

ADAPTIVE LARGE NEIGHBOURHOOD SEARCH HEURISTIC ON VEHICLE
ROUTING PROBLEM WITH DRONES AND TIME WINDOWS

by

Arifcan Yigit

B.S., Industrial Engineering, Boğaziçi University, 2016

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Industrial Engineering
Boğaziçi University

2020

ACKNOWLEDGEMENTS

First of all, I would like to thank Prof. Necati Aras for his support, patience and guidance throughout my entire graduate study. This work would not be completed without his advices and suggestions. Secondly, I would like to express my gratitude to Ümit Bilge and Sibel Salman for taking part in my thesis jury and providing valuable comments.

I would like to Yücel Gürel for answering my questions on every little detail of everything with his vast knowledge and enthusiasm.

I will never be able to find correct words to express my gratitude to my family. I would like to thank them for trusting and encouraging me at times when I feel doubtful about my decisions.

I also thank TUBITAK for their financial support during my graduate study.

ABSTRACT

ADAPTIVE LARGE NEIGHBOURHOOD SEARCH HEURISTIC ON VEHICLE ROUTING PROBLEM WITH DRONES AND TIME WINDOWS

Last-mile deliveries are the most costly and time-consuming part of the supply chain. Advancements in drone technology resulted in cheaper and faster drones capable of parcel delivery. However, their range is quite limited for drone-only delivery. Using synchronized drones and trucks could lower costs and delivery times by combining their superior features. This thesis addresses Vehicle Routing Problem with Drones and Time Windows (VRPDTW). The problem is formulated with waiting time restrictions on drones and cost minimization objective. Due to NP-hard nature of the problem, exact solution methods are inefficient even for small instances. Therefore, we develop an Adaptive Large Neighbourhood Search (ALNS) heuristic for finding near optimal solutions. Numerical experiments are conducted to measure the effectiveness of the heuristic using small and medium-sized instances generated randomly. Results show that the proposed heuristic is able to find optimal or near optimal solutions in small instances.

ÖZET

ZAMAN KISITLI DRONELU ARAÇ ROTALAMA PROBLEMİ İÇİN UYARLANABİLİR BÜYÜK KOMŞULUK ARAMASI SEZGİSELİ

Son kullanıcıya ürün ulaştırılması tedarik zincirinin en maliyetli ve en çok zaman alan kısmıdır. Drone teknolojisindeki gelişmeler, kargo taşımacılığına uygun ucuz ve hızlı droneleri mümkün kıldı. Ancak, dronelerin uçuş menzillerinin oldukça kısıtlı olması sadece dronelerle taşımacılığa engel oluyor. Senkronize drone ve araçların kullanılması araçların üstün taraflarını birleştirerek daha düşük maliyet ve teslimat süresi sağlayabilir. Bu tezde Zaman Kısıtlı Droneli Araç Rotalama Problemi üzerinde çalışılmıştır. Problem bekleme süresi kısıtları ve maliyet düşürme hedefi ile kurgulanmıştır. NP-Hard olmasından dolayı tam çözüm yöntemleri küçük örnekler için bile verimsiz kalmaktadır. Hızlı şekilde en iyiye yakın sonuç bulmak için Uyarlanabilir Büyük Komşuluk Arama sezgiseli geliştirdik. Küçük ve orta ölçekli veri setleri oluşturulup sezgiselin kullanışlılığını ölçmek için sayısal deneyler yapıldı. Sonuçlar önerilen sezgiselin küçük setlerde en iyi ya da en iyiye yakın çözümler bulabildiğini gösterdi.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZET	v
LIST OF FIGURES	vii
LIST OF TABLES	ix
LIST OF SYMBOLS	x
LIST OF ACRONYMS/ABBREVIATIONS	xii
1. INTRODUCTION	1
2. LITERATURE REVIEW	3
2.1. Travelling Salesman Problem with Drones	3
2.2. Vehicle Routing Problem with Drones	8
3. MATHEMATICAL MODEL	11
3.1. Assumptions and Notation	11
3.2. Mathematical Formulation	14
4. SOLUTION METHOD BASED ON AN ADAPTIVE LARGE NEIGHBOR- HOOD SEARCH	19
4.1. Adaptive Large Neighborhood Search	19
4.1.1. Initial Solution Generation	21
4.1.2. Destroy Moves	21
4.1.3. Repair Moves	24
4.1.4. The Design of the ALNS	26
5. COMPUTATIONAL ANALYSIS	31
5.1. Instance Generation	31
5.2. Parameter Tuning	31
5.3. Experimental Results	35
6. CONCLUSION	45
REFERENCES	46
APPENDIX A: ALNS RESULTS FOR RELAXED SET	52

LIST OF FIGURES

Figure 1.1.	Feasible VRP (a) and VRPD (b) solutions comparison.	2
Figure 4.1.	Pseudocode for initial solution generation	21
Figure 4.2.	Pseudocode for Regret-2 Move.	25
Figure 4.3.	Pseudocode of ALNS Heuristic.	27
Figure 5.1.	Average % deviation from best objective value found by CPLEX for various r values.	32
Figure 5.2.	Average % deviation from best objective value found by CPLEX for various d_f values.	33
Figure 5.3.	Average % deviation from best objective value found by CPLEX for various ρ values.	34
Figure 5.4.	Average % deviation from best objective value found by CPLEX for various η values.	34
Figure 5.5.	Two vehicles VRPTW solution for 40-4 instance by ALNS phase one.	40
Figure 5.6.	Two vehicles VRPDTW solution for 40-4 instance by ALNS phase two.	40
Figure 5.7.	Parameter sensitivity using medium-sized instances.	42

Figure 5.8. The total cost per drone sortie comparison on 12-customers instances. 44

LIST OF TABLES

Table 3.1.	Index Sets	13
Table 3.2.	Indices in the model of VRPDTW	13
Table 3.3.	Parameters in the model of VRPDTW	13
Table 3.4.	Decision variables in the model of VRPDTW	14
Table 5.1.	Results obtained by CPLEX for small instances.	35
Table 5.2.	ALNS results for small instances with one vehicle.	36
Table 5.3.	ALNS results for small instances with two vehicles.	37
Table 5.4.	CPLEX results for small instances with relaxed time windows. . .	38
Table 5.5.	CPLEX results for medium-sized instances solved as VRPTW. . .	39
Table 5.6.	ALNS results for medium-sized instances.	41
Table 5.7.	ALNS with long phase one results for medium-sized instances. . . .	43
Table A.1.	ALNS results for small instances with relaxed time windows and with one vehicle.	52
Table A.2.	ALNS results for small instances with relaxed time windows and with two vehicles.	53

LIST OF SYMBOLS

c_{ij}	Vehicle travelling cost between nodes i and j
c'_{ijk}	Drone travelling cost on (i, j, k) tuple
C	Set of customers
C_R	Cooling ratio parameter for Simulated Annealing
d	Destruction degree parameter showing how many nodes on a solution will be destroyed and reinserted
d_f	Ratio for adjusting destruction degree parameter considering the number of customers
e	Drone endurance
et_j	Earliest service start time at customer j
i	Index for launching nodes
(i, j, k)	Index for feasible tuples for drone flight
j	Index for customers
k	Index for recovery nodes
lt_j	Latest service start time at customer j
$maxIter$	Maximum iteration number algorithm could do without improving incumbent solution
N	Set of all nodes
N_l	Set of feasible launching nodes, customers and starting depot
N_r	Set of feasible recovery nodes, customers and ending depot
n	The number of customers
$nonIter$	Number of iterations since the last incumbent solution found
p_{ij}^v	Binary variable showing if truck v visits node i before node j
P	Set of tuples feasible for drone flight
q_j	Demand of customer j
Q	Vehicle capacity
r	Move repetition parameter
s_j	Service starting time at customer j

$timeLimit$	Maximum time limit for termination of algorithm
T	Temperature parameter of simulated annealing
T_{len}	Number of turns a certain temperature will be used
T_{max}	Maximum duration a drone can wait at a launching node after truck leaves <i>or</i> at service node after service completion
u_j^v	Visiting sequence of node j with vehicle v
v	Index for vehicles
V	Set of vehicles
x_{ij}^v	Binary variable showing if vehicle v traversed from i to j
y_{ijk}^v	Binary variable showing if drone belonging to vehicle v launched from i , served at j and coupled with vehicle at k
δ	Noise percentage
η	Noise percentage step size
ρ	Reaction factor for weight adjustments
τ_{ij}	Traversal time between nodes i and j by vehicle
τ'_{ij}	Traversal time between nodes i and j by drone

LIST OF ACRONYMS/ABBREVIATIONS

ALNS	Adaptive Large Neighbourhood Search
CP	Constraint Programming
DASP	Drone Assignment and Scheduling Problem
DSP	Drone Scheduling Problem
FSTSP	Flying-sidekick Travelling Salesman Problem
GRASP	Greedy Randomized Adaptive Search Procedure
HGA	Hybrid Genetic Algorithm
HGVNS	Hybrid General Variable Neighbourhood Search
HVRPD	Hybrid Vehicle-Drone Routing Problem
IP	Integer Programming
LNS	Large Neighbourhood Search
MIP	Mixed Integer Programming
PDSTSP	Parallel Drone Scheduling Travelling Salesman Problem
PDPTW	Pickup and Delivery Problem with Time Windows
SA	Simulated Annealing
TA	Threshold Acceptance
TSP	Travelling Salesman Problem
TSP-D	Travelling Salesman Problem with Drone
TSP-DS	Travelling Salesman Problem with a Drone Station
VNS	Variable Neighbourhood Search
VRP	Vehicle Routing Problem
VRPB	Vehicle Routing Problem with Backhauls
VRPD	Vehicle Routing Problem with Drones
VRPDTW	Vehicle Routing Problem with Drones and Time Windows
VRPTW	Vehicle Routing Problem with Time Windows

1. INTRODUCTION

The volume of retail e-commerce sales worldwide has increased almost 50% from 2017 to 2019 and is expected to increase steadily [1]. The high growth of sales causes significant market competition and leads companies to focus on last-mile delivery since transportation of goods from the warehouse to the customer is the most costly and time-consuming part of supply chain management.

In 2013, Amazon CEO Jeff Bezos announced plans about 30-minute delivery using drones [2]. This service announcement leads other companies and researchers to consider opportunities of drone delivery. Although Amazon has not started the drone service yet, certain companies like UPS and Alphabet's Wing have started their drone operations [3].

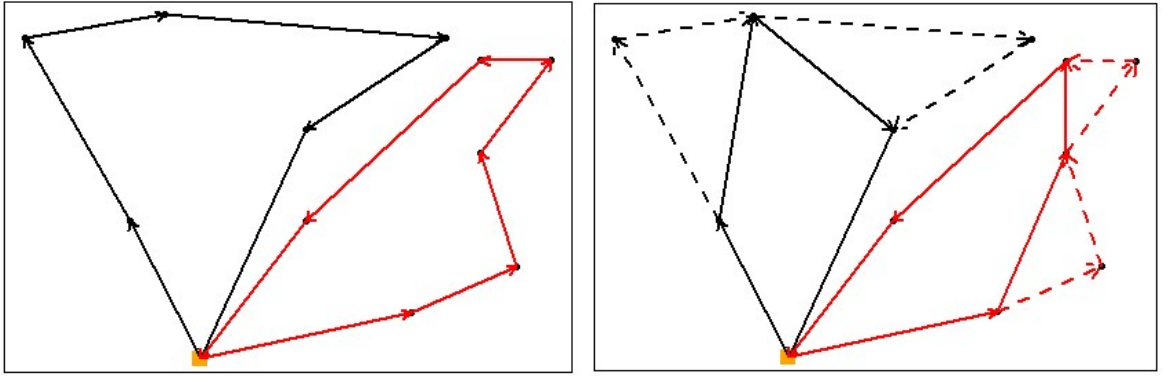
Drones are cheaper and faster than classical vehicles. However, their weight-carrying capacity is quite limited compared to trucks. Jeff Bezos stated about 86% of all parcels was under five pounds, the maximum weight a drone can carry [2], thus drones could handle most packages. However, their flight range is also restricted due to battery limitations. Returning to the depot after every delivery causes inefficiency and only allows small-range restricted service to near-depot areas.

Using trucks as a moving base for carrying drones to save battery and get closer to customers could provide most benefit from both vehicles, as first proposed by Murray and Chu [4]. They introduced an extension of TSP with a coordinated drone which is carried by truck and then launched with a parcel loaded from truck. There have been numerous variants of drone inclusion in last mile delivery afterwards such as drone-only delivery problem, TSP with drone (TSP-D) and VRP with drones (VRPD).

The comparison of regular VRP to VRPD can be seen in Figure 1.1. The first panel shows a feasible solution for a VRP problem with two vehicles. The circles denote

customer nodes and the square denotes depot. In the second panel, dashed lines denote drone flights while solid lines show truck routes.

In this thesis, we consider the VRPD with Time Windows (VRPDTW) and propose a mathematical programming formulation based on Murray and Chu model [4]. We allow the drone to wait on ground subject to certain time limits and try to minimize the total cost. VRPDTW is an extension of the VRPTW which is known to be NP-Hard. Due to complexity of the problem, exact solution methods are inefficient even for small instances. Therefore, we present an Adaptive Large Neighbourhood Search (ALNS) metaheuristic for the solution.



(a) VRP solution

(b) VRPD solution

Figure 1.1. Feasible VRP (a) and VRPD (b) solutions comparison.

The remainder of this thesis is organized as follows. In Chapter 2, we provide a literature review of drone delivery related to TSP-D and VRPD. In Chapter 3, the mathematical model is presented and explained in detail. We present the ALNS related literature and describe our ALNS algorithm for VRPDTW in Chapter 4. Numerical experiments are covered in Chapter 5. In Chapter 6, we provide concluding remarks and possible extensions.

2. LITERATURE REVIEW

Drone usage in parcel delivery is a recently vivid area of research. Most studies consider unmanned aerial vehicle (UAV) as truck-based agents but these models can be extended to robots also as in Boysen *et al.* [5].

The first drone-truck combined delivery is presented by Murray and Chu [4]. Their original model consists of one truck and one drone providing synchronized service. The problem is called TSP-D and then extended to multi vehicles, VRP with drones. Formulations vary mainly in the number of drones per truck, number of depots, the objective of cost or makespan minimization and vehicle synchronization, combined or separate deliveries, allowing drones to be collected by trucks at the same node they were launched. We will present a literature review that focuses on drone-truck cooperative parcel delivery.

2.1. Travelling Salesman Problem with Drones

Murray and Chu [4] introduce the truck-drone combined delivery model. They formulate two Mixed Integer Programming (MIP) models: the Flying-sidekick TSP (FSTSP) and the Parallel Drone Scheduling TSP (PDSTSP) with the objective of minimizing vehicle return time to depot. In FSTSP, drone uses truck as a moving-base until it comes close enough to a service node for a safe flight. Then the drone is loaded with a parcel from truck and takes off, completes its service, and returns to the truck at a recovery node which is different from the launching node. The drone follows Euclidean distances and the truck follows Manhattan distances. In PDSTSP, a truck and a set of drones are considered as heterogeneous and separate vehicles. In this setting, the problem resembles VRP as all vehicles start from depot and return to depot on their own. They introduce heuristics for both problems by swapping customers between the truck and drone, calculating the objective improvements, and then performing the best swapping. Ponza [6] also works on the FSTSP model with minor changes in the

formulation. He implements a Simulated Annealing (SA) metaheuristic and shows that SA provides good quality solutions.

Agatz *et al.* [7] reformulate the FSTSP with Integer Programming (IP) and with cost minimization objective. Another difference is including cyclic drone services. In other words, the drone can return to the node where it was launched. In this formulation, drone and truck have to follow the same road network unlike the FSTSP. They propose greedy and exact partitioning algorithms and also iterative improvement procedure. First, a TSP solution was found for all nodes, then drone service scheduling started. Their work also estimated approximate cost improvements using drone-truck combined delivery instead of truck-only delivery.

Ha *et al.* [8] consider the FSTSP with an objective of cost minimization. Their objective includes waiting costs for both vehicles as well as travel costs. They introduce a Greedy Randomized Adaptive Search Procedure (GRASP) and Local Search (LS) method which they call TSP-LS. Both heuristics start with a TSP solution, drone deliveries are added thereafter. In their later work, Ha *et al.* [9] use a hybrid genetic algorithm (HGA) for the same model. Starting from a TSP solution, deliveries are divided between truck and drone. The feasibility of the new solution is evaluated and improved using LS methods. A penalization cost term is added on objective function to account for drone endurance violations in resulting solutions.

Freitas and Penna [10] work on the original FSTSP formulation and propose a General Variable Neighbourhood Search (GVNS) heuristic. They find their initial solution by solving the problem as a TSP and then the changing optimal solution by adding drone deliveries considering improvement from customer swapping between vehicle types like Murray and Chu [4]. They implement an LS procedure using seven different moves.

Yurek and Ozmutlu [11] decompose the TSP-D into two parts. The maximum number of drone service is limited in TSP-D due to one-launch and/or one-retrieval

on a node. They utilize this fact and explore all the truck routes for reduced sets of customers excluding maximum amount of drone customers. After finding the best truck route, a mathematical model is solved for drone deliveries. Also, a heuristic is proposed to decrease the workload by finding a sub-optimal truck route.

Ferrandez *et al.* [12] study a TSP-D model where a truck can carry multiple drones. Also, they allow drones to be launched and retrieved at the same truck stop which is not necessarily a customer node like in FSTSP. They initially cluster customers using K-means to find drone-served nodes. Then by using centroids of clusters and truck-served nodes, a genetic algorithm is implemented to find the truck route. Formulations for the number of optimal clusters and total delivery time are derived from simulation results.

Chang and Lee [13] study a similar model as Ferrandez *et al.* [12]. In their model, the truck does not perform any delivery and total delivery time is minimized. Customers are clustered using K-means algorithm initially. The optimal TSP route which traverses cluster centroid is found. Lastly, a non-linear model is solved to find whether changes in cluster centroid locations can improve the solution.

Tu *et al.* [14] work on TSP with multiple drones and a cost minimization objective. They restrict the waiting time of truck and drone at recovery nodes solely for retrieval operations. Both GRASP and ALNS algorithms are implemented and results show that ALNS performs better.

Marinelli *et al.* [15] mention that allowing drone retrievals only at customer nodes causes inefficiency for drones. They extend TSP-D formulation to allow drones to be launched and recovered along the truck's path to customers. The drone still needs to be loaded and retrieved manually, therefore truck has to stop for these operations. They devise an iterative heuristic which first solves TSP. Initially focusing on customer nodes for drone launch/retrieval, they improve their findings considering en-route operations by adding en-route launch and retrieval nodes.

Luo *et al.* [16] deal with a two-echelon routing problem including a coordinated truck and drone couple. In the model, certain target nodes are only reachable by drone. Predetermined stop points are introduced. Drone launch/retrieval could take place in customer nodes or stop points, which can be seen as a restricted case of en-route problem [15]. An IP formulation is presented, and two different heuristics are examined.

Dorling *et al.* [17] consider a model of multiple drones serving from a single depot. They suggest considering energy consumption of drones during flights, which is affected by parcel weight and delivery duration. The cost function is modified to include drone conditions, fully or partially recharged, and energy consumption of drones.

Jeong *et al.* [18] also focus on the energy consumption of the drone. Their model included time-dependent and completely no-fly zones, where the drone should go around if the zone is on its path during the restricted time. An evolutionary based algorithm with several truck-related moves is developed.

Poikonen and Golden [19] address TSP-D with delivery only by multi-visit drones carried and supplied by a truck. Drone energy consumption is also included in the model. An IP was presented with delivery time minimization objective. The optimal TSP solution is used to observe visiting time and sequence to customers. Each drone is assigned to multiple customers using TSP solution and the lowest delivery time solution is found by the proposed heuristic.

Boysen *et al.* [20] introduce the drone scheduling problem (DSP). In DSP, truck route is already determined for a subset of all customers and focus is on drone service planning for remaining customers. They consider multiple drones and allow that a drone can be retrieved at the same node it is launched. Six different variants of the problem are introduced and complexity discussions for each case are provided.

Kim and Moon [21] propose TSP with a Drone Station (TSP-DS) where drones are kept at a station different from the depot. In the model, trucks carry parcels to both customers and this station where they are distributed by drones. They use decomposition techniques on the problem to minimize the last return time and the number of drones at the station.

Boysen *et al.* [5] construct a similar model to TSP-DS with autonomous truck-based robots instead of drones. There are multiple robot stations, and trucks work as a supply agent solely to robots or robot stations. Also, time windows for customer services are considered. Each customer is assigned a weight and weighted late deliveries are minimized.

The minimization of makespan and operational costs are the main objectives in researches. However, Wang *et al.* [22] formulated the TSP-D as bi-objective and studied a genetic algorithm. Wen *et al.* [23] also used multi-objective for drone-only delivery with multi-visit drones and examined an evolutionary algorithm.

Dell'amico *et al.* [24] study certain modifications on FSTSP for a more efficient and robust formulation. Drone is allowed to wait on ground for energy conservation. Objective function was selected to minimize total duration spent by drone launching/recovery and spent by truck traversing/waiting. They also propose a new formulation where drone sorties expressed with two variables instead of one tuple. Experiments show their model is more efficient than FSTSP.

Heuristics are not the only proposed solution methods. Bouman *et al.* [25] develop a dynamic programming approach. First, truck sub-routes having shortest distances between any truck-related nodes are found. Then, drone sorties assigned to promising ones and shortest truck route traverse through these sub-routes are found. Poikonen *et al.* [26] apply branch and bound algorithm with several heuristics for scalability. Carlsson and Song [27] use a continuous approximation for the TSP-D to reduce the complexity of the problem.

Kitjacharoenchai *et al.* [28] consider multiple TSP (mTSP) with drones where drone endurance is discarded. They made it possible for a drone returning to a truck different from the one where it is launched. Customers are first clustered with K-means, then an mTSP route is found by Nearest Neighbourhood algorithm. Later, drone services are added to improve the solution.

Murray and Raj [29] extend the FSTSP formulation to accommodate multiple trucks with multiple drones. A three-phase heuristic is implemented where they first find an mTSP solution for a subset of customers, then assign drones to remaining customers and lastly plan scheduling for all assignments.

2.2. Vehicle Routing Problem with Drones

Wang *et al.* [30] introduce VRPD with a makespan minimization objective. They find theoretical bounds on objective function improvements compared to VRP by examining the number of vehicles, the number of drones per vehicle and the ratio of drone velocity to truck velocity. Poikonen *et al.* [31] extend this research by including drone battery constraints. They also give theoretical bounds for cost minimization objective for the cases in the former research.

Campbell *et al.* [32] deal with deriving expected travel distance and cost for both truck and drone with continuous approximation. Using parametric experimentation, they determine the possible gain that can be obtained by truck-drone combined delivery design and multiple drones per truck.

Wang and Sheu [33] extend TSP-DS model to the VRPD with multi-visit drones stationed at many drone-hubs. Vehicles start from the warehouse and reach the hubs. In their model, trucks can supply the hubs with parcels for drone delivery starting from the hubs or can carry drones from hubs to provide a safe flight range. They propose arc-based and path-based formulations under these assumptions. They apply a branch-and-price method for path-based formulation to find exact solutions.

Daknama and Kraus [34] introduce Vehicle Routing with Drones (VRD). Their model allows more than one drone on a truck and does not consider capacity constraints. Drones can be retrieved by other trucks, and they can even carry parcel between trucks. A graph model is considered, and the features of a solution are explored. Then, a nested local search algorithm is used to find a solution starting from the mTSP solution.

Karak and Abdelghany [35] introduce Hybrid Vehicle-Drone Routing Problem (HVRDP), where customers are only served by drones capable of multi-visit and pick up. In HVRDP, vehicles carry multiple drones and do not perform delivery operations. Drones can be launched and recovered from the same node. Three different heuristics are studied to minimize vehicle cost, drone cost and both. Improvement of both routes instead of sequential approaches is proved to be more effective.

Li *et al.* [36] consider heterogeneous delivery problem with trucks and drones. Several depots are supplied with trucks from one warehouse, and then used for drone or truck delivery. They use continuum approximation model for estimating the distance travelled and costs incurred.

Schermer *et al.* [37] develop a matheuristic for VRPD with multiple drones. Similar to DSP of Boysen *et al.* [20], they model Drone Assignment and Scheduling Problem (DASP) for given truck routes with two distinct MIP formulations. After generating truck routes, they use a matheuristic to solve DASP. Their experiments show that the matheuristic generates good quality solutions.

Schermer *et al.* [38] study vehicles with en route drone operation capabilities for VRPD and give MIP formulation. Trucks can carry multi drones and could stop at any point to launch and retrieve drones. A tabu search and VNS combined heuristic is developed for the makespan minimization. Their experiments conclude drone usage is more effective when en-route operations are allowed.

Ulmer and Thomas [39] consider the same-day delivery problem with unsynchronized drones and trucks which can be seen as multiple truck extension of PDSTSP. While vehicles are on their routes, new customer requests are assumed to be coming. The acceptance of new requests and vehicle type to assign the accepted requests are decided with dynamic programming. They use parametric policy function approximation for solution.

Ham [40] extends the PDSTSP with more than one truck and depot, then use Constraint Programming (CP) for its solution. The model is extended by allowing pick-up and delivery, time windows and multi-visit drones. Drones can still carry one parcel at a time but can pick up and deliver packages from customers without returning to depot first. Ham proposes three formulations with various constraints and presented the performance quality of CP.

To our best knowledge, drone-truck synchronized VRPD with time windows for parcel delivery is only formulated by Pugliese and Guerriero [41]. They imposed a maximum waiting time limit for drone on any customer node, assign more than one drones to each vehicle and allow drones to make multi-visit trips. However, they are able to find exact solutions to small instances only.

3. MATHEMATICAL MODEL

In this chapter, a mixed-integer linear programming (MILP) model for VRPDTW is presented. The model is an extended version of FSTSP [4] and similar to VRPD model of Sacramento *et al.* [42], which is also based on FSTSP. Our model differs from the latter in having time-windows, allowing drones to wait on ground and lacking a tour completion time limit for trucks.

3.1. Assumptions and Notation

The problem considers last-mile parcel delivery by utilizing trucks and drones in coordination. A drone is faster and cheaper than a truck but has limited flight range and payload capacity. Integrated truck and drone services for combining their advantageous features produce better results. Vehicles are heterogeneous, but need synchronization, which makes the problem more difficult than VRPTW with heterogeneous trucks. Trucks are used as a moving base for drones. A truck carries parcels for drone delivery and the drone when it is not flying. When a truck arrives at a launching node, the parcel is loaded to the drone that is sent to related customer. Carried by truck, the drone saves energy and can safely perform a sortie to the customer and return to the truck within its flight range. The truck also performs deliveries on its own. If the drone is launched earlier, then the truck should go to recovery node to collect the drone.

The drone flight is denoted by an (i, j, k) tuple. i shows the launching node, j is the customer node to be served and k shows the node where the drone is recovered by the truck. Following the FSTSP notation, a dummy ending depot, node $n + 1$ where n denotes the number of customers, is created at the same location of the starting depot. Distinction of ending and starting depot allows us to define launching set $N_l = \{0\} \cup C$ and recovery set $N_r = C \cup \{n + 1\}$, which are required to determine the feasible tuples. The feasible tuple set P can be defined by i, j, k belonging to sets N_l, C ,

and N_r respectively. They are distinct from each other and require flight duration not exceeding drone endurance, e . P can be stated as $\{(i, j, k) : i \in N_l, j \in C, j \neq i, k \in N_r, k \neq i, k \neq j, \tau_{ij} + \tau_{jk} \leq e\}$. The set of all nodes is expressed as $N = \{0, n+1\} \cup C$.

The assumptions of the problem are as follows:

- All customers are served either by drone or truck and may only be visited once.
- The drone can only be launched from and recovered at customer nodes or depot.
- There could be at most one drone launch or one drone recovery for any given node. However, a node can be both retrieval and launch node. The truck waiting at a node for multi-visit drone services is prevented.
- Launch and recovery operations occur immediately. Considering these operations occur during small time intervals compared to drone flight or truck travel time, their omission is sensible, but they can also be included in drone travel times.
- Trucks and drones can leave and return to the depot at different times.
- Drones should return to the vehicle it was launched or the depot.
- A truck can wait at nodes for an unlimited time with or without drone. However, waiting of drone when it is separated from the truck is subject to a maximum time limit. A drone can wait at launch node after truck leaves and at service node after serving customer on ground for T_{max} amount of time. [24, 41] Drone also can wait at a recovery node but total waiting time for any sortie cannot exceed $2T_{max}$.
- All parcel weights are under drone capacity, so drone could serve any customer.
- Time window constraints are not soft and should always be satisfied.

The definitions of index sets, indices, parameters, and decision variables are given in Table 3.1–3.4 respectively.

Table 3.1. Index sets in the model of VRPDTW.

Sets	Definition
C	Set of customers
N_l	Set of nodes suitable for drone launching
N_r	Set of nodes suitable for drone recovery
P	Set of tuples feasible for drone flight
V	Set of vehicles

Table 3.2. Indices in the model of VRPDTW.

Indices	Definition
i	Index for launching nodes
(i, j, k)	Index for tuples which drone can traverse
j	Index for customers
k	Index for recovery nodes
v	Index for vehicle/drone combination

Table 3.3. Parameters in the model of VRPDTW.

Parameters	Definition
c_{ij}	Vehicle travelling cost between nodes i and j
c'_{ijk}	Drone travelling cost on (i, j, k) tuple
e	Drone endurance
et_j	Earliest time when a service can start at customer node j
lt_j	Latest time when a service can start at customer node j
Q	Vehicle capacity
q_j	Demand of customer j
T_{max}	Maximum drone waiting duration
τ_{ij}	Traversal time between nodes i and j by vehicle
τ'_{ij}	Traversal time between nodes i and j by drone

Table 3.4. Decision variables in the model of VRPDTW.

Variables	Definition
p_{ij}^v	Binary variable which is 1 if vehicle v visits node i before node j
s_k^v	Service start time at node k by vehicle or drone v
t_k^v	Arrival time at node k by vehicle v
$t_k'^v$	Arrival time at node k by drone v
u_k^v	Visiting sequence of node k by vehicle v
x_{ij}^v	Binary variable showing if vehicle v traverse from node i to node j
y_{ijk}^v	Binary variable which is 1 if (i, j, k) tuple is traversed by drone v

3.2. Mathematical Formulation

The objective function in Eq. (3.1) minimizes the total travel cost. The first part stands for truck travel cost and the second part for drone travel cost. Constraints in Eq. (3.2) ensure that all customer nodes are visited by either a truck or a drone. The trucks have to leave the customer by constraints in Eq. (3.3). All trucks should leave depot by constraints in Eq. (3.4) and return to the depot after serving at least one customer, by constraints in Eq. (3.5). Constraints in Eq. (3.6) enforce that the total load should not exceed the truck capacity.

Constraints in Eq. (3.7) guarantee that there could be only one drone launch at a given node. Similarly, constraints in Eq. (3.8) ensure that there could be only one drone recovery at a node. Constraints in Eq. (3.9) and Eq. (3.10) are for route synchronization. A truck has to visit nodes i and k by constraints in Eq. (3.9) if its drone has been assigned for service with launching node i and recovery node k . Drones can leave depot separately from trucks. Drone leaving is not enforced like truck but favored by its lower costs and higher speeds. If drone serves its first customer by launching from the depot, then constraint in Eq. (3.10) forces the truck to visit the recovery node.

$$\min \sum_{v \in V} \sum_{i \in N_l} \sum_{j \in N_r} c_{ij} x_{ij}^v + \sum_{v \in V} \sum_{i \in N_l} \sum_{j \in C} \sum_{k \in N_r} c'_{ijk} y_{ijk}^v \quad (3.1)$$

$$\text{s.t. } \sum_{v \in V} \sum_{\substack{i \in N_l \\ i \neq j}} x_{ij}^v + \sum_{v \in V} \sum_{\substack{i \in N_l \\ i \neq j}} \sum_{\substack{k \in N_r \\ (i,j,k) \in P}} y_{ijk}^v = 1 \quad j \in C \quad (3.2)$$

$$\sum_{\substack{i \in N_l \\ i \neq j}} x_{ij}^v = \sum_{\substack{k \in N_r \\ k \neq j}} x_{jk}^v \quad j \in C, v \in V \quad (3.3)$$

$$\sum_{j \in C} x_{0j}^v = 1 \quad v \in V \quad (3.4)$$

$$\sum_{j \in C} x_{j,n+1}^v = 1 \quad v \in V \quad (3.5)$$

$$\sum_{j \in C} q_j \left(\sum_{\substack{i \in N_l \\ i \neq j}} x_{ij}^v + \sum_{\substack{k \in N_r \\ (i,j,k) \in P}} y_{ijk}^v \right) \leq Q \quad v \in V \quad (3.6)$$

$$\sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{k \in N_r \\ (i,j,k) \in P}} y_{ijk}^v \leq 1 \quad i \in N_l, v \in V \quad (3.7)$$

$$\sum_{\substack{i \in N_l \\ i \neq k}} \sum_{\substack{j \in C \\ (i,j,k) \in P}} y_{ijk}^v \leq 1 \quad k \in N_r, v \in V \quad (3.8)$$

$$2y_{ijk}^v \leq \sum_{\substack{h \in N_l \\ h \neq i}} x_{hi}^v + \sum_{\substack{l \in C \\ l \neq k}} x_{lk}^v \quad i \in N_l, j \in \{C : j \neq i\}, k \in \{N_r : (i,j,k) \in P\}, v \in V \quad (3.9)$$

$$y_{0jk}^v \leq \sum_{\substack{h \in N_l \\ h \neq k}} x_{hk}^v \quad j \in C, k \in \{N_r : (0,j,k) \in P\}, v \in V \quad (3.10)$$

$$t_i'^v \geq t_i^v - M \left(1 - \sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{k \in N_r \\ (i,j,k) \in P}} y_{ijk}^v \right) \quad i \in C, v \in V \quad (3.11)$$

$$t_i'^v \leq t_i^v + M \left(1 - \sum_{\substack{j \in C \\ j \neq i}} \sum_{\substack{k \in N_r \\ (i,j,k) \in P}} y_{ijk}^v \right) \quad i \in C, v \in V \quad (3.12)$$

$$t_k'^v \geq t_k^v - M \left(1 - \sum_{\substack{i \in N_l \\ i \neq k}} \sum_{\substack{j \in C \\ (i,j,k) \in P}} y_{ijk}^v \right) \quad k \in \{N_r : k \neq n+1\} \quad (3.13)$$

$$t_k'^v \leq t_k^v + M \left(1 - \sum_{\substack{i \in N_l \\ i \neq k}} \sum_{\substack{j \in C \\ (i,j,k) \in P}} y_{ijk}^v \right) \quad k \in \{N_r : k \neq n+1\} \quad (3.14)$$

$$t_k^v \geq s_i^v + \tau_{ik} - M(1 - x_{ik}^v) \quad i \in N_l, k \in \{N_r : k \neq i\}, v \in V \quad (3.15)$$

$$t_j'^v \geq t_i'^v + \tau_{ij}' - M \left(1 - \sum_{\substack{k \in N_r \\ (i,j,k) \in P}} y_{ijk}^v \right) \quad j \in C, i \in \{N_r : i \neq j\}, v \in V \quad (3.16)$$

$$t_k^v \geq s_j^v + \tau_{jk}' - M(1 - \sum_{\substack{i \in N_l \\ (i,j,k) \in P}} y_{ijk}^v) \quad j \in C, k \in \{N_r : k \neq j\}, v \in V \quad (3.17)$$

$$M(y_{ijk}^v + x_{id}^v - 2) + (t_j^v - \tau_{ij}') - (t_d^v - \tau_{id}) \leq T_{max} \\ k \in N_r, j \in \{C : j \neq k\}, i \in \{N_l : (i, j, k) \in P\}, d \in N_l, v \in V \quad (3.18)$$

$$M(y_{ijk}^v - 1) + (t_k^v - \tau_{jk}') - s_j^v \leq T_{max} \\ k \in N_r, j \in \{C : j \neq k\}, i \in \{N_l : (i, j, k) \in P\}, v \in V \quad (3.19)$$

$$t_k^v - (t_j^v - \tau_{ij}') \leq e + 2T_{max} + M(1 - y_{ijk}^v) \\ k \in N_r, j \in \{C : j \neq k\}, i \in \{N_l : (i, j, k) \in P\}, v \in V \quad (3.20)$$

$$t_l^v \geq t_k^v - M(3 - \sum_{\substack{j \in C \\ (i,j,k) \in P \\ j \neq l}} y_{ijk}^v - \sum_{\substack{m \in C \\ m \neq i \\ m \neq k \\ m \neq l}} \sum_{\substack{n \in N_r \\ (l,m,n) \in P \\ n \neq i \\ n \neq k}} y_{lmn}^v - p_{il}^v) \\ i \in N_l, k \in \{N_r : k \neq i\}, l \in \{C : l \neq i, l \neq k\}, v \in V \quad (3.21)$$

$$s_k^v \geq t_k^v - M(1 - \sum_{i \in N_l} x_{ik}^v) \quad k \in N_r, v \in V \quad (3.22)$$

$$s_j^v \geq t_j^v - M(1 - \sum_{i \in N_l} \sum_{k \in N_r} y_{ijk}^v) \quad j \in C, v \in V \quad (3.23)$$

$$s_j^v \leq t_j^v + M(1 - \sum_{i \in N_l} \sum_{k \in N_r} y_{ijk}^v) \quad j \in C, v \in V \quad (3.24)$$

$$s_j^v \geq et_j \quad j \in C \quad (3.25)$$

$$s_j^v \leq lt_j \quad j \in C \quad (3.26)$$

$$u_j^v - u_k^v + 1 \leq (n+2)(1 - x_{ij}^v) \quad j \in C, k \in \{N_r : k \neq j\}, v \in V \quad (3.27)$$

$$u_k^v - u_i^v \geq 1 - (n+2)(1 - \sum_{\substack{j \in C \\ (i,j,k) \in P}} y_{ijk}^v) \quad i \in C, k \in \{N_r : k \neq i\}, v \in V \quad (3.28)$$

$$u_i^v - u_j^v \geq 1 - (n+2)p_{ij}^v \quad i \in C, j \in \{C : j \neq i\}, v \in V \quad (3.29)$$

$$u_i^v - u_j^v \leq -1 + (n+2)(1 - p_{ij}^v) \quad i \in C, j \in \{C : j \neq i\}, v \in V \quad (3.30)$$

$$p_{ij}^v + p_{ji}^v = 1 \quad i \in C, j \in \{C : j \neq i\}, v \in V \quad (3.31)$$

$$p_{0j}^v = 1 \quad j \in C, v \in V \quad (3.32)$$

$$t_0^v = 0 \quad v \in V \quad (3.33)$$

$$t_0^v = 0 \quad v \in V \quad (3.34)$$

$$s_0^v = 0 \quad v \in V \quad (3.35)$$

$$t_i^v \geq 0 \quad i \in N, v \in V \quad (3.36)$$

$$t_i'^v \geq 0 \quad i \in N, v \in V \quad (3.37)$$

$$s_i^v \geq 0 \quad i \in N, v \in V \quad (3.38)$$

$$x_{ik}^v \in \{0, 1\} \quad i \in N_l, k \in \{N_r : k \neq i\}, v \in V \quad (3.39)$$

$$y_{ijk}^v \in \{0, 1\} \quad i \in N_l, j \in \{C : j \neq i\}, k \in \{N_r : (i, j, k) \in P\}, v \in V \quad (3.40)$$

$$1 \leq u_k^v \leq n + 2 \quad k \in N_r, v \in V \quad (3.41)$$

$$p_{ij}^v \in \{0, 1\} \quad i \in N_l, j \in \{C : j \neq i\}, v \in V \quad (3.42)$$

Constraints in Eq. (3.11)–(3.14) ensure the time synchronization of truck and drone routes. Constraints in Eq. (3.11) and Eq. (3.12) state both vehicles arrive at launch node i at the same time. Constraints in Eq. (3.13) and Eq. (3.14) guarantee the same for recovery nodes.

Truck arrival time to a node should be greater than total of its service starting time, service duration and travel time from the previous node by constraints in Eq. (3.15). Drone arrival time to a customer should be greater than its arrival time to launch node and travel time by constraints in Eq. (3.16). Since launch node arrival time is equal to that of truck by constraints in Eq. (3.11)–(3.12), this equation allows drone launch without waiting for truck service starting in that node. Allowing drone launch without waiting for earliest service time for truck node gives trucks incentive for travelling to the next node in their route. Constraints in Eq. (3.17) ensure drone arrival to a recovery node take place after service completion and travel time between both nodes.

Constraint in Eq. (3.18) states drone can wait at a launching node on ground after truck departure for a maximum amount of time T_{max} . Second term states departure time of drone from launch node and third term states the same for truck. Constraint in Eq. (3.19) allows drone to wait at the customer node after completing its service.

P set includes drone endurance in its definition but only in terms of physical distance. Constraints in Eq. (3.20) guarantee drone service assignment is feasible considering drone battery, truck operations and drone waiting times. The drone waiting time limit on recovery node was selected as $2T_{max}$ for convenience since constraints in Eq. (3.18)–(3.20) impose $2T_{max}$ as total maximum waiting time limit for drone on ground. If drone waited on both launching and service nodes for T_{max} time unit, it cannot wait at recovery node since these waiting times are addressed in constraint in Eq. (3.20) by $2T_{max}$ term. If drone did not wait or partially wait prior to recovery node, then it can wait truck arrival for the remaining amount of total waiting time limit.

Constraints in Eq. (3.21) define the relation between different drone sorties and prevent simultaneous drone sorties. If a drone makes two flights following the paths (i, j, k) and (l, m, n) where truck visits node i before node l , then the drone arrival time to node l should be greater than arrival time to node k . In other words, a drone cannot be assigned to flight of (l, m, n) if it has not completed its earlier (i, j, k) service flight.

Constraints in Eq. (3.22)–(3.26) handle time windows. Constraints in Eq. (3.22) ensure that truck service start time is greater than its arrival time. A truck can reach a node earlier and wait for service time but drone cannot reach earlier and wait as a truck does. Constraints in Eq. (3.23)–(3.24) force that a drone starts service at the exact arrival time. Constraints in Eq. (3.25)–(3.26) guarantee that customers are served during their time windows.

Constraints in Eq. (3.27) are for sub-tour elimination. Constraints in Eq. (3.28) ensure that relative position of drone launch node comes before retrieval node. Constraints in Eq. (3.29)–(3.32) defines variable p_{ij} . Constraints in Eq. (3.33)–(3.35) assign initial depot times to time-related variables. Constraints in Eq. (3.36)–(3.42) define ranges for variables.

4. SOLUTION METHOD BASED ON AN ADAPTIVE LARGE NEIGHBORHOOD SEARCH

VRPDTW is an NP-hard problem and the number of feasible drone flight tuples significantly affects the difficulty of the mathematical model in terms of the number of decision variables and constraints. Therefore, finding an optimal solution to even small-sized instances is difficult. Hence, propose an ALNS algorithm to find near optimal solutions.

The ALNS method is a general framework to combine different kinds of selection and insertion techniques as designated by Pisinger and Ropke [43,44]. It utilizes various moves and allows exploring neighborhoods non-hierarchically, thus enables complex local search. At each iteration, a part of the current solution is changed via removing certain number of nodes and reinserting them back. The removal move and the insertion move are determined probabilistically which are updated through the algorithm to account for selected element's success in finding good solutions. The moves which select the nodes for removal are called as destroy moves and which reinsert them into the solution are called repair moves. We will discuss them in detail in Section 4.1.2 and 4.1.3, respectively.

4.1. Adaptive Large Neighborhood Search

Large neighborhood search (LNS) is a metaheuristic introduced first by Shaw [45] for VRP and VRPTW. LNS iterates through a removal and a subsequent re-insertion operator. In each step, a number of customers are removed from the solution and the number of customers are increased gradually to avoid local optima. These customers are selected by considering their relative distance and whether or not they are assigned to the same vehicle. For re-insertion, Shaw uses branching heuristics with limited discrepancy search for scalability. LNS is based on Greedy Acceptance (GA) and includes a degree of randomness in the removal procedure.

Schrimpf *et al.* [46] develop Ruin and Recreate (R&R) metaheuristic by combining three different ruin operators with the best insertion recreate operator. The authors also determine the number of customers to be removed randomly contrary to Shaw’s gradual increase. Their work compares Random Walk (RW), GA and Threshold Acceptance (TA). Experiments show that TA performs better.

Ropke and Pisinger [43] employ more than one move for both removal and repair operator for Vehicle Routing Problem with Backhauls (VRPB). Also, they introduce a monitoring and learning layer to search the solution space better. By keeping track of the frequency an operator improves the solution, the layer adjusts the probabilities with which each removal and repair operator is selected. They implement the so called Adaptive LNS (ALNS) algorithm to Pickup and Delivery Problem with Time Windows and compare to LNS [47]. Results show that ALNS outperforms LNS. They further improve the idea of selection probabilities by considering the quality of generated solutions [44, 48]. Every move operator is selected by using their past success and their selection probabilities are updated to favor better ones, hence they coin the term adaptive for this new heuristic. SA is used as acceptance method in their papers.

Lutz [49] gives a very detailed review of the ALNS metaheuristic. He discusses various destroy/repair moves and solution acceptance criteria. He applies the algorithm with TA method on Rich Pickup and Delivery Problem with Time Windows. This thesis also proposes adding noise, randomization and adjustments for methods which provide solutions of higher quality but also spend more time.

Sacramento *et al.* [42] adapt ALNS to VRP with drones. They add a maximum time limit for route completion time and use total cost minimization as the objective. ALNS starts with a VRPD solution by adding drone services to a feasible VRP solution and then improve the solution by adding drone sorties.

4.1.1. Initial Solution Generation

We aim at finding an initial feasible solution by considering truck deliveries only. The number of vehicles is estimated by the ratio of total demand to truck capacity, $\left\lceil \frac{\sum_j q_j}{Q} \right\rceil$. Customers are sorted by their lt_j and then assigned to truck routes sequentially as presented in Figure 4.1. However, this approach may fail to produce feasible solutions due to time windows and customer locations. In case of infeasibility, we add one more vehicle to the set and repeat the algorithm. Improvements and route merging are carried out by ALNS in the first phase of the algorithm.

```

Sort customer list with EDD algorithm
while solution is infeasible do
  for each vehicle in vehicle set do
    Assign next customer to vehicle route
    if There is no unassigned customer then
      Terminate loop
    end if
  end for
  Check feasibility of the solution
  if Solution is not feasible then
    Increase the number of trucks by one
  end if
end while

```

Figure 4.1. Pseudocode for initial solution generation.

4.1.2. Destroy Moves

A subset of customers is removed from the solution in every iteration by destroy moves. In this chapter, a solution refers to the collection of all vehicle routes while a route refers to individual truck routes. The size of the subset to be removed is controlled by a parameter named destruction degree, d . We choose d from an interval

with equal selection probabilities. The interval is a sequence from one to $\lceil n \cdot d_f \rceil$, where d_f is a ratio for determining d proportionally to the problem size.

The solution altering mechanism first selects one route or two routes with equal probabilities. The number of routes and vehicle types determine the nature of the move. For instance, if two routes are selected, then d number of customers are exchanged or relocated. For one route, if a truck and a drone is selected, then d number of customers are removed from the truck service and inserted into drone service. One of following five destroy moves is selected based on their weights and vehicle types to remove d number of customers from the solution. The weights are utilized in selection probability calculations and explained in Subsection 4.1.4.

Random Removal: d number of customers are randomly removed from truck or drone route.

Longest Waiting Removal: The problem involves time windows which may cause a truck to arrive a customer node earlier than its service start time, which implies idle time until service. Selecting the nodes in which a truck spent most of the time waiting and then reinserting them into the route or assigning them to drone could produce a more efficient solution.

Least Desirable Removal: Desirability is defined in terms of distance for trucks and duration for drones. For trucks, customers are sorted by total distance between next node and previous node. The nodes for which the longest distances are traversed are removed from the solution. For drones, sorties that have the longest flight duration are removed with the aim of having more sorties with shorter flight distance and less idle time. The sortie duration is calculated by subtracting the launching time from the recovery time, both flight and waiting times are taken into account.

String Removal: This move removes truck customers that are served in a sequence. A random customer is selected and d nodes including and following that customer is removed from solution. The move can be considered as a simplified version of the related removal move of Shaw [45] or the cluster removal of Sacramento *et al.* [42] since truck customers are ordered in a sequence if they are closely located or have similar time windows.

Outer Nodes Removal: With a cost minimization objective related only to distance, it is assumed that a low-cost, good-quality solution should have well-separated clusters of customers. The destroy move can be considered as a simplified k -means approach. We find the centroid of the truck route by averaging its x - and y -coordinates, calculate and sort distances to the centroid and then select the furthest nodes to the centroid. This move primarily aims to find outlier truck customers and then inserting them into other truck routes or into a drone service. In inter-route operations, better separation of clusters is aimed while in intra-route operations, this move aims at a shorter truck route.

In terms of removing the worst performing nodes, *Longest Waiting Removal* resembles *Least Desirable Removal*. However, the main consideration is time spent on a customer node and this duration has no direct effect on the objective function since our model lacks a penalty term for waiting times. It still proves to be effective in our experimentations by allowing more drone services while leading to less truck waiting for certain instances.

The destruction degree d and destroy moves are strongly affected by each other. For instance, *String Removal* should not be chosen for drones. Also, if there are only five customers and $d = 6$, then it is not possible to remove six unique customers. We address the former by labeling moves based on their usability on truck or drone, hence preventing selection of *String Removal* or *Outer Nodes Removal* for drone-served nodes. For the latter, each move checks the value of d . If d is found not applicable, then the move sets parameter d to its maximum meaningful value.

As a post-processing step, if the move is applied on truck customers, we check the drone sorties starting from or ending at removed truck customers similar to Sacramento *et al.* [42]. We remove any drone customers attached to removed truck customers. If such drone customers are kept intact, intra-route operations can change the sequence of launching and recovery nodes or violate drone endurance. Drone may start its flight from one truck and lands to another for inter-route relocations, which violates assumptions. We select a random subset of removed customers as drone customers, not necessarily at the same number as removed drone customer. As a result, two separate list of customers are delivered to repair moves. One list contains nodes for insertion by selected repair method. The other list contains nodes for drone reassignment if there are any.

4.1.3. Repair Moves

Repair moves are used to reinsert removed nodes to create a complete solution. We use three different repair moves and apply them iteratively. In case drone customers have to be reassigned along with removed trucks customers, we first insert truck customers to create more sortie options. Later, random insertion or cheapest insertion for drone operations is selected with their respective probabilities.

Random Insertion: For truck, customers are reinserted into routes randomly. For drones, we search for a feasible launch node and recovery node couple pseudo-randomly where launch node precedes recovery node and there is no scheduled flight between those nodes. Drone is assigned to first suitable pair found. Infeasibility occurs when all pairs are evaluated and there is no feasible sortie.

Cheapest Insertion: Also called *Nearest Insertion*, the heuristic repairs solution by inserting nodes to increase the total cost least. For trucks, a repair move calculates the cost of inserting a node between every truck-node pairs. Then, the least costly insertion position is determined and the node is inserted there. For drones, similar to *Random Insertion* for drones, feasible launch nodes and recovery nodes are found

among available ones. Then, sortie distances are calculated for each launch and recovery node combination. The sortie with the smallest cost is selected and inserted into drone service.

Regret-2 Move: The last repair move is only performed on truck routes and tries to improve the greedy approach seen in *Cheapest Insertion*. Regret-2 move [47] determines the regret value by computing the difference between the costs of the cheapest and second cheapest solution for all unassigned nodes. Then the node with the maximum regret value is inserted in the best solution and the process is repeated. The algorithm can be generalized as Regret- k where k refers to that k -th cheapest solution is compared to the best instead of the second one as in Regret-2. The implementation of Regret-2 is outlined in Figure 4.2.

```

while there are unassigned customer nodes do
  for each customer unassigned do
    for each possible insertion place on truck route do
      Insert customer to the specific place for new route;
      Calculate the cost of new route;
    end for
    Order resulting costs in ascending order;
    Take difference of second and best elements of the sequence;
    Record the node, best route and difference information;
  end for
  Find the maximum difference value over all nodes;
  Insert the node found into best route and update unassigned nodes;
end while

```

Figure 4.2. Pseudocode for Regret-2 Move.

4.1.4. The Design of the ALNS

The initial solution includes only truck routes as mentioned in Section 4.1.1. We use ALNS for both improving the initial VRPTW solution and scheduling drone services. We exploit the flexible structure of ALNS and design our algorithm in two phases. The first phase is a fast improvement phase for the reduced problem of VRPTW. With the omission of drone delivery, we use only truck-related destroy and repair moves. In this phase, inter-route node relocation/exchanges and intra-route improvements are possible. Also, we eliminate any excessive vehicle that initial solution algorithm could introduce. After 200 non-improving iterations (*nonIter*) or one fourth of the time limit (*timeLimit*) pass, the first phase is terminated.

In the second phase, the sets of destroy and repair moves are extended by adding drone variants. The final weights of the first phase are discarded and therefore ALNS starts with equal destroy and repair probabilities for VRPDTW. The two-phase structure is not a nested design, truck routes can still be improved in the second phase. Additionally, drone customers are inserted and removed. Drone customer removals occur either because of drone moves or removal of sortie starting/ending nodes by truck moves. The pseudocode of the ALNS heuristic is provided in Figure 4.3.

Destroy and repair moves are selected by using the probability, $P(m)$, which reflect the past success of related heuristic in finding good quality solutions. Equation 4.1 is used for the move selection probability. $w(m)$ is the weight vector and records solution quality information. Each $w(m)$ entry is a score which shows the earlier success of the move in finding incumbent, better or feasible solutions. Weights are increased the most for finding the incumbent solution and the least for a feasible solution. Initially, all weights are set equal to one which leads to equal selection probabilities for each destroy and repair heuristic.

```

Form an initial feasible route such that each customer is served by one vehicle;
Compute the initial solution cost;
Initialize simulated annealing parameters  $T$  and  $T_{len}$ ;
Assign equal probability to destroy and repair moves separately;
while  $nonIter \leq maxIncum$  or  $t \leq timeLimit$  do
    Choose a destruction degree parameter,  $d$ ;
    Select one or two routes;
    Select a destroy and repair move respectively;
    Remove  $d$  customer from current route(s) by the selected destroy method;
    if Any truck customer with a connected drone flight is removed then
        Remove connected drone customer also;
    end if
    Reinsert removed customers to route(s) by selected repair method;
    Update destroy and repair methods numbers used in the segment;
    Check feasibility of new solution;
    if New solution is feasible then
        Find the cost of new solution;
        Check if new solution cost is better than current one;
        if New solution cost is better or  $Random(0, 1) \leq e^{-costChange/T}$  then
            Update current solution by new solution;
            if New solution cost is better than incumbent cost then
                Update incumbent solution and cost;
                Set  $nonIter$  to 0;
                Set noise parameter to 0;
            else
                Update  $nonIter$ ;
                if  $nonIter \bmod 1000 = 0$  then
                    Increase noise parameter by 0.001;
                end if
            end if
        end if
    end if

```

Figure 4.3. Pseudocode of ALNS Heuristic.

```

        end if
    end if
end if
Update destroy and repair method scores for the segment;
if Total of destroy moves used in segment is equal to segment length then
    Update destroy and repair method probabilities by their scores in the last
    segment;
    Set destroy and repair methods numbers used in the segment to 0;
end if
if Last temperature  $T$  was used for  $T_{len}$  iterations then
    Update  $T$  as  $T * coolRatio$ 
    Update  $T_{len}$  as  $\max\{T_{len} * coolRatio, 500\}$ 
end if
if Drone deliveries not started yet and ( $nonIter = 200$  or  $t \geq timeLimit/4$ )
then
    Start drone deliveries
end if
end while

```

Figure 4.3. Pseudocode of ALNS Heuristic – (cont.)

$$P(m) = \frac{w(m)}{\sum w(m)} \quad (4.1)$$

Updating weights in every iteration causes excessive workload and early convergence of selection probabilities. Using the same weights for a certain number of iterations enables the heuristic to focus more on solution finding. It also helps having a more effective local search by utilizing various moves in earlier iterations since convergence of selection probabilities favors specific moves later. The number of iterations

which uses the same weights is called a segment. The weights and selection probabilities remain the same in a segment while the number of times a move used ($u(m)$) and its success in the segment ($s(m)$) are recorded. A reaction parameter (ρ) is also used to adjust how fast new success information should be incorporated into the weight vector. At the end of a segment, weights are updated by using previous weight vector, solution quality information, number of times heuristic used and reaction factor. The weight adjustment is shown in Equation 4.2.

$$w(m) = \begin{cases} (1 - \rho)w(m) + \rho \frac{s(m)}{u(m)}, & \text{if } u(m) > 0 \\ (1 - \rho)w(m), & \text{if } u(m) = 0 \end{cases} \quad (4.2)$$

Weights are used to calculate selection probabilities as in Equation 4.1 but there are two exceptions for selecting moves with the stated probability. In case there is no drone-served customer, assigning a drone customer to truck would be infeasible. Secondly, if drone repair moves detect there is no feasible sortie, truck-to-drone insertion results in non-improving solutions. Therefore, we check the number of customers in selected vehicle types and use a smaller set of moves to be selected in both cases. The probabilities are still calculated with Equation 4.1.

New solutions are accepted by SA procedure. Better solutions are always accepted, while a worse solution with cost S' is accepted with a probability based on its deviation from the current solution cost (S) and temperature (T) by equation $\exp(-(S' - S)/T)$. The temperature controls the ratio of accepting worse solutions and adjusted high initially to promote diversification and decreased gradually by using cooling ratio (C_R) for detailed searches towards the end. T adjustments are made at the end of certain number of iterations, T_{len} . T_{len} is increased by a fraction of $1 - C_R$ for each turn until a maximum length, which is set as 500 iterations.

The time limit (*timeLimit*) and the number of non-improving iterations (*nonIter*) are two parameters used for termination criterion. The time limit is set to five minutes and non-improving iteration limit is set to 10,000 iterations.

ALNS has a bias towards earlier success of destroy and repair moves. If a certain move is proved to be useful initially while another one is not, their weights are adjusted accordingly. Initial success and failure of moves can be reversed throughout the heuristic, however weight adjustment is not fast enough to use later information if weight discrepancies are high. We apply three different procedures to stop or at least relieve such effects. The first one is adding a noise term on the objective whose value depends on a percentage of the solution cost. The noise percentage (δ) starts from zero and increases at each tenth of the non-improving iteration limit by a fixed amount equal to the noise percentage step size (η). The second measure is the minimum probability limit for allowing all moves to be used any time. The last one is applying the selected destroy and repair method more than once in an iteration and choosing the best solution. This parameter is called repetition parameter, (r) and allows a more robust comparison of moves.

5. COMPUTATIONAL ANALYSIS

5.1. Instance Generation

Two types of problem instances which differ in the distribution of customer locations. Instances are generated randomly over a 40x40 km grid. In random method, coordinates of the depot and customers are selected uniformly. The second method aims at clustering customers. A random customer is selected as the seed, and the remaining customers are selected from the 10x10 km area centered on the seed customer. The depot location is determined independent of customer locations over 40x40 km grid.

Time windows are generated between two and six hours of length, and longer time windows are generated with higher probabilities. The nodes, which are close to each other, are assigned to overlapped time windows. The grid is divided into four equal regions where each customer in a region is assigned to a at most two hours apart service start time, thereby guaranteeing time window overlap together with at least two hours of service interval.

All customer demands are considered to be equal and in unit size. Vehicle capacity Q is set to $\left\lceil \frac{\sum_j q_j}{v} \right\rceil$ to use all predetermined number of vehicles v . One and two vehicles are used for small instances. For medium instances, two and four vehicles are used.

5.2. Parameter Tuning

The ALNS has numerous parameters which makes a grid search for parameter optimization difficult. Thus, we first use the parameter values which we found reasonable while designing the algorithm. Then, we adjust the most significant ones iteratively. For comparison, we solve small instances with eight, 12 and 16 customers with one vehicle and two vehicles.

The repetition parameter, r , is used to find several solutions in an iteration using the same destroy and repair moves. Then the best solution found is passed to SA for approval or rejection. A high value of r leads to more robust destroy and repair move comparisons but also slows the algorithm down. Hence, we select r as the first parameter to tune. The average percentile deviation from optimal objective value can be seen in Figure 5.1 for different r values. With one vehicle, the smallest deviation is achieved for $r = 5$. With two vehicles, $r = 3$ and $r = 5$ give almost the same results. Thus we select $r = 5$.

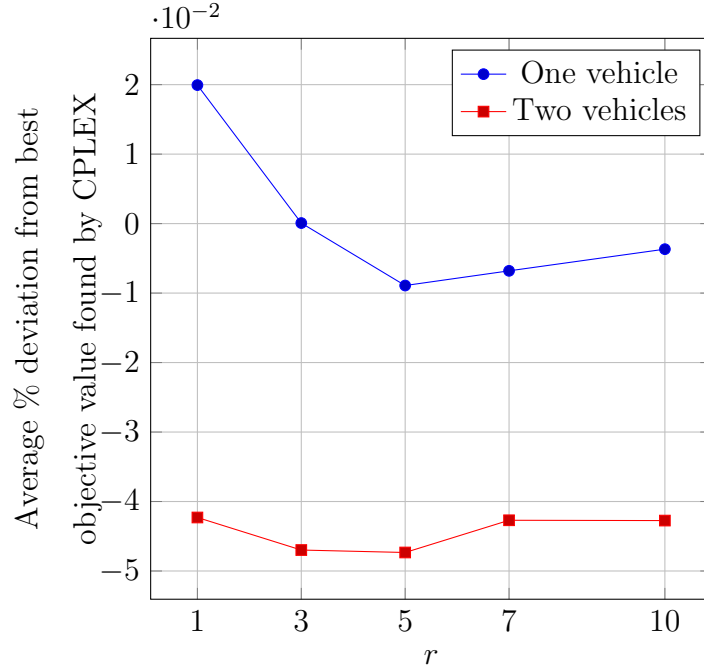


Figure 5.1. Avg. % dev. from best obj. value found by CPLEX for various r values.

The destruction degree parameter, d , determines the number of nodes that are removed and reinserted in every iteration. While a larger number of removal and reinsertion is better for exploring the solution space and escaping from local minima, it also increases the computational workload. d_f controls the range of d , as d goes from one to $\lceil n \cdot d_f \rceil$, where n is the number of customers. Therefore, we select d_f as the second parameter for tuning. As can be seen in Figure 5.2, instances with one vehicle are solved the worst for $d_f = 0.1$ and $d_f = 0.4$ values. However, instances with two vehicles show less sensitivity to d_f . We set the d_f value to 0.2.

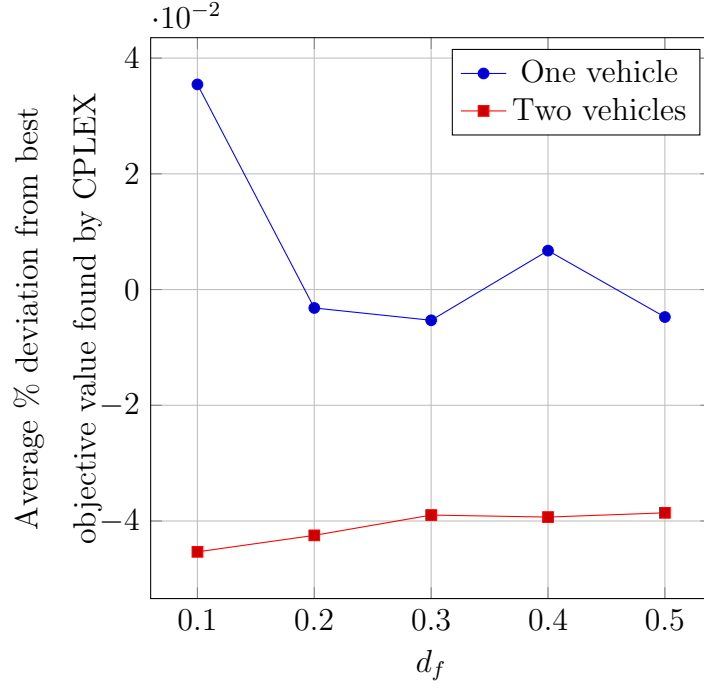


Figure 5.2. Avg. % dev. from best obj. value found by CPLEX for various d_f values.

The reaction factor, ρ , controls how fast weights should be adjusted based on the solution quality of the last segment. Also, ρ decreases the weights of moves which are not used in the segments or which produce infeasible or worse solutions. The results are shown in Figure 5.3. There is no clear pattern for instances with one vehicle solutions, and instances with two vehicles show slightly better results with $\rho = 0.2$ and $\rho = 0.3$ values. We set $\rho = 0.2$.

The last parameter for fine-tuning is noise percentage step size η . A noise term is used which is normally distributed and whose mean is proportional to the noise percentage (δ) of the current solution cost. η determines the step size of change in δ . δ is set to zero initially and after every incumbent solution. At every tenth of *nonIter* limit, δ is increased by η . We experiment with no noise parameter and four different levels. Except 0.002 level for instances with one vehicle, the algorithm is nearly insensitive to noise as seen in Figure 5.4. We select $\eta = 0.001$ as a result.

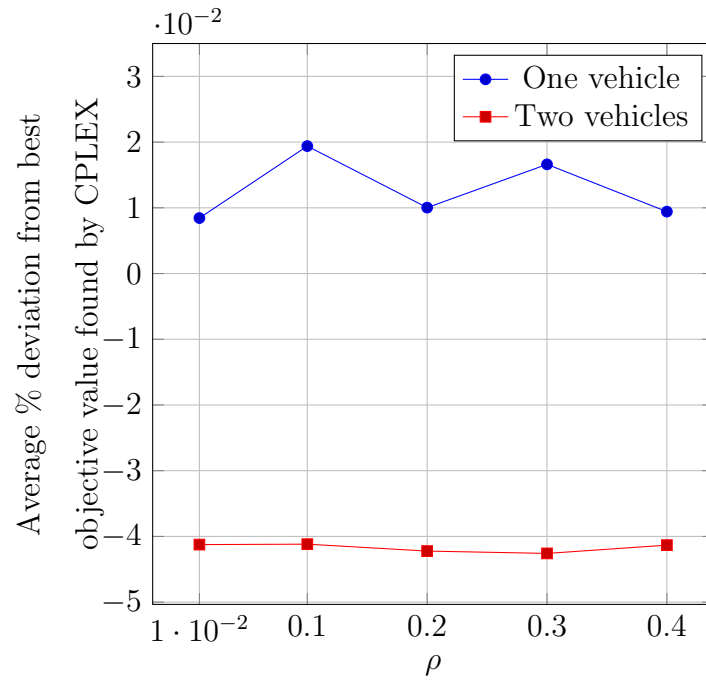


Figure 5.3. Avg. % dev. from best obj. value found by CPLEX for various ρ values.

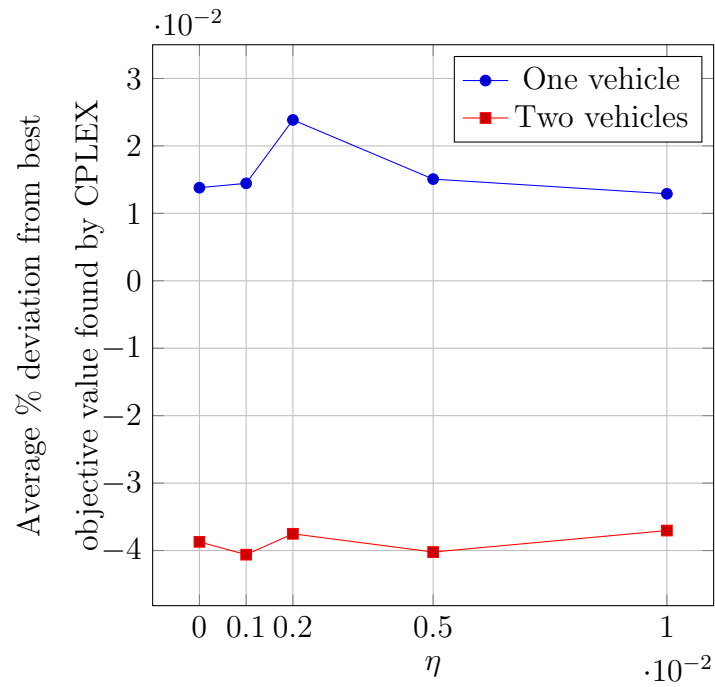


Figure 5.4. Avg. % dev. from best obj. value found by CPLEX for various η values.

5.3. Experimental Results

Table 5.1 presents the instances used for parameter tuning. Instance name combines the number of customers and unique instance id. The type designates the instance generation method. Each instance is solved both with one vehicle and two vehicles with CPLEX 12.7.1 on GAMS suite 24.8.5. An Intel i7-4700HQ CPU @2.4GHz and 8 GB ram environment is used for both CPLEX and the ALNS heuristic. The ALNS is coded in the R-language. Relative optimality gap is set to zero. Six hours resource limit is given for the instances in the Table 5.1 since they are used for parameter tuning. For all the other instances, three hours resource limit is given. For some instances, the resource limit is achieved before the optimal solution is found. Therefore, the values on the table are suboptimal for certain instances with certain number of vehicles. *Opt. Gap* presents the optimality gap between the best possible solution and the best solution obtained, and it is provided by CPLEX.

Table 5.1. Results obtained by CPLEX for small instances.

Instance	Type	$v = 1$	Opt. Gap(%)	CPU Time(s)	$v = 2$	Opt. Gap(%)	CPU Time(s)
8-1	Random	497.07	0	38	597.01	0	108
8-2	Random	1099.32	0	5	1349.94	0	18
8-3	Clustered	394.47	0	83	455.24	0	191
8-4	Clustered	869.20	0	16	1588.74	0	841
12-1	Random	1430.88	0	315	1672.90	0	4239
12-2	Random	1274.58	0	693	1343.04	0	1726
12-3	Clustered	610.34	0	12997	1019.14	31.18	21600
12-4	Clustered	824.45	0	4497	1368.57	13.51	21600
16-1	Random	1788.55	0	5650	1893.13	16.92	21600
16-2	Random	1721.95	19.16	21600	1848.82	31.92	21600
16-3	Clustered	1719.03	42.96	21600	1756.74	22.34	21600
16-4	Clustered	1057.98	43.45	21600	1625.78	62.80	21600

In Table 5.2, we provide the comparison of the ALNS results with those obtained CPLEX for one vehicle. We run the ALNS five times for each instance, and *Best* represents the best objective value while *Mean* represents the average objective value over five runs. Deviations are defined as $100x((Heuristic\ cost/Solver\ cost) - 1)$. *Total Solution Time* shows the total duration for five ALNS runs. ALNS can find optimal or near optimal solutions in most instances. For the last three 16 customers instances, ALNS is able to find better solutions than the CPLEX can find within three hours.

Table 5.2. ALNS results for small instances with one vehicle.

Instance	Best	Mean	% Dev. of Best from CPLEX	% Dev. of Mean from CPLEX	Total Sol. Time(s)
8-1	497.07	497.07	0	0	381
8-2	1099.32	1099.32	0	0	326
8-3	394.47	394.47	0	0	409
8-4	869.20	869.20	0	0	478
12-1	1438.62	1494.41	0.54	4.44	852
12-2	1282.08	1380.61	0.59	8.32	1084
12-3	628.45	683.50	2.97	11.99	693
12-4	852.98	882.74	3.46	7.07	981
16-1	1788.55	1853.12	0	3.61	1020
16-2	1466.76	1605.25	-14.82	-6.78	1176
16-3	1446.21	1479.08	-15.87	-13.96	957
16-4	974.43	983.02	-7.90	-7.09	1046

Table 5.3 compares ALNS results with those obtained by CPLEX for two vehicles. The solver cannot find the optimal values for half of the instances within three hours. ALNS find optimal values or better values than the reported cost excluding one instance. Total time that the ALNS used is greatly increased as in the solver. The ALNS produces more stable results compared to one vehicle instances as best and mean deviations are very close to each other.

Table 5.3. ALNS results for small instances with two vehicles.

Instance	Best	Mean	% Dev. of Best from CPLEX	% Dev. of Mean from CPLEX	Total Sol. Time(s)
8-1	597.01	597.54	0	0.09	675
8-2	1349.94	1349.94	0	0	467
8-3	455.24	457.70	0	0.54	430
8-4	1588.74	1588.74	0	0	403
12-1	1675.22	1700.02	0.14	1.62	1065
12-2	1343.04	1350.93	0	0.59	1246
12-3	899.05	916.48	-11.78	-10.07	1269
12-4	1311.45	1321.46	-4.17	-3.44	1048
16-1	1885.61	1889.97	-0.40	-0.17	1279
16-2	1650.54	1664.21	-10.72	-9.99	1436
16-3	1633.81	1663.57	-7.00	-5.30	1500
16-4	1122.20	1135.98	-30.97	-30.13	1500

We generate a second set of small VRPDTW instances where time windows are relaxed. Without changing any other parameter for instance generation, we increase the length of time windows by a factor of two. Thus, customers have at least four hours service interval. Certain customers can be visited any time during the day. The results by CPLEX for these instances with one vehicle and two vehicles are presented in Table 5.4. For the instances with eight customers, CPLEX find the optimal solution easily. However, CPU times for solutions of instances with 12 and 16 customers are increased significantly as in the first set. For the instances with 16 customers and with one vehicle, only one optimal solution is obtained by CPLEX within three hours. For solutions with two vehicles, the problem size CPLEX can obtain an optimal solution is reduced to 12 customers.

Table A.1 and A.2 reveals the performance of ALNS on the second set with relaxed time windows. Results show that ALNS is able to find optimal values or to improve best objective values found by CPLEX within three hours for the most instances with

relaxed time windows. However, total solution time is increased significantly for the instances with two vehicles, and average deviation from the objective values of CPLEX is slightly increased compared to instances with one vehicle. Adding more moves to handle customer distribution among vehicles may lead to better clustering and may decrease the time heuristic uses, thereby leading to a faster and more robust heuristic.

Table 5.4. CPLEX results for small instances with relaxed time windows.

Instance	Type	$v = 1$	Opt. Gap(%)	CPU Time(s)	$v = 2$	Opt. Gap(%)	CPU Time(s)
8-5	Random	1208.83	0	5	1832.78	0	14
8-6	Random	1182.41	0	5	1450.72	0	31
8-7	Random	832.65	0	9	1259.94	0	96
8-8	Clustered	351.14	0	28	431.55	0	71
8-9	Clustered	313.66	0	50	350.89	0	110
8-10	Clustered	654.05	0	15	597.01	0	72
12-5	Random	1123.32	0	56	1458.46	0	2607
12-6	Random	717.03	0	7027	1090.89	32.24	10800
12-7	Random	738.45	0	1704	1154.04	21.00	10800
12-8	Clustered	605.10	0	8201	903.69	30.47	10800
12-9	Clustered	625.77	7.88	10800	900.91	30.04	10800
12-10	Clustered	527.59	0	1947	782.37	29.29	10800
16-5	Random	581.76	30.79	10800	708.57	42.33	10800
16-6	Random	914.50	20.75	10800	1550.18	31.97	10800
16-7	Random	703.89	14.41	10800	1186.86	21.93	10800
16-8	Clustered	619.17	19.28	10800	884.82	40.89	10800
16-9	Clustered	1271.60	11.43	10800	1540.58	39.65	10800
16-10	Clustered	1119.70	0	8935	1454.97	29.88	10800

We cannot get feasible solutions of the instances with more than 16 customers within three hours by using CPLEX. Instead, we solve the problem as VRPTW for medium-sized instances. We generate instances with 20, 30 and 40 customers and solve them with CPLEX and with two and four vehicles. The results for are presented in Table 5.5. Then, we use the ALNS to compare the heuristic results of truck-drone

combined delivery solutions and truck-only solutions of CPLEX. Table 5.6 provides the results by the ALNS. ALNS outperforms best objective value obtained by CPLEX within three hours in more than half of the instances. However, the heuristic is not able to improve certain instances even by including drone deliveries. Local minima can be the reason and adjusting temperature of SA proportional to number of customers can help avoiding these local minima.

Table 5.5. CPLEX results for medium-sized instances solved as VRPTW.

Instance	Type	$v = 2$	Opt. Gap(%)	CPU Time(s)	$v = 4$	Opt. Gap(%)	CPU Time(s)
20-1	Random	2713.35	0	78	3318.17	0	2506
20-2	Random	2312.90	0	795	2835.53	6.45	10800
20-3	Clustered	1684.74	0	913	2688.71	9.59	10800
20-4	Clustered	1234.06	0	201	1929.55	11.00	10800
30-1	Random	2953.89	0	9021	2796.71	6.21	10800
30-2	Random	3018.35	0	9725	3842.79	28.53	10800
30-3	Clustered	1869.87	0	1464	2573.16	6.58	10800
30-4	Clustered	1917.97	0	2157	2732.04	5.21	10800
40-1	Random	5569.41	47.78	10800	4085.16	41.56	10800
40-2	Random	7261.79	57.40	10800	5256.04	18.41	10800
40-3	Clustered	2236.97	14.86	10800	3007.62	26.62	10800
40-4	Clustered	2258.13	28.40	10800	1735.15	27.50	10800

ALNS phase one and two solutions for instance 40-4 with two vehicles are presented in Figure 5.5 and Figure 5.6 for better comparison. The square represents the depot, solid lines and dashed lines denote truck and drone paths respectively. The ALNS starts with an initial solution that contains only truck services and improves that solution in the first phase without drones. In the second phase, drone services are started. Improvements of truck routes are also continued as it can be seen from that customers are distributed differently in two solutions. Drones are generally sent to outer customers while truck visits the ones closer to the centre since unit cost of truck traveling is higher than that of drone.

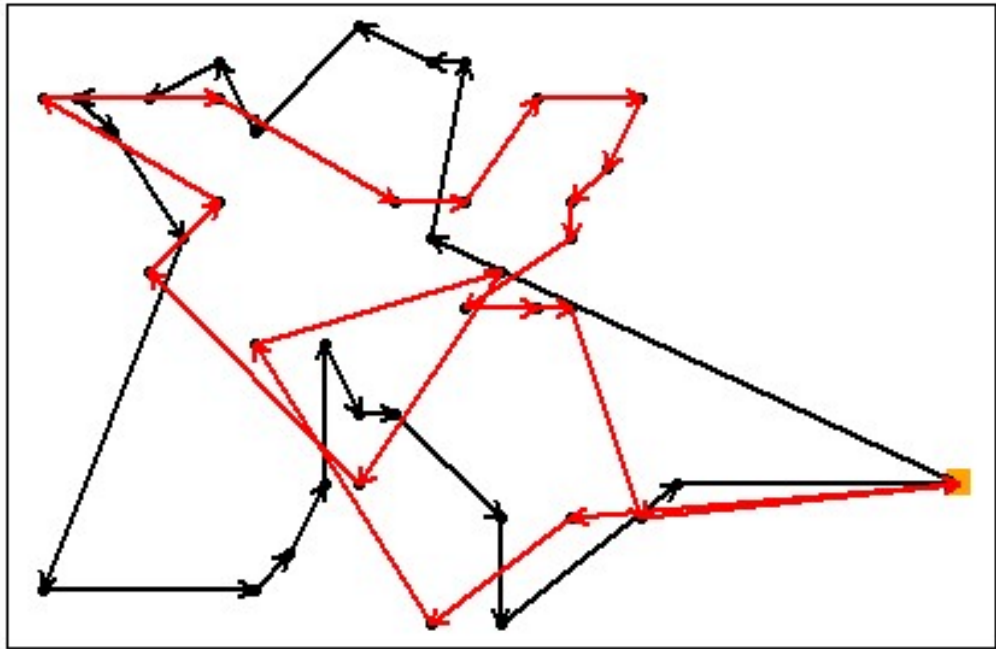


Figure 5.5. Two vehicles VRPTW solution for 40-4 instance by ALNS phase one.

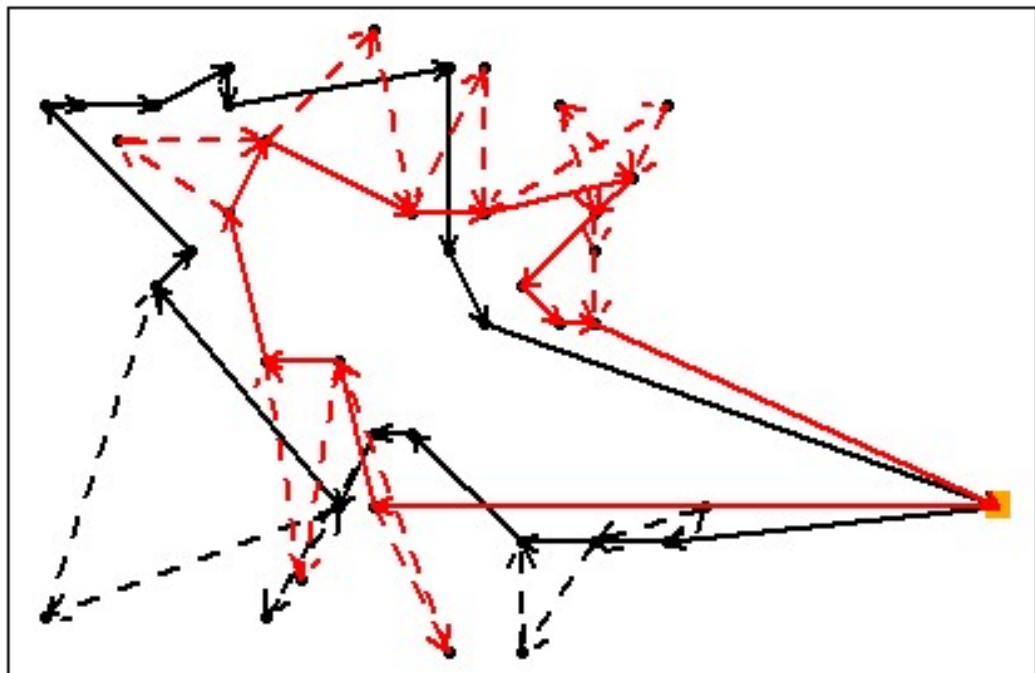


Figure 5.6. Two vehicles VRPDTW solution for 40-4 instance by ALNS phase two.

Table 5.6. ALNS results for medium-sized instances.

v	Instance	Best	Mean	% Dev. of Best from CPLEX	% Dev. of Mean from CPLEX	Total Sol. Time(s)
2	20-1	2452.61	2630.40	-9.61	-3.06	1298
	20-2	2079.89	2187.94	-10.07	-5.40	1374
	20-3	1466.76	1500.68	-12.94	-10.93	1290
	20-4	1103.90	1131.07	-10.55	-8.35	1304
	30-1	3113.73	3168.23	5.41	7.26	1500
	30-2	3163.19	3406.53	4.80	12.86	1500
	30-3	1992.20	2153.39	6.54	15.16	1412
	30-4	1927.57	2081.85	0.50	8.54	1500
	40-1	2316.98	2437.04	-58.4	-56.24	1500
	40-2	2454.87	2509.96	-66.19	-65.44	1500
	40-3	2291.78	2413.81	2.45	7.91	1500
	40-4	1439.87	1525.54	-36.24	-32.44	1500
4	20-1	3336.18	3562.42	0.54	7.36	1361
	20-2	2636.55	2776.15	-7.02	-2.09	1466
	20-3	2550.02	2584.00	-5.16	-3.89	1411
	20-4	1920.68	1984.97	-0.46	2.87	1405
	30-1	3000.00	3143.80	7.27	12.41	1500
	30-2	3348.60	3516.50	-12.86	-8.49	1500
	30-3	2610.12	2700.05	1.44	4.93	1500
	30-4	2757.08	2867.46	0.92	4.96	1500
	40-1	3295.44	3528.92	-19.33	-13.62	1500
	40-2	5160.82	5293.29	-1.81	0.71	1500
	40-3	2847.99	2935.62	-5.31	-2.39	1500
	40-4	2294.54	2351.57	32.24	35.53	1500

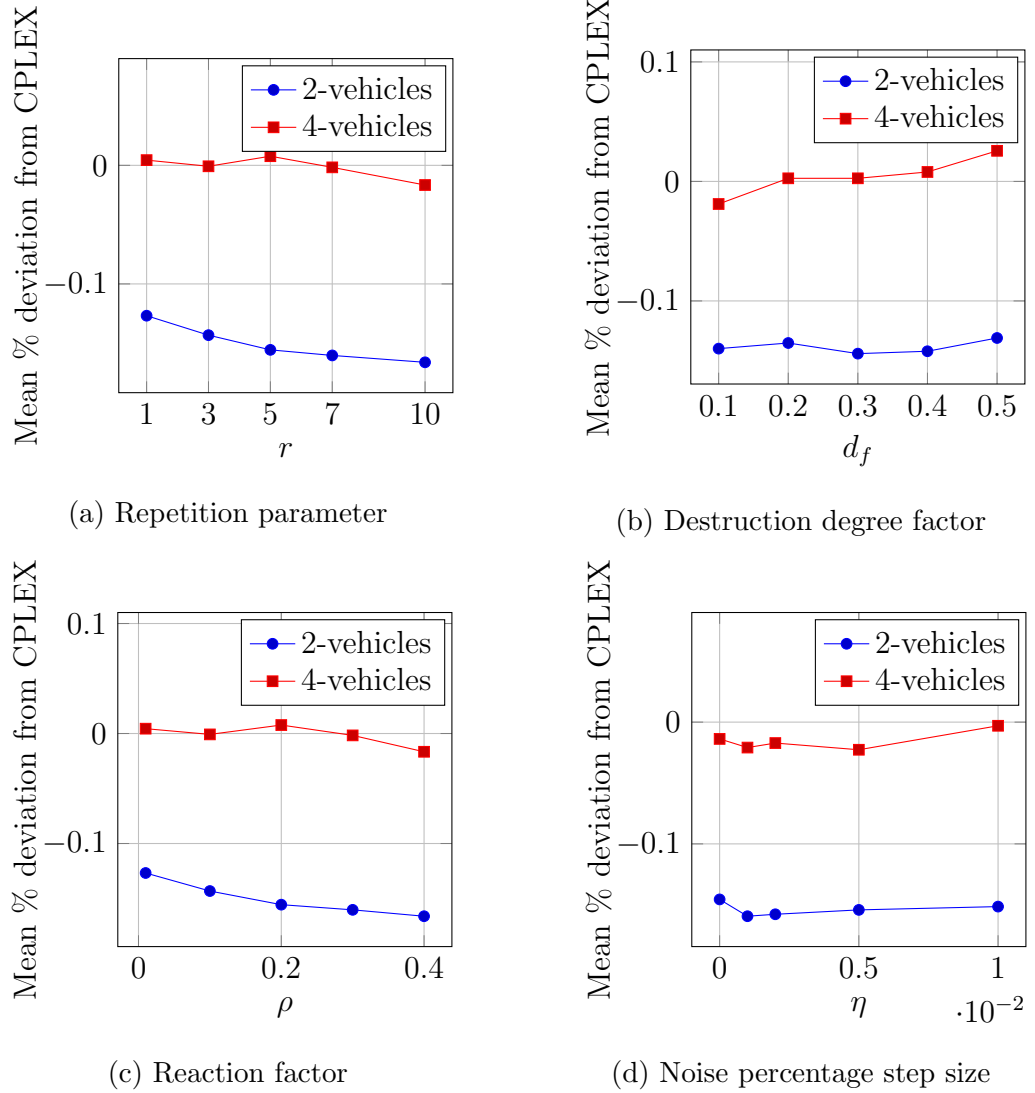


Figure 5.7. Parameter sensitivity using medium-sized instances.

We adjust four different parameters by experimenting over small-sized instances as described in the previous section. The conclusion was that ALNS shows limited sensitivity to parameter values. We experiment on the medium-sized instances with the same parameter set and values to see whether the number of customers and vehicles affect parameter sensitivity. The results can be seen in Figure 5.7. Except the noise percentage step size, the parameters showed a more clear pattern and more sensitivity on medium-sized instances compared to small-sized instances.

We implement ALNS algorithm in two phases. Phase one is a fast truck-only solution improvement stage for VRPTW. We research the effect of initializing phase

two with better truck routes by slightly changing the ALNS implementation. The non-improving iteration limit on phase one is eliminated and phase one is given three minutes to improve initial solution. The phase two is given five minutes and thus eight minutes time limit in total is set on each instance. Table 5.7 reveals that solutions are improved compared to default implementation results in Table 5.6 for most of the instances by using a better truck route for initialization of drone service assignment.

Table 5.7. ALNS with long phase one results for medium-sized instances.

v	Instance	Best	Mean	% Dev. of Best from CPLEX	% Dev. of Mean from CPLEX	Total Sol. Time(s)
2	20-1	2397.2482	2453.5073	-11.65	-9.58	2400
	20-2	1851.0226	2141.4495	-19.97	-7.41	1770
	20-3	1473.257	1504.0621	-12.55	-10.72	1578
	20-4	1095.4388	1125.4972	-11.23	-8.80	2156
	30-1	2946.7748	3100.9233	-0.24	4.98	2400
	30-2	2947.1864	3159.1546	-2.36	4.67	2400
	30-3	1997.424	2044.818	6.82	9.36	2327
	30-4	1963.4772	2044.5581	2.37	6.60	2400
	40-1	2280.2916	2356.9865	-59.06	-57.68	2400
	40-2	2416.2826	2441.6428	-66.73	-66.38	2400
	40-3	2287.174	2385.8164	2.24	6.65	2400
	40-4	1436.3374	1504.8188	-36.39	-33.36	2400
4	20-1	3321.5004	3361.4392	0.11	1.30	2400
	20-2	2533.5944	2614.9069	-10.65	-7.78	2034
	20-3	2455.2236	2538.9518	-8.68	-5.57	1851
	20-4	1907.151	1953.3238	-1.16	1.23	2381
	30-1	3017.0222	3071.1176	7.88	9.81	2400
	30-2	3367.405	3542.9592	-12.37	-7.80	2400
	30-3	2588.739	2669.2022	0.61	3.73	2400
	30-4	2795.5882	2885.9632	2.33	5.63	2400
	40-1	3352.408	3474.9834	-17.94	-14.94	2323
	40-2	5038.229	5322.6309	-4.14	1.27	2400
	40-3	2541.0306	2736.84	-15.51	-9.00	2400
	40-4	2275.365	2332.8513	31.13	34.45	2361

We further experiment on 12-customers instances to observe the effect of the number of drone sorties on the total cost. We add drone sortie limit constraints to our VRPDTW model to obtain the exact amount of certain drone sorties in a solution. Then, we normalize the results for comparison by dividing with the no-sortie solution cost. The results are provided in Figure 5.8. Adding drone sorties initially decrease the total cost. However, the marginal effect also decreases. The total cost increases after a certain number of drone sorties. Despite being cheaper and faster, drone sorties cause tight schedules due to their endurance and waiting time limits. Time windows handling and truck-drone synchronization in existence of multiple drone sorties can be the reason of negative effect of adding drone sorties after a certain point.

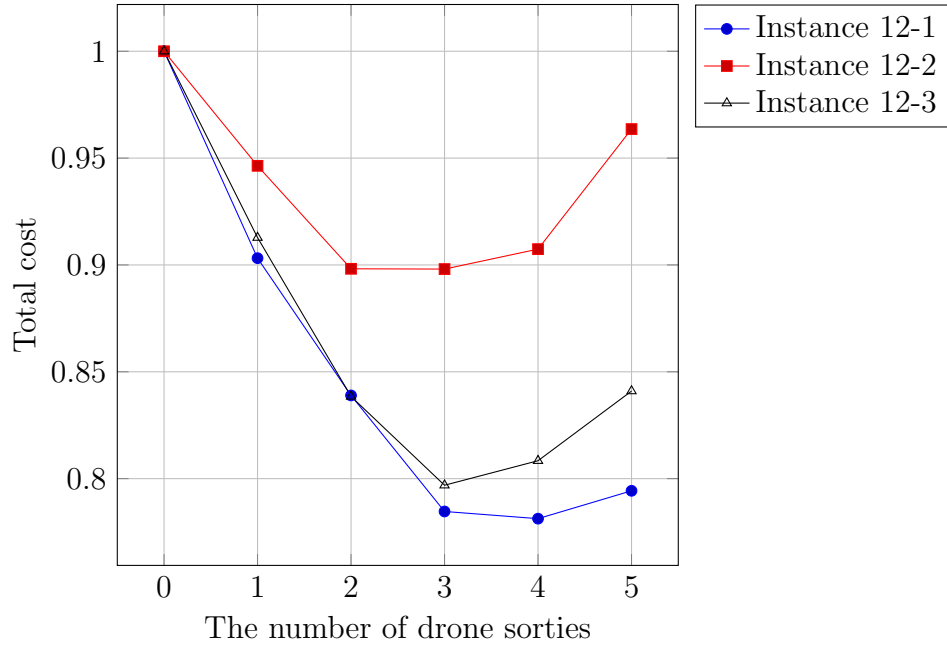


Figure 5.8. The total cost per drone sortie comparison on 12-customers instances.

6. CONCLUSION

In this thesis, we formulate a Vehicle Routing Problem with Drones and Time Windows (VRPDTW) model by extending FSTSP. We allow drones to wait on ground for energy conservation limited to a maximum duration. Due to NP-hardness of the problem, using solvers to find exact solutions is inefficient for small sized instances. For medium sized instances like 30-40 customers, finding a feasible solution using solvers under three hours is also not possible. Therefore, we design an ALNS heuristic to find a near optimal solution fast.

We experiment and tune parameters on small instances. ALNS proves it can find optimal or near optimal solutions on small instances. For medium sized instances, we solve them with VRPTW and compare the results with ALNS. The heuristic find VRPDTW solutions which improves VRPTW solution on most instances. We show that increasing the number of drone sorties on a solution do not always decrease the total cost due to synchronization difficulties.

Drone-truck combined VRPDTW has promising outcomes for cost reduction, hence needs further considerations. One extension of our problem can be limiting truck's waiting time and penalizing waiting times of both vehicles. Energy consumption during drone load carrying can be integrated to model. Drone batteries per truck can be limited which would restrain drone flights further. Another extension can allow multiple drones per truck in a setting where drone number is scarce and distribution of drones among vehicles should be decided beforehand.

ALNS can be improved by considering a route selection mechanism similar to destroy and repair moves selection. If alterations on a route do not lead to better solutions, then the route can be selected less.

REFERENCES

1. Lipsman, A., *Global Ecommerce 2019*, <https://www.emarketer.com/content/global-ecommerce-2019>, accessed in June 2020.
2. Gross, D., *Amazon's drone delivery: How would it work?*, <https://edition.cnn.com/2013/12/02/tech/innovation/amazon-drones-questions/>, accessed in June 2020.
3. Intelligence, B. I., *Why Amazon, UPS and even Domino's is investing in drone delivery services*, <https://www.businessinsider.com/drone-delivery-services>, accessed in June 2020.
4. Murray, C. C. and A. G. Chu, "The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery", *Transportation Research Part C: Emerging Technologies*, Vol. 54, pp. 86–109, 2015.
5. Boysen, N., S. Schwerdfeger and F. Weidinger, "Scheduling last-mile deliveries with truck-based autonomous robots", *European Journal of Operational Research*, Vol. 271, No. 3, pp. 1085–1099, 2018.
6. Ponza, A., *Optimization of drone-assisted parcel delivery*, Master's Thesis, University of Padova, 2016.
7. Agatz, N., P. Bouman and M. Schmidt, "Optimization approaches for the traveling salesman problem with drone", *Transportation Science*, Vol. 52, No. 4, pp. 965–981, 2018.
8. Ha, Q. M., Y. Deville, Q. D. Pham and M. H. Ha, "On the min-cost traveling salesman problem with drone", *Transportation Research Part C: Emerging Technologies*, Vol. 86, pp. 597–621, 2018.

9. Ha, Q. M., Y. Deville, Q. D. Pham and M. H. Hà, “A hybrid genetic algorithm for the traveling salesman problem with drone”, *Journal of Heuristics*, Vol. 26, No. 2, pp. 219–247, 2020.
10. de Freitas, J. C. and P. H. V. Penna, “A variable neighborhood search for flying sidekick traveling salesman problem”, *International Transactions in Operational Research*, Vol. 27, No. 1, pp. 267–290, 2020.
11. Yurek, E. E. and H. C. Ozmutlu, “A decomposition-based iterative optimization algorithm for traveling salesman problem with drone”, *Transportation Research Part C: Emerging Technologies*, Vol. 91, pp. 249–262, 2018.
12. Ferrandez, S. M., T. Harbison, T. Weber, R. Sturges and R. Rich, “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, 2016.
13. Chang, Y. S. and H. J. Lee, “Optimal delivery routing with wider drone-delivery areas along a shorter truck-route”, *Expert Systems with Applications*, Vol. 104, pp. 307–317, 2018.
14. Tu, P. A., N. T. Dat and P. Q. Dung, “Traveling salesman problem with multiple drones”, *Proceedings of the Ninth International Symposium on Information and Communication Technology*, pp. 46–53, 2018.
15. Marinelli, M., L. Caggiani, M. Ottomanelli and M. Dell’Orco, “En route truck–drone parcel delivery for optimal vehicle routing strategies”, *IET Intelligent Transport Systems*, Vol. 12, No. 4, pp. 253–261, 2017.
16. Luo, Z., Z. Liu and J. Shi, “A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle”, *Sensors*, Vol. 17, No. 5, p. 1144, 2017.

17. Dorling, K., J. Heinrichs, G. G. Messier and S. Magierowski, “Vehicle routing problems for drone delivery”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 47, No. 1, pp. 70–85, 2016.
18. Jeong, H. Y., B. D. Song and S. Lee, “Truck-drone hybrid delivery routing: Payload-energy dependency and No-Fly zones”, *International Journal of Production Economics*, Vol. 214, pp. 220–233, 2019.
19. Poikonen, S. and B. Golden, “Multi-visit drone routing problem”, *Computers & Operations Research*, Vol. 113, p. 104802, 2020.
20. Boysen, N., D. Briskorn, S. Fedtke and S. Schwerdfeger, “Drone delivery from trucks: Drone scheduling for given truck routes”, *Networks*, Vol. 72, No. 4, pp. 506–527, 2018.
21. Kim, S. and I. Moon, “Traveling salesman problem with a drone station”, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 49, No. 1, pp. 42–52, 2018.
22. Wang, K., B. Yuan, M. Zhao and Y. Lu, “Cooperative route planning for the drone and truck in delivery services: A bi-objective optimisation approach”, *Journal of the Operational Research Society*, Vol. 71, No. 9, pp. 1–18, 2019.
23. Wen, T., Z. Zhang and K. K. Wong, “Multi-objective algorithm for blood supply via unmanned aerial vehicles to the wounded in an emergency situation”, *Plos One*, Vol. 11, No. 5, p. e0155176, 2016.
24. Dell’Amico, M., R. Montemanni and S. Novellani, “Drone-assisted deliveries: new formulations for the flying sidekick traveling salesman problem”, *arXiv preprint arXiv:1906.02962*, 2019.
25. Bouman, P., N. Agatz and M. Schmidt, “Dynamic programming approaches for the traveling salesman problem with drone”, *Networks*, Vol. 72, No. 4, pp. 528–542,

2018.

26. Poikonen, S., B. Golden and E. A. Wasil, “A branch-and-bound approach to the traveling salesman problem with a drone”, *INFORMS Journal on Computing*, Vol. 31, No. 2, pp. 335–346, 2019.
27. Carlsson, J. G. and S. Song, “Coordinated logistics with a truck and a drone”, *Management Science*, Vol. 64, No. 9, pp. 4052–4069, 2018.
28. Kitjacharoenchai, P., M. Ventresca, M. Moshref-Javadi, S. Lee, J. M. Tanchoco and P. A. Brunese, “Multiple traveling salesman problem with drones: Mathematical model and heuristic approach”, *Computers & Industrial Engineering*, Vol. 129, pp. 14–30, 2019.
29. Murray, C. C. and R. Raj, “The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones”, *Transportation Research Part C: Emerging Technologies*, Vol. 110, pp. 368–398, 2020.
30. Wang, X., S. Poikonen and B. Golden, “The vehicle routing problem with drones: Several worst-case results”, *Optimization Letters*, Vol. 11, No. 4, pp. 679–697, 2017.
31. Poikonen, S., X. Wang and B. Golden, “The vehicle routing problem with drones: Extended models and connections”, *Networks*, Vol. 70, No. 1, pp. 34–43, 2017.
32. Campbell, J. F., D. Sweeney and J. Zhang, *Strategic design for delivery with trucks and drones*, Technical Report, University of Missouri, Columbia, MO, USA, 2017.
33. Wang, Z. and J.-B. Sheu, “Vehicle routing problem with drones”, *Transportation Research Part B: Methodological*, Vol. 122, pp. 350–364, 2019.
34. Daknama, R. and E. Kraus, “Vehicle routing with drones”, *arXiv preprint arXiv:1705.06431*, 2017.

35. Karak, A. and K. Abdelghany, “The hybrid vehicle-drone routing problem for pick-up and delivery services”, *Transportation Research Part C: Emerging Technologies*, Vol. 102, pp. 427–449, 2019.
36. Li, Y., G. Zhang, Z. Pang and L. Li, “Continuum approximation models for joint delivery systems using trucks and drones”, *Enterprise Information Systems*, Vol. 14, No. 4, pp. 406–435, 2020.
37. Schermer, D., M. Moeini and O. Wendt, “A matheuristic for the vehicle routing problem with drones and its variants”, *Transportation Research Part C: Emerging Technologies*, Vol. 106, pp. 166–204, 2019.
38. Schermer, D., M. Moeini and O. Wendt, “A hybrid VNS/Tabu search algorithm for solving the vehicle routing problem with drones and en route operations”, *Computers & Operations Research*, Vol. 109, pp. 134–158, 2019.
39. Ulmer, M. W. and B. W. Thomas, “Same-day delivery with heterogeneous fleets of drones and vehicles”, *Networks*, Vol. 72, No. 4, pp. 475–505, 2018.
40. Ham, A. M., “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, 2018.
41. Pugliese, L. D. P. and F. Guerriero, “Last-mile deliveries by using drones and classical vehicles”, *International Conference on Optimization and Decision Science*, pp. 557–565, Springer, 2017.
42. Sacramento, D., D. Pisinger and S. Ropke, “An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones”, *Transportation Research Part C: Emerging Technologies*, Vol. 102, pp. 289–315, 2019.
43. Ropke, S. and D. Pisinger, “A unified heuristic for a large class of vehicle routing problems with backhauls”, *European Journal of Operational Research*, Vol. 171,

No. 3, pp. 750–775, 2006.

44. Pisinger, D. and S. Ropke, “A general heuristic for vehicle routing problems”, *Computers & operations research*, Vol. 34, No. 8, pp. 2403–2435, 2007.
45. Shaw, P., “Using constraint programming and local search methods to solve vehicle routing problems”, *International conference on principles and practice of constraint programming*, pp. 417–431, Springer, 1998.
46. Schrimpf, G., J. Schneider, H. Stamm-Wilbrandt and G. Dueck, “Record breaking optimization results using the ruin and recreate principle”, *Journal of Computational Physics*, Vol. 159, No. 2, pp. 139–171, 2000.
47. Ropke, S. and D. Pisinger, “An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows”, *Transportation Science*, Vol. 40, No. 4, pp. 455–472, 2006.
48. Pisinger, D. and S. Ropke, “Large neighborhood search”, *Handbook of Metaheuristics*, pp. 399–419, Springer, 2010.
49. Lutz, R., *Adaptive Large Neighborhood Search*, Bachelor Thesis, Universitat Ulm, 2015.

APPENDIX A: ALNS RESULTS FOR RELAXED SET

Table A.1. ALNS results for small instances with relaxed time windows and with one vehicle.

Instance	Best	Mean	% Dev. of Best from CPLEX	% Dev. of Mean from CPLEX	Total Sol. Time(s)
8-5	1208.83	1208.83	0	0	433
8-6	1182.41	1182.41	0	0	415
8-7	832.65	832.65	0	0	584
8-8	351.14	351.14	0	0	479
8-9	313.66	313.66	0	0	685
8-10	654.05	654.05	0	0	421
12-5	1123.32	1123.32	0	0	770
12-6	717.03	719.27	0	0.31	1313
12-7	746.82	746.83	1.13	1.13	1144
12-8	605.10	609.14	0	0.67	1017
12-9	605.10	606.47	-3.30	-3.08	1392
12-10	527.59	528.58	0	0.19	1128
16-5	525.76	531.36	-9.62	-8.66	1233
16-6	876.26	879.23	-4.18	-3.86	1178
16-7	671.15	672.15	-4.65	-4.51	1500
16-8	619.35	628.24	0.03	1.46	989
16-9	1290.96	1373.93	1.52	8.05	985
16-10	1135.47	1151.64	1.41	2.85	1207

Table A.2. ALNS results for small instances with relaxed time windows and with two vehicles.

Instance	Best	Mean	% Dev. of Best from CPLEX	% Dev. of Mean from CPLEX	Total Sol. Time(s)
8-5	1832.78	1832.78	0	0	372
8-6	1450.71	1450.71	0	0	409
8-7	1259.94	1259.94	0	0	461
8-8	431.55	432.32	0	0.18	972
8-9	350.89	350.89	0	0	826
8-10	597.01	597.54	0	0.09	903
12-5	1458.47	1459.21	0	0.05	1223
12-6	1066.79	1082.87	-2.21	-0.74	1500
12-7	1154.67	1164.45	0.05	0.90	1500
12-8	899.04	912.54	-0.51	0.98	1329
12-9	894.03	901.60	-0.76	0.08	1067
12-10	776.76	785.99	-0.72	0.46	1461
16-5	681.80	716.63	-3.78	1.14	1500
16-6	1485.83	1504.40	-4.15	-2.95	1500
16-7	1192.26	1199.79	0.46	1.09	1500
16-8	915.86	924.97	3.51	4.54	1500
16-9	1540.58	1551.57	0	0.71	1500
16-10	1407.84	1460.02	-3.24	0.35	1500