 1, pp. 281–297. Oakland, CA, USA.
- Macrina, G., Pugliese, L.D.P., Guerriero, F. and Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, vol. 120, p. 102762.
- Marinelli, M., Caggiani, L., Ottomanelli, M. and Dell’Orco, M. (2018). En route truck–drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems*, vol. 12, no. 4, pp. 253–261.
- Martin, G.T. (1966). Solving traveling salesman problem by integer linear programming. *Operations Research*, p. B71.
- Miliotis, P. (1976). Integer programming approaches to the travelling salesman problem. *Mathematical Programming*, vol. 10, no. 1, pp. 367–378.
- Miliotis, P. (1978). Using cutting planes to solve the symmetric travelling salesman problem. *Mathematical Programming*, vol. 15, no. 1, pp. 177–188.
- Min, H. (1989). The multiple vehicle routing problem with simultaneous delivery and pick-up points. *Transportation Research Part A: General*, vol. 23, no. 5, pp. 377–386.
- Mitchell, M. (1998). *An introduction to genetic algorithms*. MIT press.
- Murray, C.C. and Chu, A.G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, vol. 54, pp. 86–109.
- Naddef, D. and Thienel, S. (2002). Efficient separation routines for the symmetric traveling salesman problem ii: separating multi handle inequalities. *Mathematical Programming*, vol. 92, no. 2, pp. 257–283.
- Padberg, M. and Rinaldi, G. (1987). Optimization of a 532-city symmetric traveling salesman problem by branch and cut. *Operations Research Letters*, vol. 6, no. 1, pp. 1–7.
- Padberg, M. and Rinaldi, G. (1991). A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *SIAM Review*, vol. 33, no. 1, pp. 60–100.
- Perera, S., Dawande, M., Janakiraman, G. and Mookerjee, V. (2020). Retail deliveries by drones: How will logistics networks change? *Production and Operations Management*, vol. 29, no. 9, pp. 2019–2034.
- Poikonen, S. and Campbell, J.F. (2021). Future directions in drone routing research. *Networks*, vol. 77, no. 1, pp. 116–126.

- Poikonen, S., Golden, B. and Wasil, E.A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, vol. 31, no. 2, pp. 335–346.
- Ponza, A. (2016). *Optimization of drone-assisted parcel delivery*. Master's thesis, Università Degli Studi di Padova.
- Qin, A.K. and Suganthan, P.N. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In: *2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785–1791. IEEE.
- Rabta, B., Wankmüller, C. and Reiner, G. (2018). A drone fleet model for last-mile distribution in disaster relief operations. *International Journal of Disaster Risk Reduction*, vol. 28, pp. 107–112.
- Raj, A. and Sah, B. (2019). Analyzing critical success factors for implementation of drones in the logistics sector using grey-dematel based approach. *Computers & Industrial Engineering*, vol. 138, pp. 106–118.
- Raj, R. and Murray, C. (2020). The multiple flying sidekicks traveling salesman problem with variable drone speeds. *Transportation Research Part C: Emerging Technologies*, vol. 120, pp. 102–813.
- Raj, R. and Murray, C. (2021). The time-dependent multiple flying sidekicks traveling salesman problem: Parcel delivery with traffic congestion. *Available at SSRN*.
- Ranieri, L., Digiesi, S., Silvestri, B. and Roccotelli, M. (2018). A review of last mile logistics innovations in an externalities cost reduction vision. *Sustainability*, vol. 10, no. 3, pp. 782–800.
- Rechenberg, I. (1973). Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution. *Frommann-Holzboog*.
- Rosenkrantz, D.J., Stearns, R.E. and Lewis, II, P.M. (1977). An analysis of several heuristics for the traveling salesman problem. *SIAM Journal on Computing*, vol. 6, no. 3, pp. 563–581.
- Ryan, H. (2016). Domino's delivers its first pizza using a drone . <https://www.nzherald.co.nz/business/news/article>, Last accessed on 1 June 2019.
- Schermer, D. (2019). Integration of drones in last-mile delivery: The vehicle routing problem with drones. In: *Operations Research Proceedings 2018*, pp. 17–22. Springer.
- Schermer, D., Moeini, M. and Wendt, O. (2018). Algorithms for solving the vehicle routing problem with drones. In: *Asian Conference on Intelligent Information and Database Systems*, pp. 352–361. Springer.

- Schermer, D., Moeini, M. and Wendt, O. (2020). A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone. *Networks*, vol. 76, no. 2, pp. 164–186.
- Scott, J. and Scott, C. (2017). Drone delivery models for healthcare. In: *Proceedings of the 50th Hawaii International Conference on System Sciences*, pp. 3297–3304.
- Slabinac, M. (2015). Innovative solutions for a “last-mile” delivery—a European experience. *Business Logistics in Modern Management*, pp. 111–129.
- Solomon, M.M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, vol. 35, no. 2, pp. 254–265.
- Spires, J. (2019 7). Amazon Prime Air - from eCommerce to drone delivery . <https://www.dronesdj.com/>, Last accessed on 20 July 2019.
- Thiels, C.A., Aho, J.M., Zietlow, S.P. and Jenkins, D.H. (2015). Use of unmanned aerial vehicles for medical product transport. *Air Medical Journal*, vol. 34, no. 2, pp. 104–108.
- Ulmer, M.W. and Thomas, B.W. (2018). Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks*, vol. 72, no. 4, pp. 475–505.
- Van den Bergh, F. and Engelbrecht, A.P. (2002). A new locally convergent particle swarm optimiser. In: *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 6–11. IEEE.
- Voudouris, C. and Tsang, E. (1999). Guided local search and its application to the traveling salesman problem. *European Journal of Operational Research*, vol. 113, no. 2, pp. 469–499.
- Wang, X., Poikonen, S. and Golden, B. (2017). The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, vol. 11, no. 4, pp. 679–697.
- Wang, Z. and Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, vol. 122, pp. 350–364.
- Wren, A. and Holliday, A. (1972). Computer scheduling of vehicles from one or more depots to a number of delivery points. *Journal of the Operational Research Society*, vol. 23, no. 3, pp. 333–344.
- Yadav, J. and Sharma, M. (2013). A review of k-mean algorithm. *International Journal of Engineering Trends and Technology*, vol. 4, no. 7, pp. 2972–2976.
- Yurek, E.E. and Ozmutlu, H.C. (2018). A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, vol. 91, pp. 249–262.

Appendices

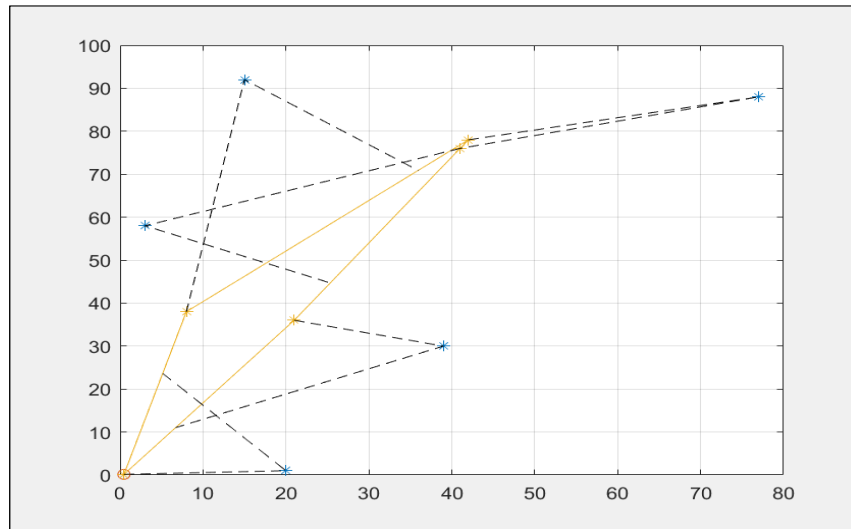
Additional drone and truck scheduling problem with interception examples

This appendix contains additional truck and drone scheduling problem with interception examples. A graphical representation of the ACS-DT1 and ACS-DT2 solutions are shown in Figures 1a, 2a, 3a, 4a, 5a, 6a, 9a, 10a, 11a, 12a, 13a, 14a, 15a, 16a, 17a, 18a, 19a, and 31a. Here customers receive parcels from drone and truck.

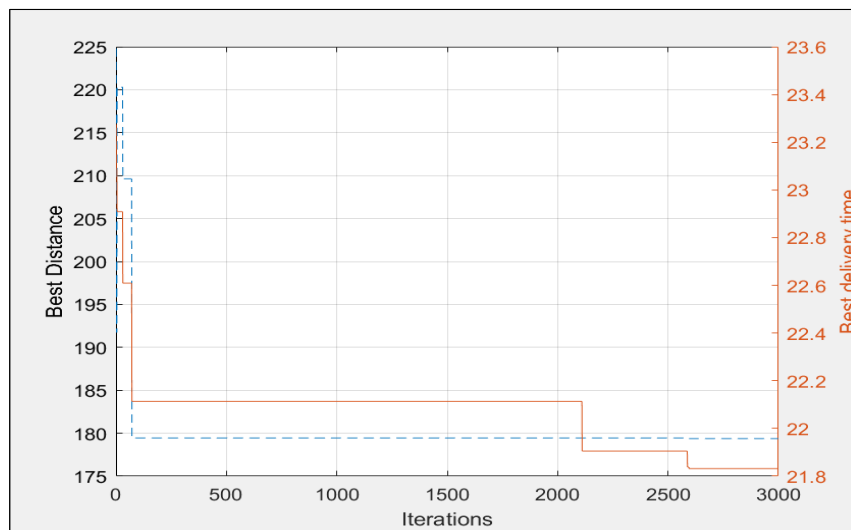
The best cost per iteration for the datasets is shown in Figure 1b, 2b, 3b, 4b, 5b, 6b, 9b, 10b, 11b, 12b, 13b, 14b, 15b, 16b, 17b, 18b, 19b, and 20b.

The ants solutions search per iteration is shown in Figure 1c, 2c, 3c, 4c, 5c, 6c, 9c, 10c, 11c, 12c, 13c, 14c, 15c, 16c, 17c, 18c, 19c, and 20c

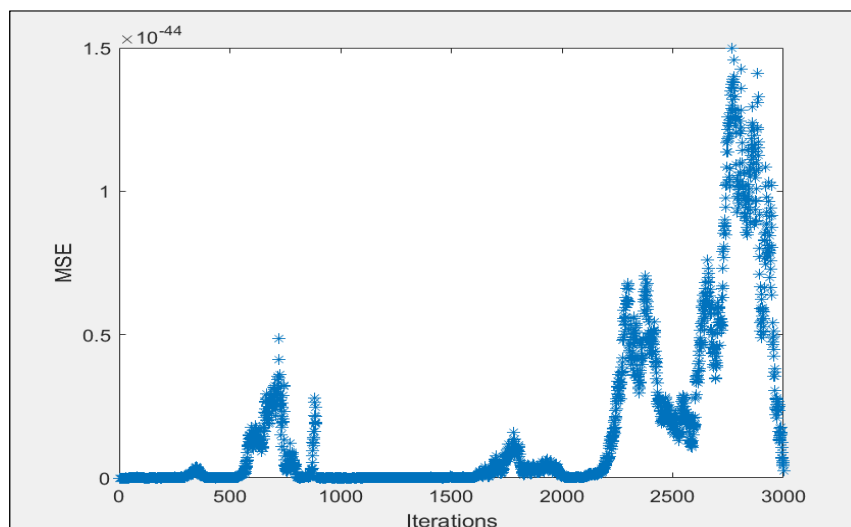
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path

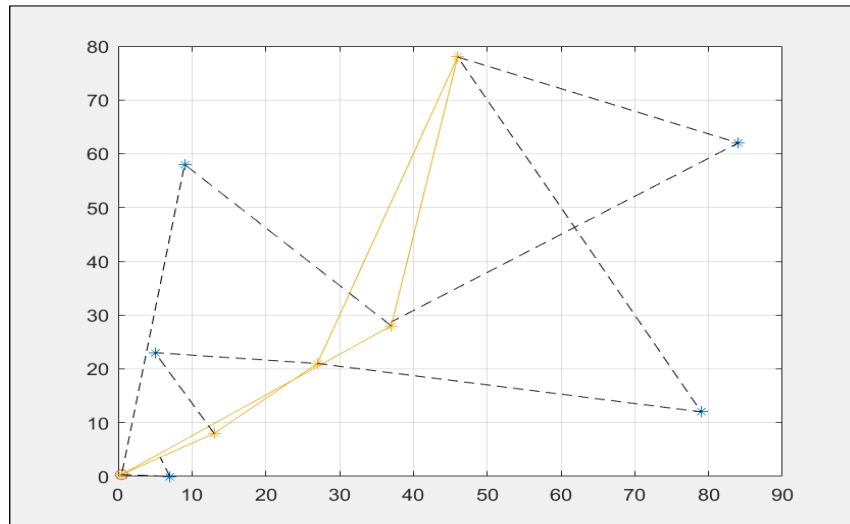


(b) Best cost per iteration

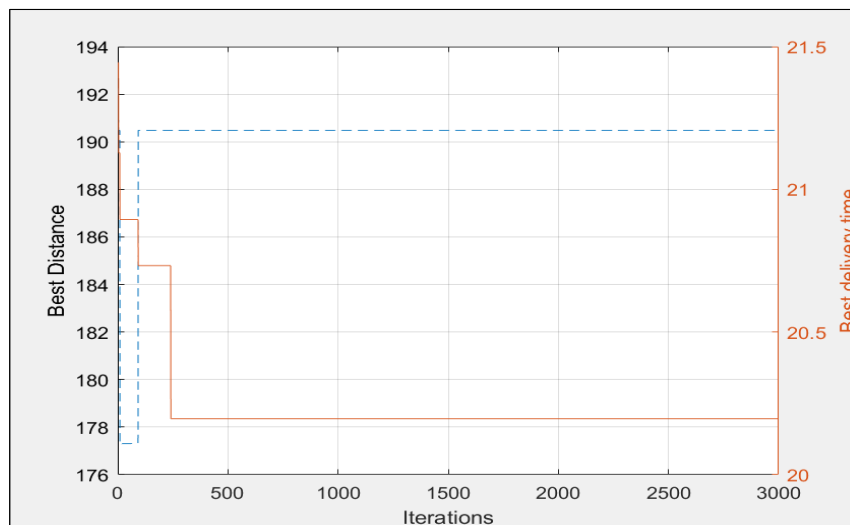


(c) Ant diversity per iteration

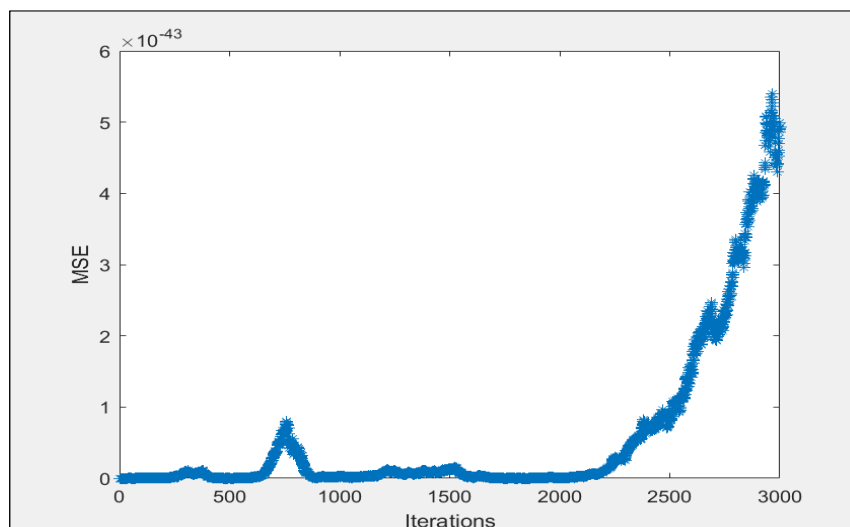
Figure 1: The drone and truck solution for 10 nodes (uniform-52-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10



(a) Best path

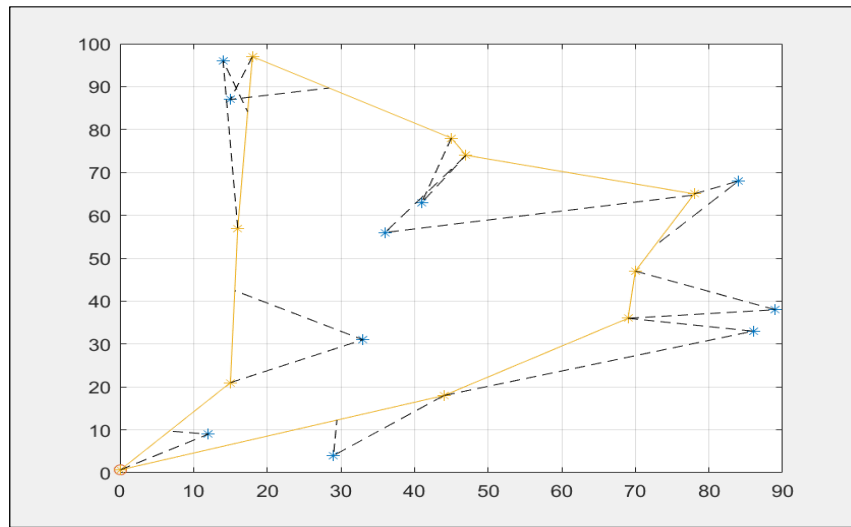


(b) Best cost per iteration

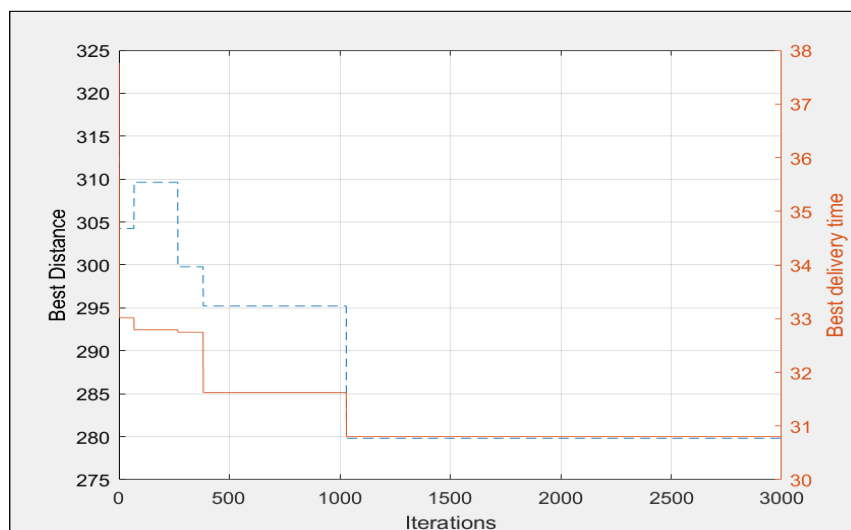


(c) Ant diversity per iteration

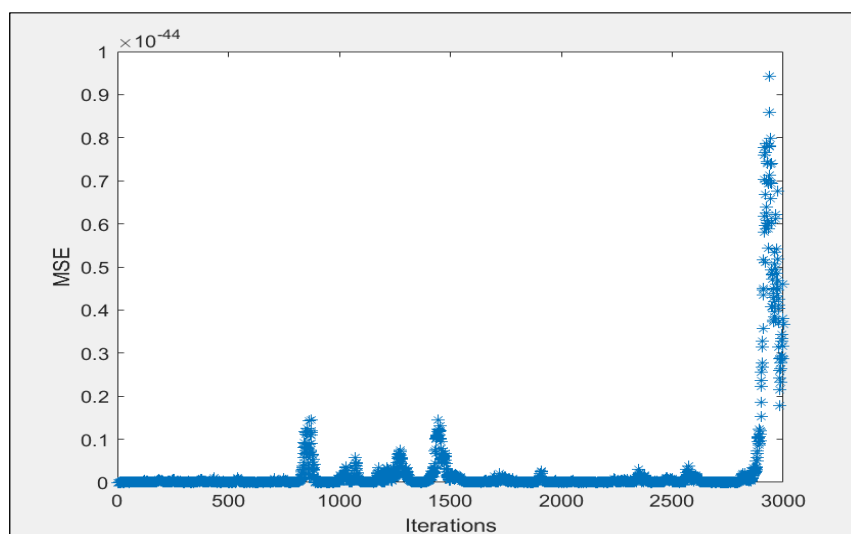
Figure 2: The drone and truck solution for 10 nodes (uniform-53-n10) with drone speed (v_d) of 20 and truck speed v_t of 10



(a) Best path

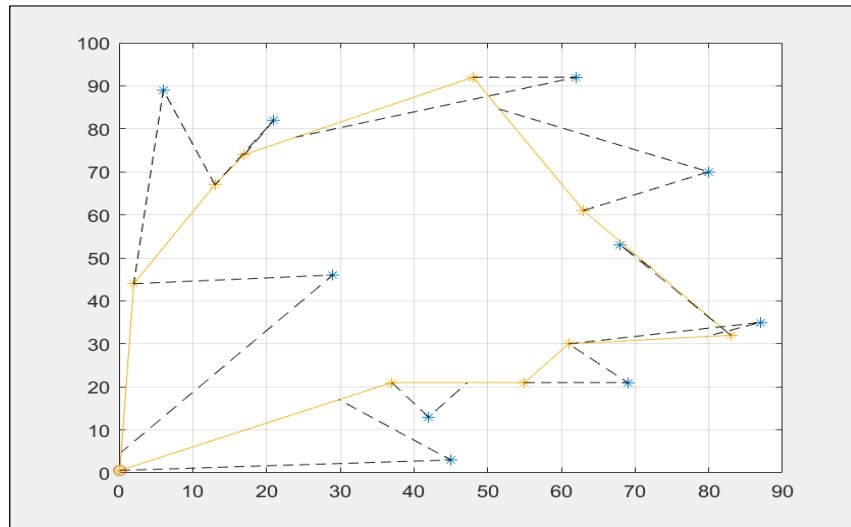


(b) Best cost per iteration

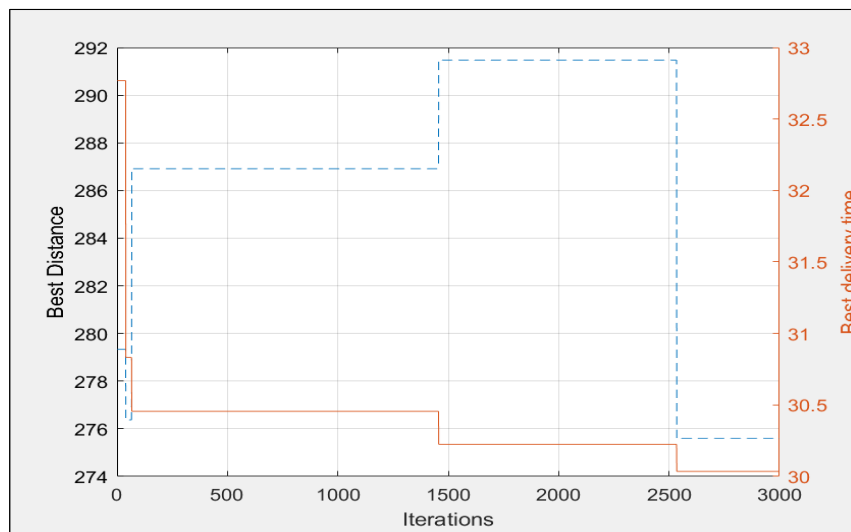


(c) Ant diversity per iteration

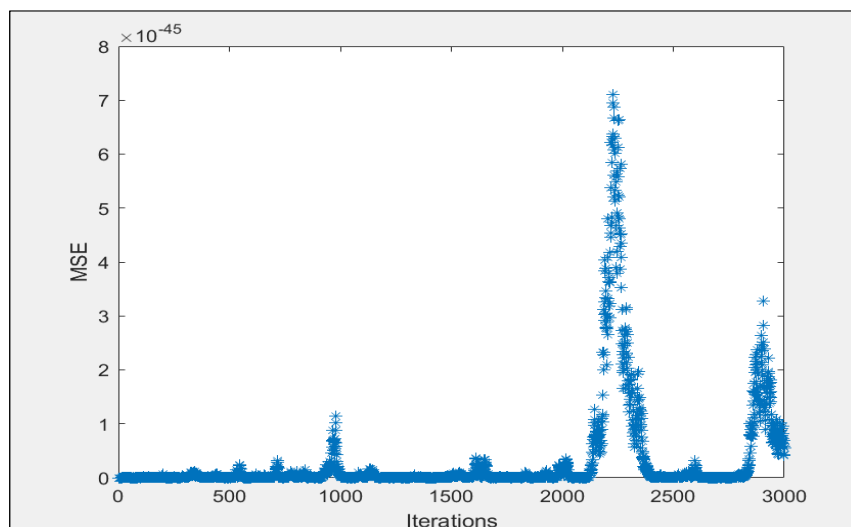
Figure 3: The drone and truck solution for 20 nodes (uniform-62-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10



(a) Best path

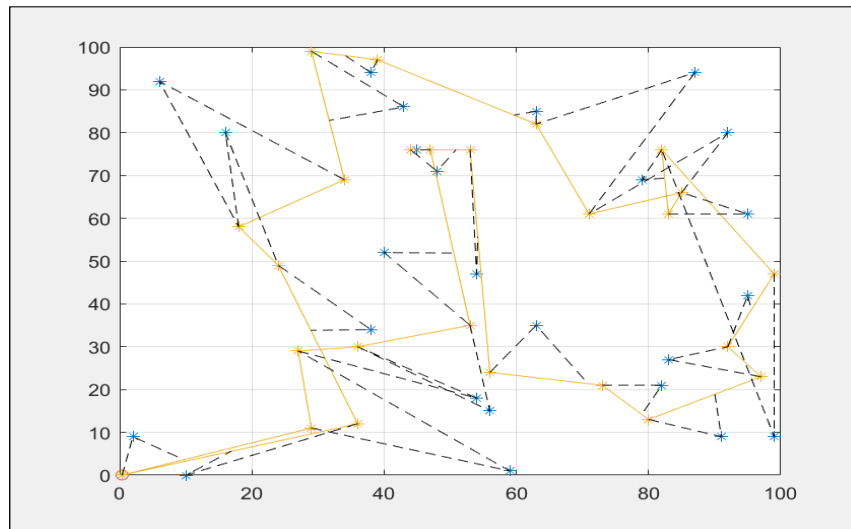


(b) Best cost per iteration

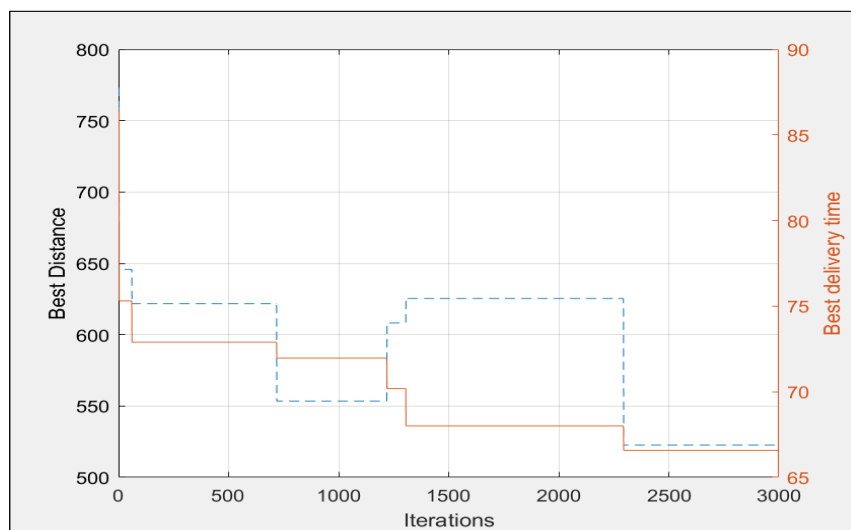


(c) Ant diversity per iteration

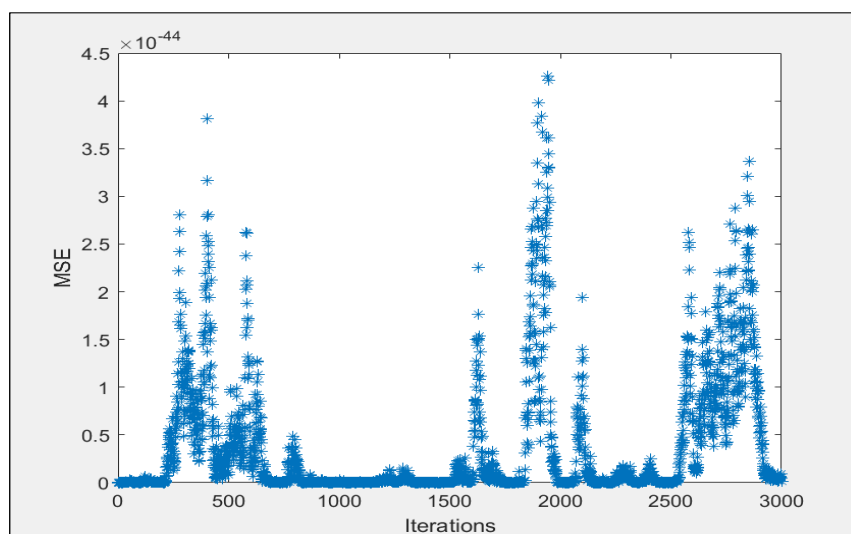
Figure 4: The drone and truck solution for 20 nodes (uniform-63-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10



(a) Best path

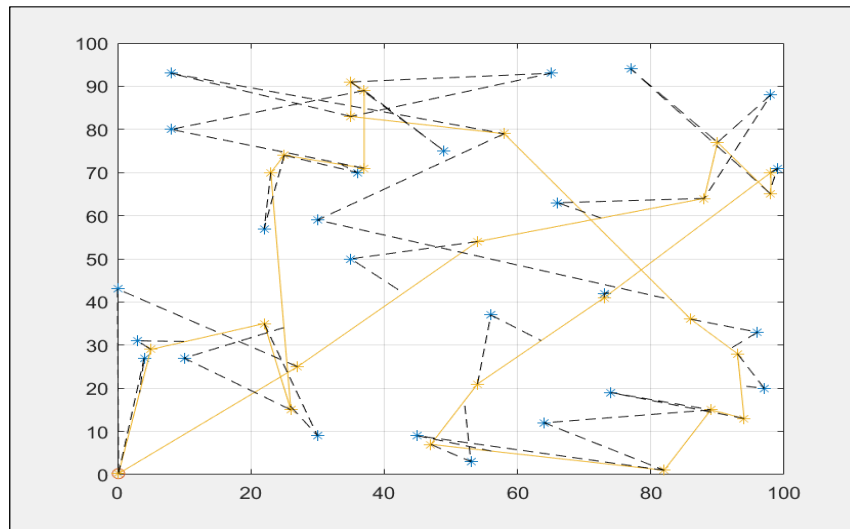


(b) Best cost per iteration

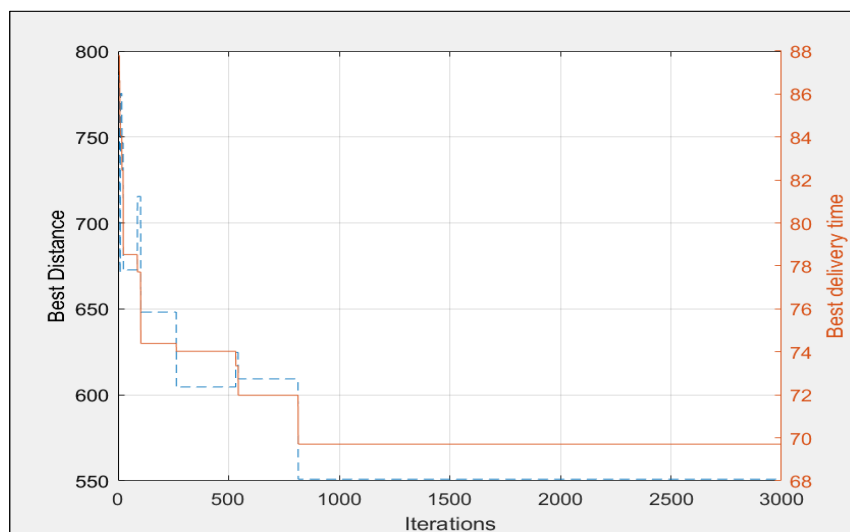


(c) Ant diversity per iteration

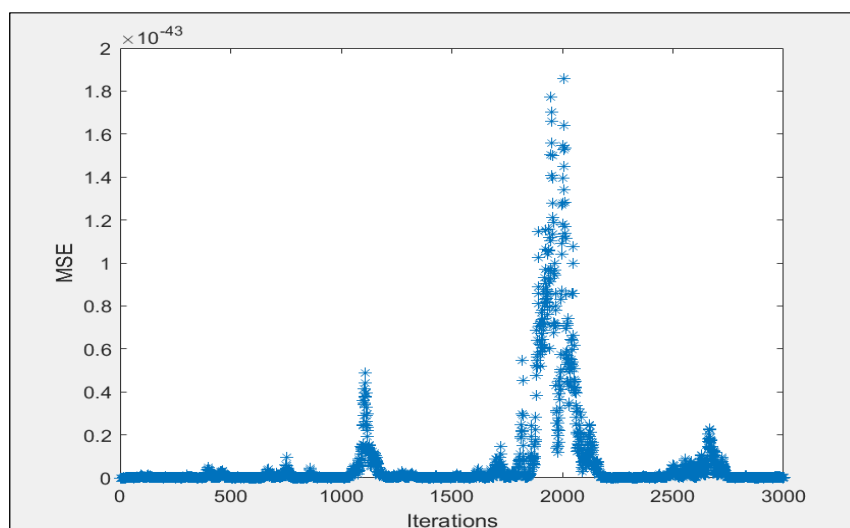
Figure 5: The drone and truck solution for 50 nodes (uniform-71-n50) with drone speed (v_d) of 20 and truck speed v_t of 10



(a) Best path



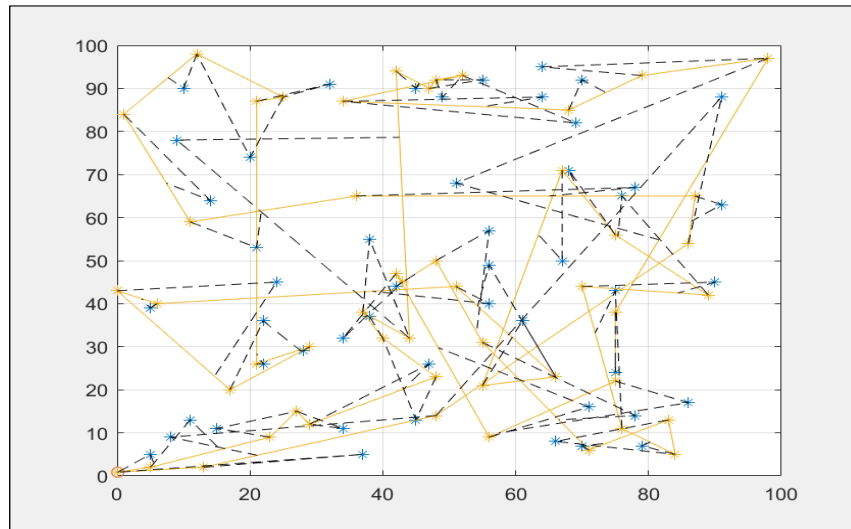
(b) Best cost per iteration



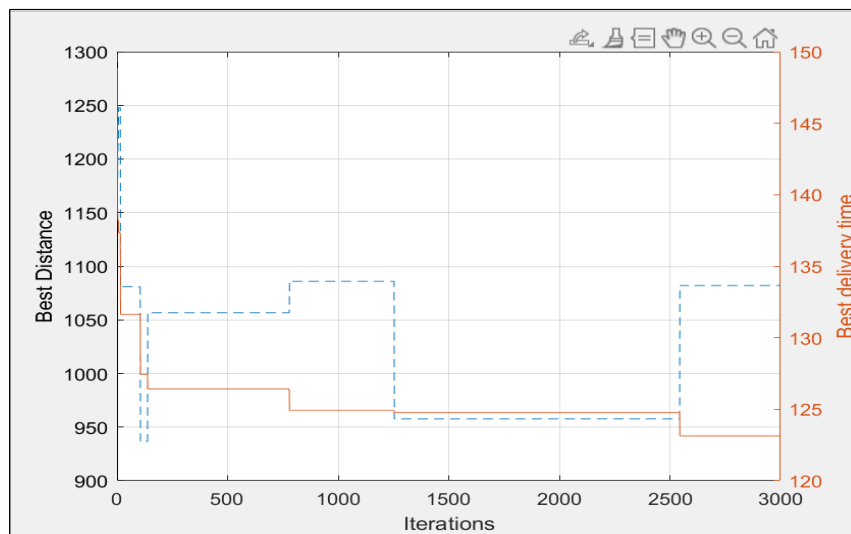
(c) Ant diversity per iteration

Figure 6: The drone and truck solution for 50 nodes (uniform-73-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10

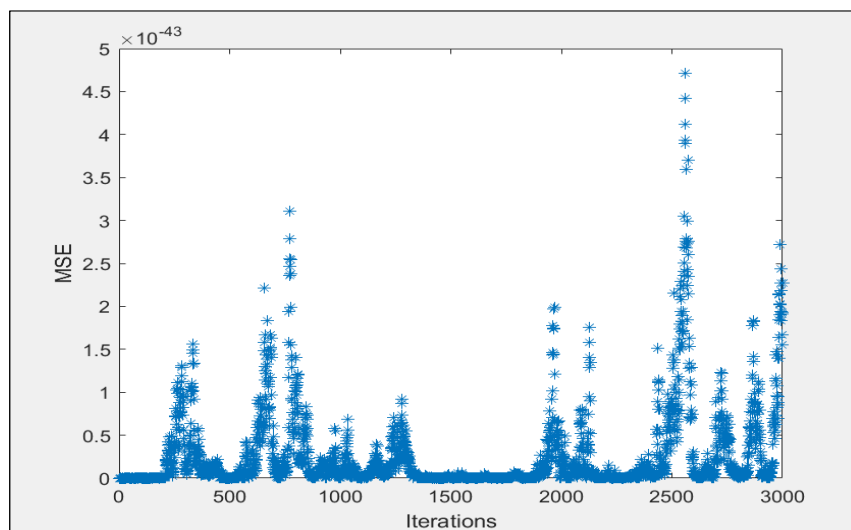
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



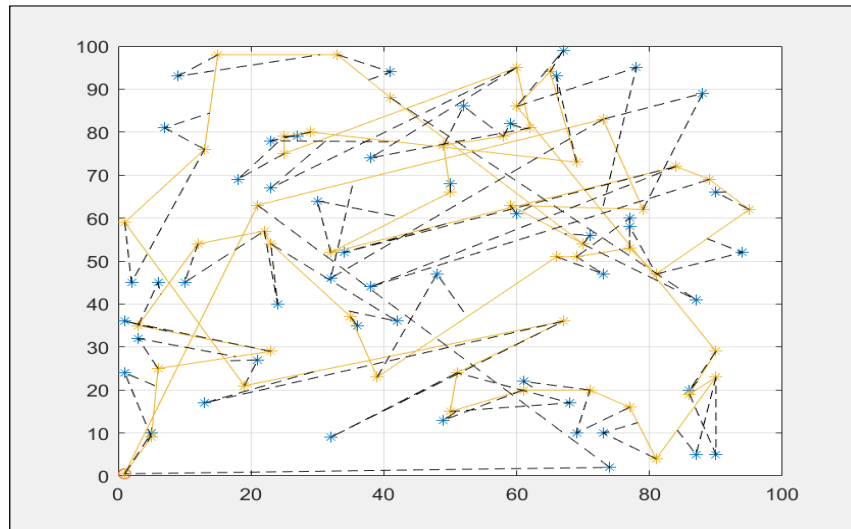
(b) Best cost per iteration



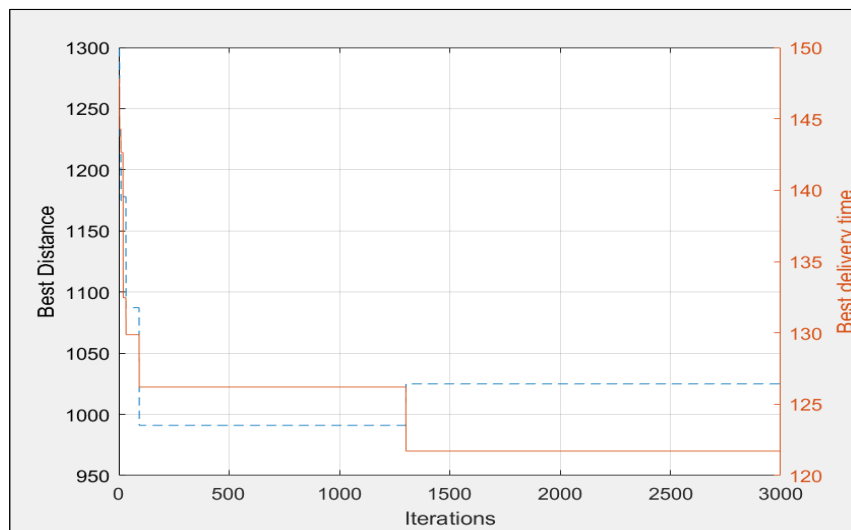
(c) Ant diversity per iteration

Figure 7: The drone and truck solution for 100 nodes (uniform-92-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10

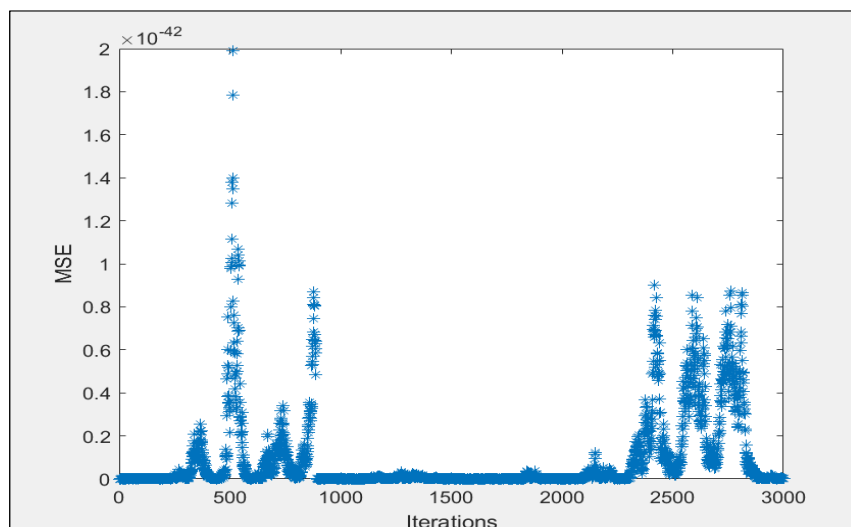
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



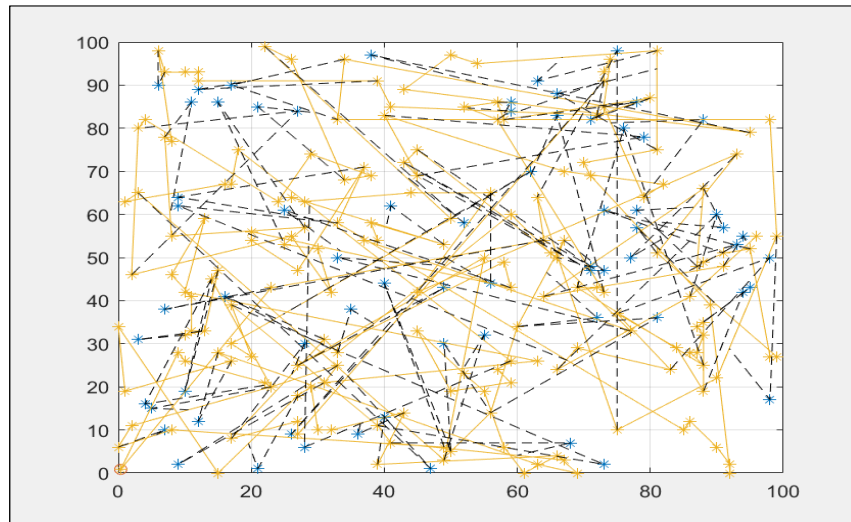
(b) Best cost per iteration



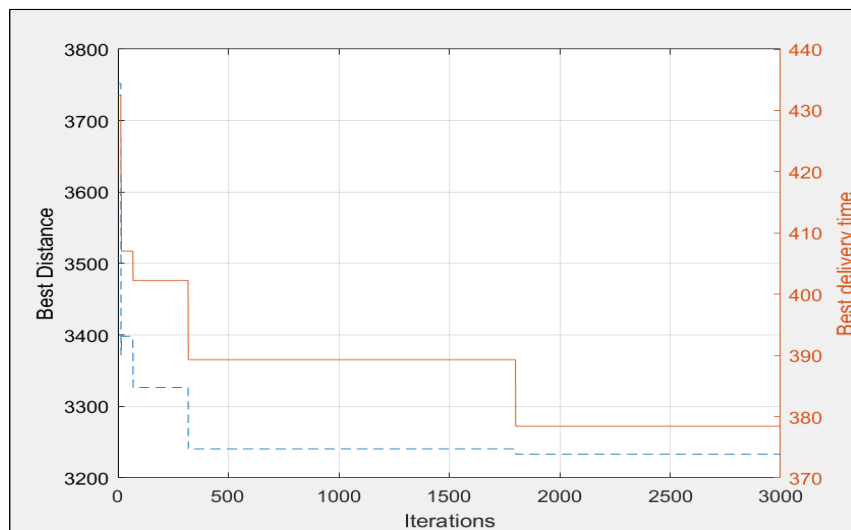
(c) Ant diversity per iteration

Figure 8: The drone and truck solution for 100 nodes (uniform-92-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10

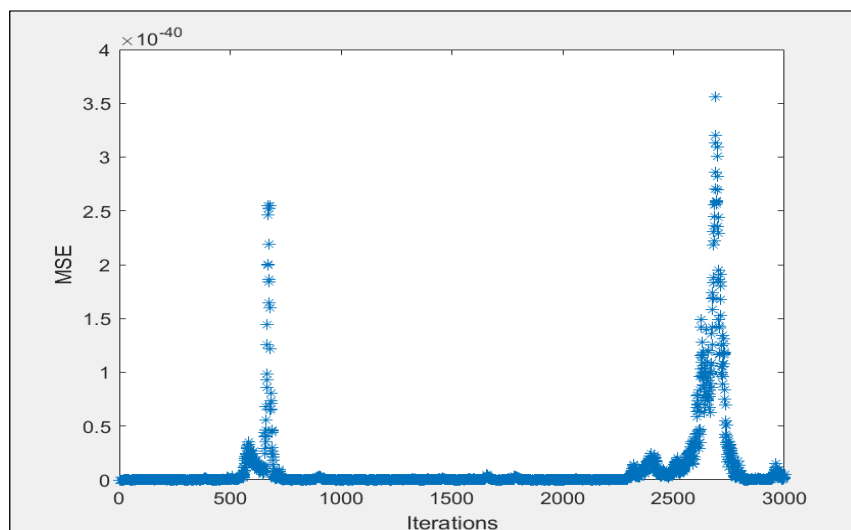
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path

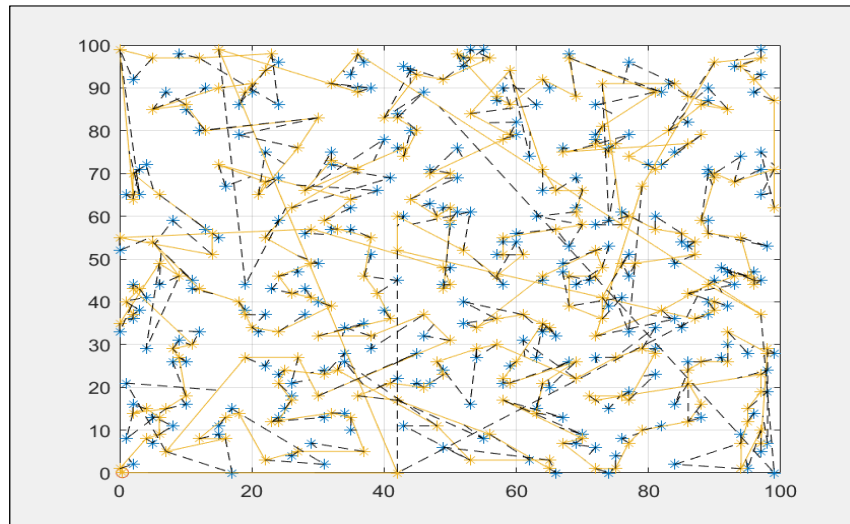


(b) Best cost per iteration

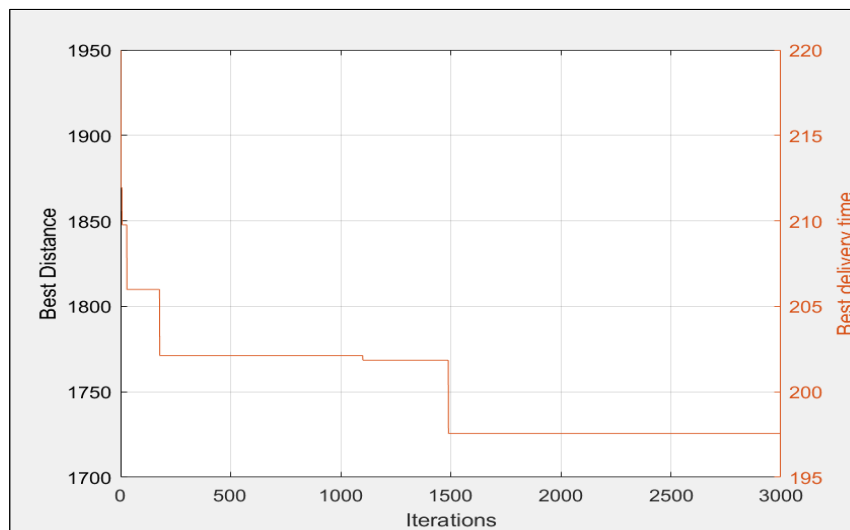


(c) Ant diversity per iteration

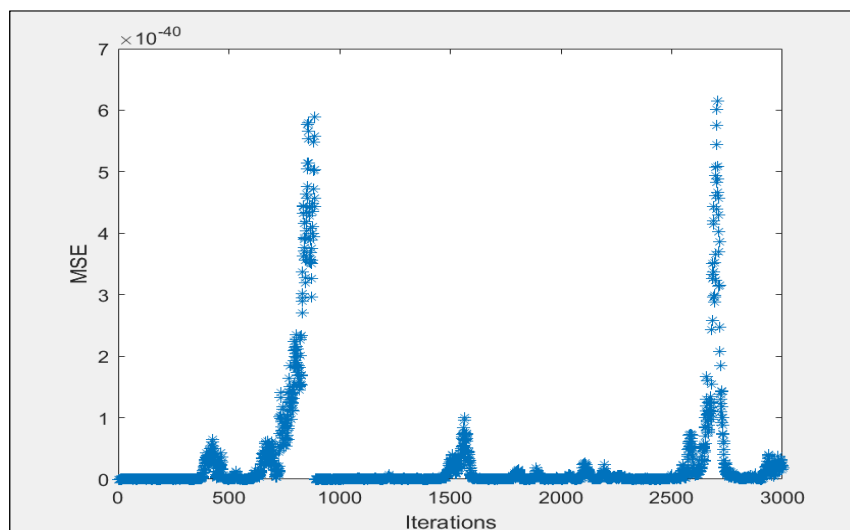
Figure 9: The drone and truck solution for 250 nodes (uniform-2-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10



(a) Best path

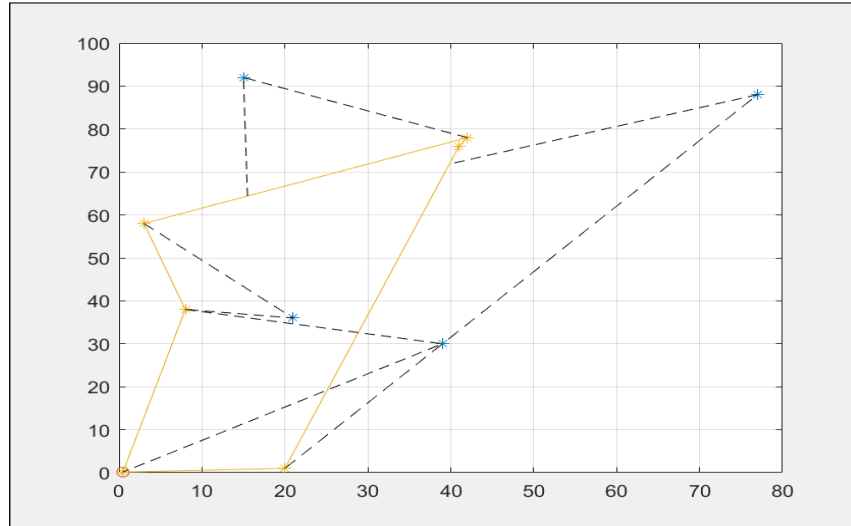


(b) Best cost per iteration

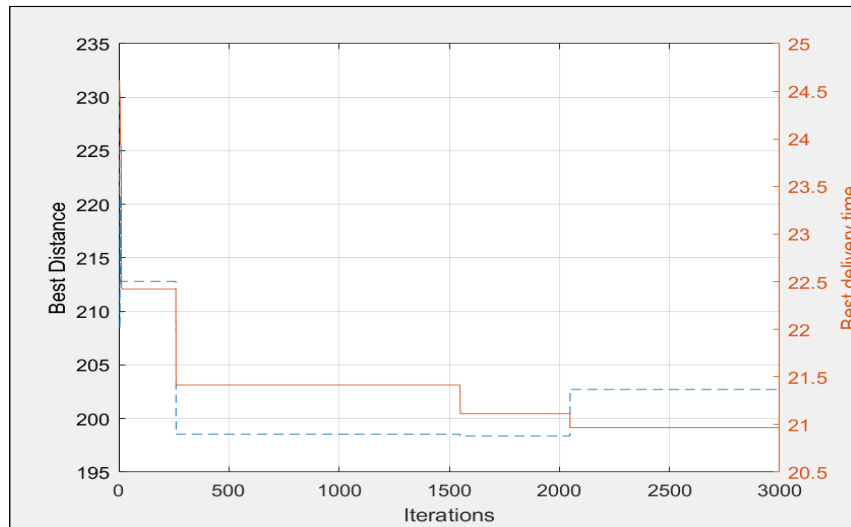


(c) Ant diversity per iteration

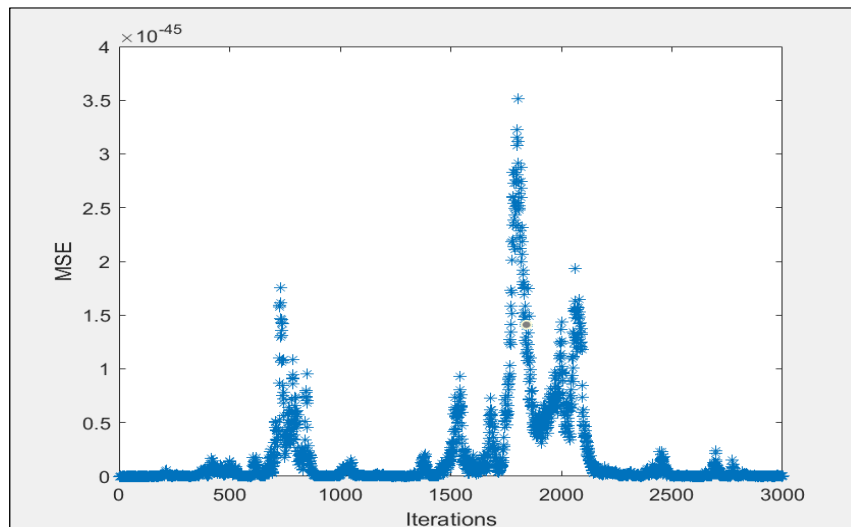
Figure 10: The drone and truck solution for 500 nodes (uniform-6-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10



(a) Best path

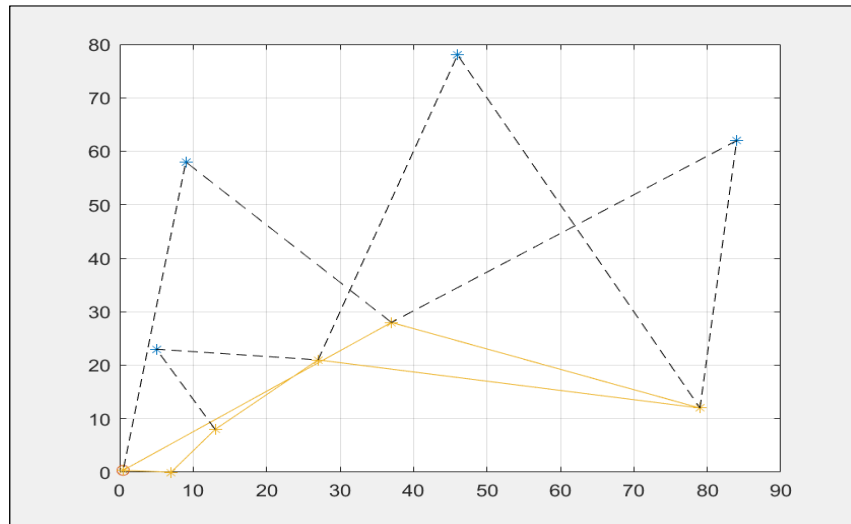


(b) Best cost per iteration

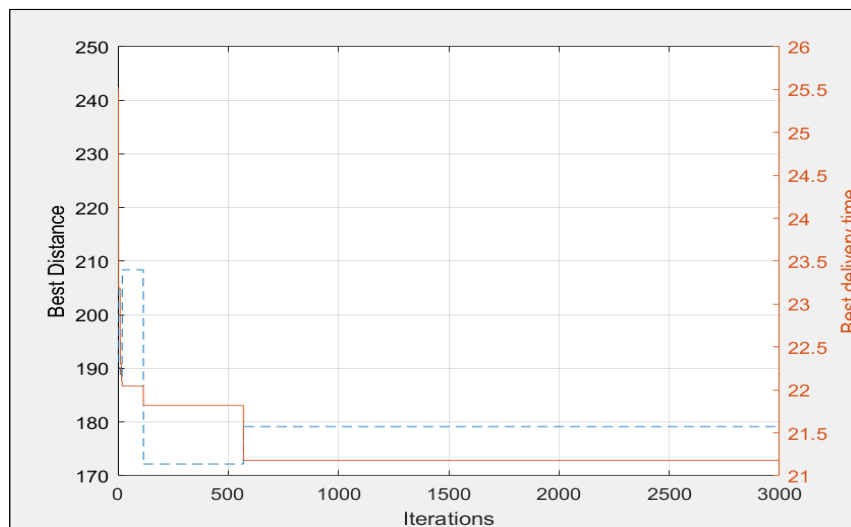


(c) Ant diversity per iteration

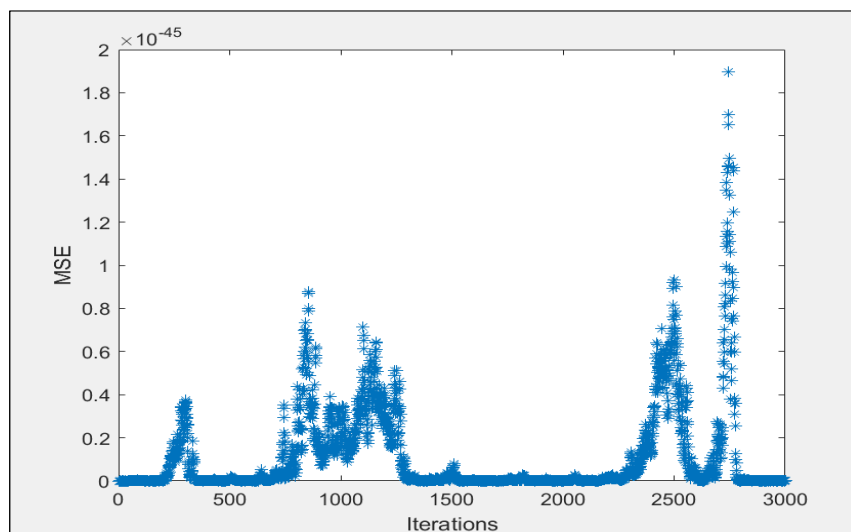
Figure 11: The drone and truck solution for 10 nodes (uniform-52-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2



(a) Best path



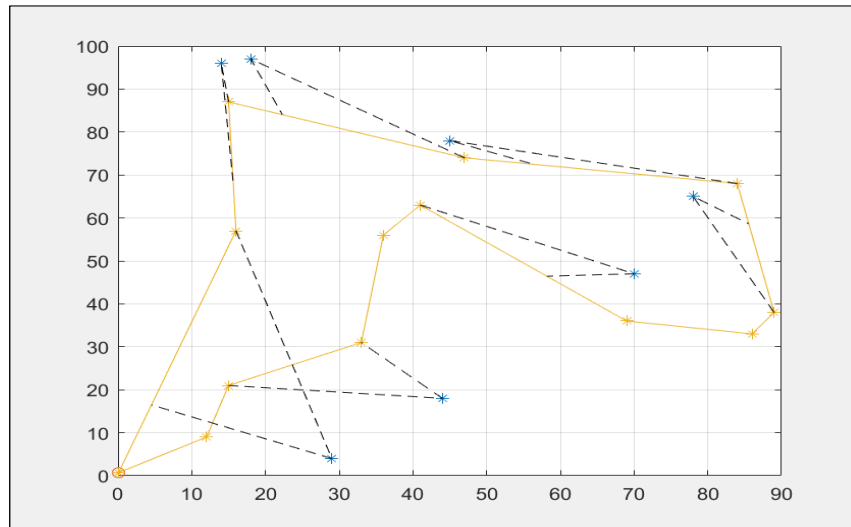
(b) Best cost per iteration



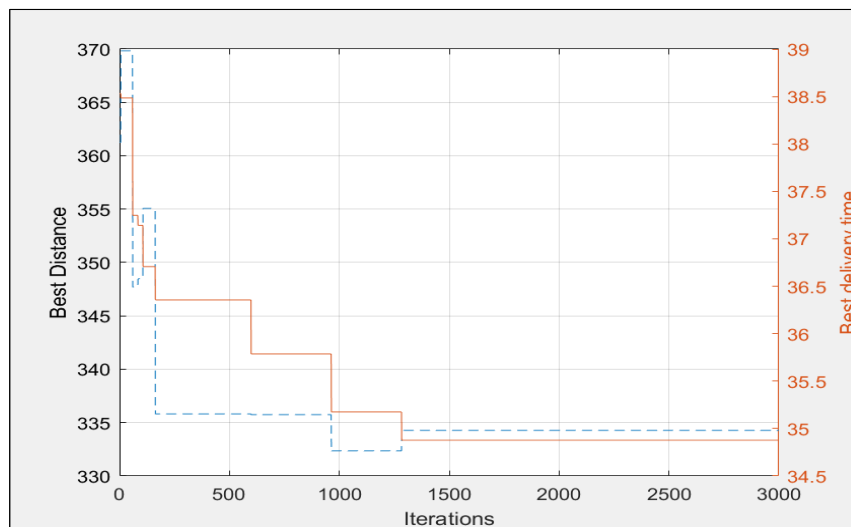
(c) Ant diversity per iteration

Figure 12: The drone and truck solution for 10 nodes (uniform-53-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

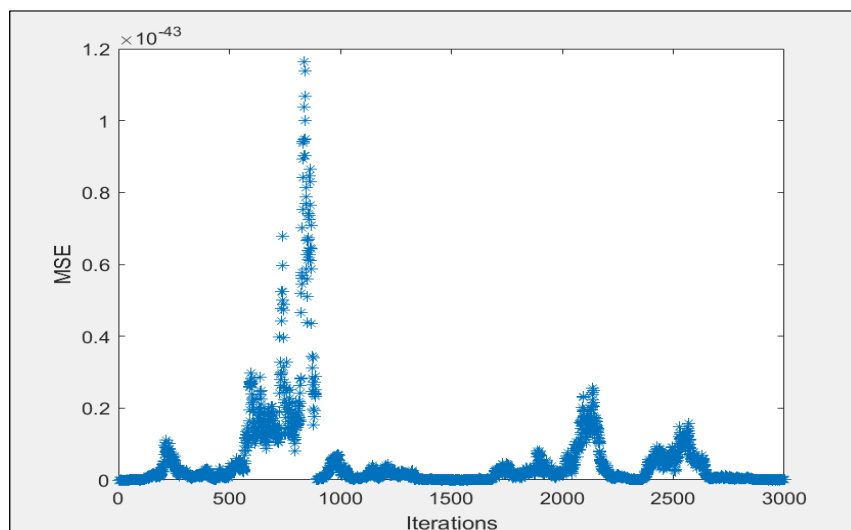
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path

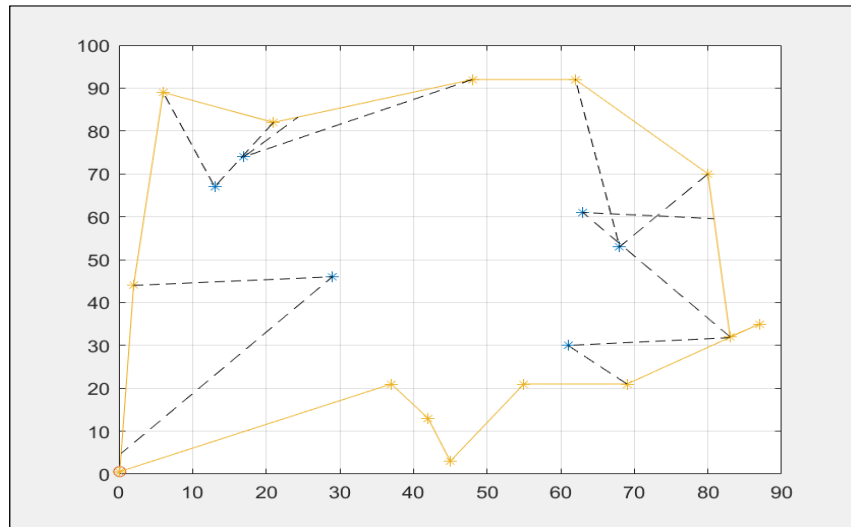


(b) Best cost per iteration

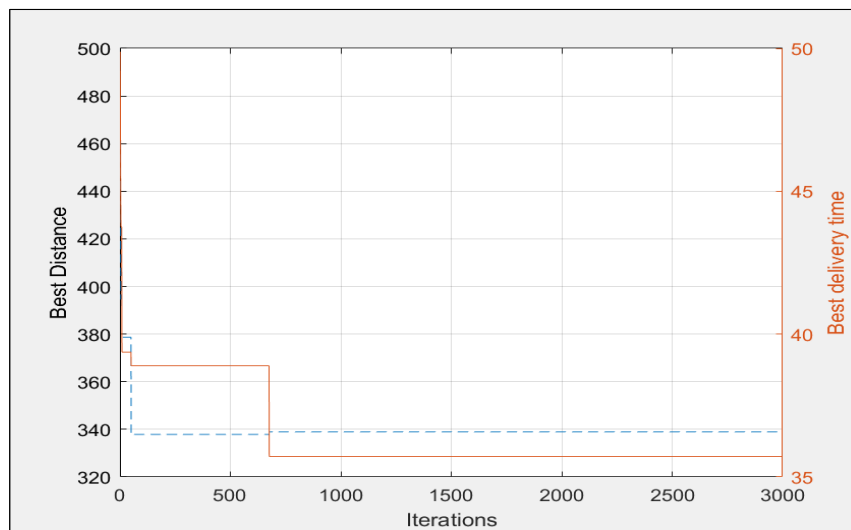


(c) Ant diversity per iteration

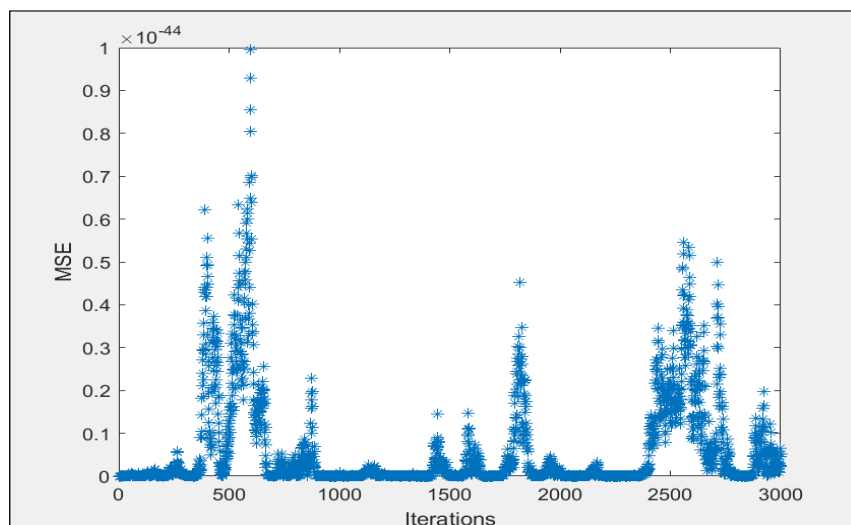
Figure 13: The drone and truck solution for 20 nodes (uniform-62-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2



(a) Best path



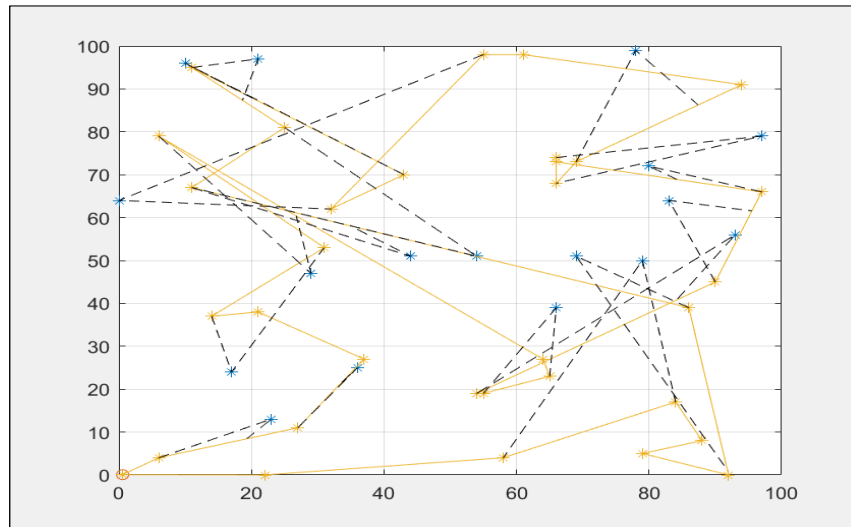
(b) Best cost per iteration



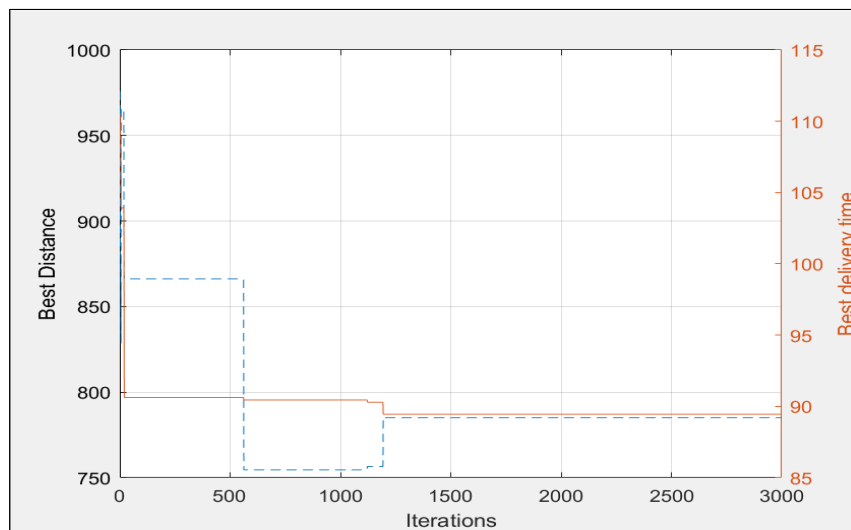
(c) Ant diversity per iteration

Figure 14: The drone and truck solution for 20 nodes (uniform-63-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

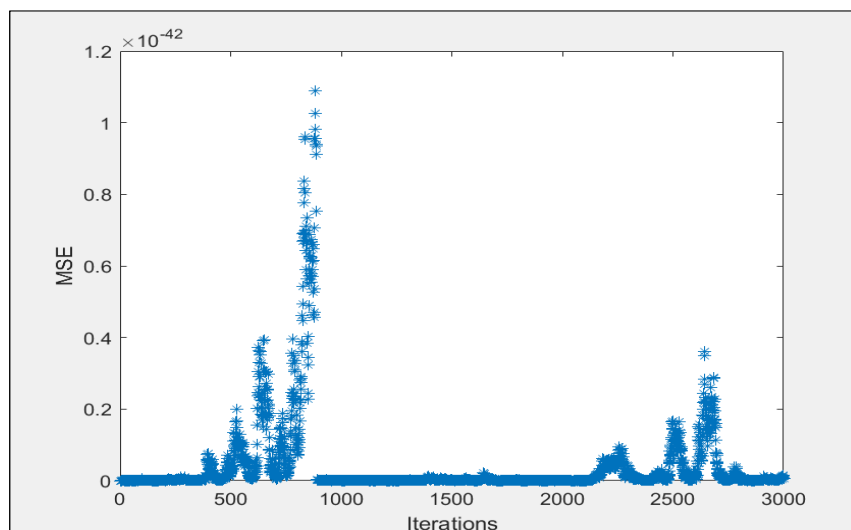
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



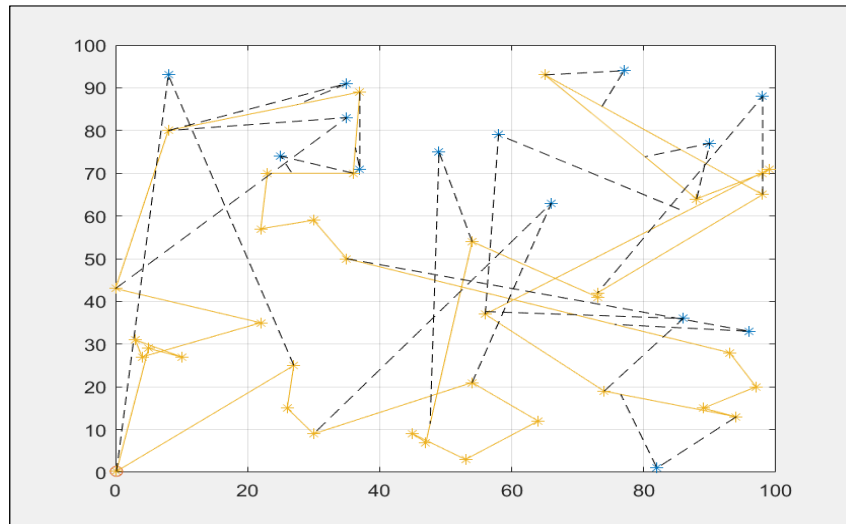
(b) Best cost per iteration



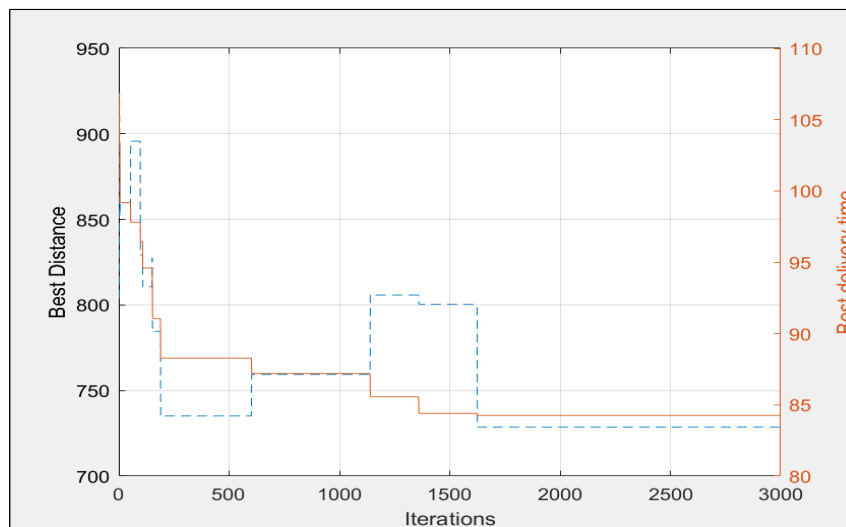
(c) Ant diversity per iteration

Figure 15: The drone and truck solution for 50 nodes (uniform-72-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

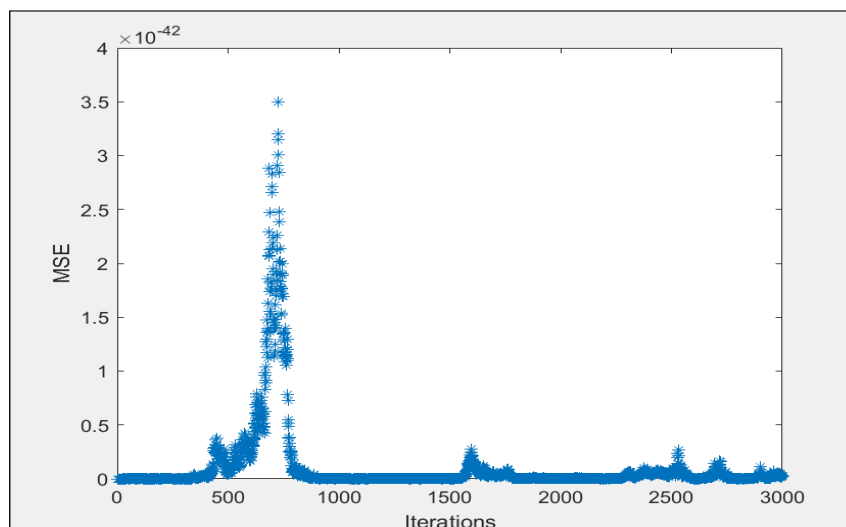
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



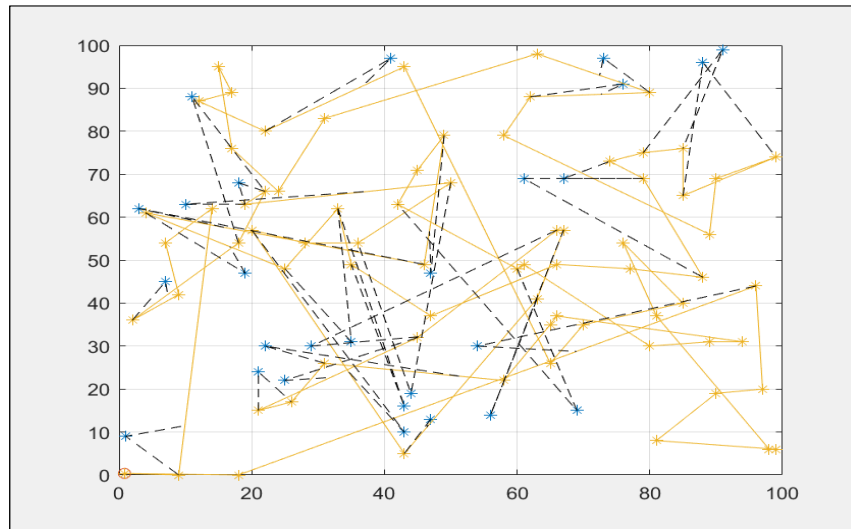
(b) Best cost per iteration



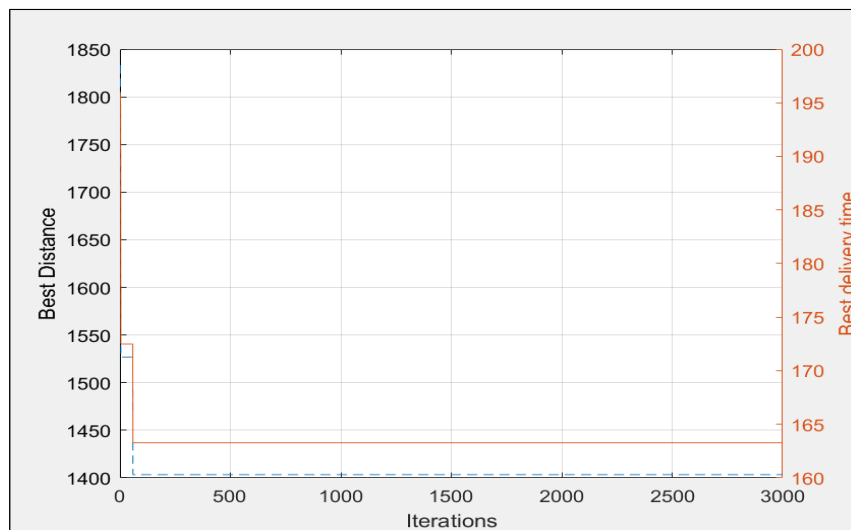
(c) Ant diversity per iteration

Figure 16: The drone and truck solution for 50 nodes (uniform-73-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

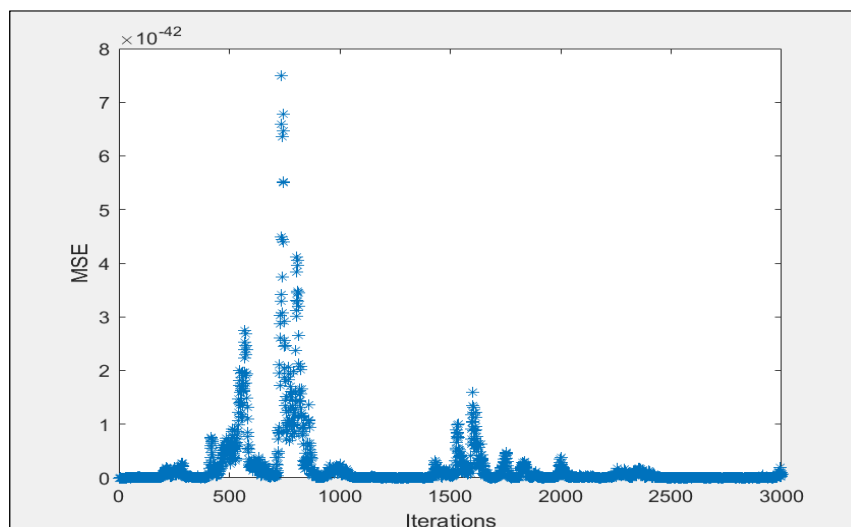
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



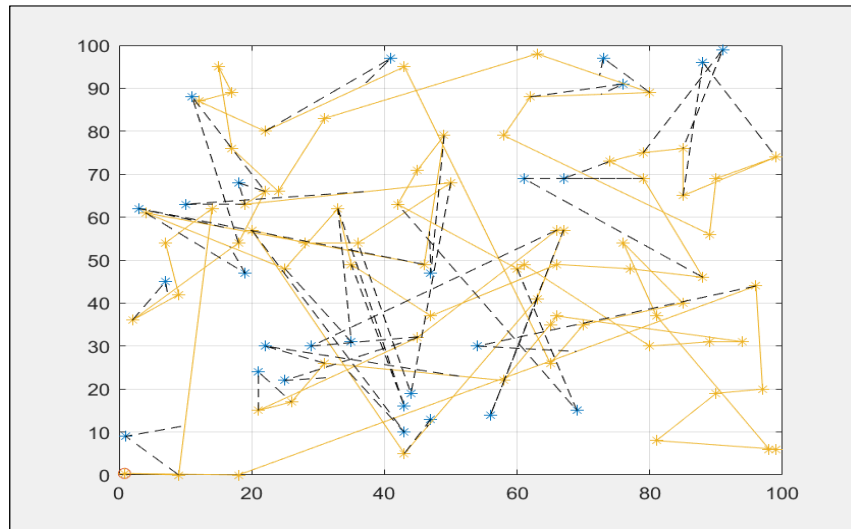
(b) Best cost per iteration



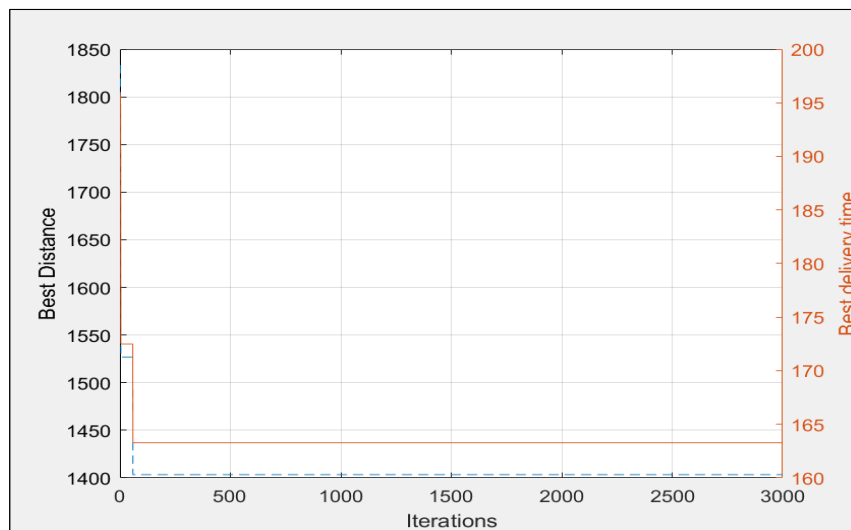
(c) Ant diversity per iteration

Figure 17: The drone and truck solution for 100 nodes (uniform-92-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

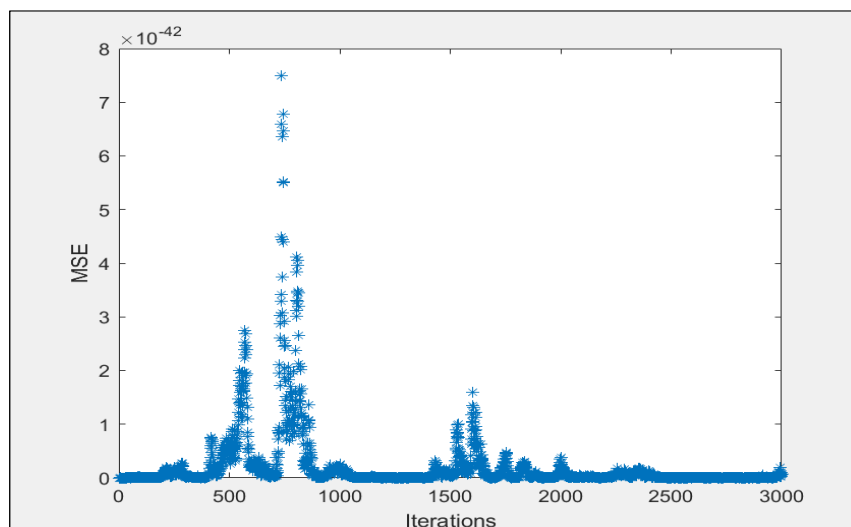
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



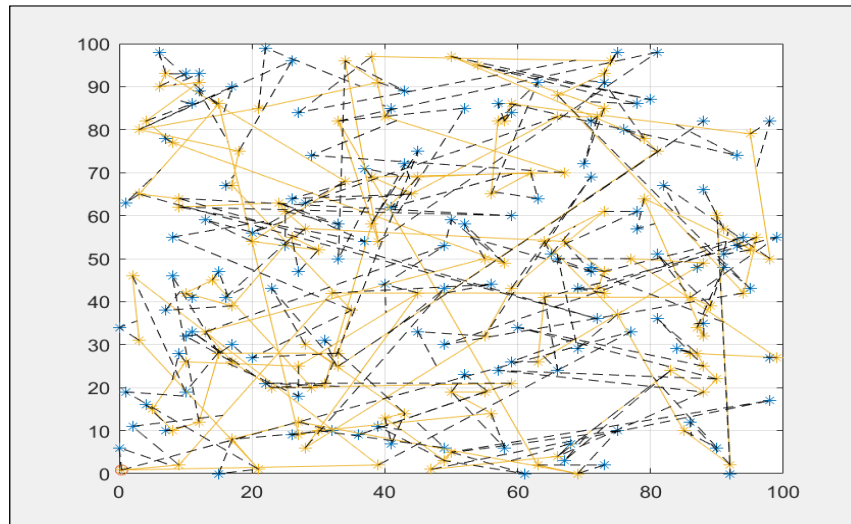
(b) Best cost per iteration



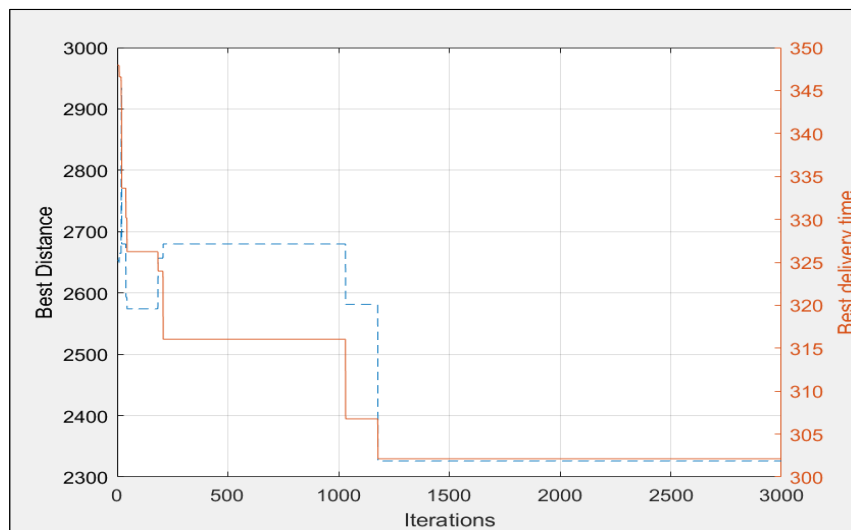
(c) Ant diversity per iteration

Figure 18: The drone and truck solution for 100 nodes (uniform-93-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

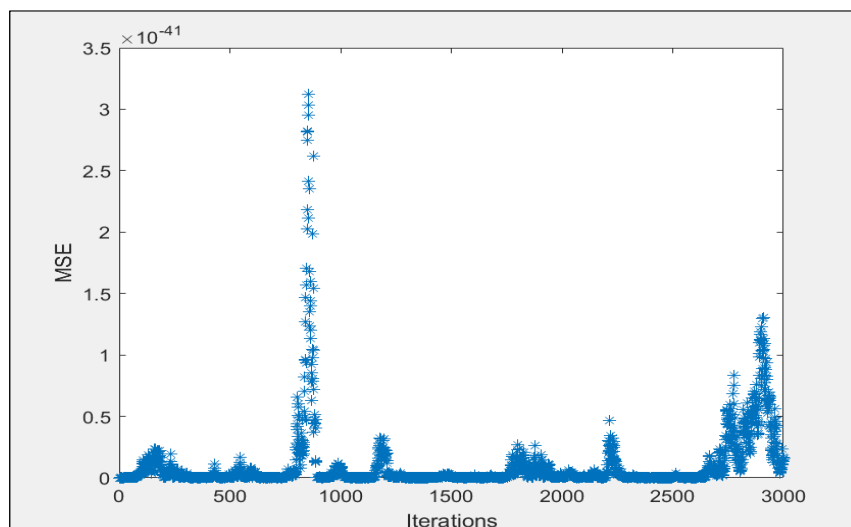
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path

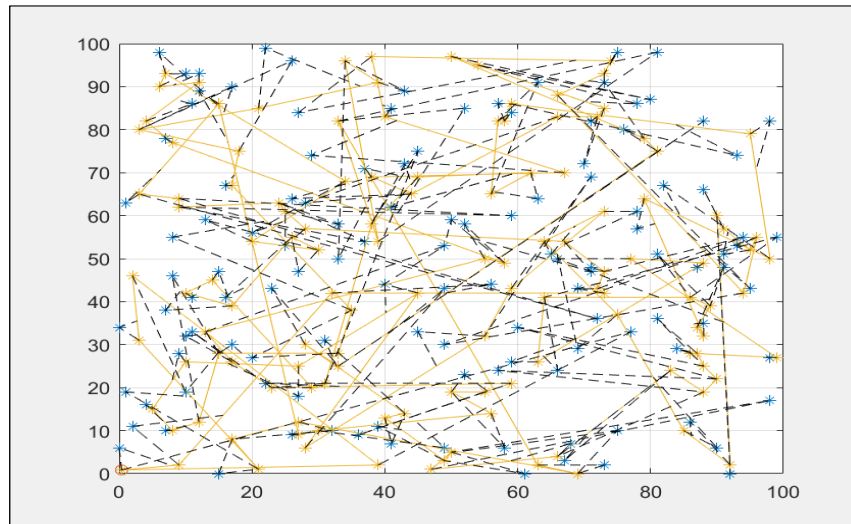


(b) Best cost per iteration

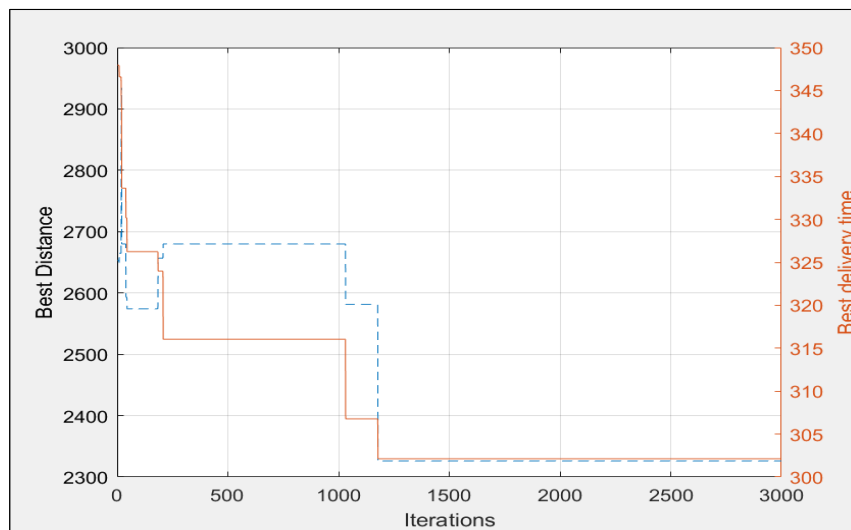


(c) Ant diversity per iteration

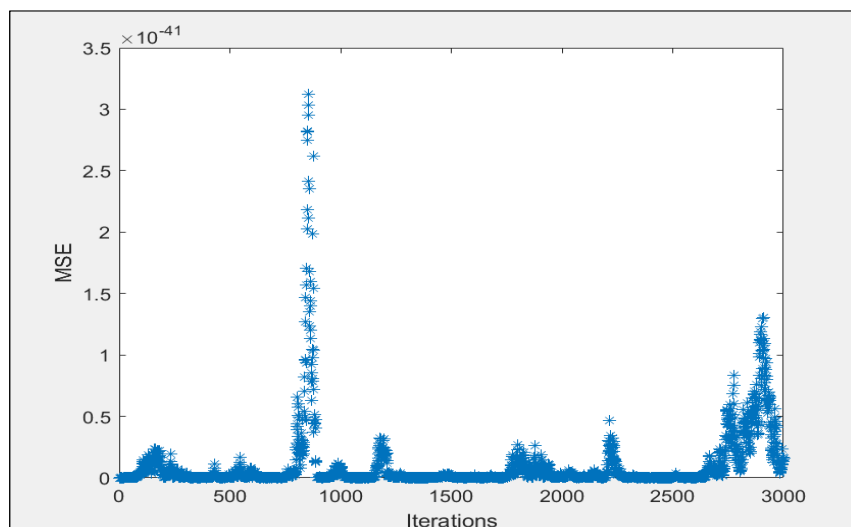
Figure 19: The drone and truck solution for 250 nodes (uniform-2-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2



(a) Best path

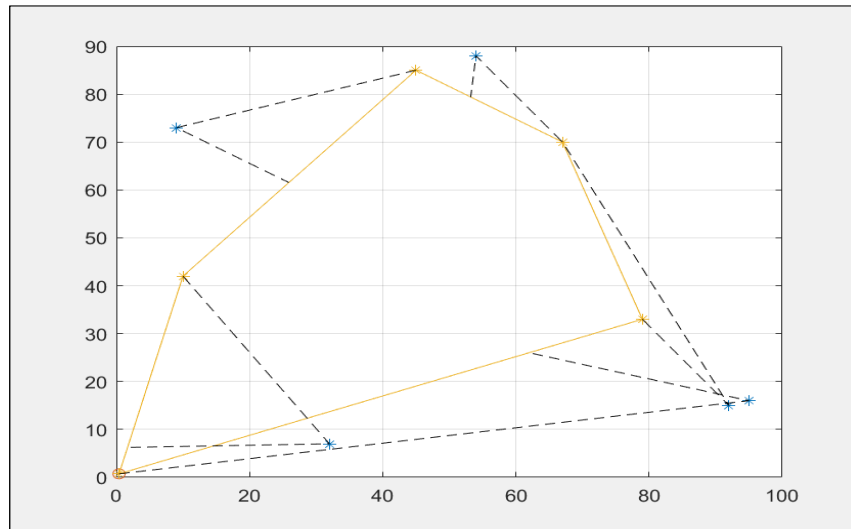


(b) Best cost per iteration

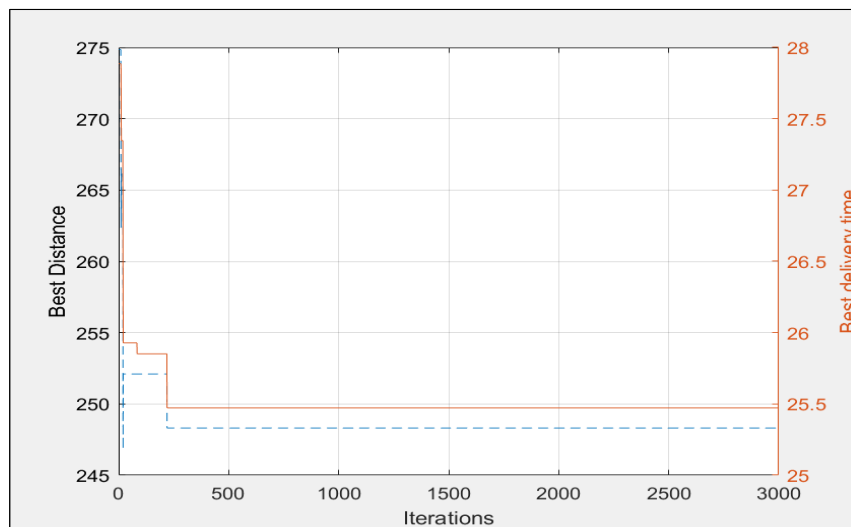


(c) Ant diversity per iteration

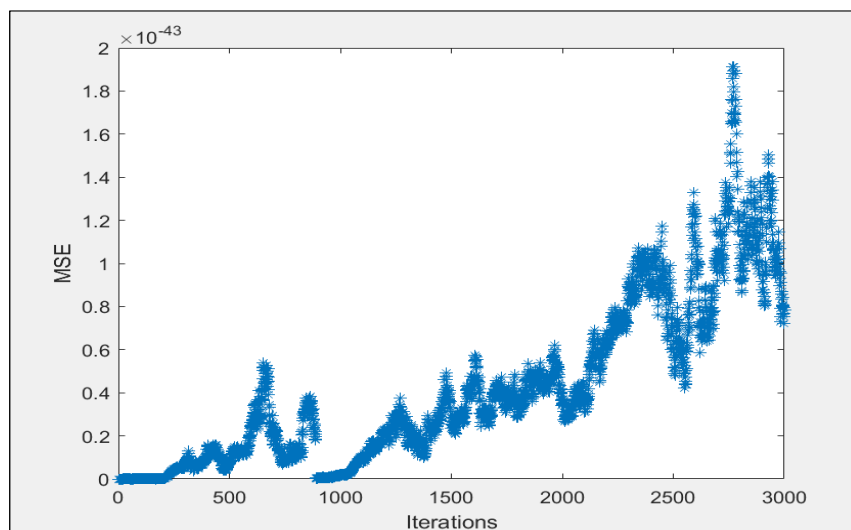
Figure 20: The drone and truck solution for 500 nodes (uniform-6-n500) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2



(a) Best path

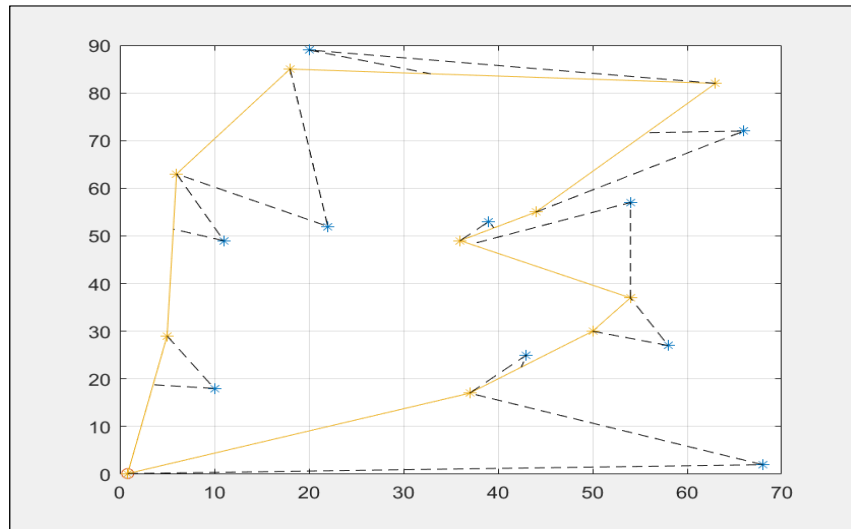


(b) Best cost per iteration

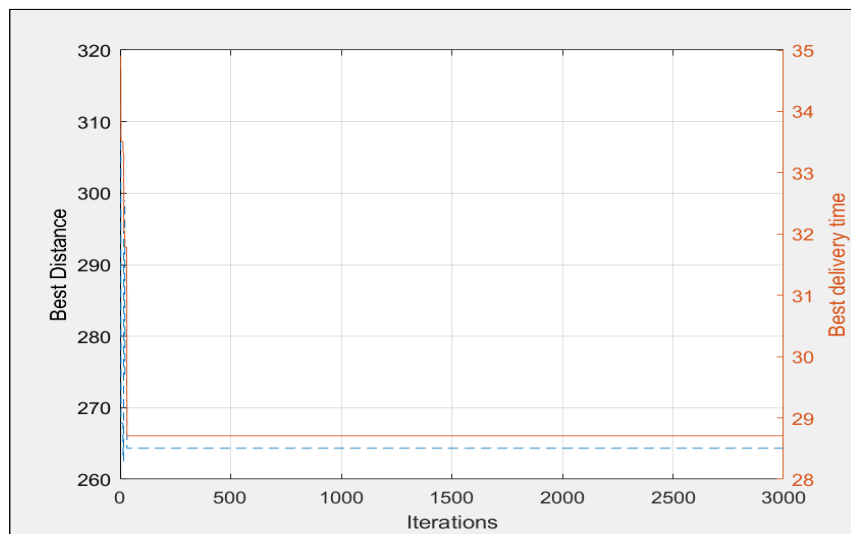


(c) Ant diversity per iteration

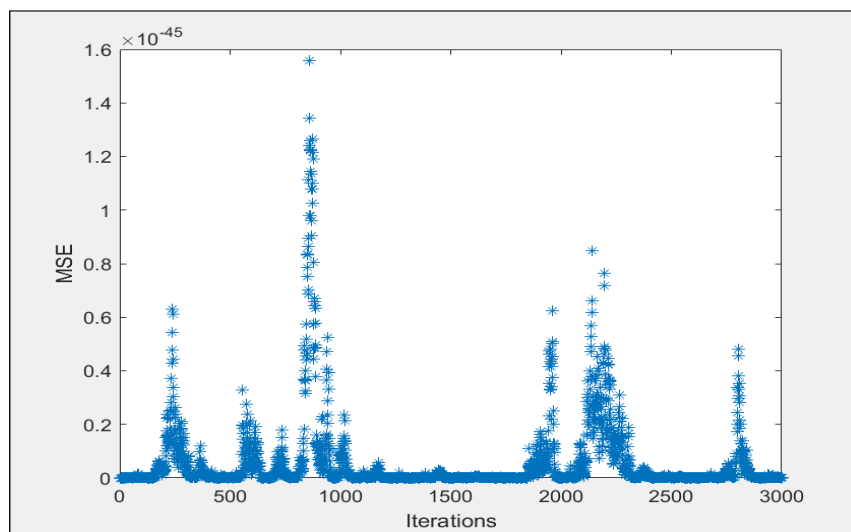
Figure 21: The drone and truck solution for 10 nodes (uniform-51-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1



(a) Best path

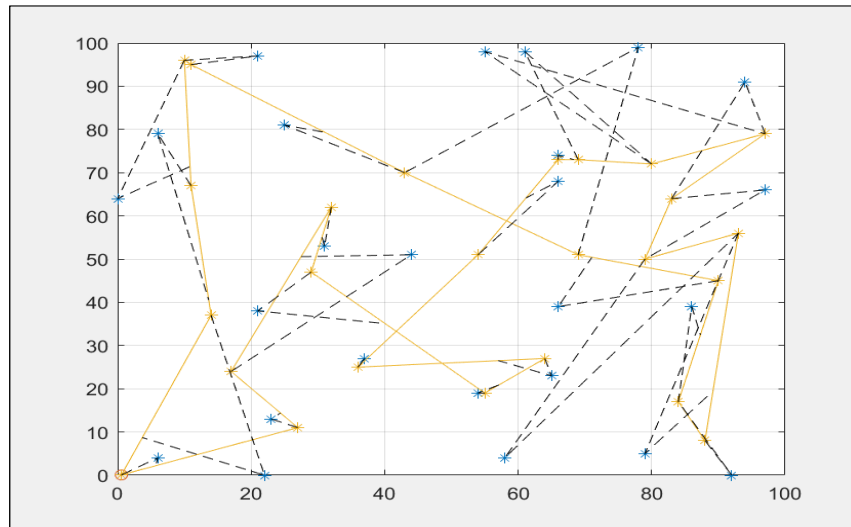


(b) Best cost per iteration

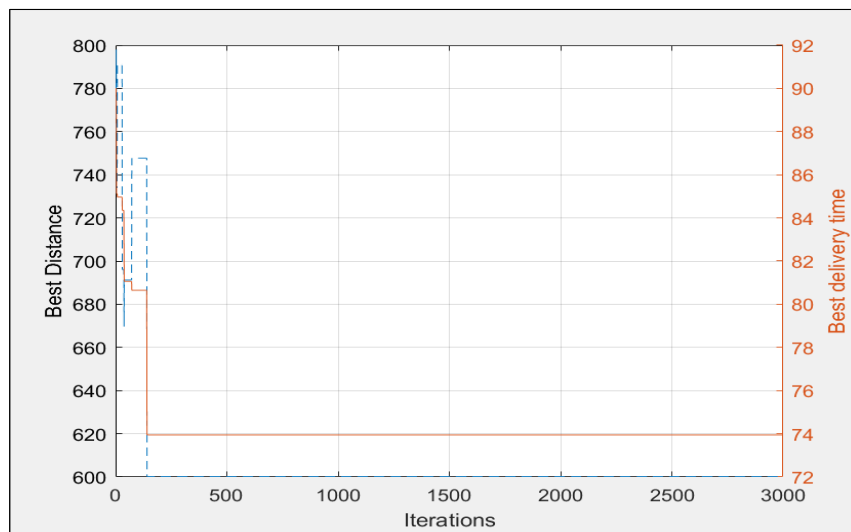


(c) Ant diversity per iteration

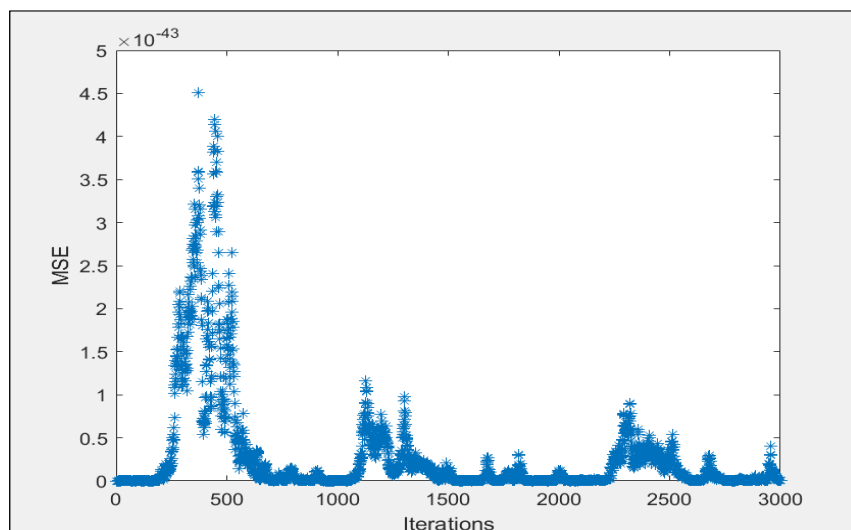
Figure 22: The drone and truck solution for 20 nodes (uniform-61-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1



(a) Best path

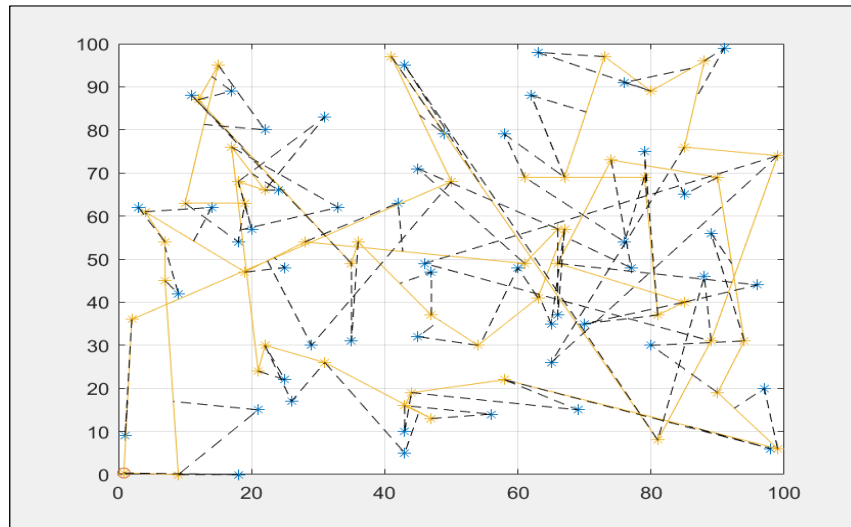


(b) Best cost per iteration

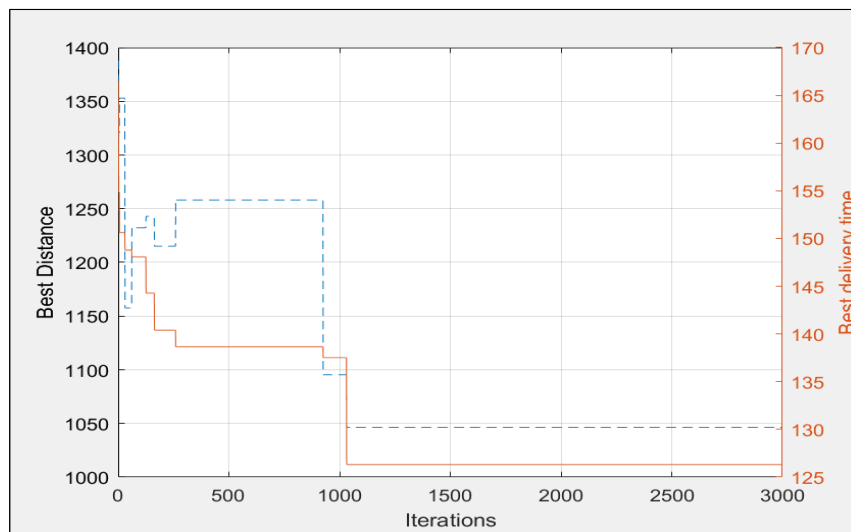


(c) Ant diversity per iteration

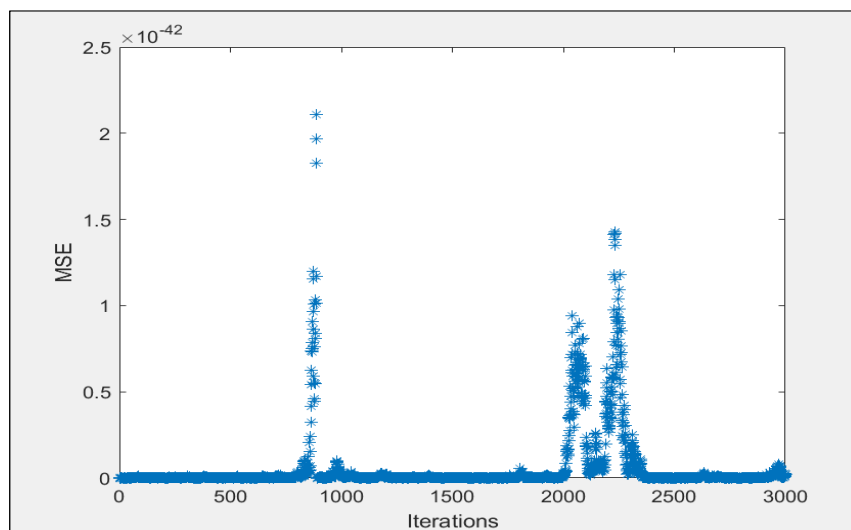
Figure 23: The drone and truck solution for 50 nodes (uniform-72-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1



(a) Best path



(b) Best cost per iteration

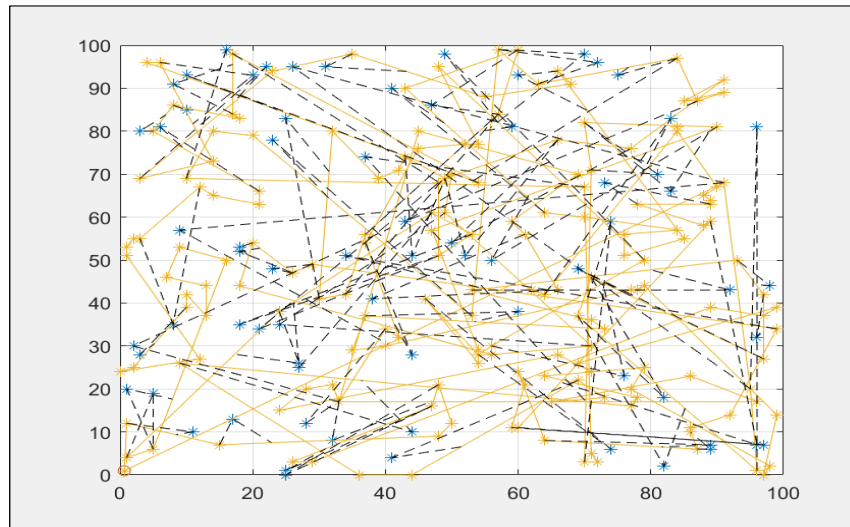


(c) Ant diversity per iteration

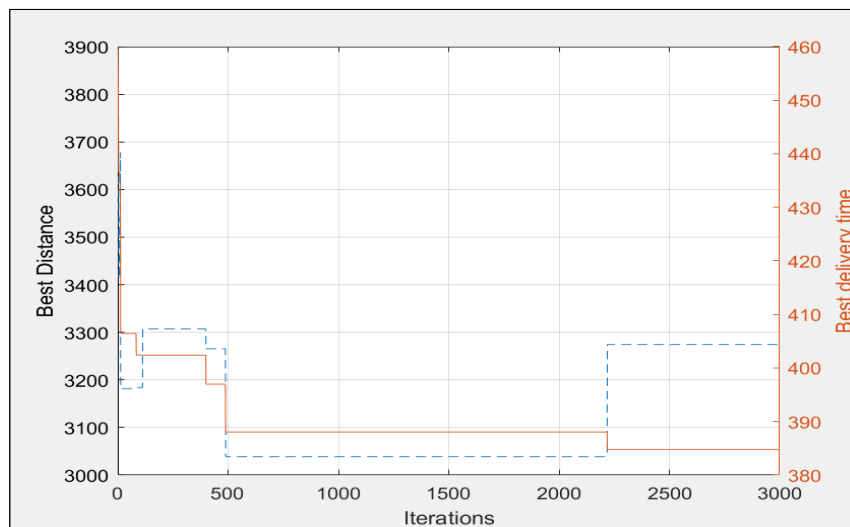
Figure 24: The drone and truck solution for 100 nodes (uniform-91-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1

ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES

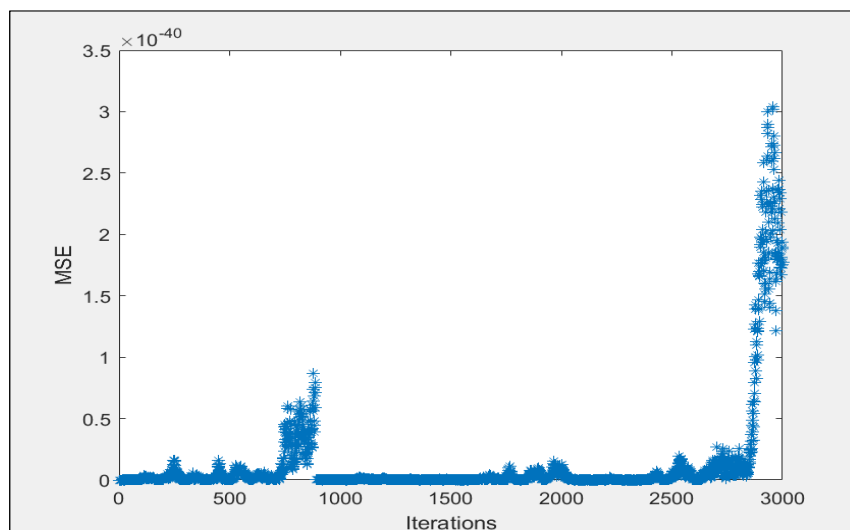
137



(a) Best path



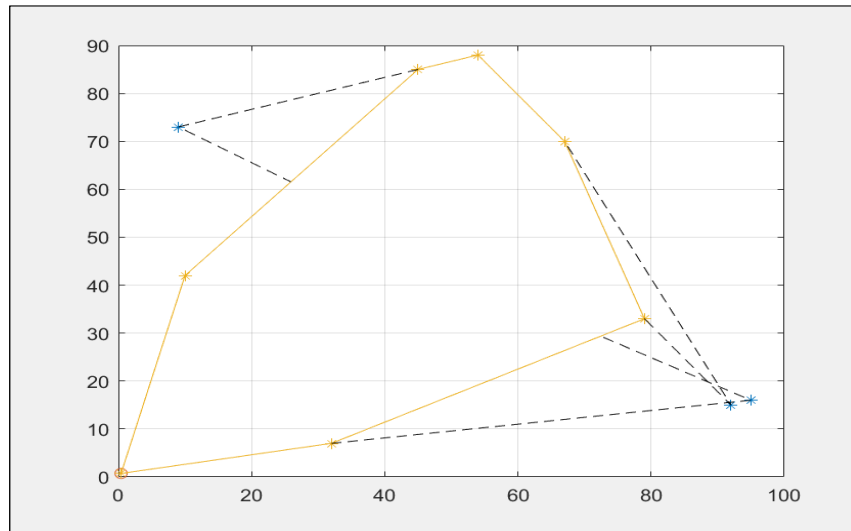
(b) Best cost per iteration



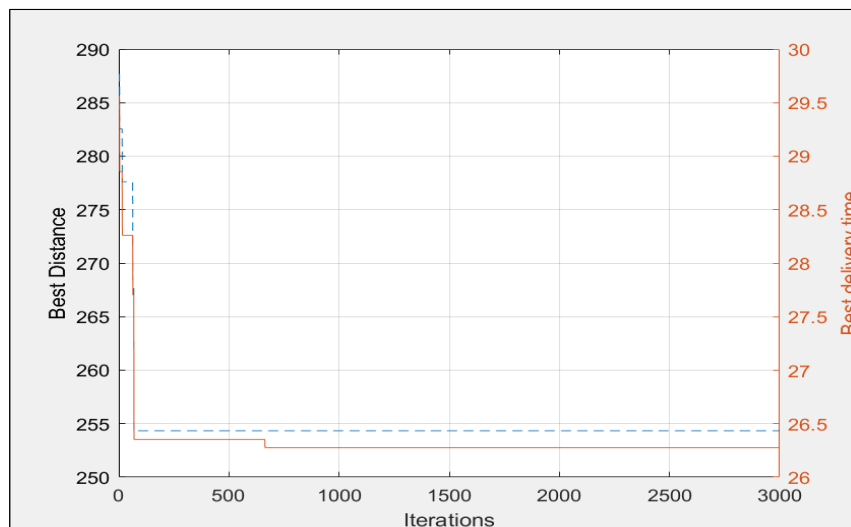
(c) Ant diversity per iteration

Figure 25: The drone and truck solution for 250 nodes (uniform-1-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT1

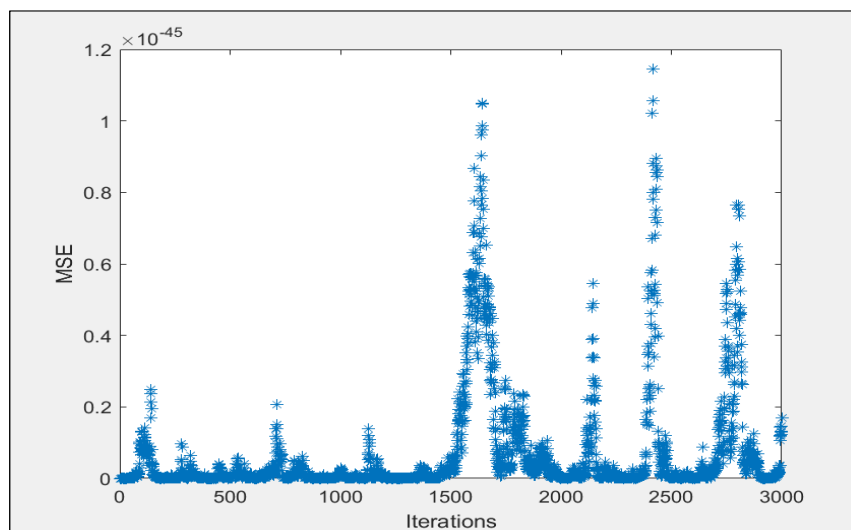
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path

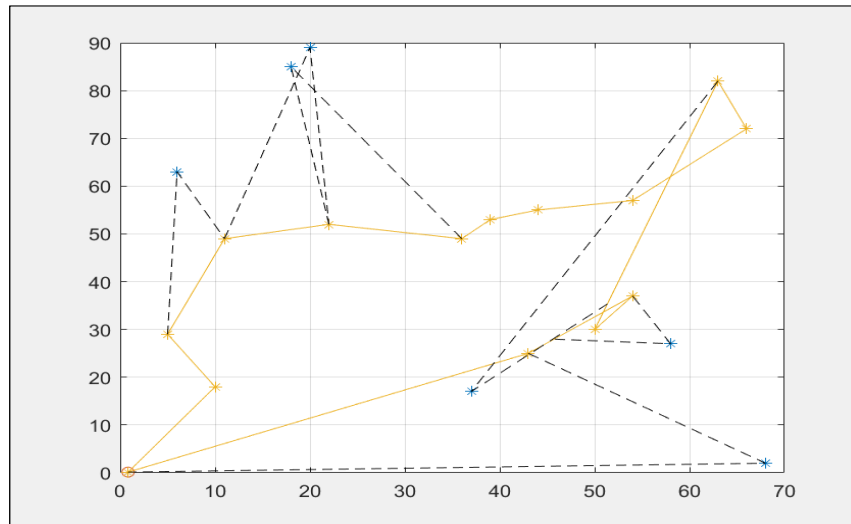


(b) Best cost per iteration

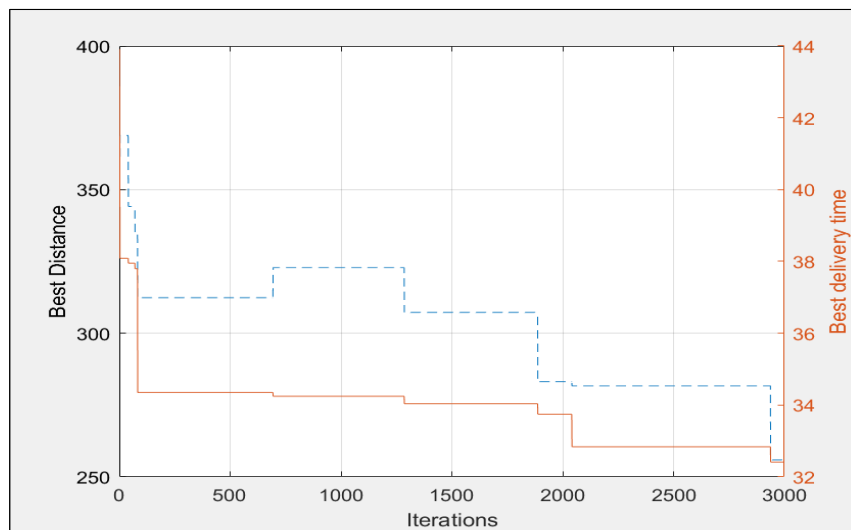


(c) Ant diversity per iteration

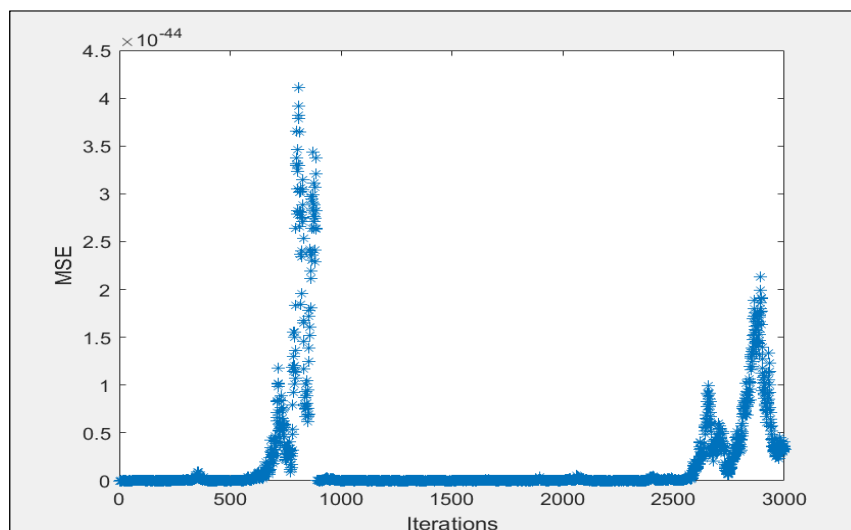
Figure 26: The drone and truck solution for 10 nodes (uniform-51-n10) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2



(a) Best path



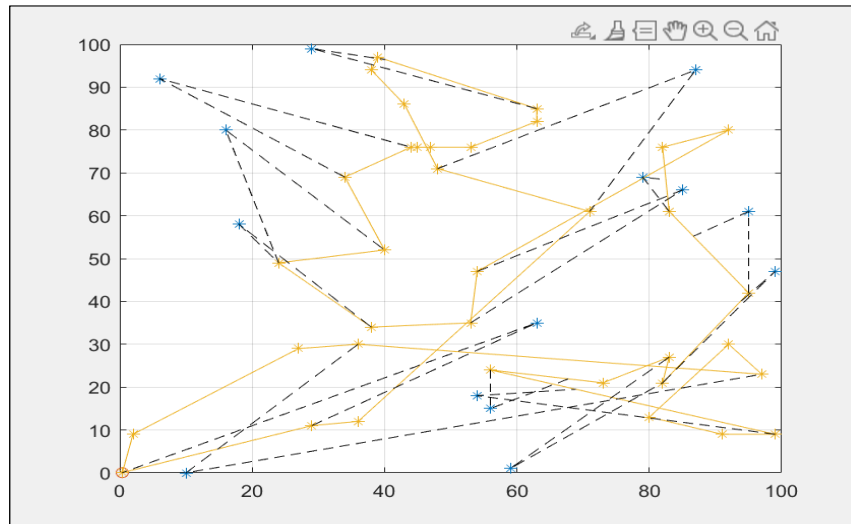
(b) Best cost per iteration



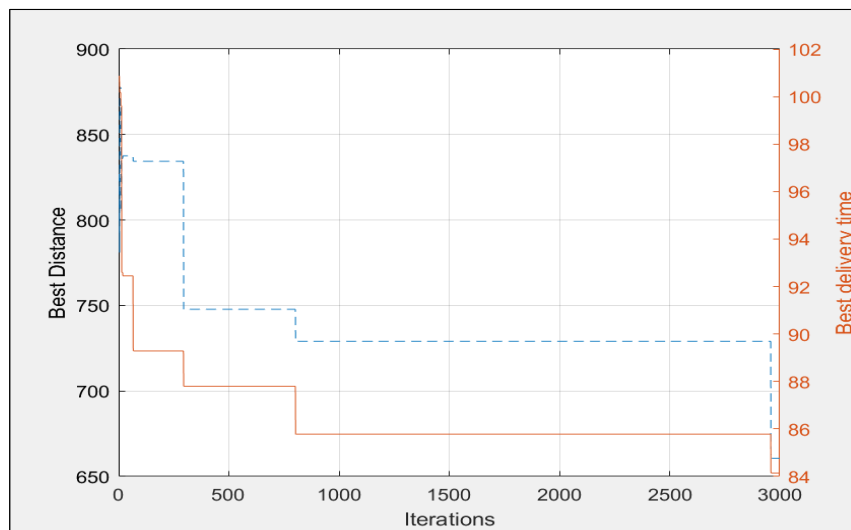
(c) Ant diversity per iteration

Figure 27: The drone and truck solution for 20 nodes (uniform-61-n20) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

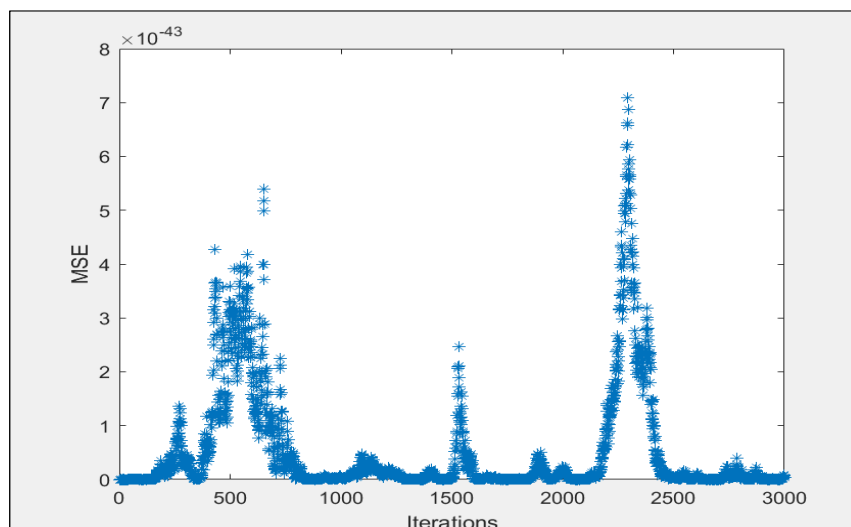
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



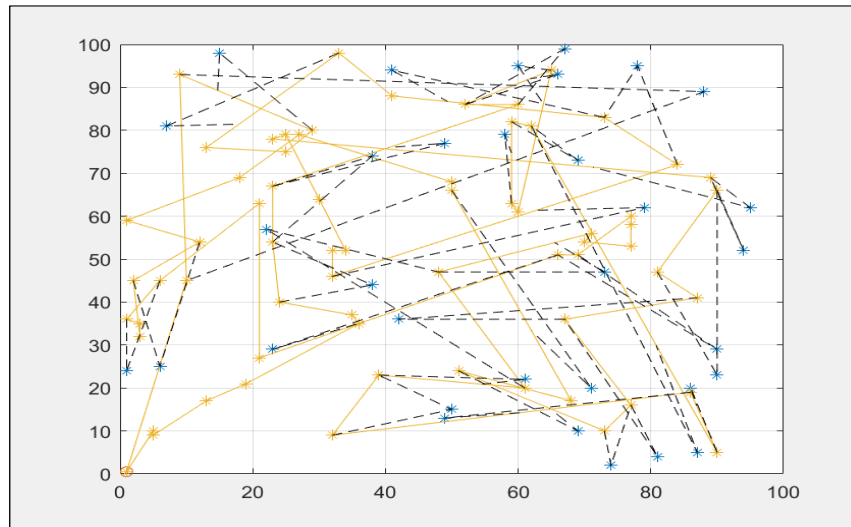
(b) Best cost per iteration



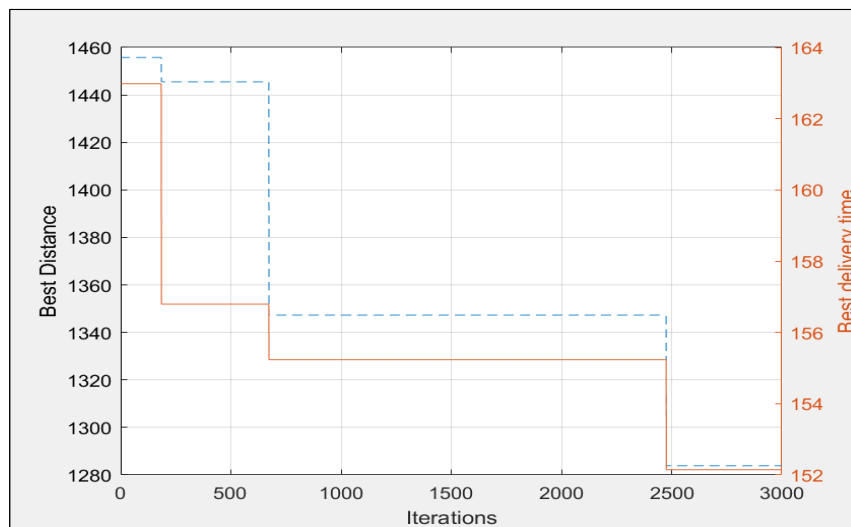
(c) Ant diversity per iteration

Figure 28: The drone and truck solution for 50 nodes (uniform-71-n50) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

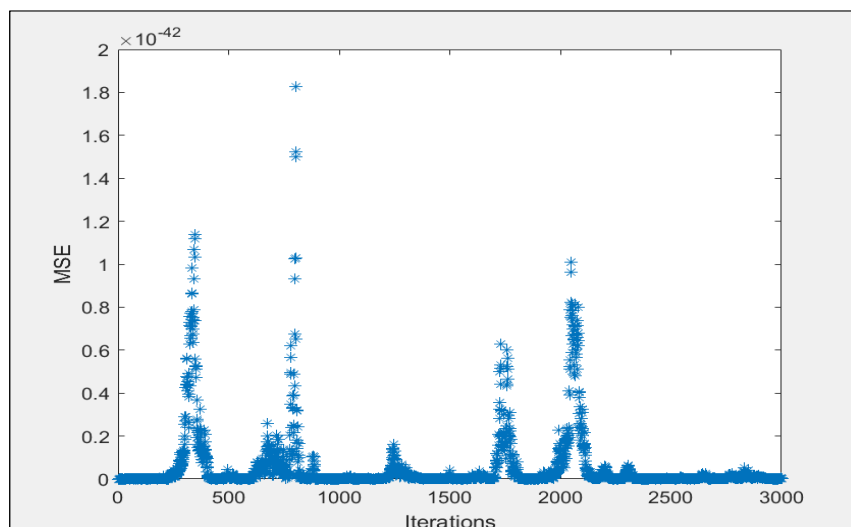
ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES



(a) Best path



(b) Best cost per iteration

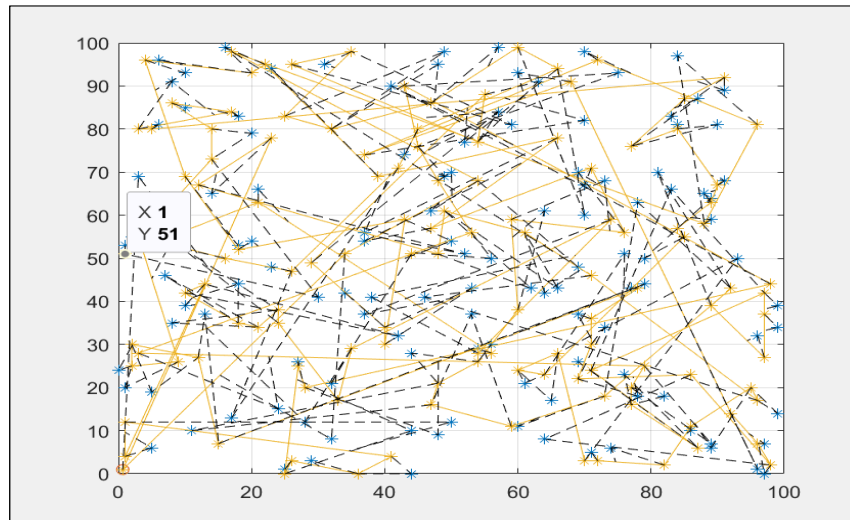


(c) Ant diversity per iteration

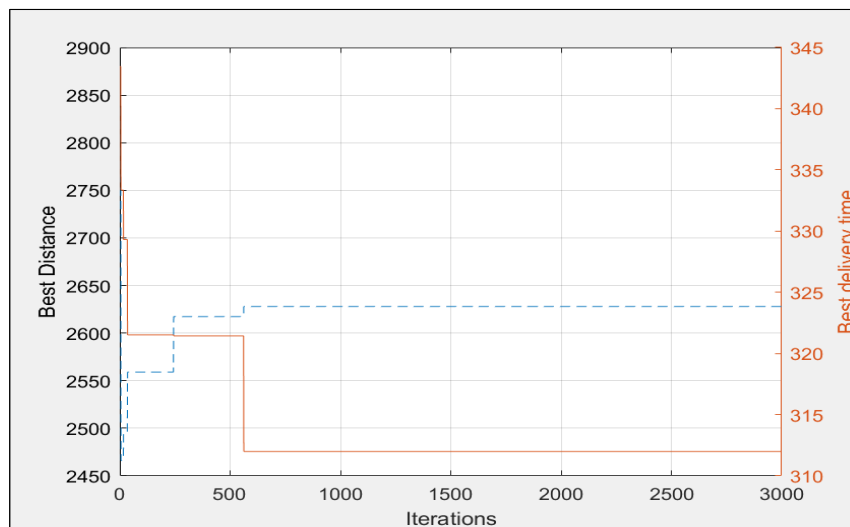
Figure 29: The drone and truck solution for 100 nodes (uniform-91-n100) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

ADDITIONAL DRONE AND TRUCK SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES

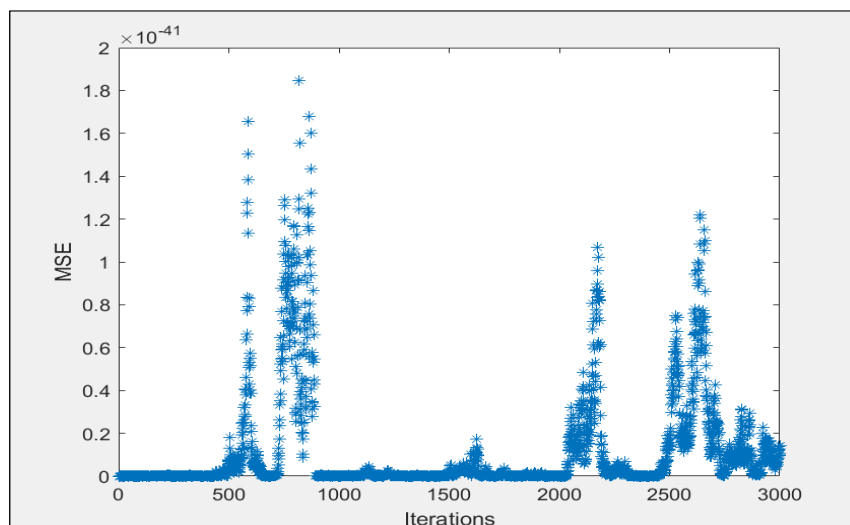
142



(a) Best path

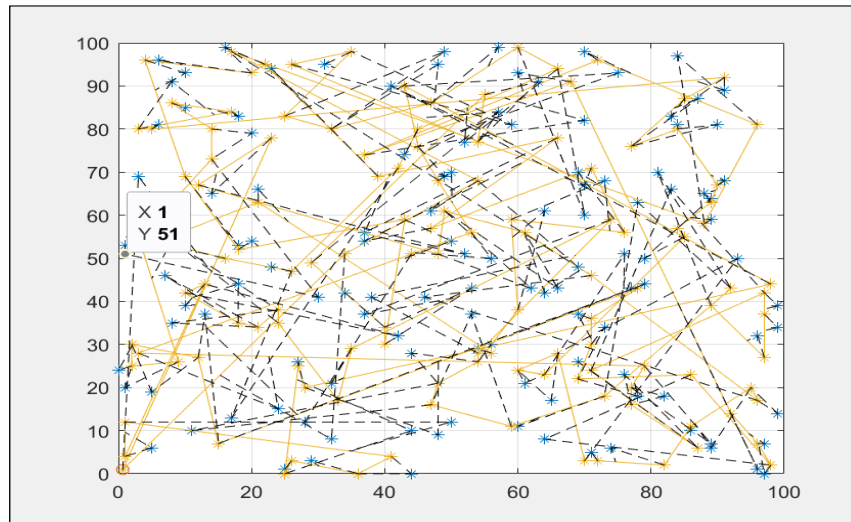


(b) Best cost per iteration

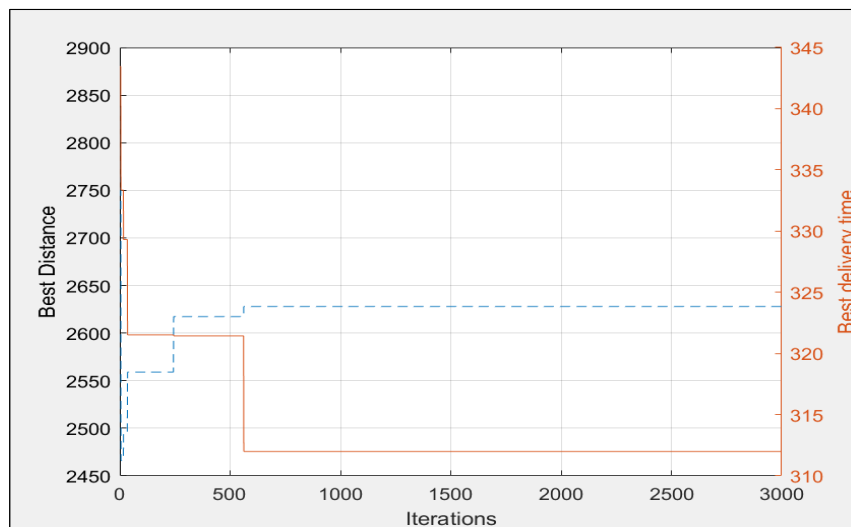


(c) Ant diversity per iteration

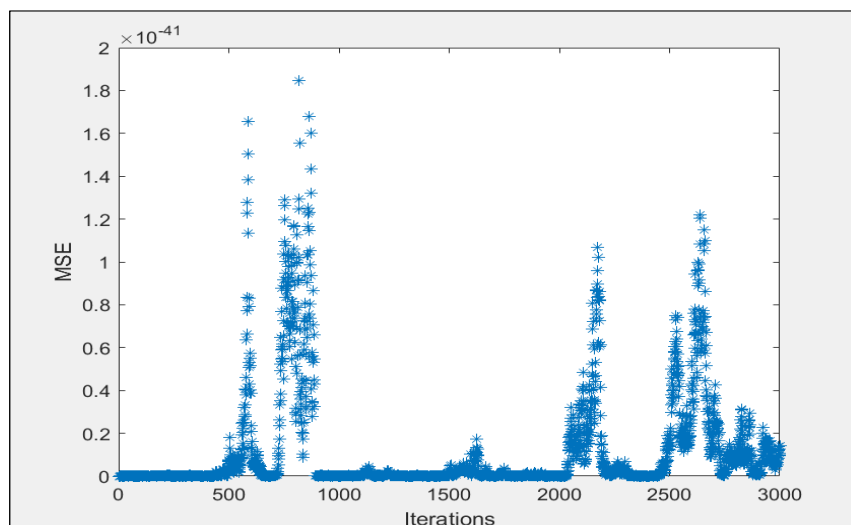
Figure 30: The drone and truck solution for 250 nodes (uniform-1-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2



(a) Best path



(b) Best cost per iteration



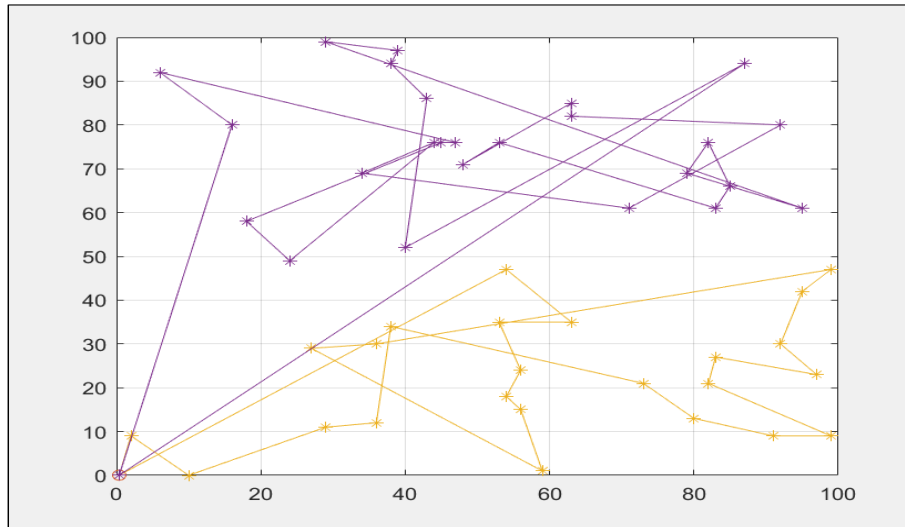
(c) Ant diversity per iteration

Figure 31: The drone and truck solution for 250 nodes (uniform-1-n250) with drone speed (v_d) of 20 and truck speed (v_t) of 10 using ACS-DT2

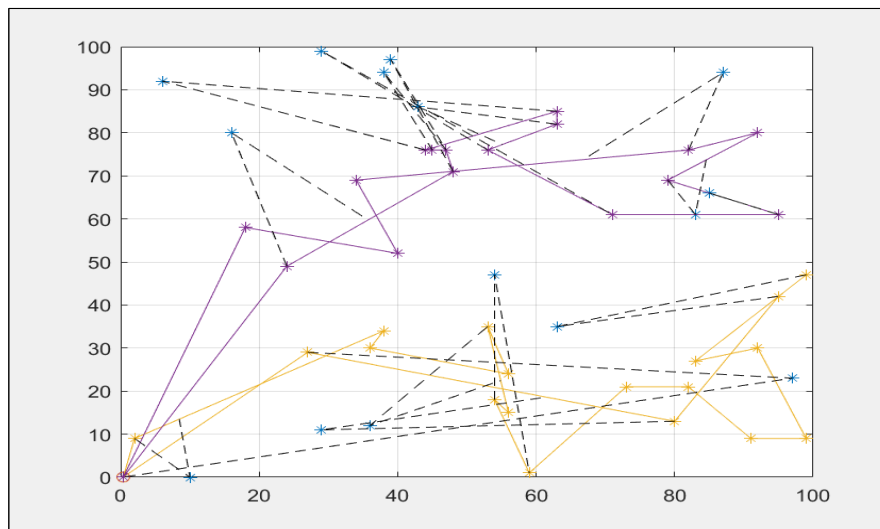
Additional multiple truck and drone scheduling problem with interception examples

This appendix contains additional multiple truck and drone scheduling problem with interception examples. A graphical representation of the truck only solution and multiple truck and drone solutions are shown in Figures 32, 35, 38, 41, 47, 44, 47, 50, 53 and, 56. Here customers receive parcels from multiple trucks and drones. The best cost per iteration for the datasets is shown in Figure 33,36, 39, 42 45, 48, 51, 54, and 57. The ants solutions search per iteration is shown n Figure 34,37, 40, 43, 46, 49, 52, 55 and 58.

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES 145



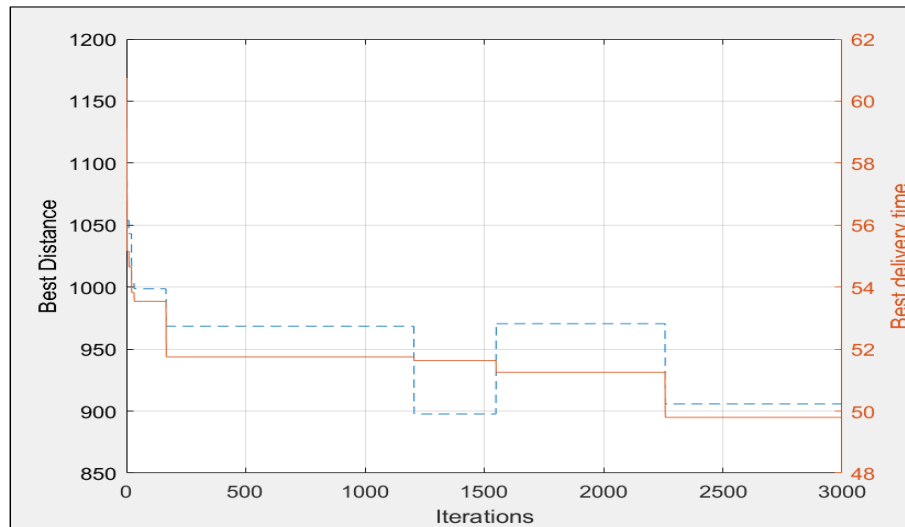
(a) Trucks-only



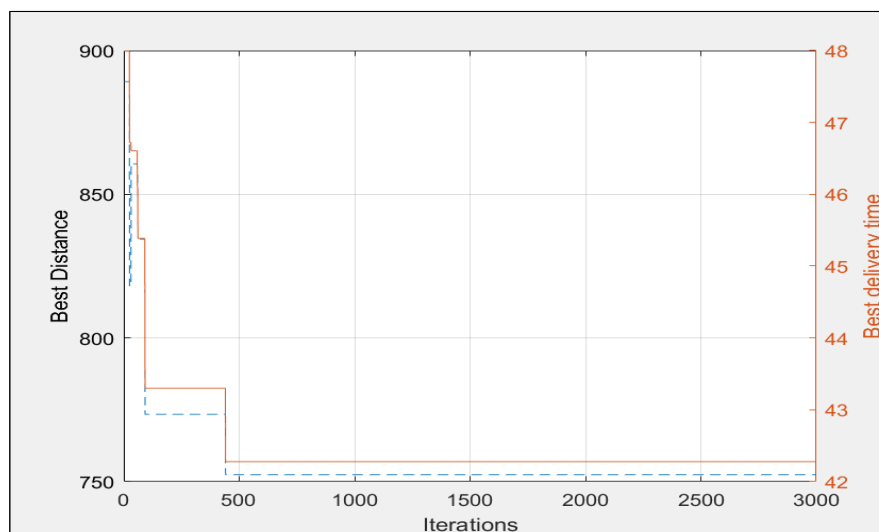
(b) Drones and trucks

Figure 32: 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-71-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 146



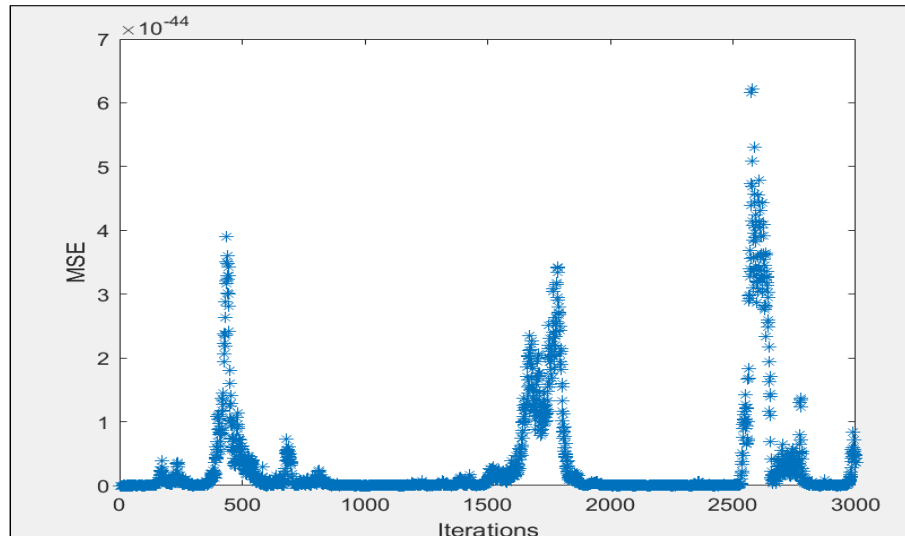
(a) Trucks-only



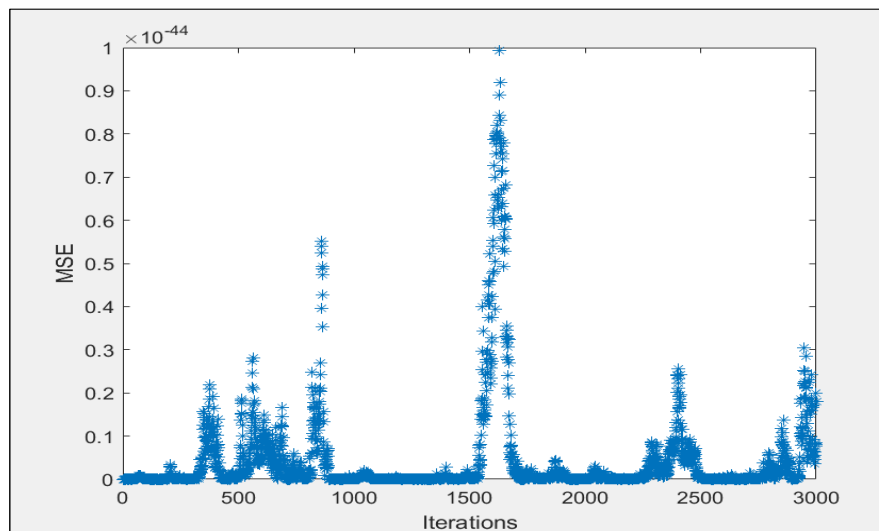
(b) Drones and trucks

Figure 33: Best solutions for 50 customers receiving deliveries from either a truck only system or a multiple drone-truck delivery system (Uniform-71-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 147



(a) Trucks-only

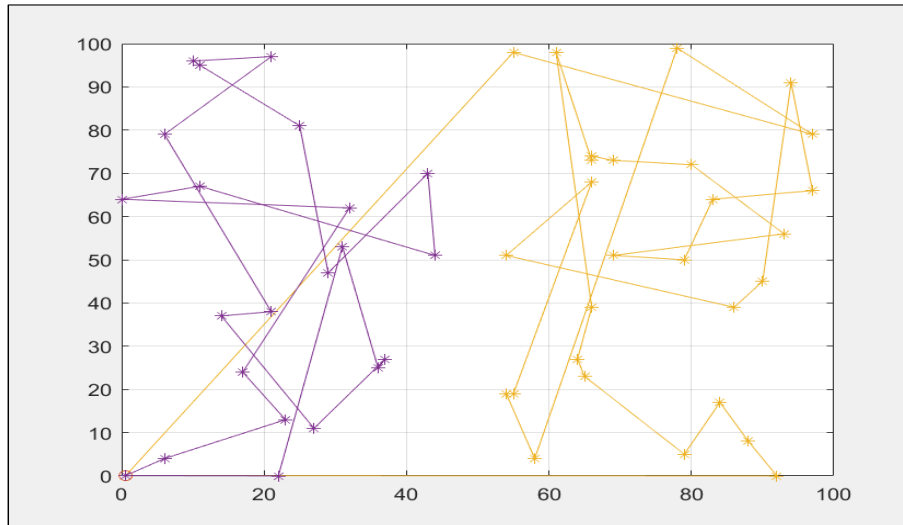


(b) Drones and trucks

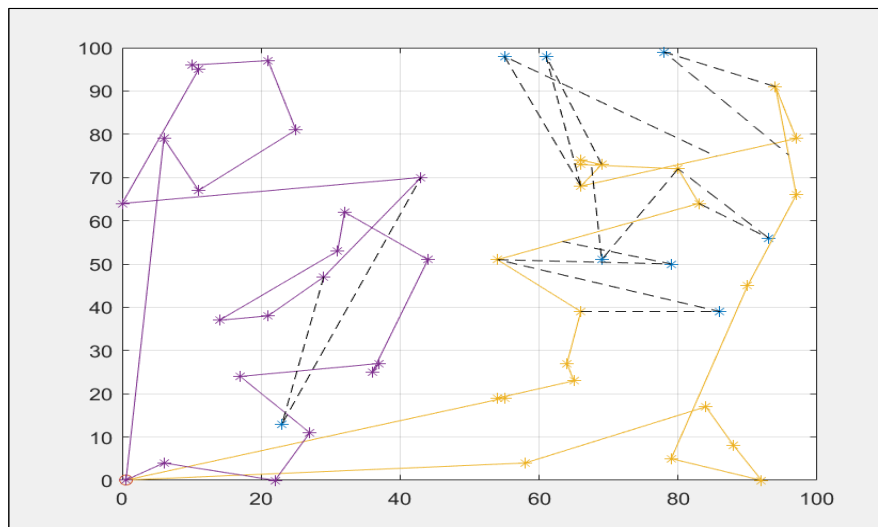
Figure 34: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-71-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

148



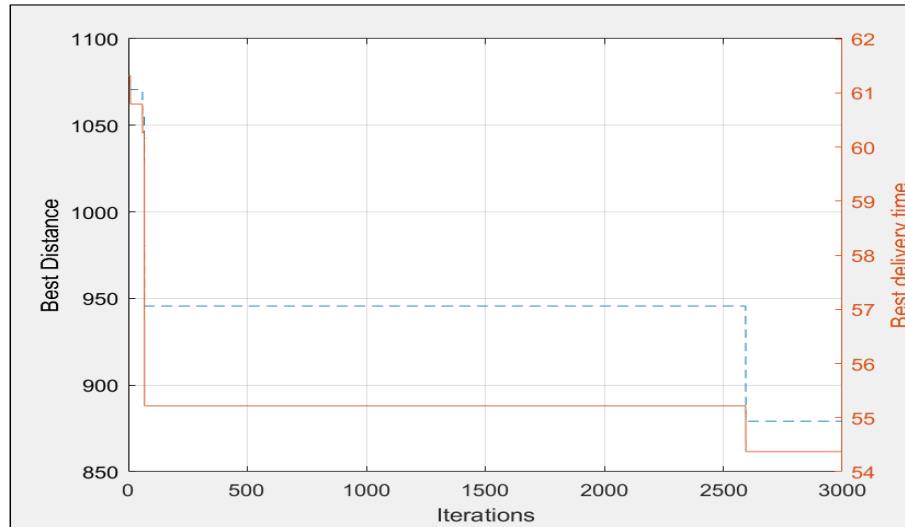
(a) Trucks-only



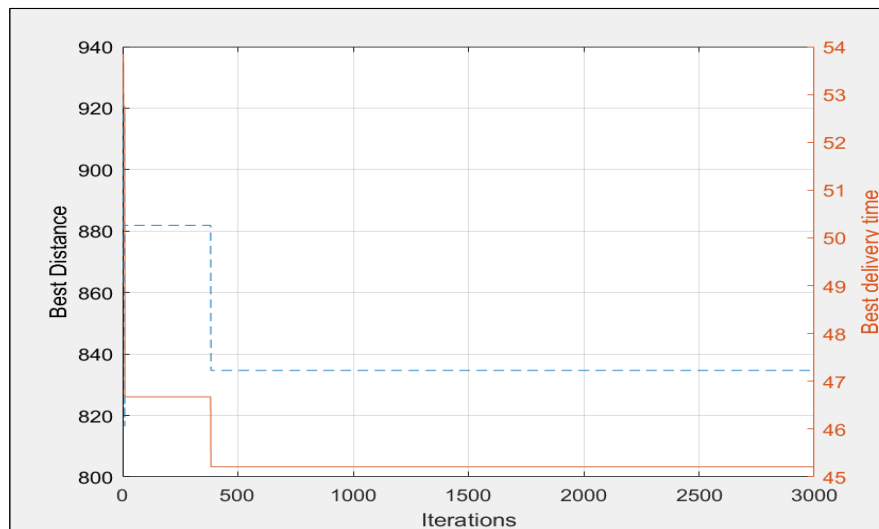
(b) Drones and trucks

Figure 35: 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-72-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 149



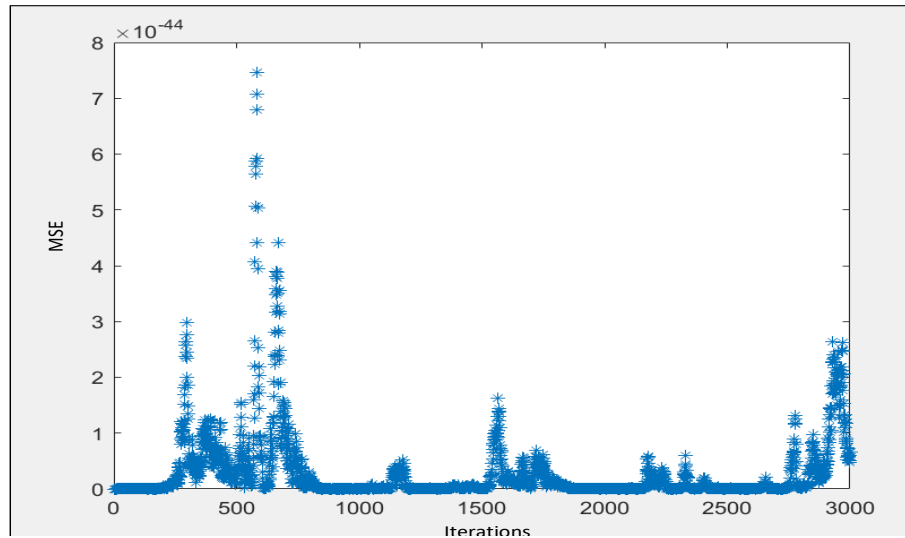
(a) Trucks-only



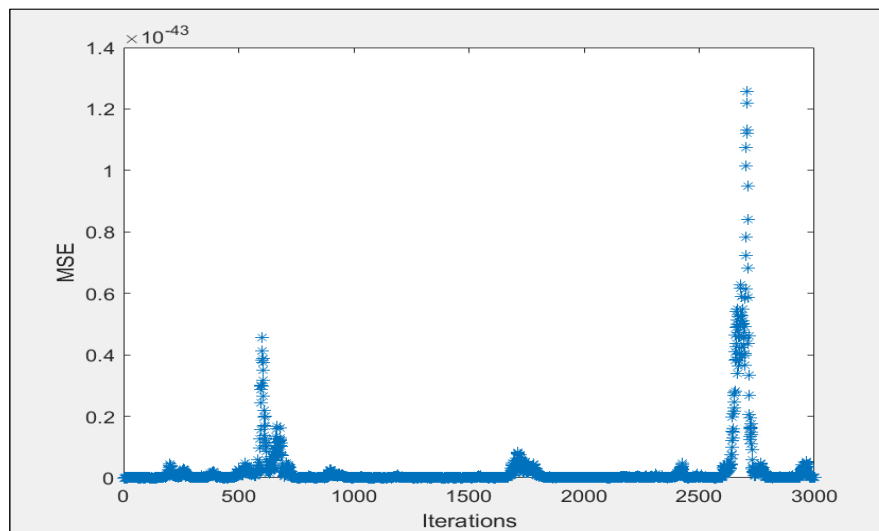
(b) Drones and trucks

Figure 36: Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-72-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 150



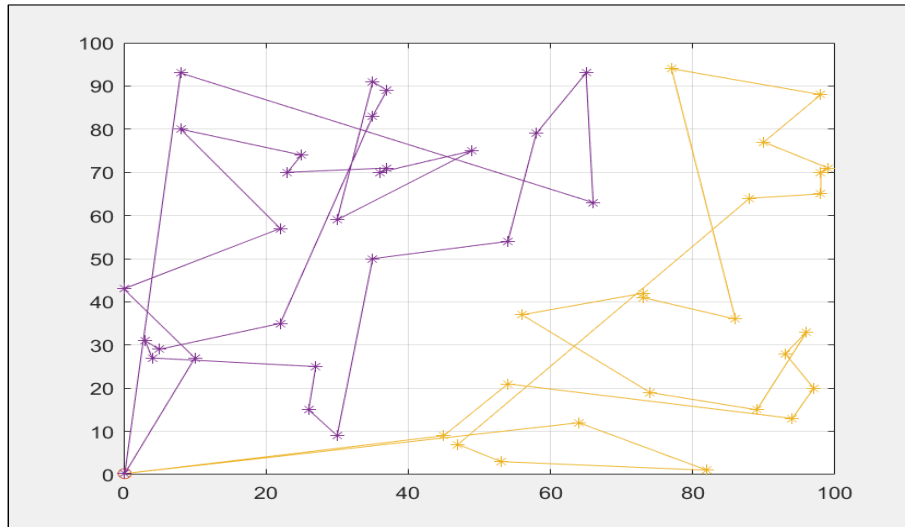
(a) Trucks-only



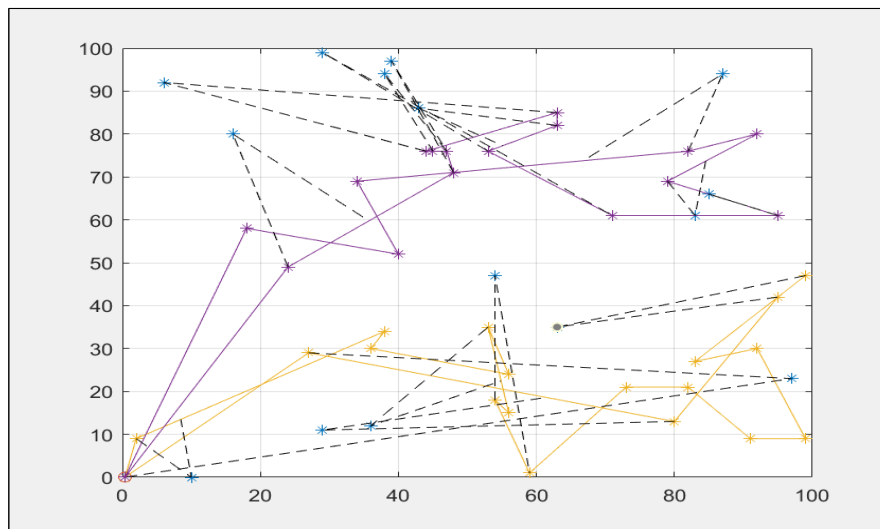
(b) Drones and trucks

Figure 37: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-72-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES 151



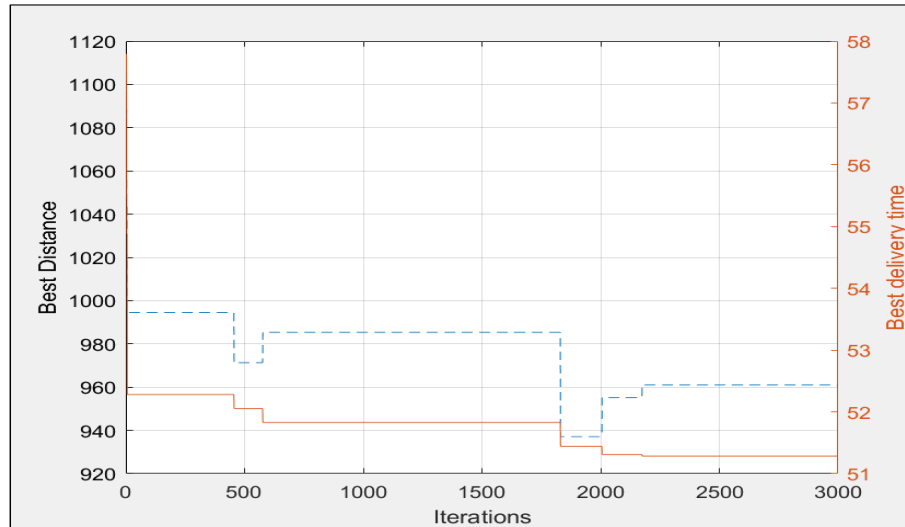
(a) Trucks-only



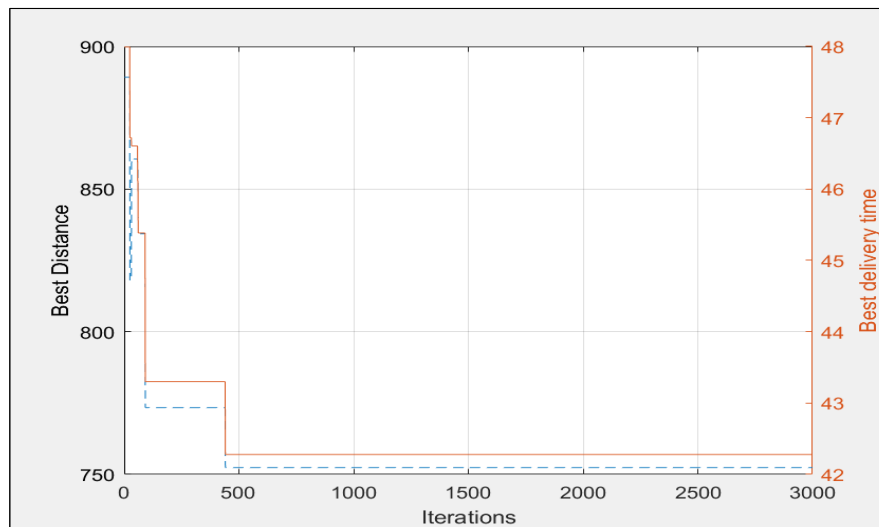
(b) Drones and trucks

Figure 38: 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-73-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 152

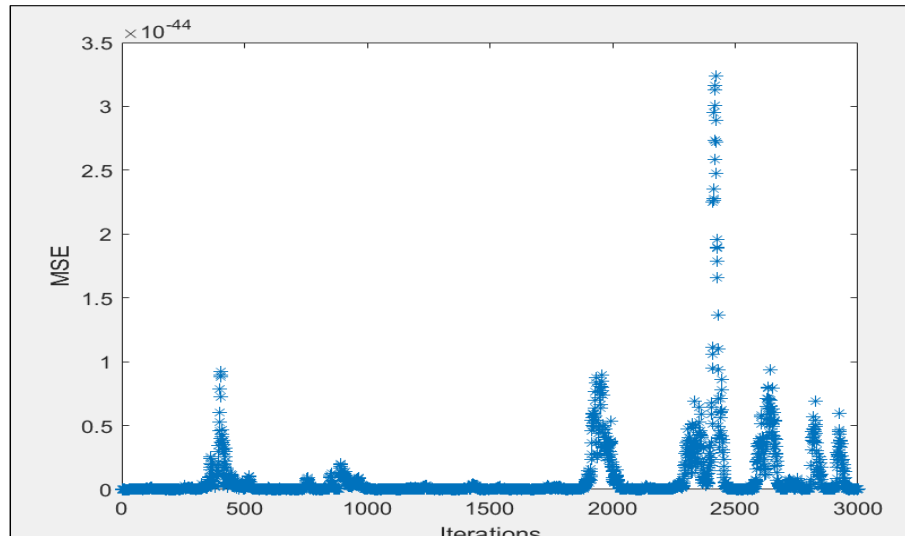


(a) Trucks-only

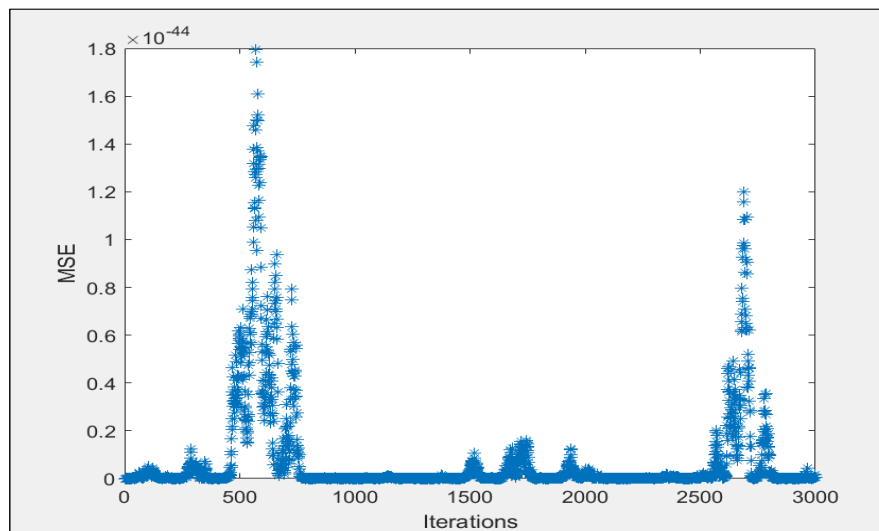


(b) Drones and trucks

Figure 39: Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-73-n50)



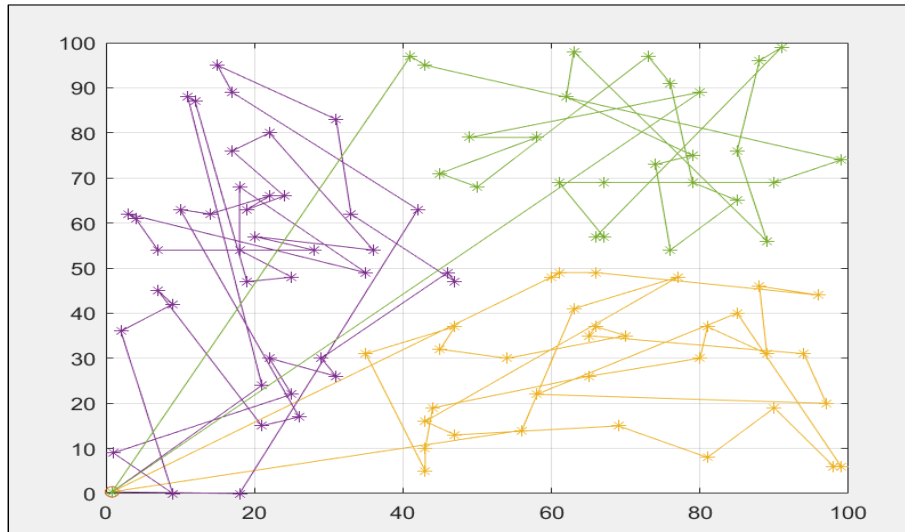
(a) Trucks-only



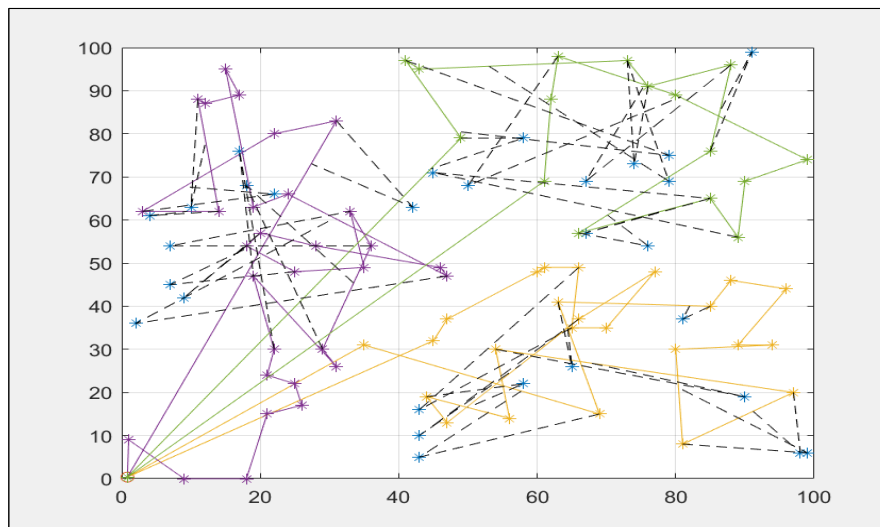
(b) Drones and trucks

Figure 40: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-73-n50)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 154



(a) Trucks-only

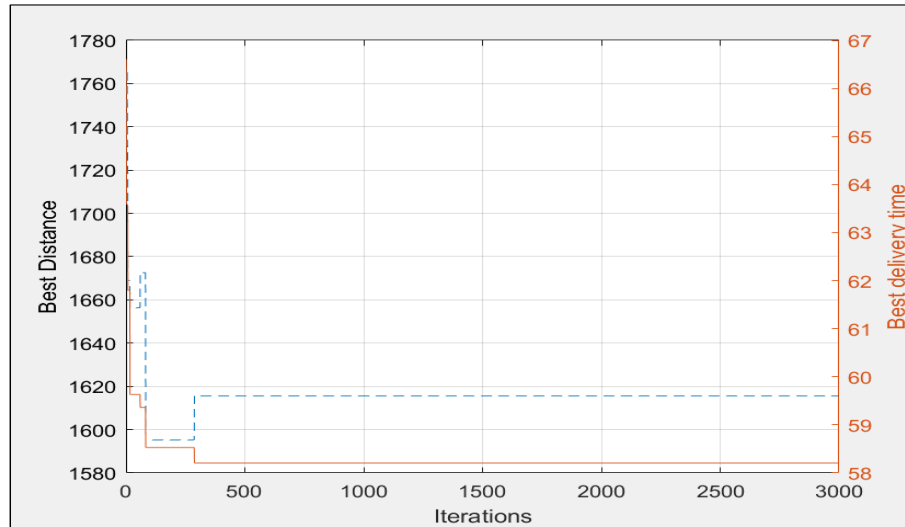


(b) Drones and trucks

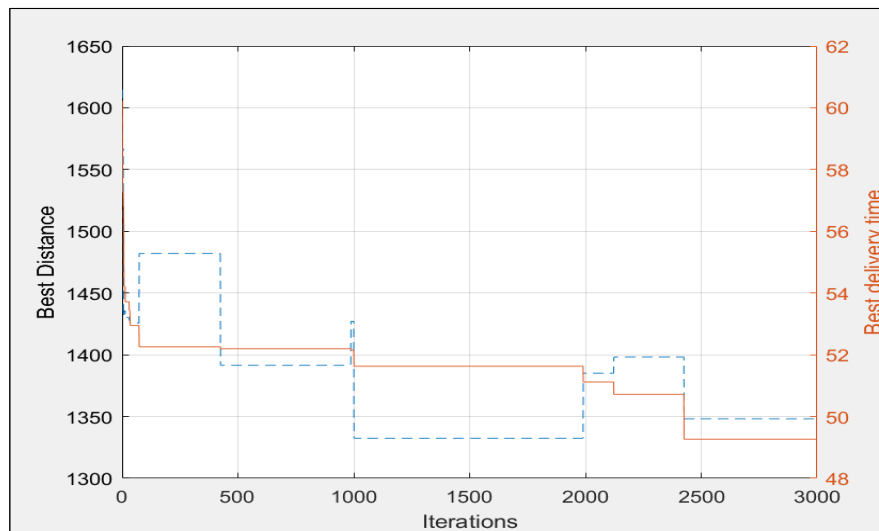
Figure 41: 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-91-n100)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

155

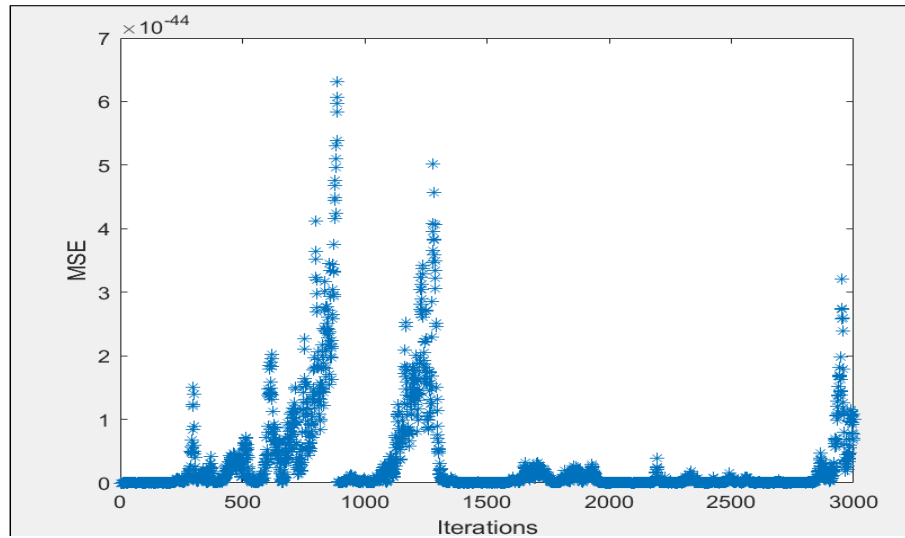


(a) Trucks-only

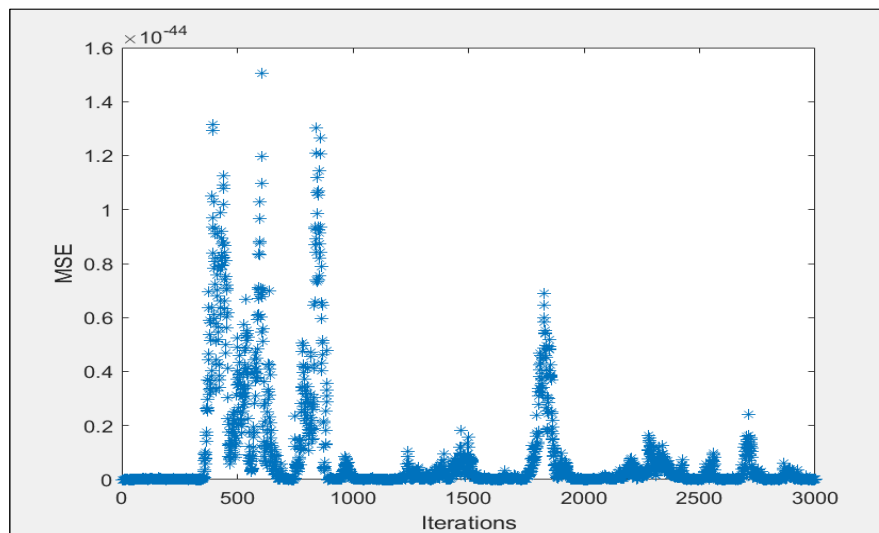


(b) Drones and trucks

Figure 42: Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-91-n100)



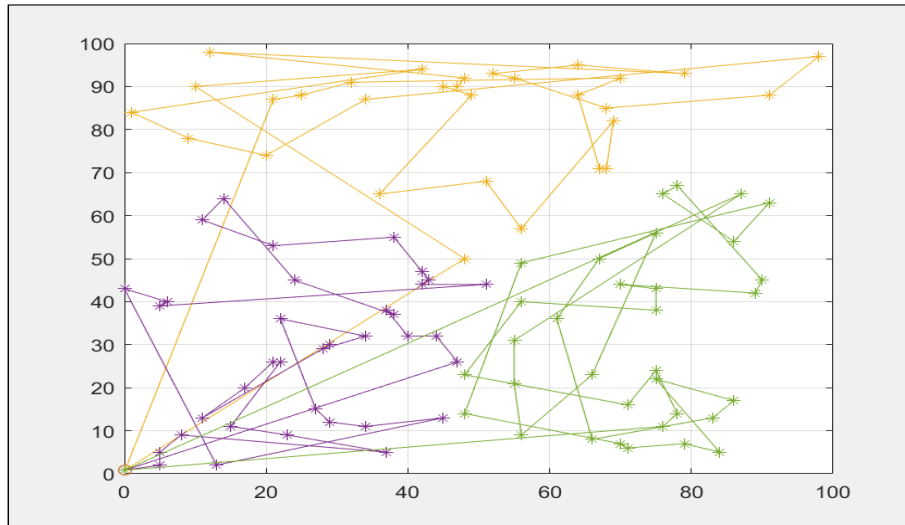
(a) Trucks-only



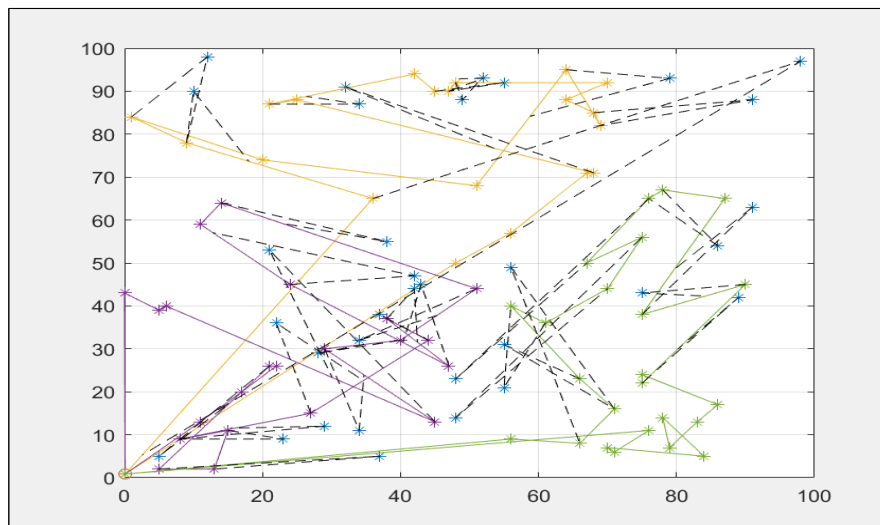
(b) Drones and trucks

Figure 43: Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-91-n100)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES 157



(a) Trucks-only

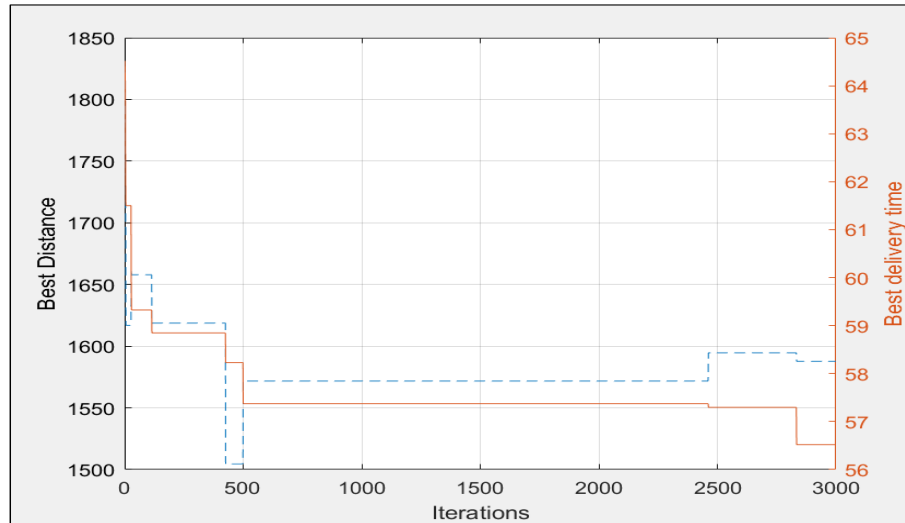


(b) Drones and trucks

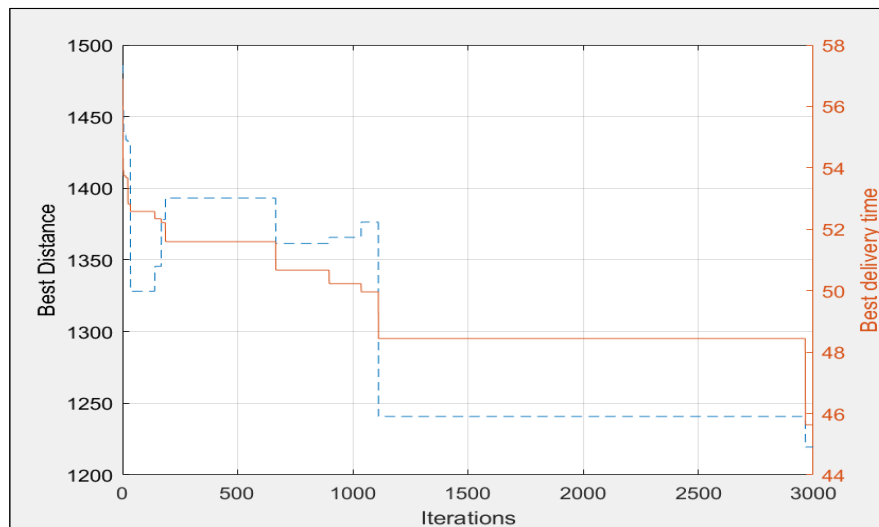
Figure 44: 100 customers receiving deliveries from either a truck only system or a multiple drone-truck delivery system (Uniform-92-n100)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

158



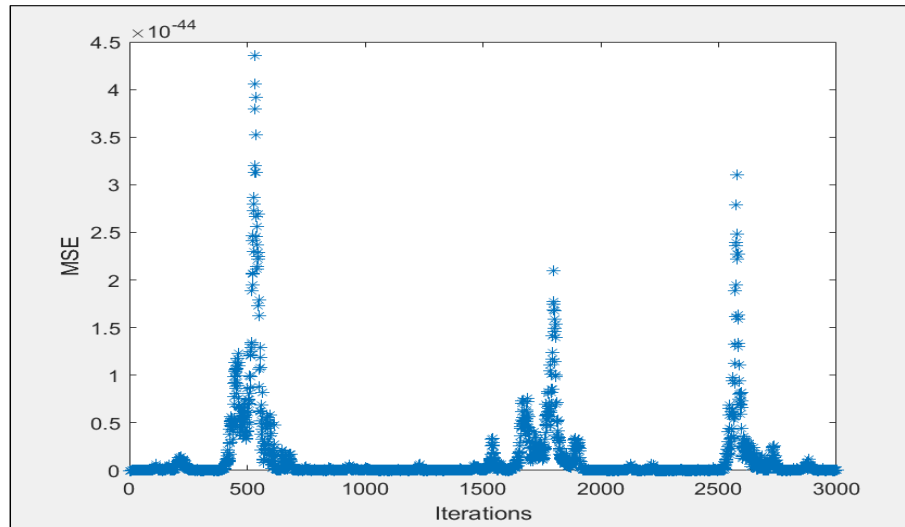
(a) Trucks-only



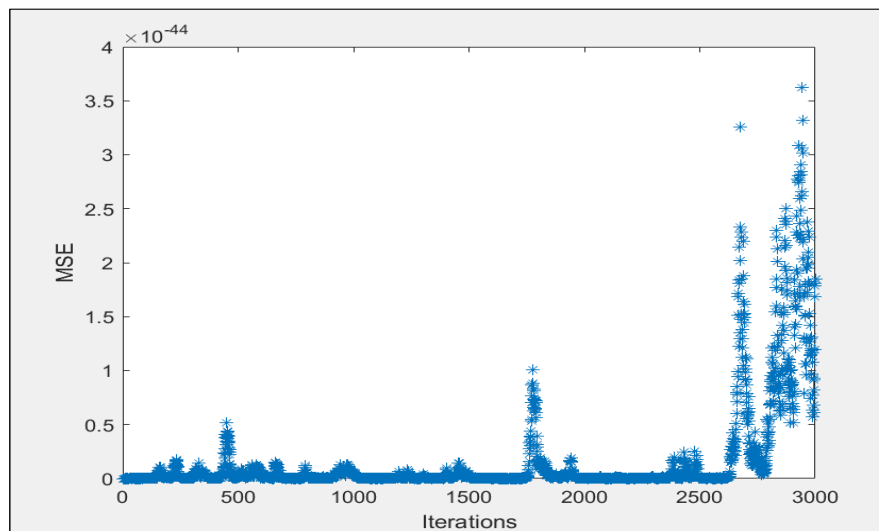
(b) Drones and trucks

Figure 45: Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-92-n100)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 159



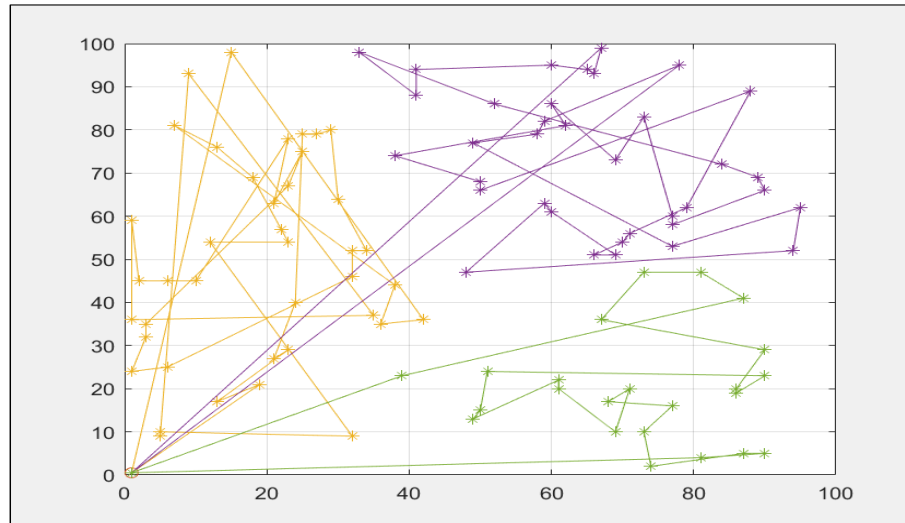
(a) Trucks-only



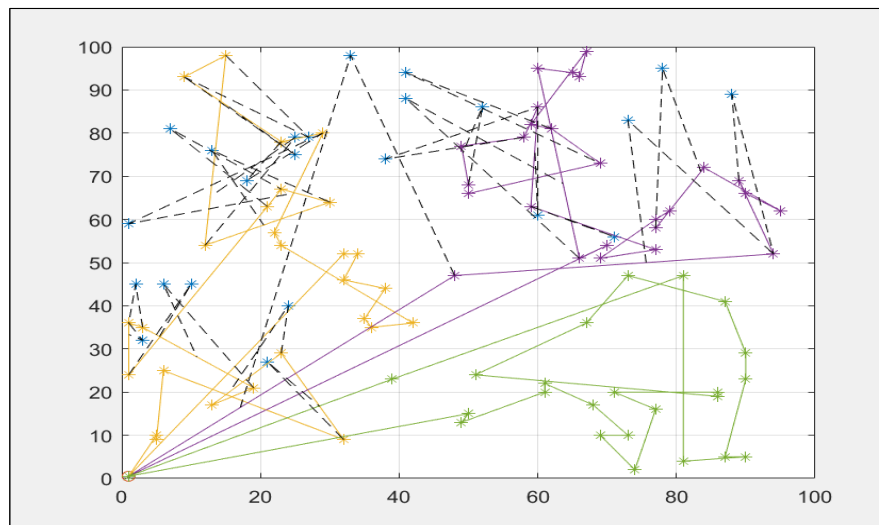
(b) Drones and trucks

Figure 46: Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-92-n100)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 160



(a) Trucks-only

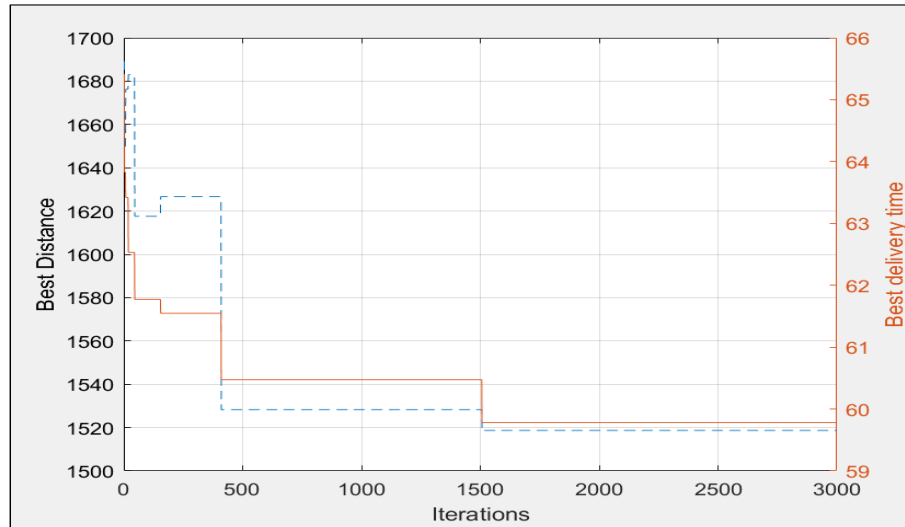


(b) Drones and trucks

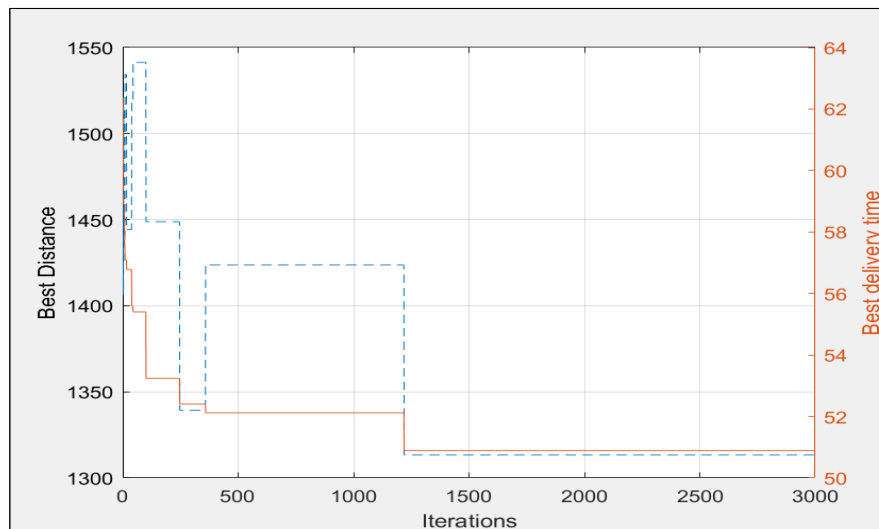
Figure 47: 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-93-n100)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

161

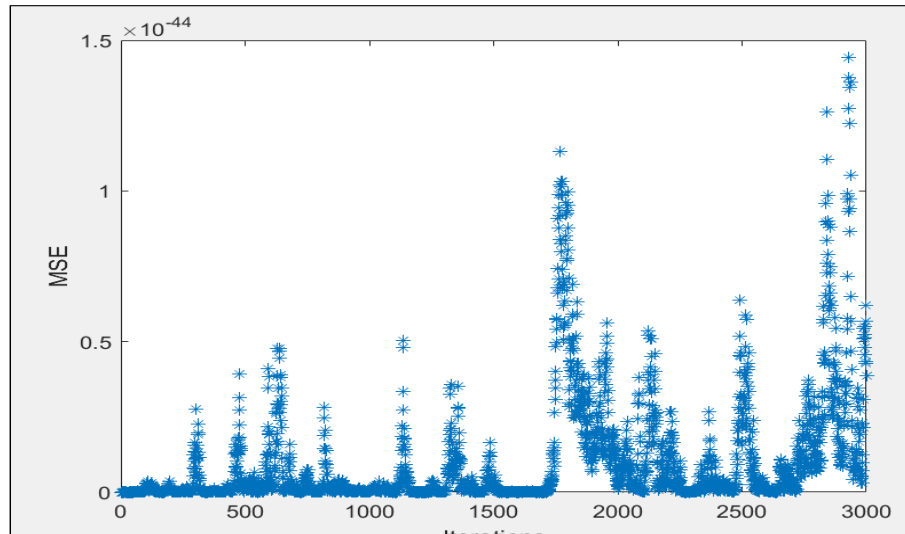


(a) Trucks-only

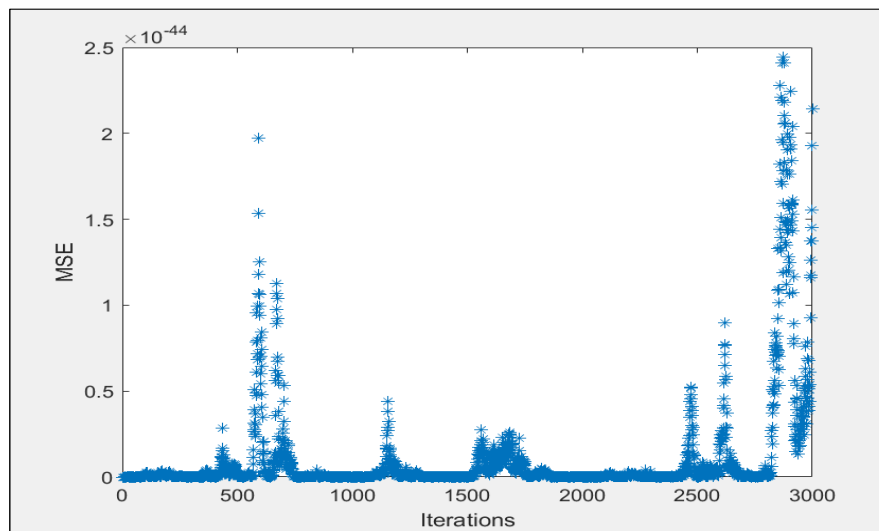


(b) Drones and trucks

Figure 48: Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-93-n100)



(a) Trucks-only

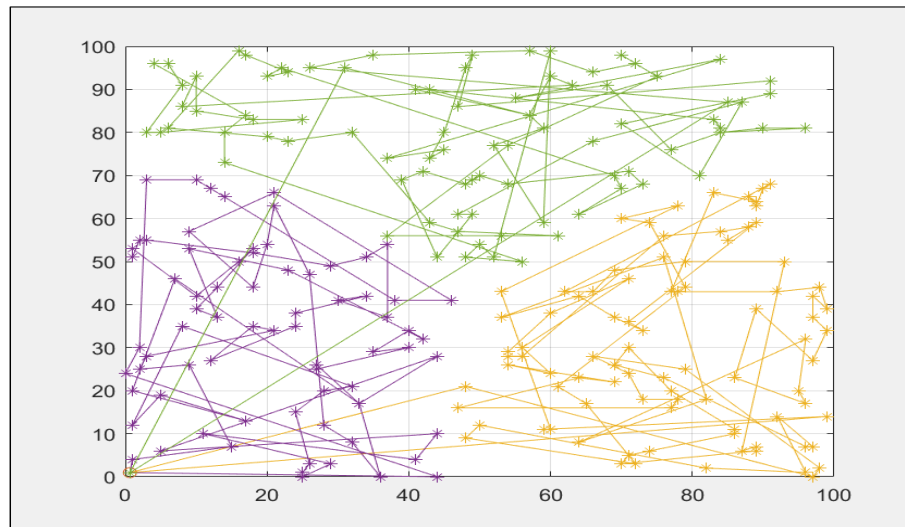


(b) Drones and trucks

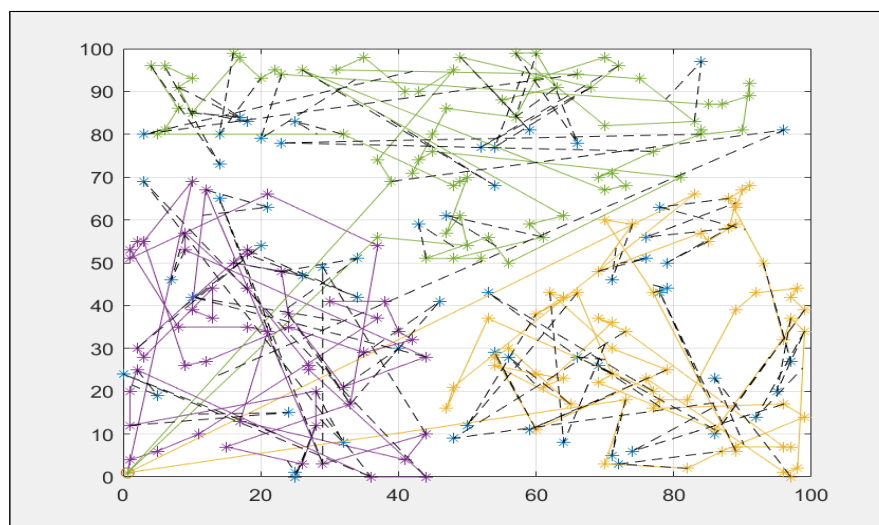
Figure 49: Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-93-n100)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

163



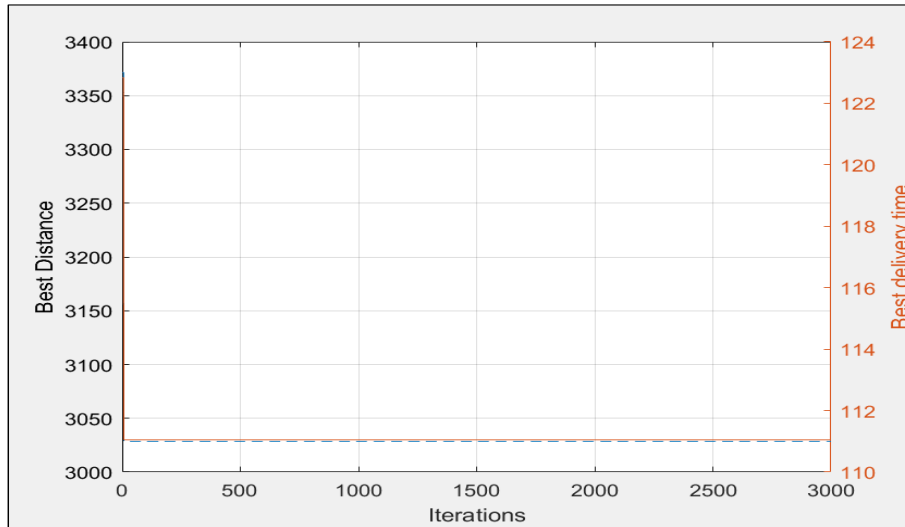
(a) Trucks-only



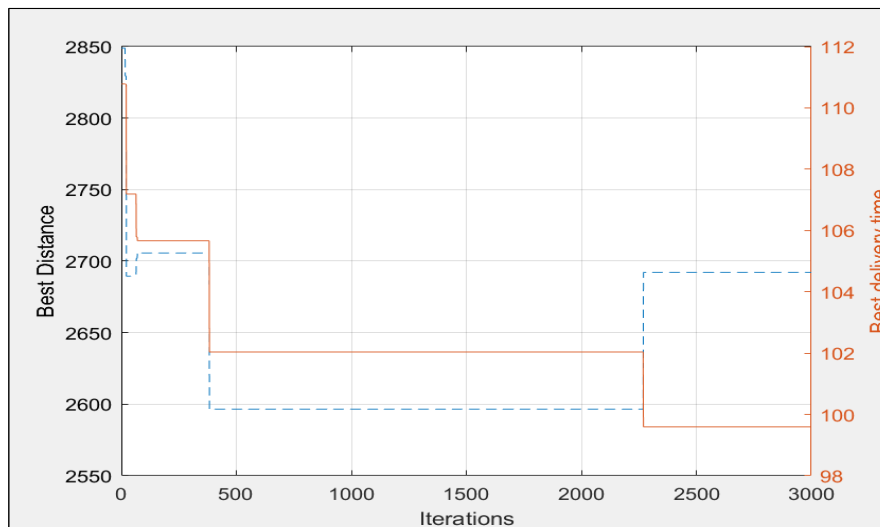
(b) Drones and trucks

Figure 50: 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-1-n250)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM WITH INTERCEPTION EXAMPLES 164

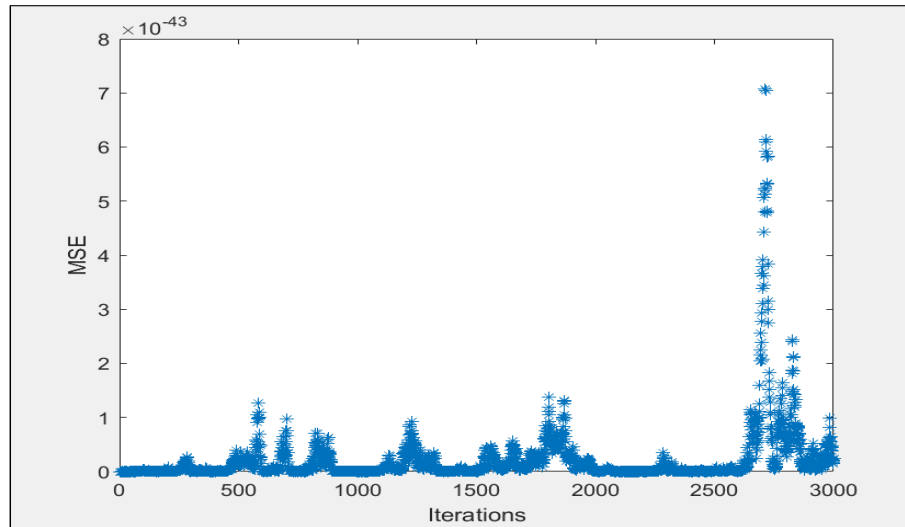


(a) Trucks-only

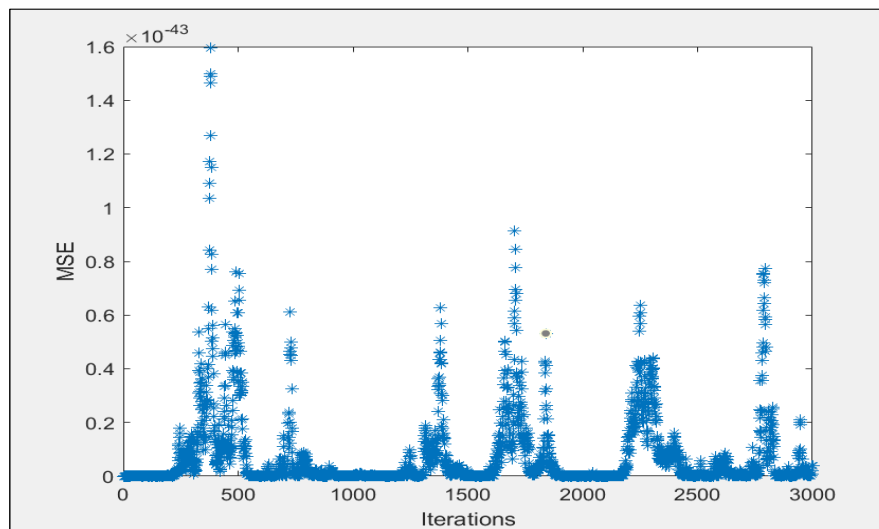


(b) Drones and trucks

Figure 51: Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-1-n250)



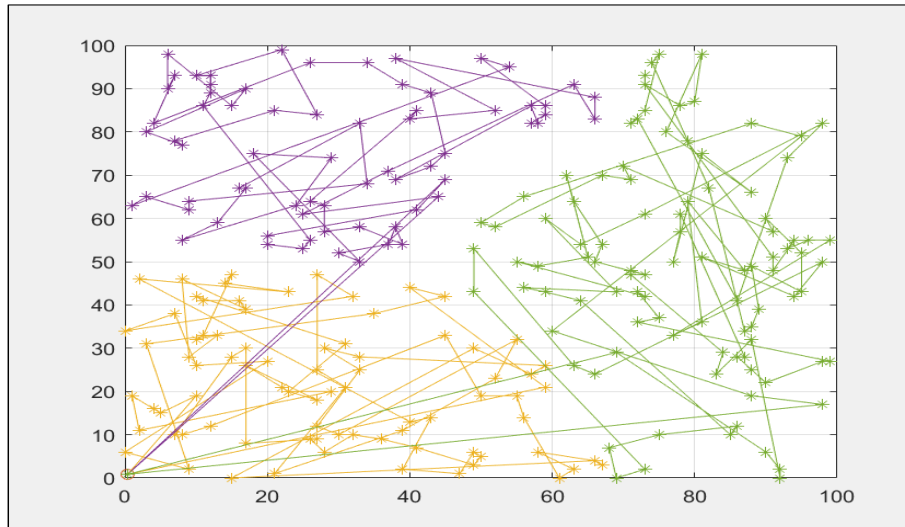
(a) Trucks-only



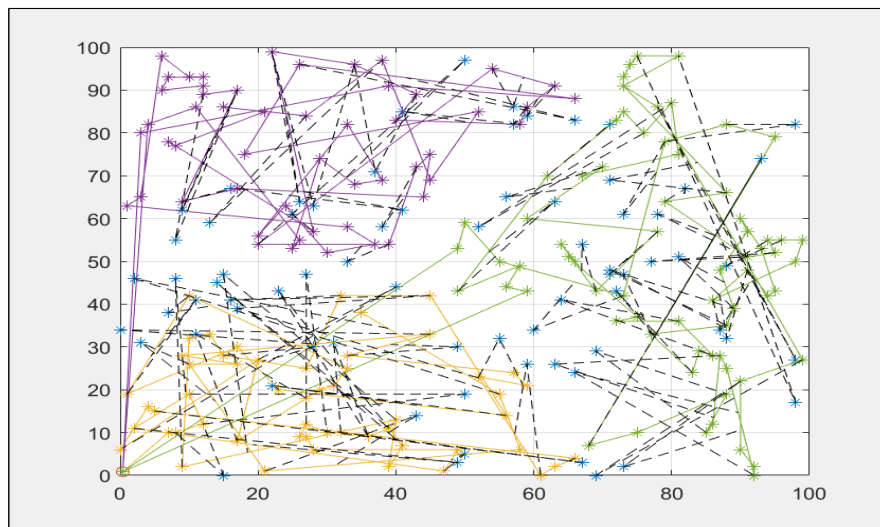
(b) Drones and trucks

Figure 52: Solutions search diversity for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-1-n250)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 166



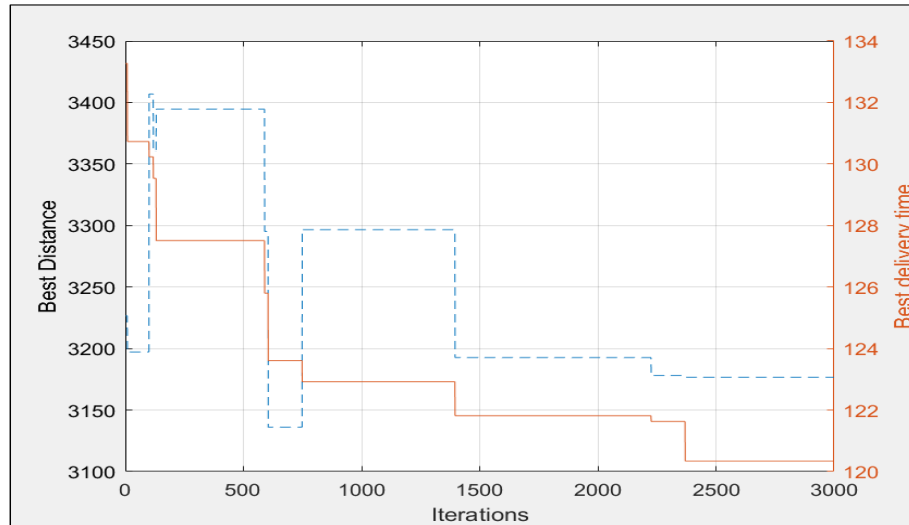
(a) Trucks-only



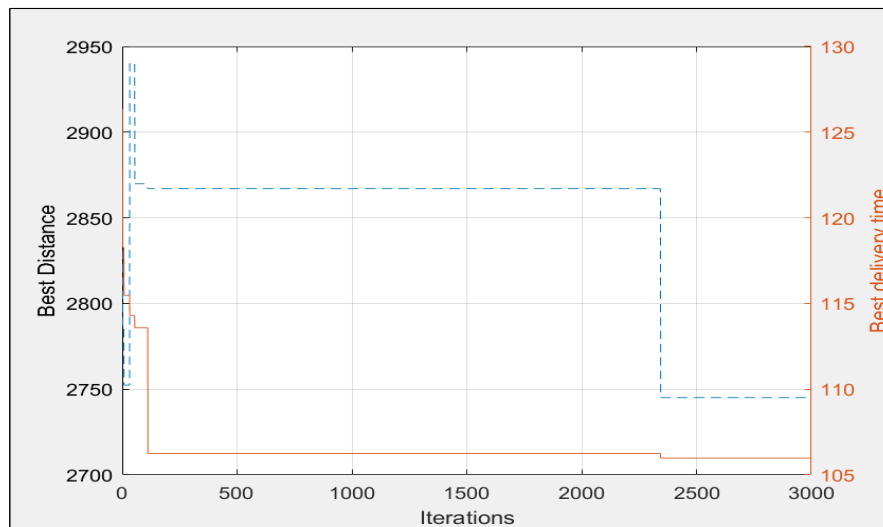
(b) Drones and trucks

Figure 53: 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-2-n250)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 167



(a) Trucks-only

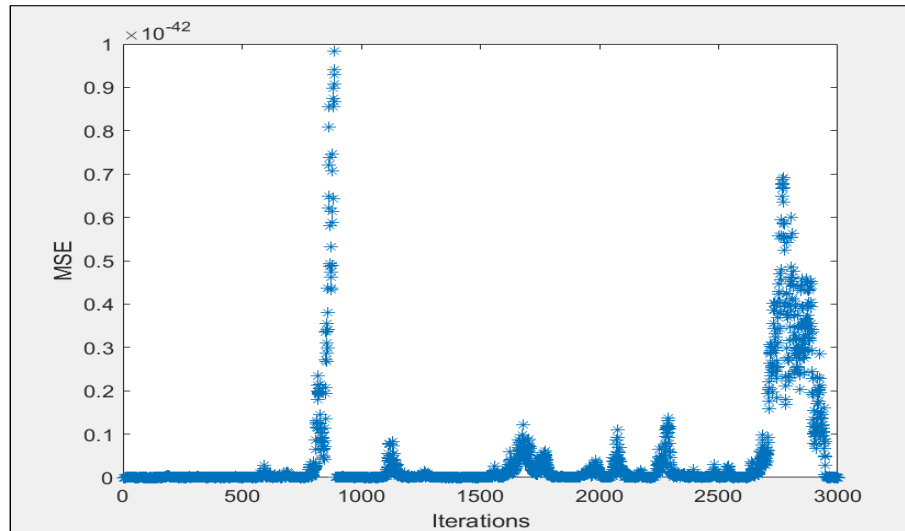


(b) Drones and trucks

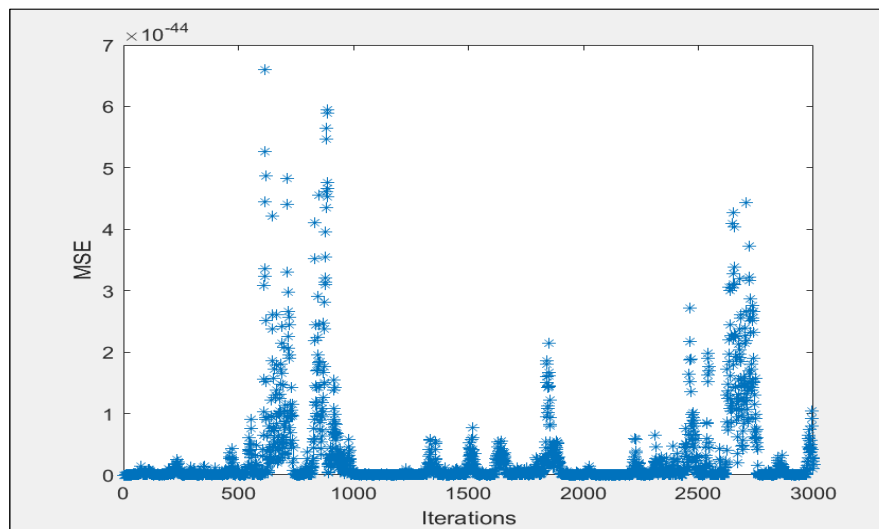
Figure 54: Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-2-n250)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

168



(a) Trucks-only

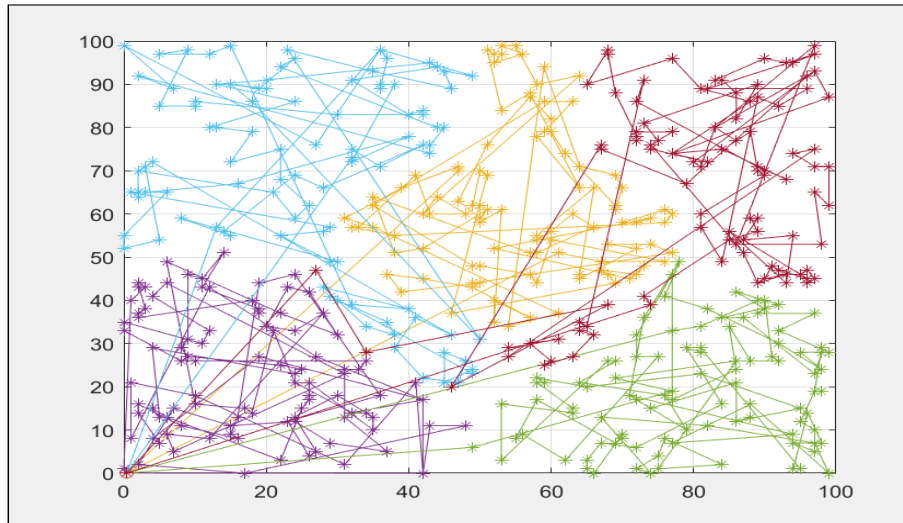


(b) Drones and trucks

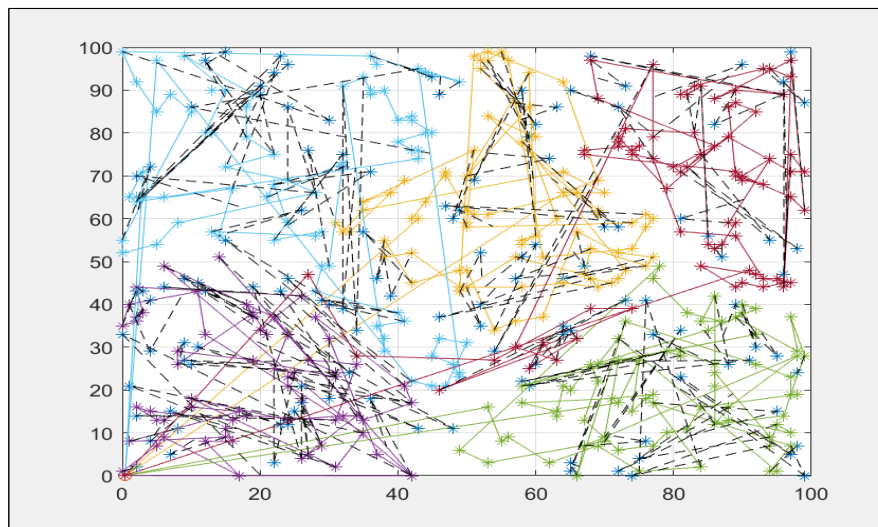
Figure 55: Solutions search diversity for 250y customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-2-n250)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

169



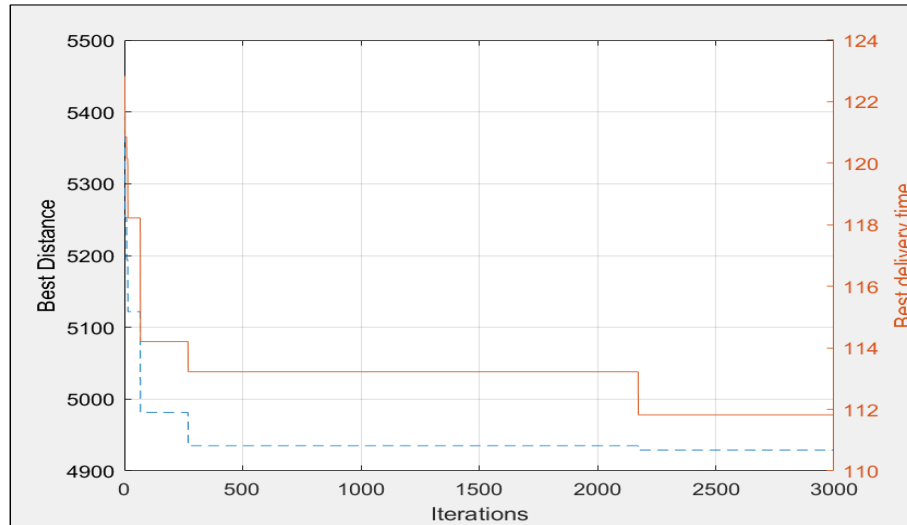
(a) Trucks-only



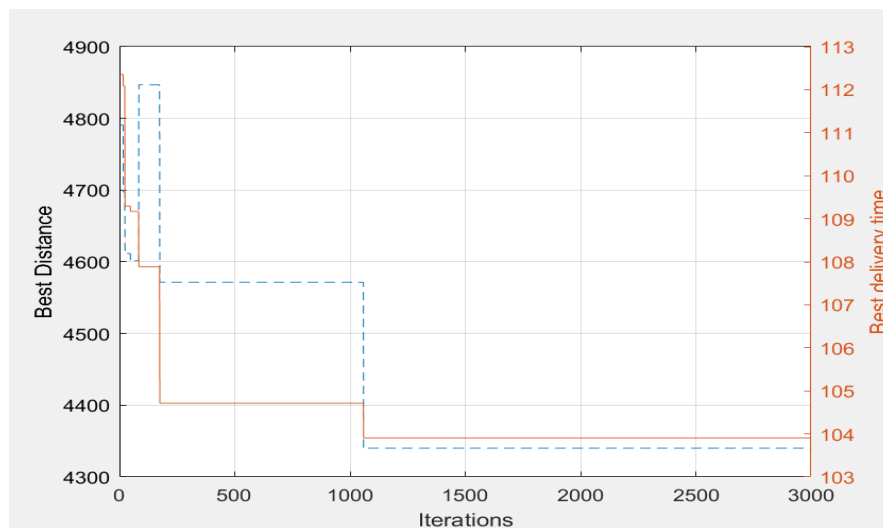
(b) Drones and trucks

Figure 56: 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-5-n500)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES 170



(a) Trucks-only

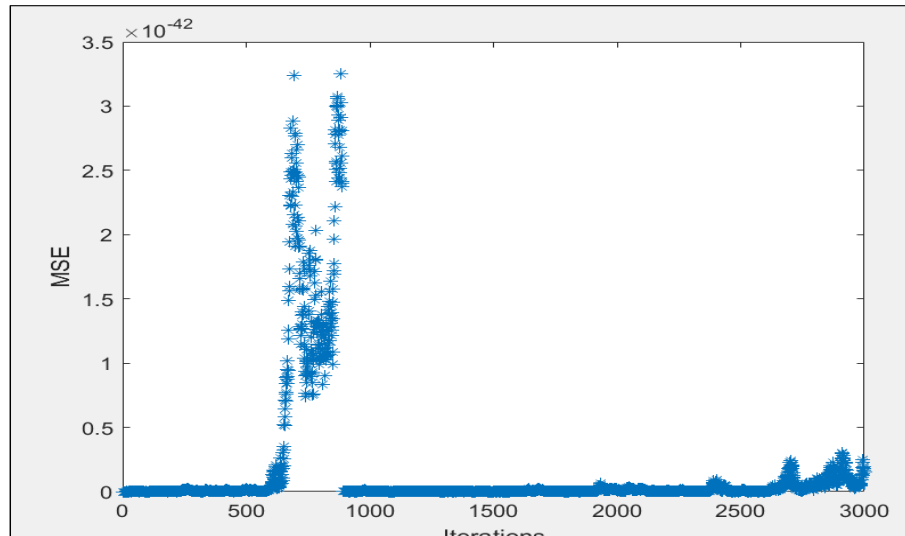


(b) Drones and trucks

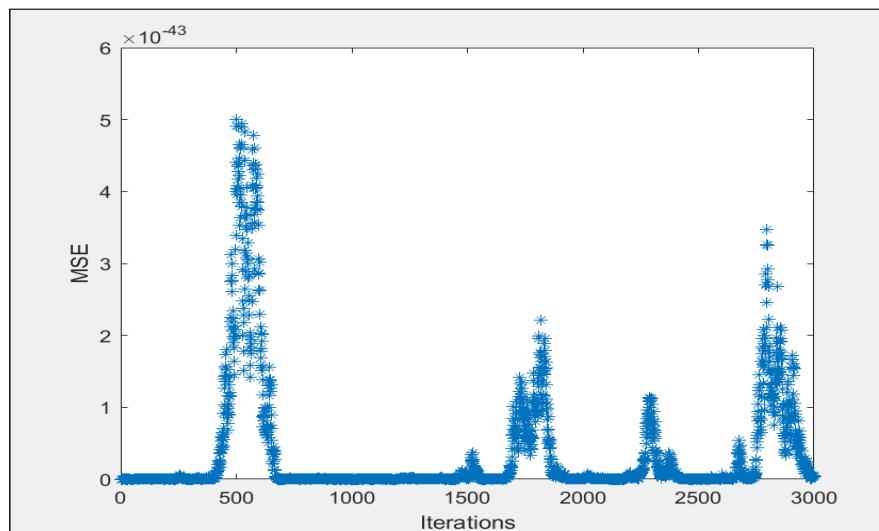
Figure 57: Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-5-n500)

ADDITIONAL MULTIPLE TRUCK AND DRONE SCHEDULING PROBLEM
WITH INTERCEPTION EXAMPLES

171



(a) Trucks-only



(b) Drones and trucks

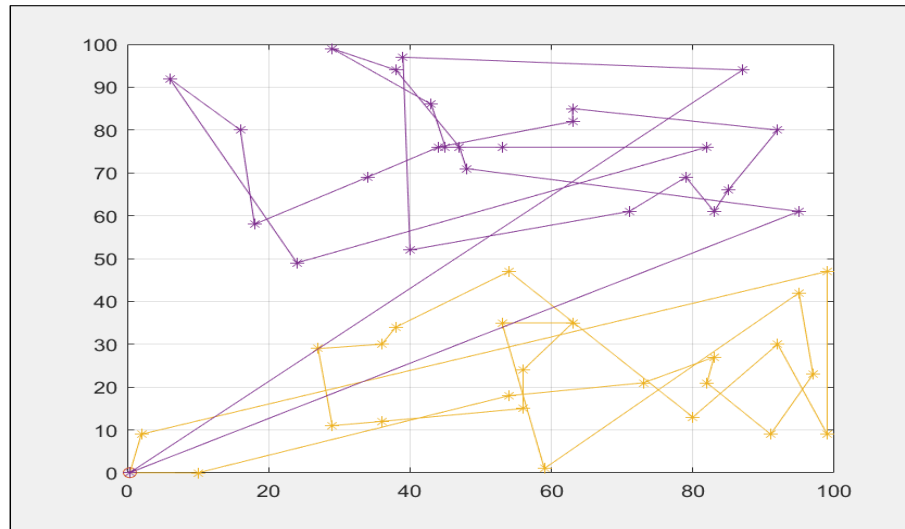
Figure 58: Solutions search diversity for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system (Uniform-5-n500)

Additional drones and trucks scheduling problem with interception and time window examples

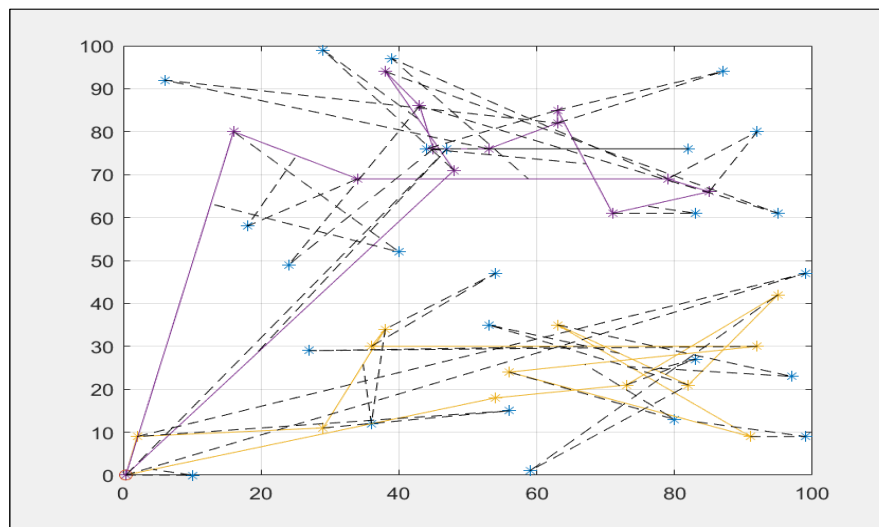
This appendix contains additional truck(s) and drone(s) scheduling problem with interception and time window constraint examples. A graphical representation of the VRPDTW solutions are shown in Figures 59, 62, 65, 65, 71, 74, 77, 80, and 83. Here customers receive parcels from drone and truck.

The best cost per iteration for the datasets is shown in Figure 60, 63, 66, 69, 72, 75, 78, 81, and 84.

The ants solutions search per iteration is shown in Figure 61, 64, 67, 70, 73, 76, 79, 82, and 85.

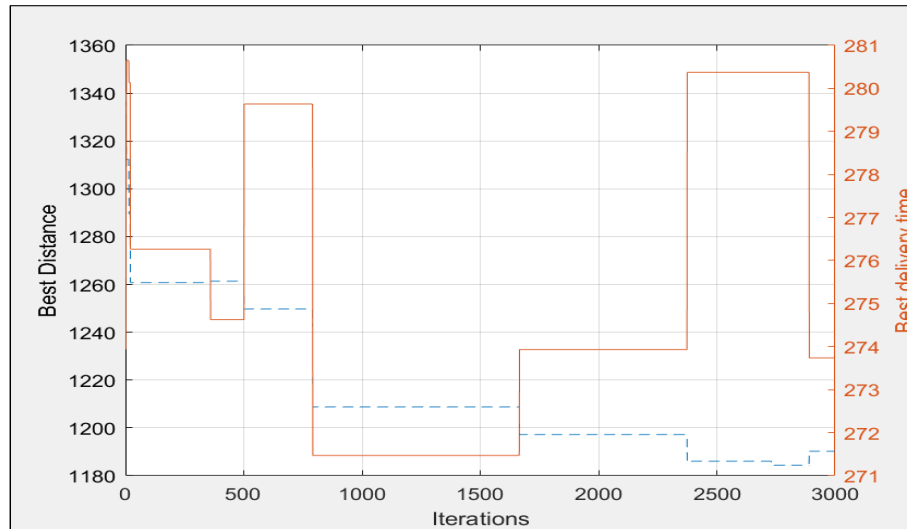


(a) Trucks-only

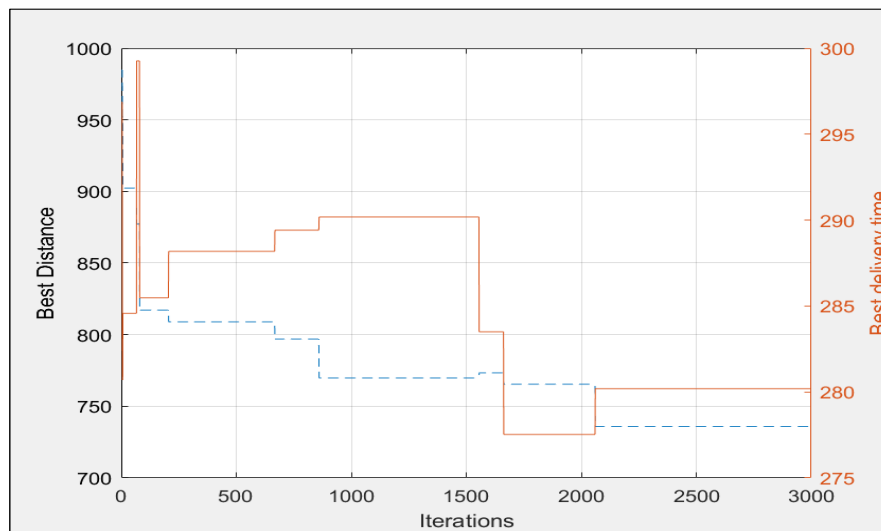


(b) Drones and trucks

Figure 59: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple trucks and drones delivery system, all with time windows (Uniform-71-n50)

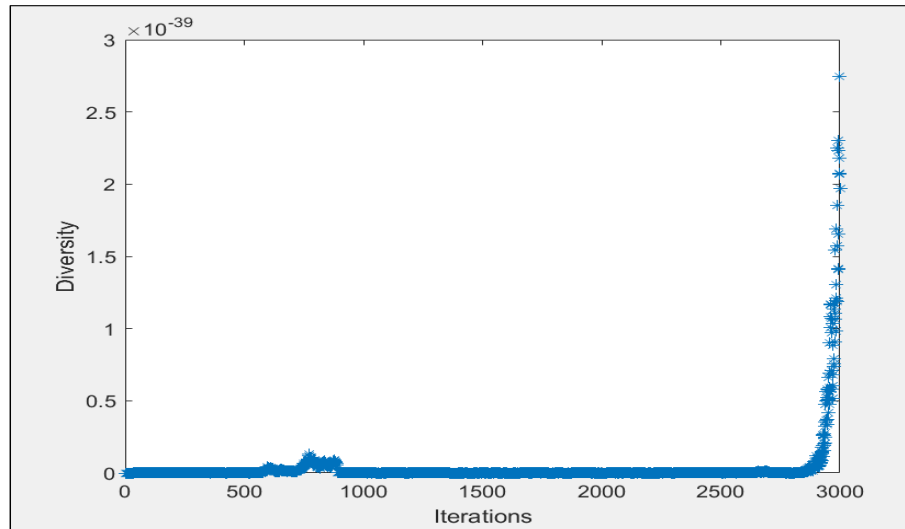


(a) Trucks-only

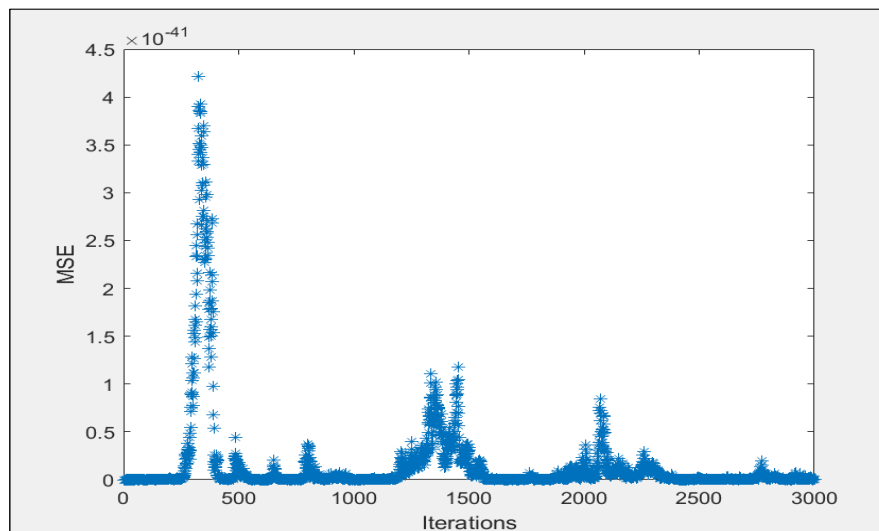


(b) Drones and trucks

Figure 60: Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-71-n50)

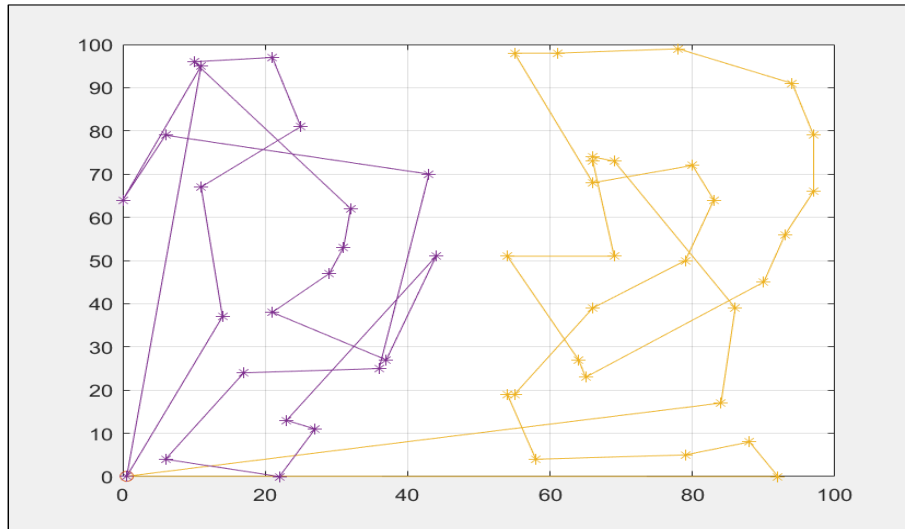


(a) Trucks-only

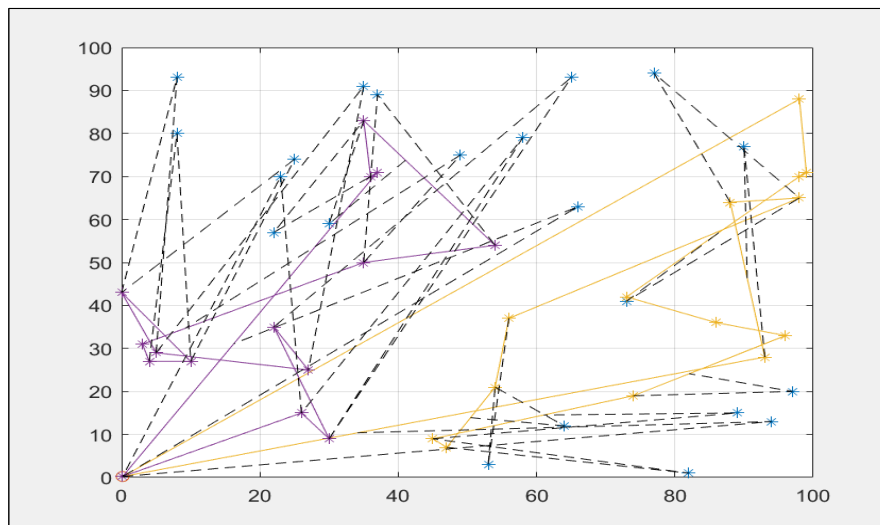


(b) Drones and trucks

Figure 61: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-71-n50)

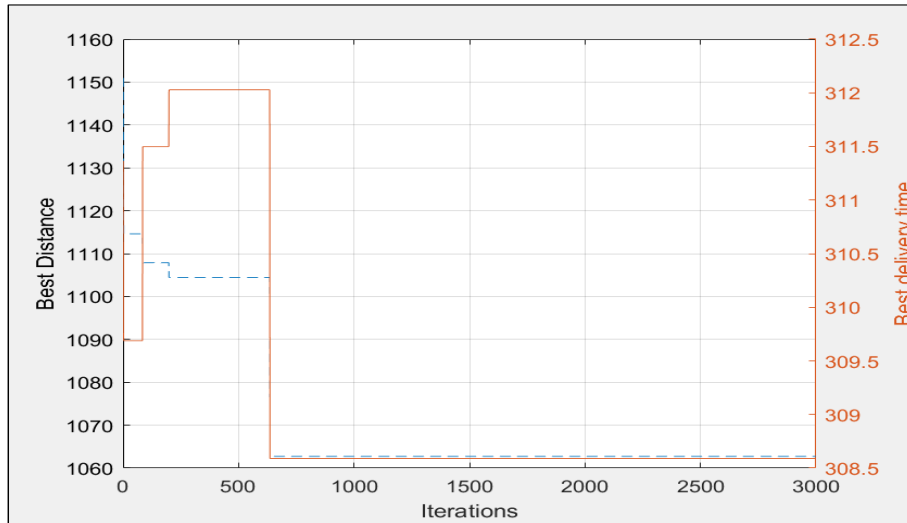


(a) Trucks-only

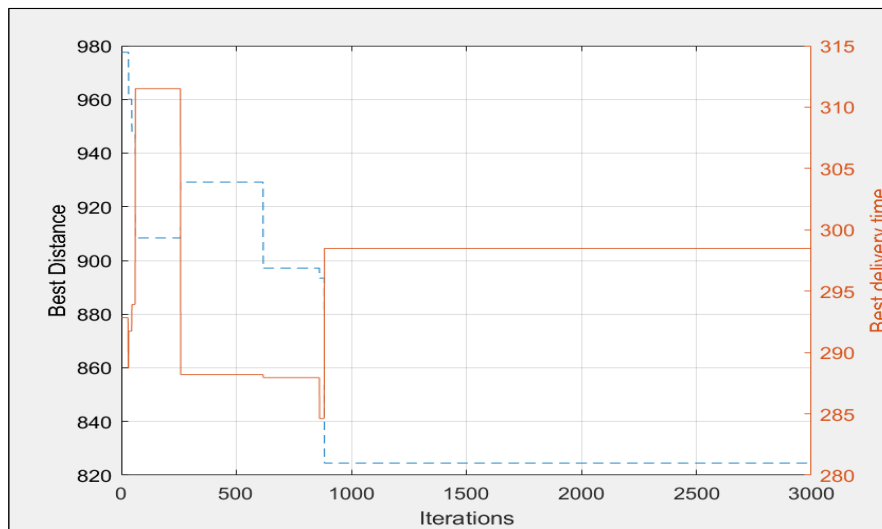


(b) Drones and trucks

Figure 62: 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-72-n50)

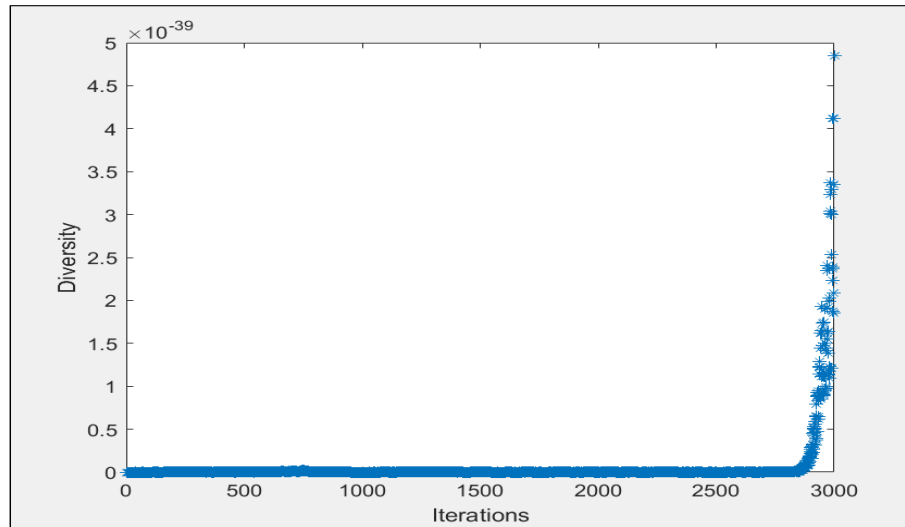


(a) Trucks only

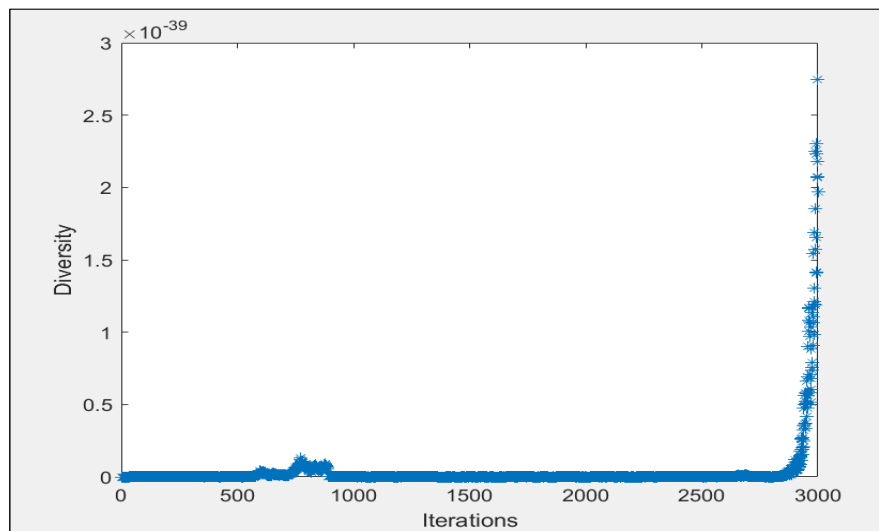


(b) Drones and trucks

Figure 63: Best solutions for 50 receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-72-n50)

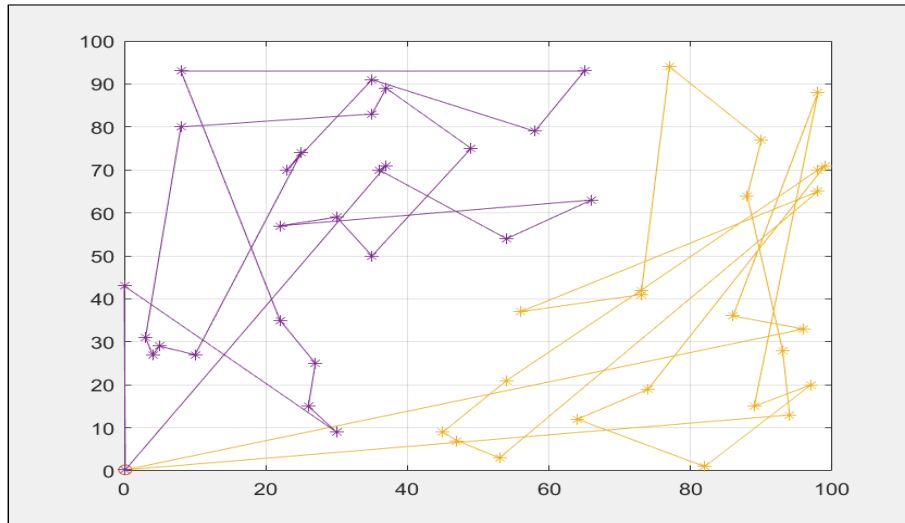


(a) Trucks-only

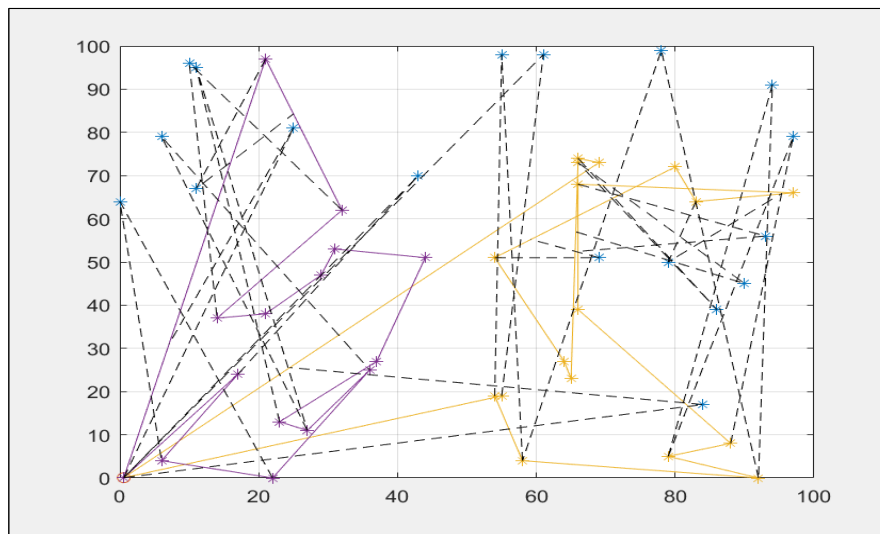


(b) Drones and trucks

Figure 64: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-72-n50)

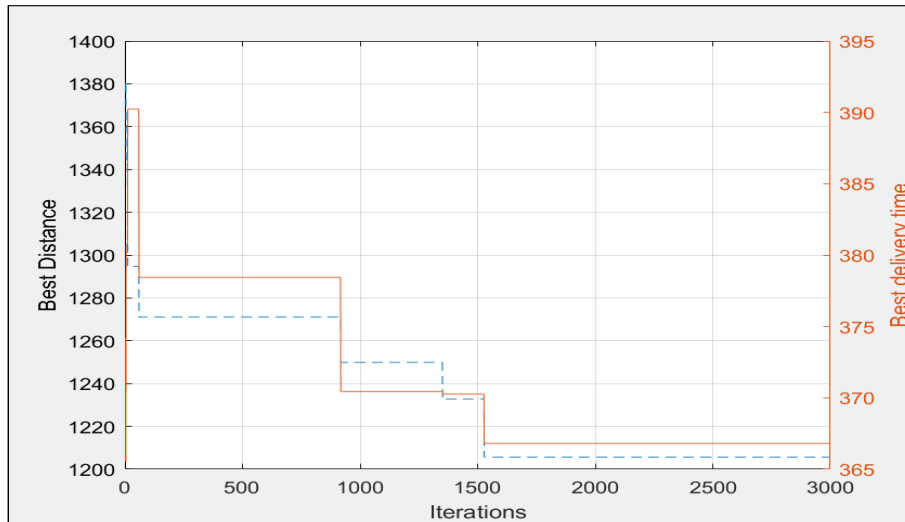


(a) Trucks-only

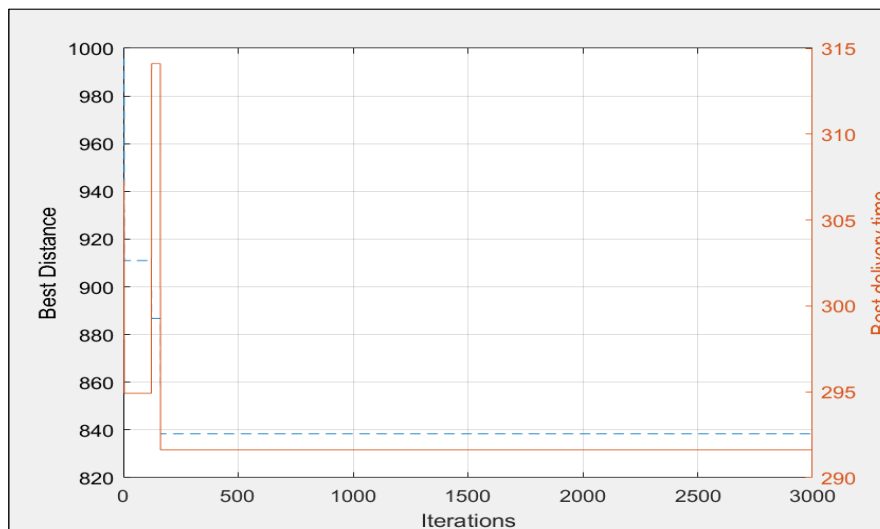


(b) Drones and trucks

Figure 65: 50 customers receiving deliveries from either a truck only system or a multiple drone-truck delivery system, all with time windows (Uniform-73-n50)

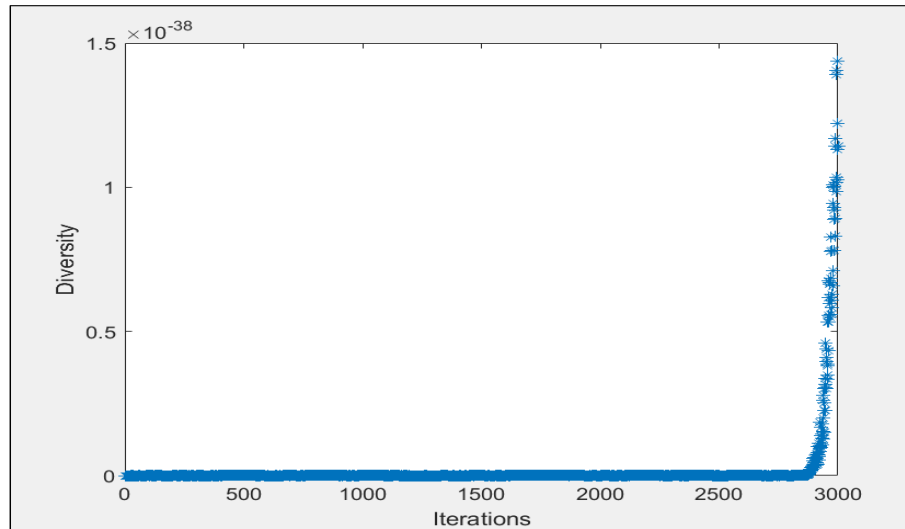


(a) Trucks-only

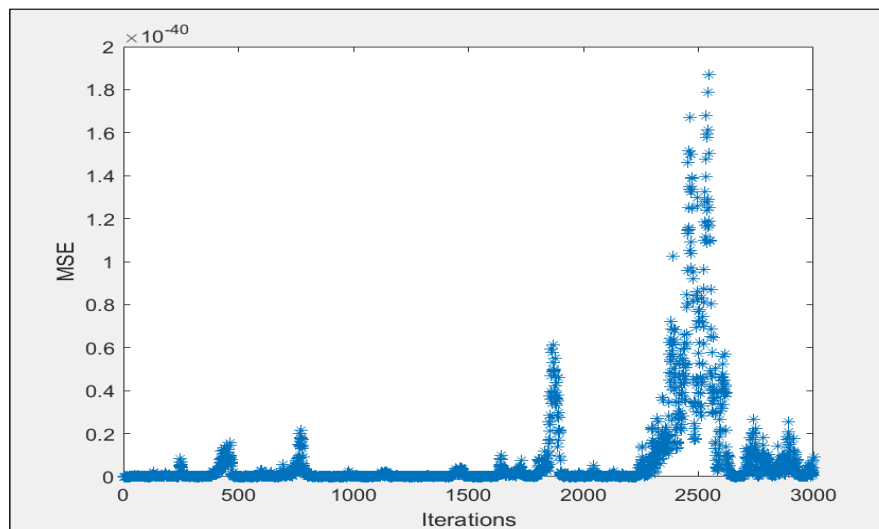


(b) Drones and trucks

Figure 66: Best solutions for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-73-n50)

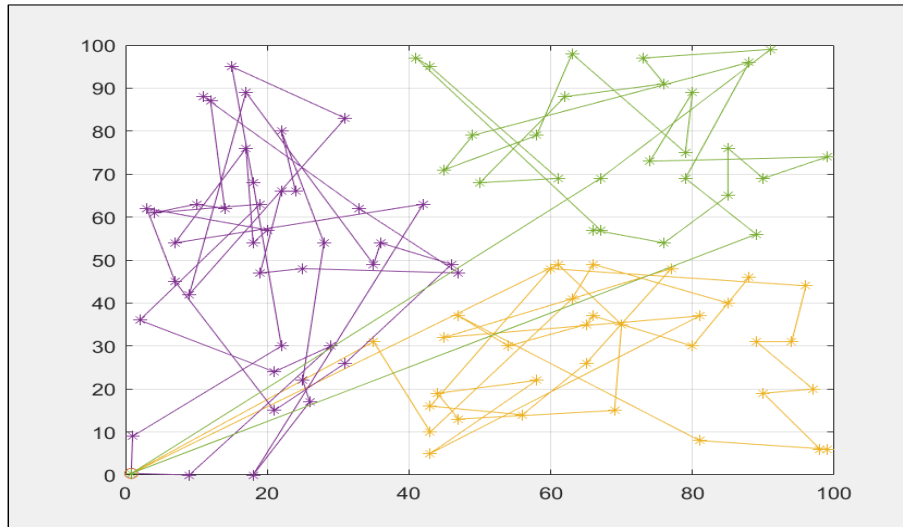


(a) Trucks-only

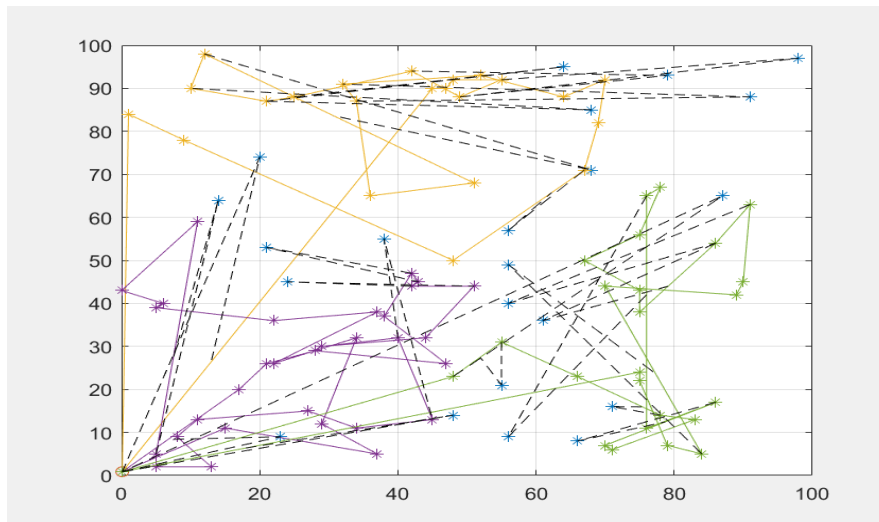


(b) Drones and trucks

Figure 67: Solutions search diversity for 50 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-73-n50)

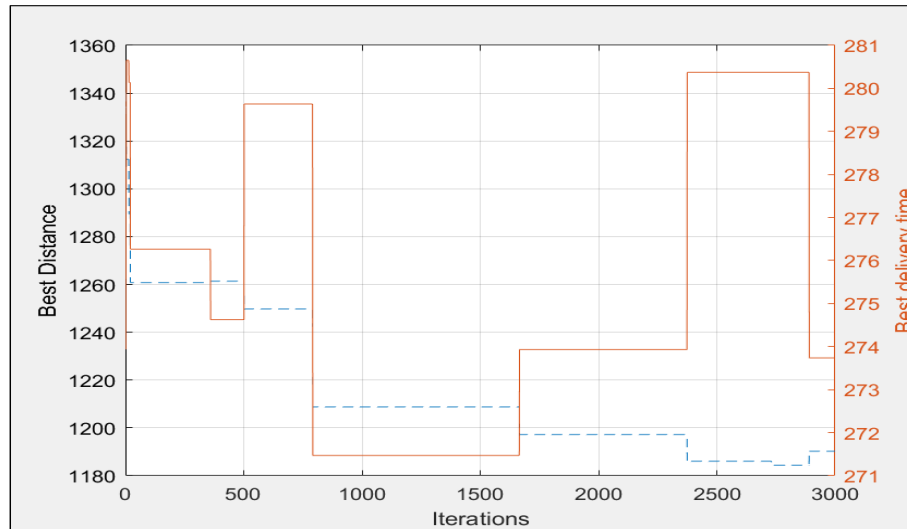


(a) Trucks-only

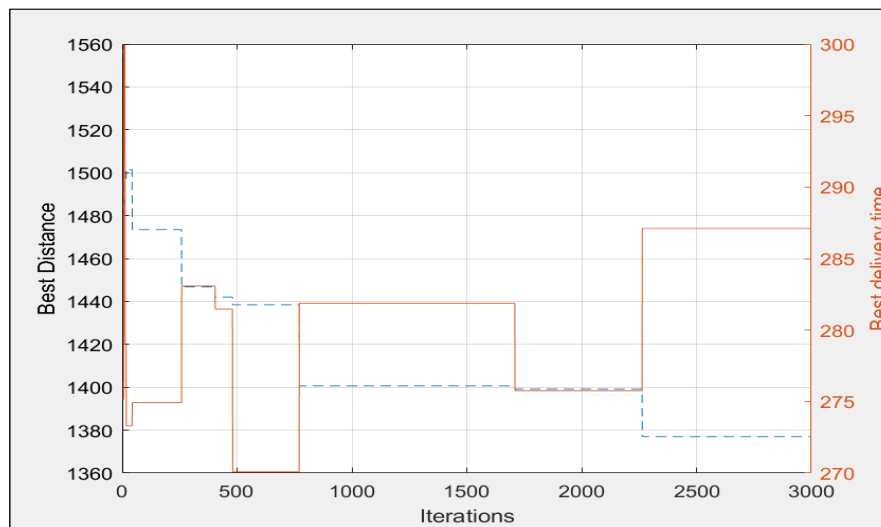


(b) Drones and trucks

Figure 68: 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-91-n100)

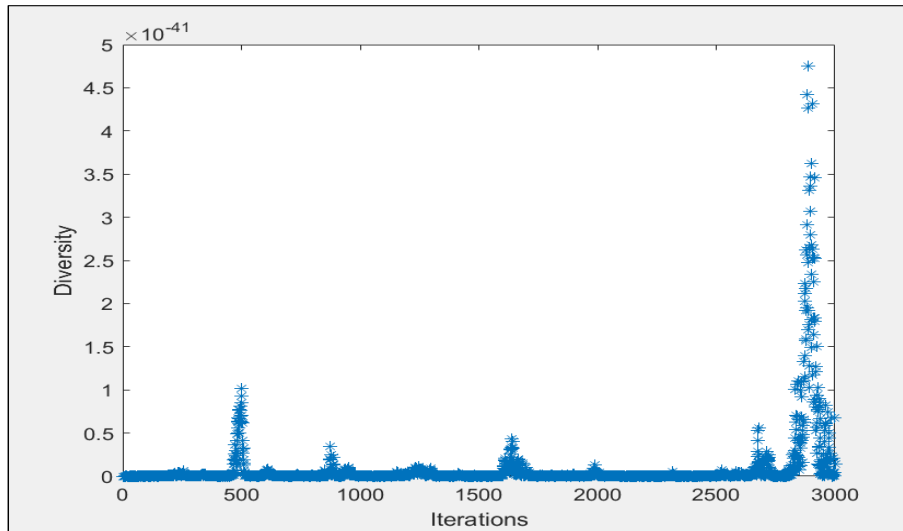


(a) Trucks-only

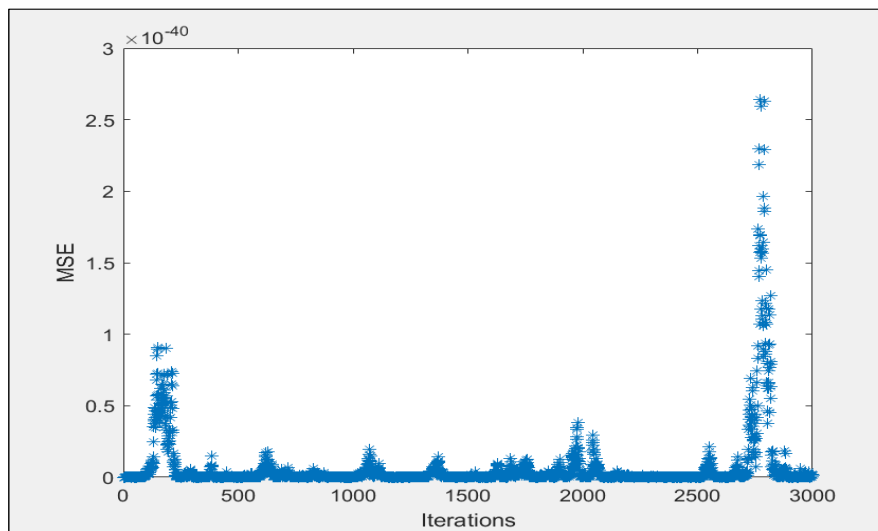


(b) Drones and trucks

Figure 69: Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-91-n100)



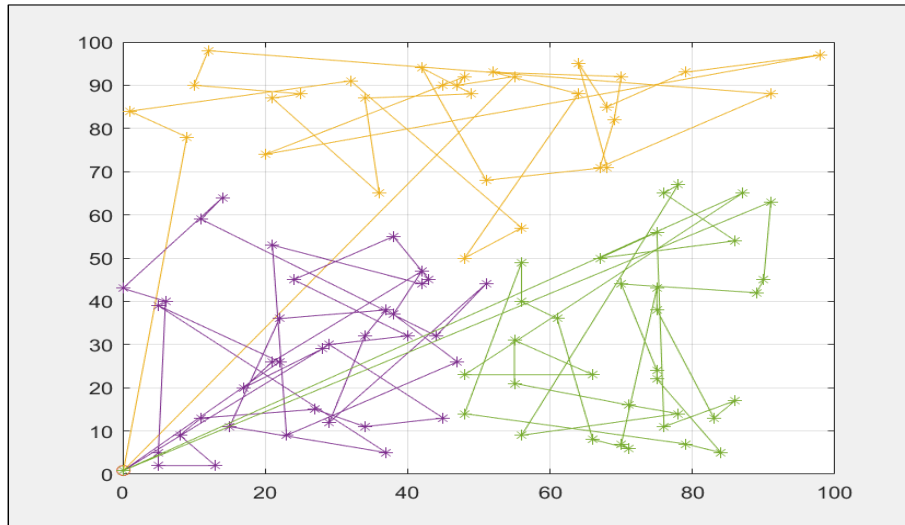
(a) Trucks-only



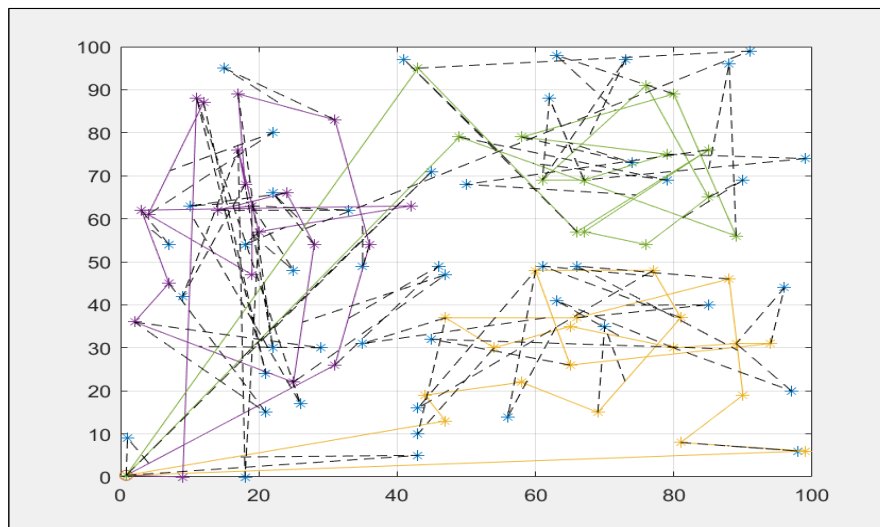
(b) Drones and trucks

Figure 70: Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-91-n100)

ADDITIONAL DRONES AND TRUCKS SCHEDULING PROBLEM WITH INTERCEPTION AND TIME WINDOW EXAMPLES

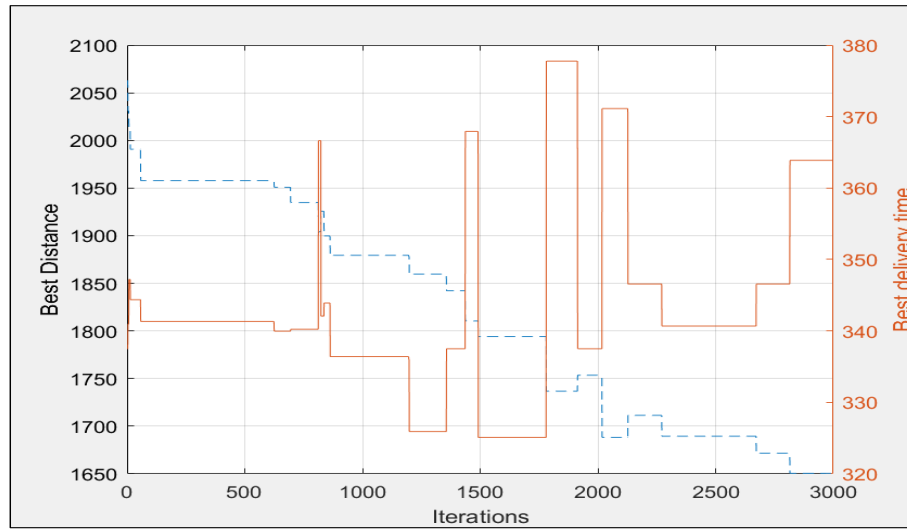


(a) Trucks-only

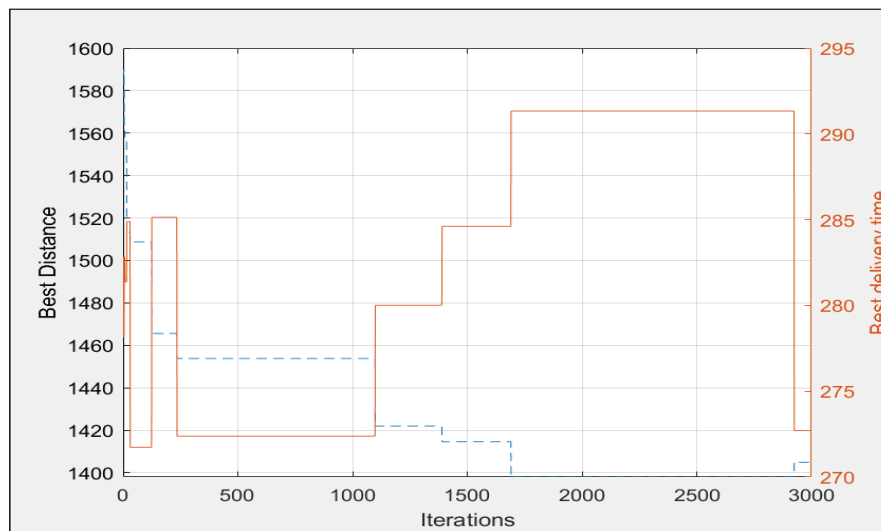


(b) Drones and trucks

Figure 71: 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-92-n100)

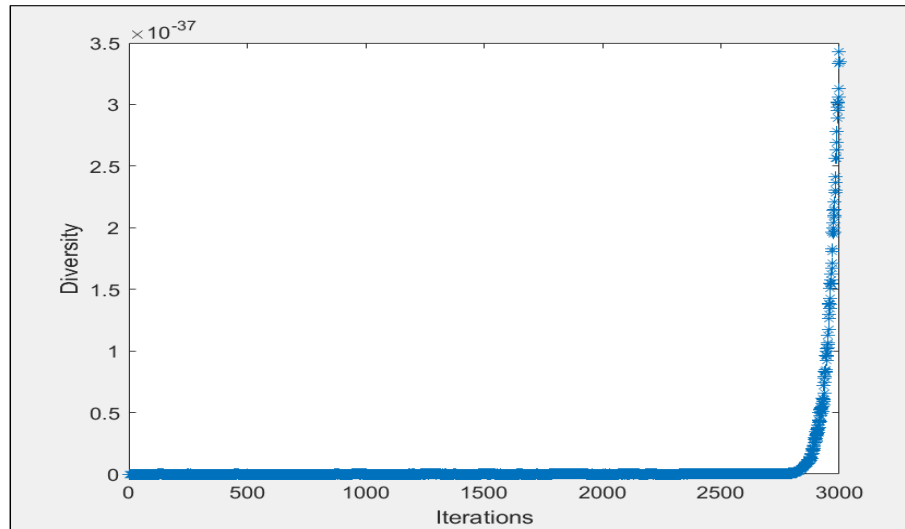


(a) Trucks-only

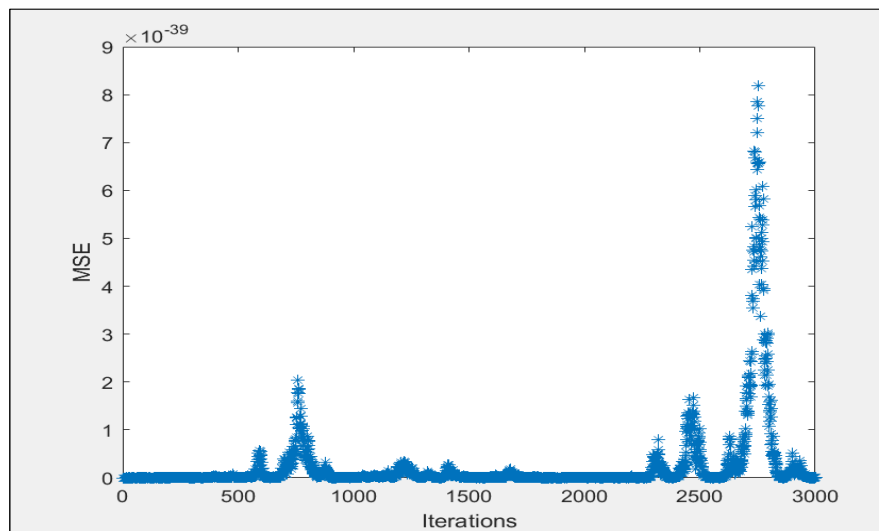


(b) Drones and trucks

Figure 72: Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-92-n100)

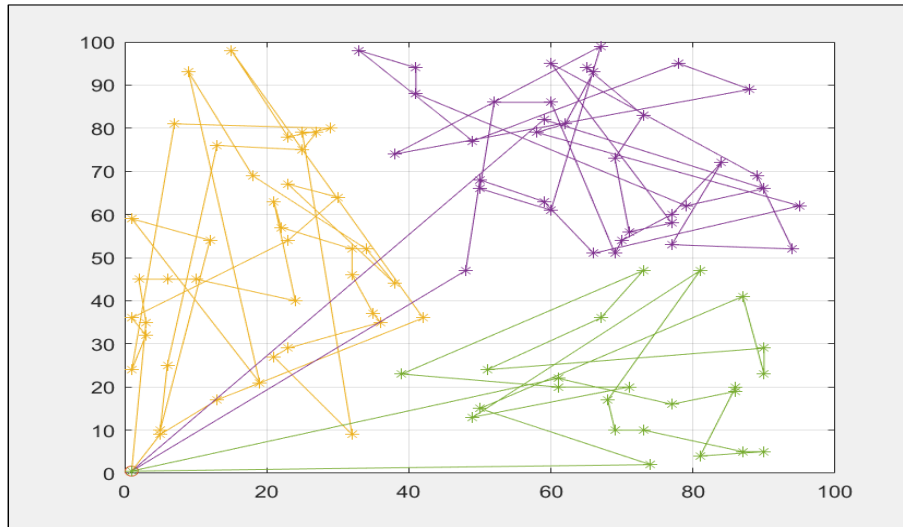


(a) Trucks-only

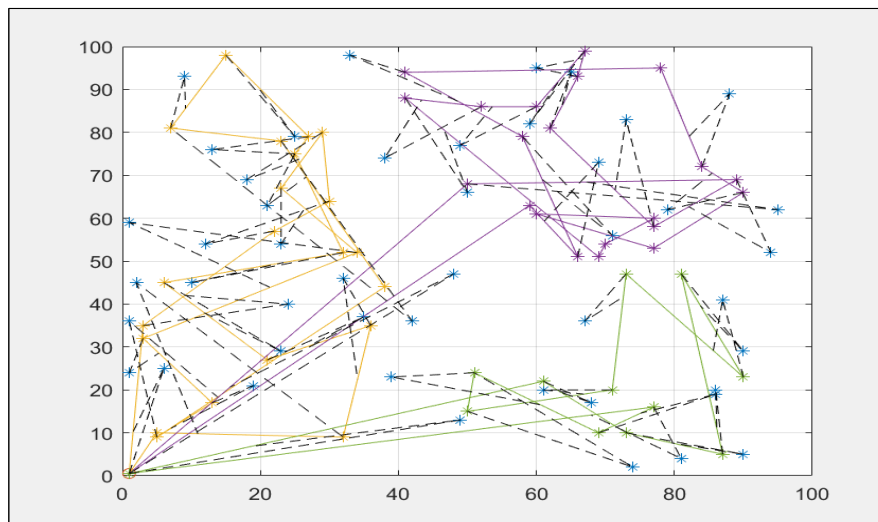


(b) Drones and trucks

Figure 73: Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-92-n100)

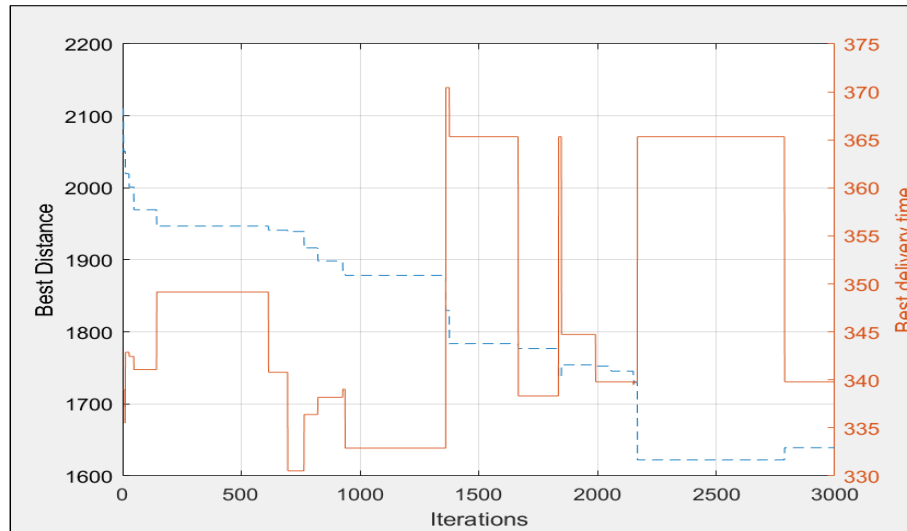


(a) Trucks-only

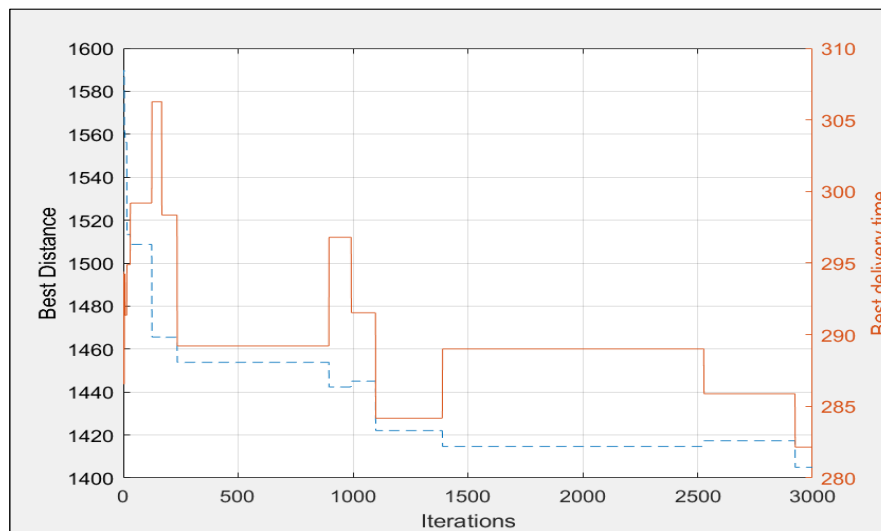


(b) Drones and trucks

Figure 74: 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)

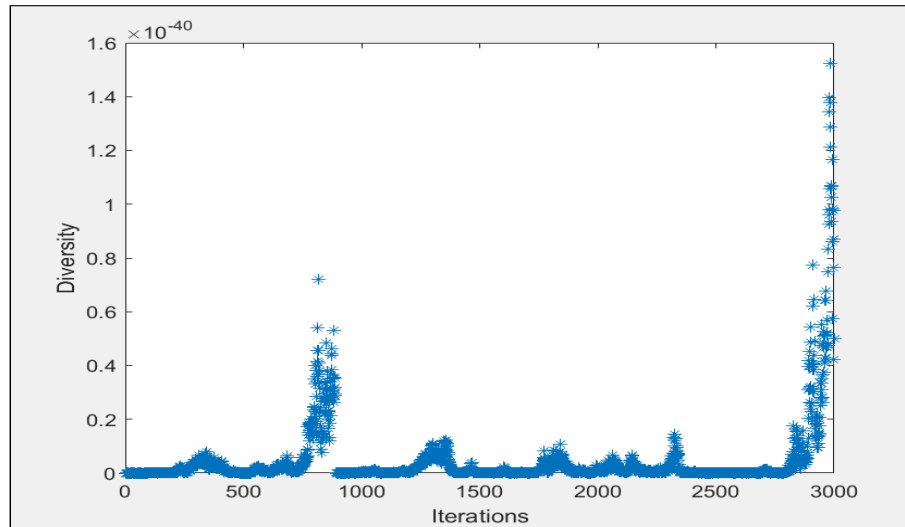


(a) Trucks-only

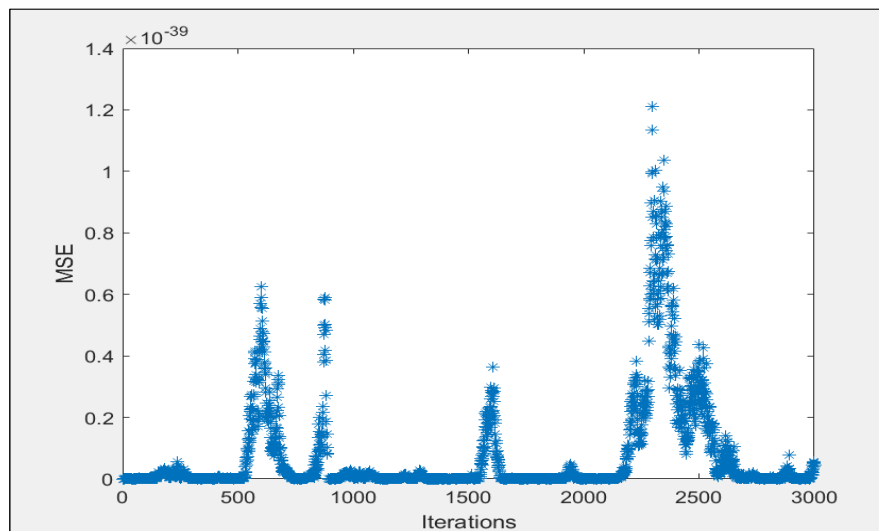


(b) Drones and trucks

Figure 75: Best solutions for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)

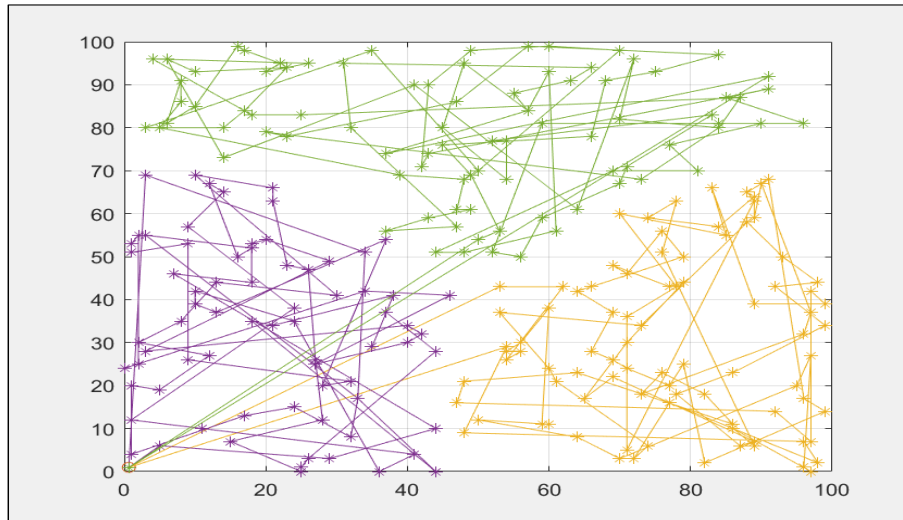


(a) Trucks-only

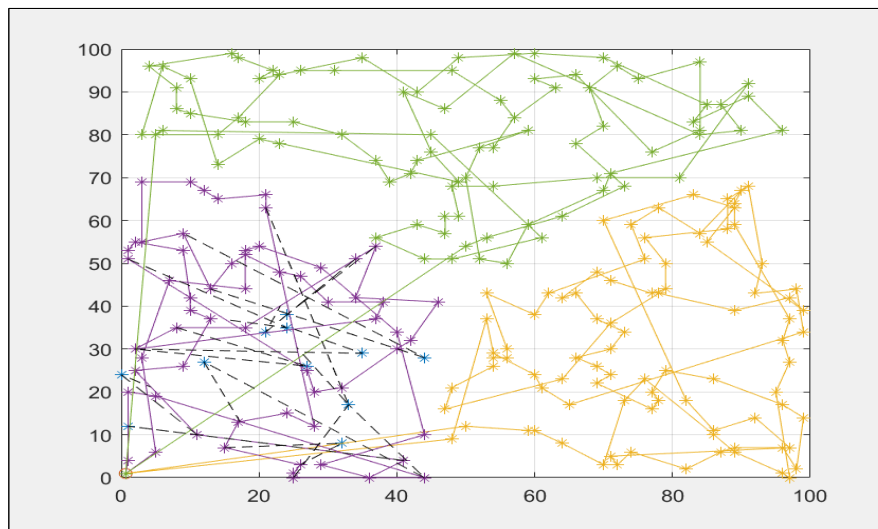


(b) Drones and trucks

Figure 76: Solutions search diversity for 100 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)

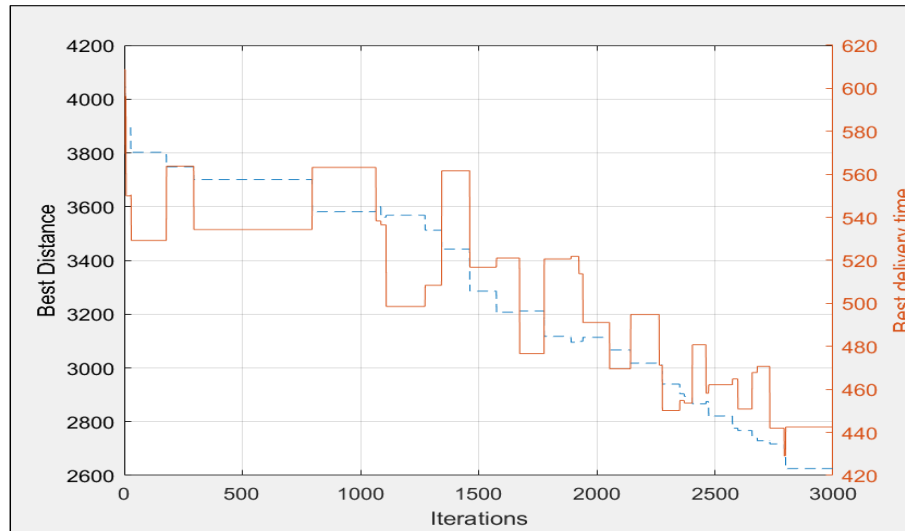


(a) Trucks-only

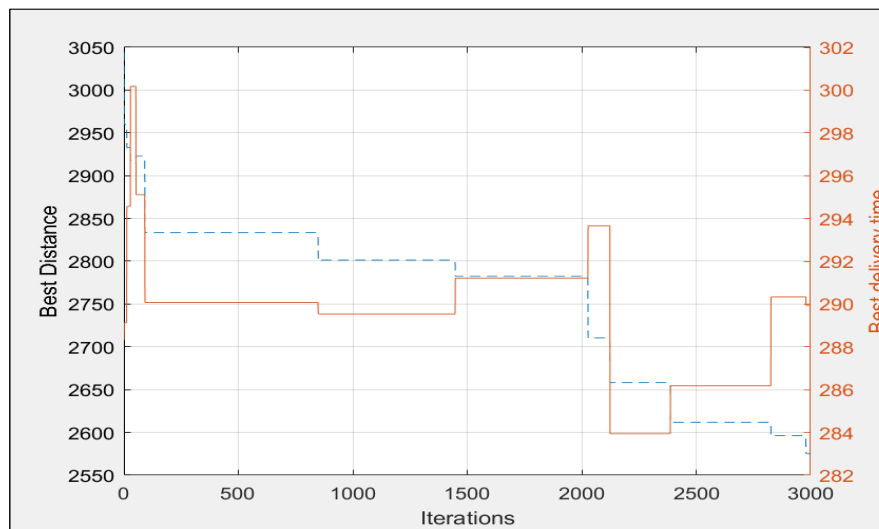


(b) Drones and trucks

Figure 77: 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-1-n250)

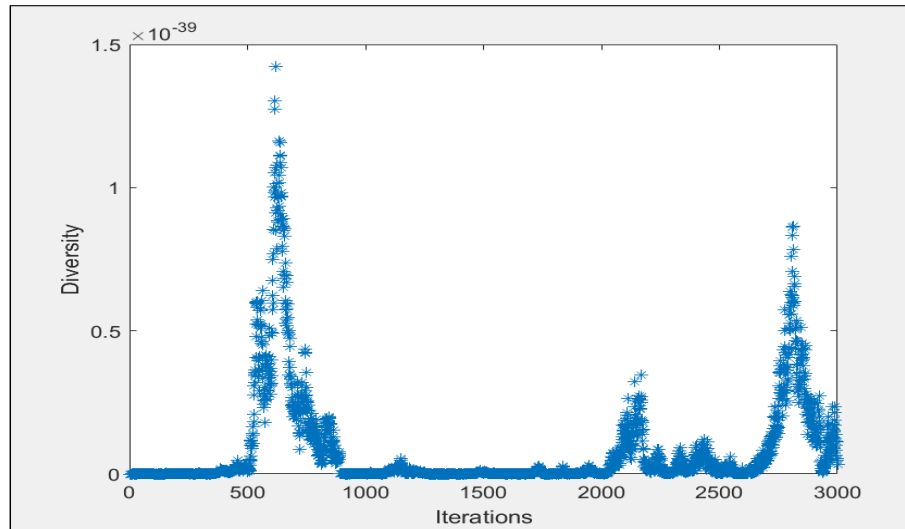


(a) Trucks-only

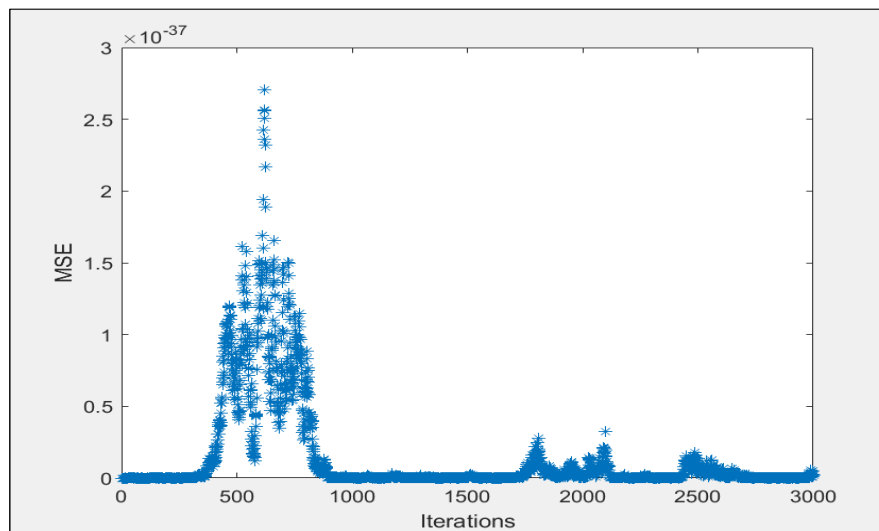


(b) Drones and trucks

Figure 78: Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-1-n250)

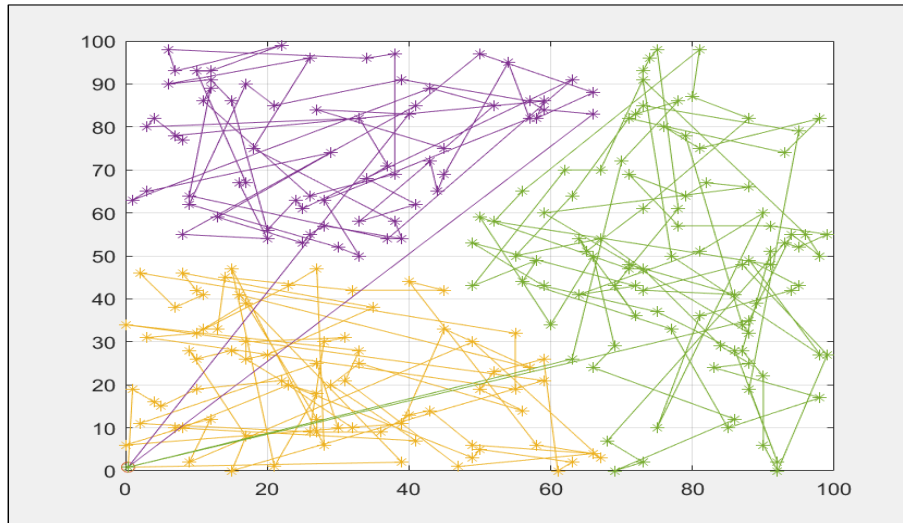


(a) Trucks-only

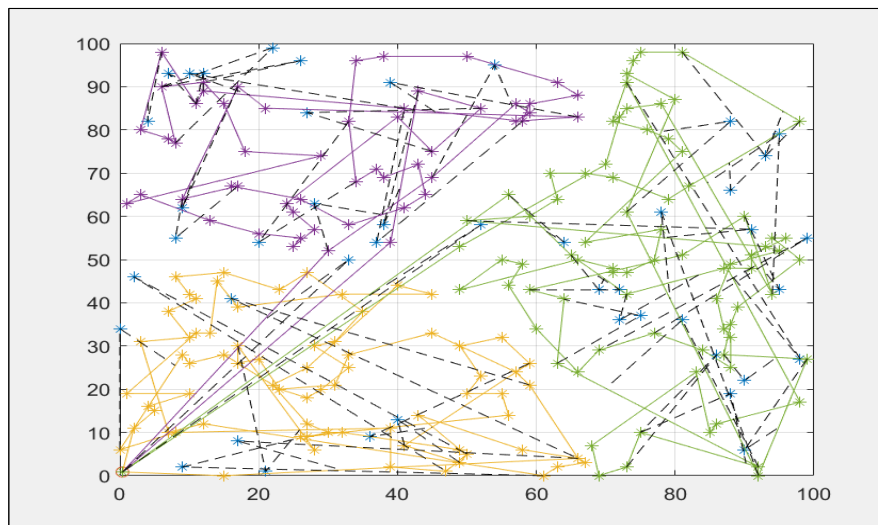


(b) Drones and trucks

Figure 79: Solutions search diversity for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-1-n250)

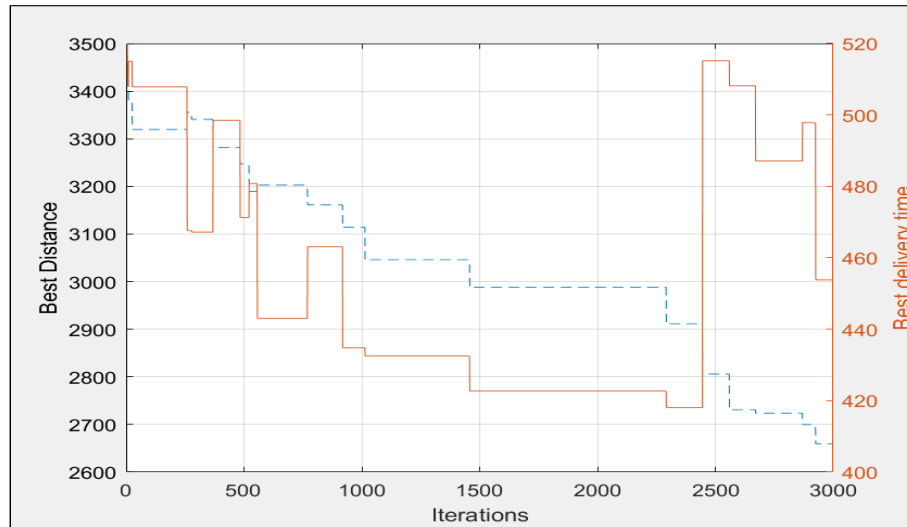


(a) Trucks-only

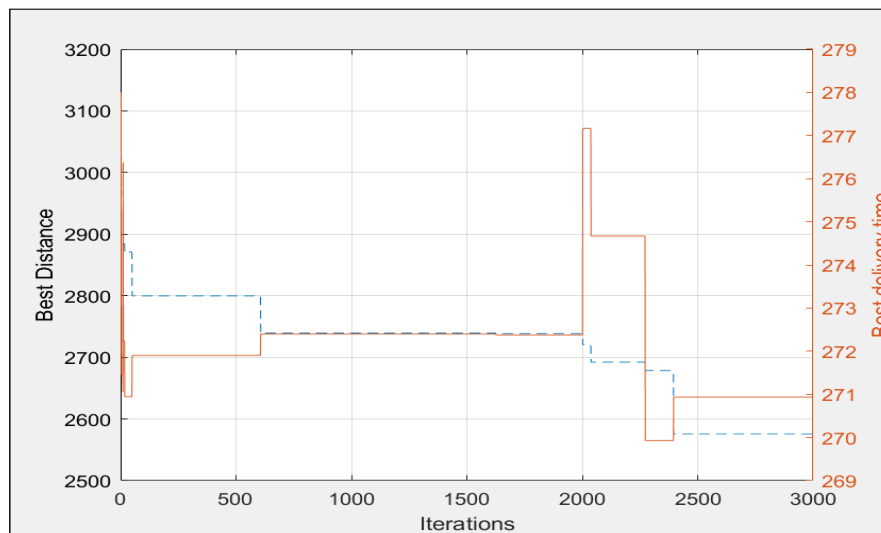


(b) Drones and trucks

Figure 80: 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)

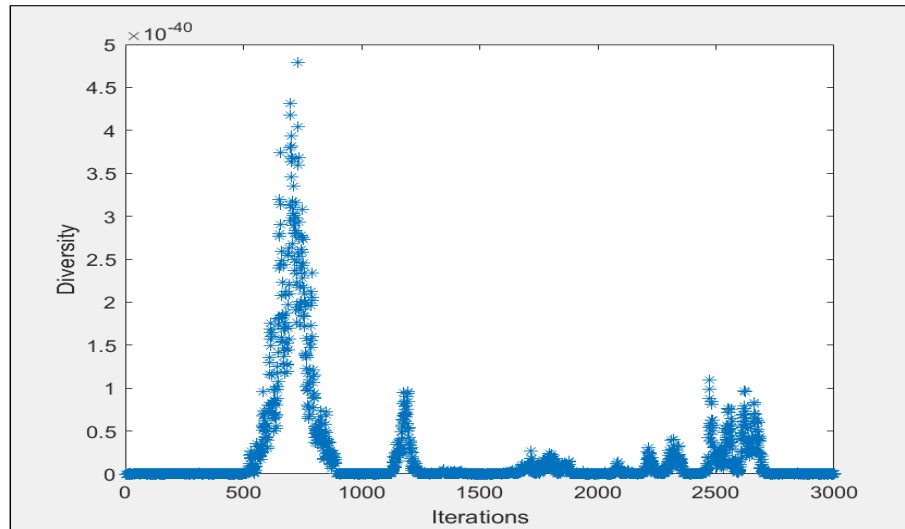


(a) Trucks-only

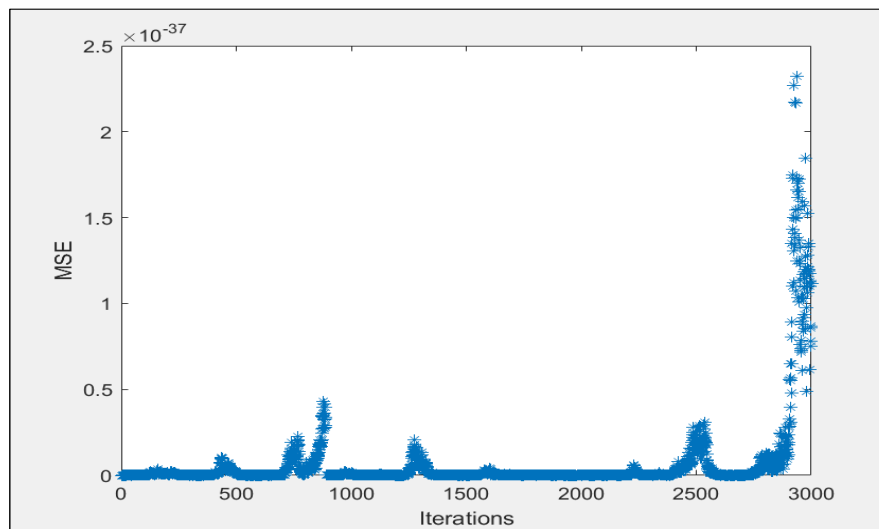


(b) Drones and trucks

Figure 81: Best solutions for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-93-n100)

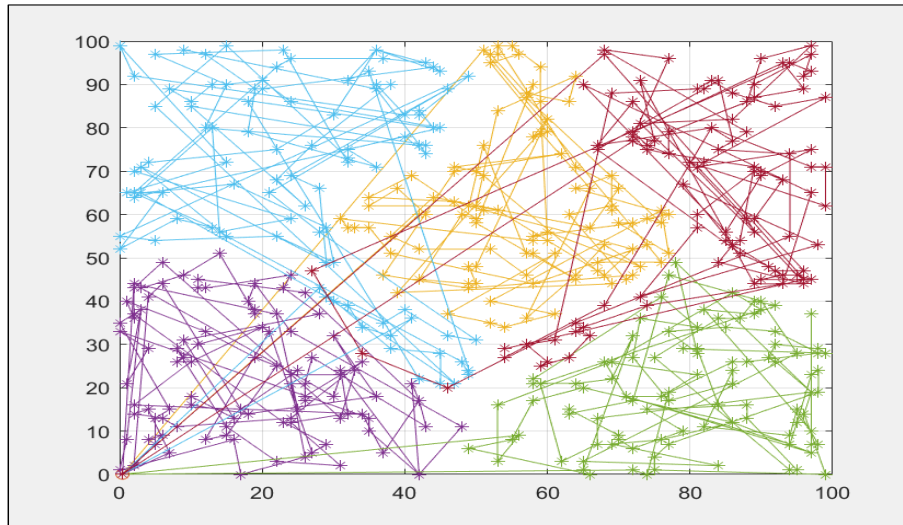


(a) Trucks-only

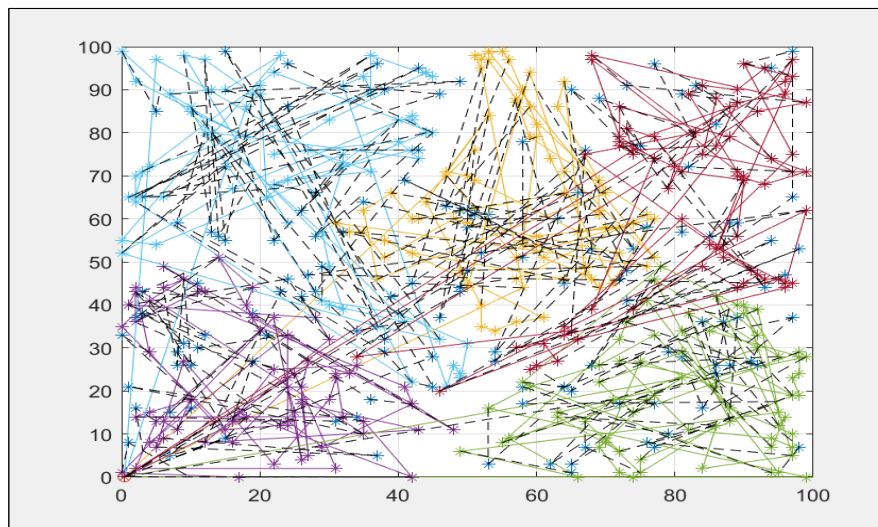


(b) Drones and trucks

Figure 82: Solutions search diversity for 250 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-2-n250)

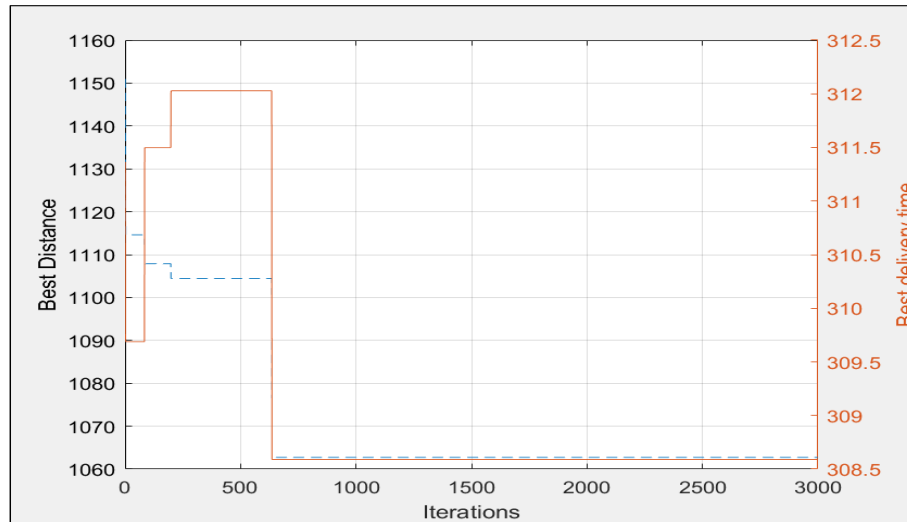


(a) Trucks-only

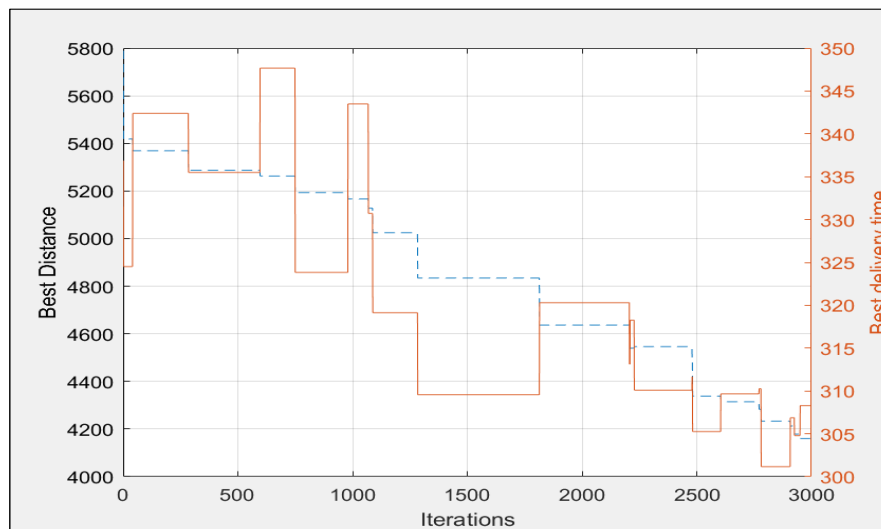


(b) Drones and trucks

Figure 83: 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-5-n500)

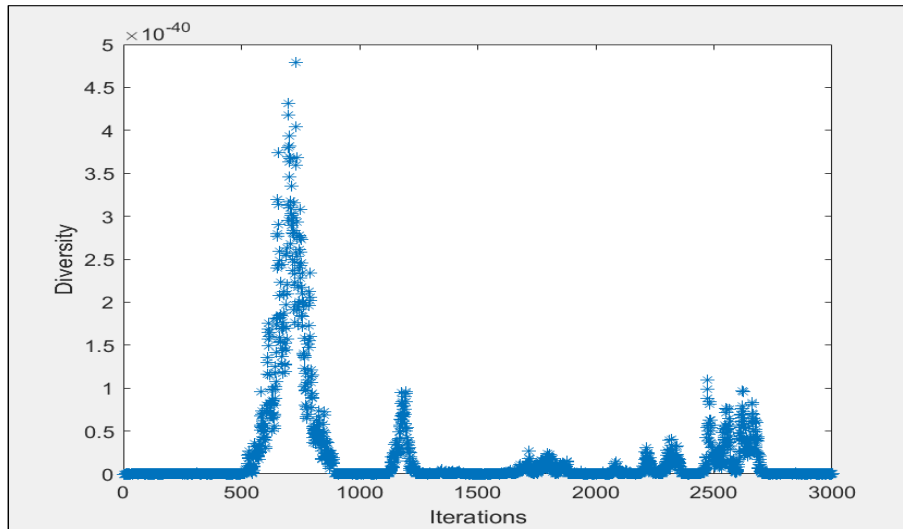


(a) Trucks-only

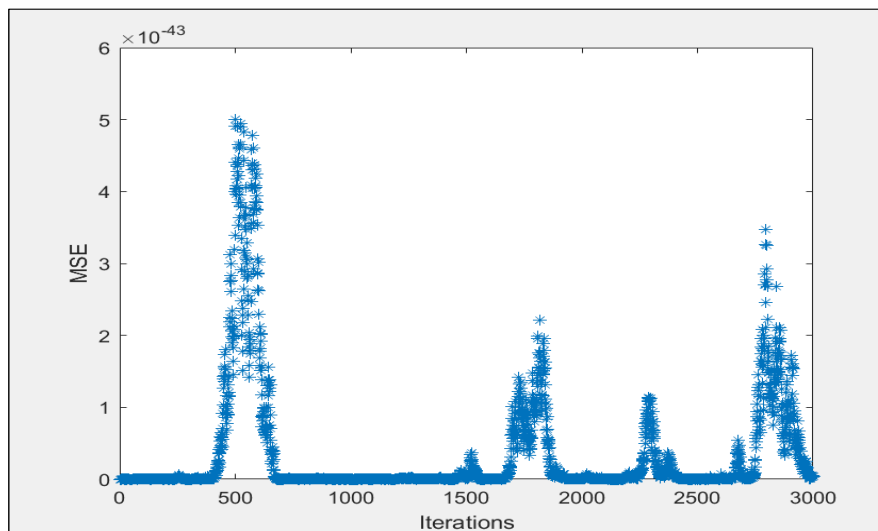


(b) Drones and trucks

Figure 84: Best solutions for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-5-n500)



(a) Trucks-only



(b) Drones and trucks

Figure 85: Solutions search diversity for 500 customers receiving deliveries from either a truck-only system or a multiple drone-truck delivery system, all with time windows (Uniform-5-n500)