



MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES



**A HYBRID METAHEURISTIC METHOD TO
TRAVELING SALESMAN PROBLEM WITH
DRONE**

NOYAN SEBLA SEZER

Ph.D. THESIS

Department of Industrial Engineering

Thesis Supervisor

Prof. Dr. Serol BULKAN

Thesis CO-Supervisor

Assist. Prof. Dr. Emre ÇAKMAK

ISTANBUL, 2023



MARMARA UNIVERSITY
INSTITUTE FOR GRADUATE STUDIES
IN PURE AND APPLIED SCIENCES



**A HYBRID METAHEURISTIC METHOD TO
TRAVELING SALESMAN PROBLEM WITH
DRONE**

NOYAN SEBLA SEZER

(724413004)

Ph.D. THESIS

Department of Industrial Engineering

Thesis Supervisor

Prof. Dr. Serol BULKAN

Thesis CO-Supervisor

Assist. Prof. Dr. Emre ÇAKMAK

ISTANBUL, 2023

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to whom have been instrumental in the completion of this thesis.

First and foremost, I am deeply grateful to my supervisor, Prof. Dr. Serol Bulkan, for his guidance, expertise, and unwavering support throughout the entire research process. I am truly fortunate to have had the opportunity to work under his mentorship.

I would also like to extend my heartfelt thanks to my co-supervisor, Assist. Prof. Dr. Emre akmak. His insightful suggestions, technical support and constant encouragement have played a significant role in refining my research.

I would like to express my sincere appreciation to the member of my thesis committee, Prof. Dr. iğdem Alabaş Uslu. Her valuable feedback, critical evaluation and suggestions have greatly contributed to this work. I would also thank to my thesis committee member, Assist. Prof. Dr. Melike Öztürk for her contributions with valuable suggestions.

I am deeply grateful to my family for their unwavering love, patience, and encouragement throughout my academic journey. I owe a special debt of gratitude to my son, Tibet, for his understanding and for providing me with moments of joy and inspiration amidst the challenges of completing this thesis.

Last but certainly not least, I would like to express my deepest appreciation to my mother. Her unwavering support, her sacrifices, and her unconditional love have been the driving force behind my entire academic pursuits. Without her support, this accomplishment would not have been possible.

AUGUST, 2023

NOYAN SEBLA SEZER

TABLE OF CONTENTS

ACKNOWLEDGEMENT.....	i
TABLE OF CONTENTS	ii
ÖZET	v
ABSTRACT	vii
CLAIM FOR ORIGINALITY	ix
SYMBOLS	x
ABBREVIATIONS	xii
LIST OF FIGURES	xiii
LIST OF TABLES	xv
1. INTRODUCTION.....	1
1.1 Use of Drones in Last-Mile Delivery	1
1.2 Overview of Traveling Salesman Problem with Drone.....	2
1.2.1 The problem description and mathematical formulation of TSP-D	5
1.3 Literature Review on TSP-D and Solution Methods.....	8
1.4 Objective of the Thesis	15
2. OVERVIEW OF THESIS METHODS.....	19
2.1 Genetic Algorithm	19
2.1.1 Encoding.....	23
2.1.2 Genetic population	24
2.1.3 Fitness evaluation	25
2.1.4 Reproduction	25
2.1.4.1 Selection.....	26
2.1.4.2 Crossover	27
2.1.4.3 Mutation	28

2.2	Ant Colony Optimization Algorithm.....	29
2.2.1	Solution construction in ACO	30
2.2.2	Pheromone update in ACO.....	32
3.	THE HYBRID METAHEURISTIC METHOD TO TRAVELING SALESMAN PROBLEM WITH DRONE.....	35
3.1	The Genetic Algorithm with Ant Search-Based Solution Method to TSP-D	35
3.1.1	Supportive structure for algorithm	37
3.1.2	Solution representation in genetic algorithm.....	38
3.1.3	Ant-Search algorithm	39
3.1.3.1	Solution representation in the ant-search algorithm.....	40
3.1.3.2	Two-pheromone framework of the ant-search algorithm	40
3.1.4	Route construction in the GA-AS algorithm.....	41
3.1.4.1	Restrictions.....	41
3.1.4.2	Selection process	42
3.1.5	Evaluation.....	52
3.1.6	Pheromone update	53
3.1.7	Parent selection.....	54
3.1.8	Crossover and mutation.....	55
4.	COMPUTATIONAL EXPERIMENTS	57
4.1	Parameter Setting.....	57
4.2	Computational Experiments on TSP-D Instances	60
4.2.1	Results of GA-AS algorithm for optimal TSP-D solutions.....	61
4.2.2	Comparison results of GA-AS algorithm with rival algorithms	62
4.3	Computational Experiments on TSP Instances.....	66
5.	DISCUSSION	71
6.	CONCLUSION	75
	REFERENCES	79

APPENDIX	85
CURRICULUM VITAE	89

ÖZET

İHA'LI GEZGİN SATICI PROBLEMİ İÇİN BİR MELEZ METASEZGİSEL YÖNTEM

İnsansız Hava Araçları (İHA) teknolojisinin son yıllardaki hızla gelişimi ve lojistik süreçlerde kullanımının önerilmesi ile İHA' lı Gezgin Satıcı Problemi (GSP-İHA) önemli bir araştırma alanı olarak ortaya çıkmıştır. Bu tezde ele alınan GSP-İHA, bir yer aracı ile bir hava aracının dağıtım sistemlerinde entegre kullanılmasıyla geleneksel gezgin satıcı probleminin bir uzantısı olan, İHA destekli teslimat rotasını optimize etmeyi amaçlayan yeni bir kombinatoriyal optimizasyon problemidir. Tez kapsamında, bu problemin çözümü için metasezgisel bir yöntem geliştirilerek optimizasyon alanına katkı sağlamak ve İHA destekli teslimat süreçlerinde benzer problemlerin çözümü için de görüş oluşturabilmek amaçlanmıştır. Bu amaç ile hibrit bir metasezgisel algoritma olan Karınca Araması Temelli Genetik Algoritma (KA-GA) çözüm yöntemi önerilmiştir. KA-GA algoritması, mevcut gelişmiş metasezgisel yöntemlerden Genetik Algoritma (GA) ve Karınca Kolonisi Optimizasyonu (KKO) algoritmasının, GSP-İHA için uyarlanmış hibrit bir uygulamasıdır. KA-GA algoritması, GA' nın standart optimizasyon adımlarına, geleneksel KKO algoritmasından esinlenen bir arama prosedürü olan Karınca Araması (KA)' nı dahil etmektedir. KA-GA algoritması, rota kurma sürecini problem özgü özellikleriyle birleştirerek, GA ile üretilen bir çözümün KA prosedürü aracılığıyla tam bir GSP-İHA turu olarak tamamlanmasına olanak sağlamaktadır. KA, karıncaların davranış metaforunu taklit ederek, müşteri noktalarını GSP-İHA turu olarak sıralama işlevini yerine getiren stokastik süreçli bir algoritmadır. Ayrıca, önerilen algoritma, klasik KKO 'nun feromon yapısından farklı bir prosedür kullanmaktadır. Bir GSP-İHA çözümünde iki araç tipi için, farklı feromon yapıları gerektiren iki tür alt rota ortaya çıktığı gözlemlendiğinden, KA-GA algoritması için ikili-feromon yapısı geliştirilmiştir. Gerçekleştirilen deneyler, GSP-İHA formülasyonunun geleneksel araç rotalamasına göre avantajlarını ve önerilen KA-GA algoritmasının GSP-İHA' yı etkili bir şekilde çözebilme yeteneğini destekleyen kanıtlar sunmaktadır. Sayısal sonuçlar, önerilen

algoritmanın test problemlerinin optimum çözümlerini tutarlı olarak elde ettiğini, belirli rakip algoritmalarından daha iyi performans gösterdiğini ortaya koymuştur ve geleneksel rotalama problemine İHA entegre etmenin faydalarını göstermiştir. Bu tez, yeni hibrit bir metasezgisel algoritma önererek GSP-İHA alanına katkı sağlamaktadır.

ABSTRACT

A HYBRID METAHEURISTIC METHOD TO TRAVELING SALESMAN PROBLEM WITH DRONE

The Traveling Salesman Problem with Drone (TSP-D) has emerged as a prominent research area in recent years, driven by the rapid advancements in drone technology and by the idea of utilizing drones in delivery. This thesis addresses the TSP-D, a relatively new combinatorial optimization problem that extends the traditional traveling salesman problem by introducing a drone that can delivery parcels to customers in addition to a ground vehicle and which seeks to optimize that drone-aided delivery route. This thesis aims to contribute to the field of optimization and provide insights for solving similar problems in domains of drone aided delivery processes. For this contribution, the thesis proposes a novel hybrid metaheuristic algorithm, the Genetic Algorithm with Ant Search-Based Solution Method (GA-AS). The GA-AS algorithm is based on the implementation of the current state-of-the-art metaheuristic algorithms, Genetic Algorithm (GA) and Ant Colony Optimization (ACO), by hybridizing and customized them for TSP-D features. The GA-AS framework follows the standard optimization steps of the GA and incorporates a new solution searching procedure, Ant Search, which draws inspiration from the traditional ACO algorithm. The GA-AS algorithm modifies the route construction process by incorporating problem-specific features, allowing a solution generated in the GA stage to be transformed into a complete TSP-D tour through the AS stage. The AS algorithm is a stochastic process imitates the ant behavior metaphor to construct tours which responsible for selecting customer nodes to be sequenced in the TSP-D tour. Besides, the proposed algorithm employs a unique procedure on the pheromone formulation, distinct from the pheromone structure of ACO. Binary-pheromone framework is discussed to be introduced in the GA-AS algorithm, since a TSP-D solution involves two types of sub-routes emerged with requiring separate pheromone frameworks for both truck and drone. The computational experiments conducted in this thesis provide substantial evidence supporting the advantages of the

TSP-D formulation and the capability of the GA-AS algorithm to effectively tackle the TSP-D. The algorithm consistently produced optimal solutions of benchmarking problems, outperformed certain rival algorithms, and demonstrated the benefits of integrating a drone into traditional truck travel. This research contributes to the field of TSP-D by introducing a novel hybrid metaheuristic algorithm.

CLAIM FOR ORIGINALITY

The considered problem in this thesis, TSP-D, has been still addressed by a limited number of studies since it is existed very recently in the literature. This thesis presents a novel approach for solving the TSP-D, the hybridization of GA and ACO, as they have not been previously explored for this problem. The research is also having its novelty through some other perspectives:

- o In the majority of TSP-D studies, classical heuristic approaches take precedence in solution while the exploration of metaheuristics in this context has been relatively limited. This thesis presents a novel approach with contributing to metaheuristic solution methodologies of TSP-D.
- o The majority of heuristic algorithms discussed in the TSP-D literature address the problem through a two-staged process. They begin with the construction of a traditional TSP tour then finalize it as a TSP-D tour commonly rely on neighborhood search procedures. In contrary, the proposed GA-AS constructs and optimizes truck and drone routes in a single stage. A method that simultaneously constructs routes has not been presented yet in the literature. The approach proposed in this thesis does not involve neighborhood search and two-staged solution construction as already documented in existing literature, consequently it fulfills the mentioned gap.
- o A lack of research on the implementation of ACO for the TSP-D has been inferenced. To the best of our knowledge, an ACO based algorithm is the first time implemented and presented to solve TSP-D in the literature.
- o A novel binary-pheromone structure of ACO algorithm is presented that can lead to implementation for future relevant problems.
- o Computation experiments indicated that the proposed GA-AS algorithm improved upon the certain best-known solutions as can be used to in further benchmarking.

This research supports the proposed hybrid metaheuristic approach as a novel contribution to the advancement of TSP-D problem-solving methodologies.

SYMBOLS

v_0	: depot
v_i	: i . customer location ($i=1, \dots, N$)
c^t	: travel time of drone
c^d	: flight time of drone
V	: set of customer locations
T	: truck tour
D	: drone tour
O	: set of operations
c_o	: cost of operation
o_k	: operation k
T_k	: truck subtour in operation k
D_k	: drone subtour in operation k
$c^t(T_k)$: travel time of truck subtour in operation k
$c^d(D_k)$: flight time of drone subtour in operation k
$t(o_k)$: completion time of operation k
$t(T, D)$: completion time of TSP-D tour
$p(i, j)$: probability of selection
P_{ij}	: pheromone level
n_{ij}	: heuristic value
N_i	: set of neighborhoods of i
α	: control parameter in ACO
β	: control parameter in ACO
L_k	: length of tour

- e : evaporation rate
- P_t : truck pheromone matrix
- P_d : drone pheromone matrix
- $P_{t_{ij}}$: pheromone amount on truck path ij
- $P_{d_{ij}}$: pheromone amount on drone path ij
- v_{ij} : selection value for candidate path ij
- d_{ij} : Euclidian distance between node i and j
- S_t : truck speed
- S_d : drone speed
- T_C : truck cost matrix
- D_C : drone cost matrix
- f_k : fitness value of chromosome k
- $TCost_k$: total cost of TSP-D tour in chromosome k
- $p(f_k)$: fitness probability of chromosome k

ABBREVIATIONS

ACO	: Ant Colony Optimization
AS	: Ant Search
FSTP	: Flying Sidekick-Traveling Salesman Problem
GA	: Genetic Algorithm
GA-AS	: The Genetic Algorithm with Ant Search-Based Solution Method
GRASP	: Greedy Randomized Adaptive Search Procedure
HCLS	: Hill-Climbing Local Search
HGA	: Hybrid Genetic Algorithm
HGVNS	: Hybrid General Variable Neighborhood Search
ILP	: Integer Linear Programming
MILP	: Mixed Integer Linear Programming
LS	: Local Search
NP-Hard	: Non-Deterministic Polynomial-Time Hardness
mFSTP	: Multiple Flying Sidekick-Traveling Salesman Problem
mTSP-mD	: Multi Drop Traveling Salesman Problem with Multiple Drone
PDSTSP	: Parallel Drone Scheduling Traveling Salesman Problem
SA	: Simulated Annealing
TSP	: Traveling Salesman Problem
TSP-D	: Traveling Salesman Problem with Drone
TSPDP	: Traveling Salesman Problem with Drone and Parking
TSP-mD	: Traveling Salesman Problem with Multiple Drone
UAV	: Unmanned Air Vehicle
VRP-D	: Vehicle Routing Problem with Drone

LIST OF FIGURES

Figure 1.1: (a) a TSP tour (b) a TSP-D tour	4
Figure 1.2: An operation example in TSP-D.....	6
Figure 2.1: Illustrations to some genetic encoding types in GA: (a) binary encoding, (b) value encoding, (c) permutation encoding	24
Figure 2.2: Illustration of real ants finding shortest path	29
Figure 3.1: Flow diagram of the GA-AS algorithm.....	36
Figure 3.2: Transportation Types appeared in a TSP-D route	38
Figure 3.3: An illustration of solution representation with binary encoding for the TSP-D	39
Figure 3.4: Solution representation in AS algorithm.....	40
Figure 3.5: An example to solution construction in GA-AS: stage 1	44
Figure 3.6: An example to solution construction in GA-AS: stage 2	44
Figure 3.7: An example to solution construction in GA-AS: stage 3	45
Figure 3.8: An example to solution construction in GA-AS: stage 4	45
Figure 3.9: An example to solution construction in GA-AS: stage 5	46
Figure 3.10: An example to solution construction in GA-AS: stage 6	46
Figure 3.11: An example to solution construction in GA-AS: stage 7	47
Figure 3.12: An example to solution construction in GA-AS: stage 8	47
Figure 3.13: An example to solution construction in GA-AS: stage 9	48
Figure 3.14: An example to solution construction in GA-AS: stage 10	48
Figure 3.15: An example to solution construction in GA-AS: stage 11	49
Figure 3.16: An example to solution construction in GA-AS: stage 12	49
Figure 3.17: An example to solution construction in GA-AS: stage 13	50
Figure 3.18: An example to solution construction in GA-AS: stage 14	50

Figure 3.19: An example to solution construction in GA-AS: stage 15	50
Figure 3.20: An example to solution construction in GA-AS: stage 16, 17, 18	51
Figure 3.21: Crossover operator in GA-AS	56
Figure 3.22: Mutation operator in GA-AS	56
Figure 4.1: S/N analysis results.....	59
Figure 4.2: illustrated TSP and TSP-D graphs of “eil51” problem.....	68

LIST OF TABLES

Table 1.1: A summary of the literature review on TSP-D	14
Table 4.1: Determined parameter levels for design of experiment	58
Table 4.2: Parameter Setting in GA-AS	59
Table 4.3: Numerical results of GA-AS on instances from Bouman et al. (2018b) with optimal solutions.	62
Table 4.4: Comparison results of GA-AS algorithm with rival algorithms on Bouman et al. (2018b) instances.....	64
Table 4.5: p -values obtained from Wilcoxon Signed Rank test.....	65
Table 4.6: Results of GA-AS algorithm on TSP instances	67

1. INTRODUCTION

1.1 Use of Drones in Last-Mile Delivery

Using Unmanned Air Vehicles (UAVs), commonly known as *Drones*, is a relatively new concept that has gained popularity over recent years due to the rapid advancements in drone technology.

The idea of using drones for delivery can be traced back to the early 2010s when drone technology began to advance rapidly, what have in fact led to increased use of drones in numerous business area. So that several companies started exploring the potential of using drones for commercial purposes. One of those areas is last-mile delivery systems in logistics. Drones offer several advantages over traditional delivery methods, such as speed, cost-effectiveness, and environmental friendliness.

Experiment with drone-based delivery in the last-mile logistics has first come along with the innovative announcement of the online retailer Amazon in 2013 on their plan to use drones to deliver packages to customers. The plan, called Prime Air, aimed to revolutionize the delivery industry by using autonomous drones to deliver packages directly to customers' doorsteps. Moreover, the idea began to draw attention at the international level since primary online retailers have declared their drone-integrated delivery processes such as Google's Project Wing, DHL's Parcelcopter project, UPS's Flight Forward project. These companies have invested heavily in drone technology and have conducted numerous tests and pilot programs to evaluate the feasibility of using drones for delivery.

The use of drones for delivery has also been driven by the rapid growth of e-commerce, which has increased the demand for fast and efficient delivery options. Drones offer a way to speed up the delivery process, reduce costs, and improve customer satisfaction.

One of the most significant benefits of using drones for delivery is speed. Drones can cover long distances quickly and efficiently, bypassing traffic and other obstacles that can slow down ground-based vehicles. This makes them ideal for urgent deliveries, such as medical supplies, where time is of the essence.

Another advantage of using drones for delivery is cost-effectiveness. Drones are relatively inexpensive to operate, and they require minimal infrastructure, making them an attractive option for companies looking to reduce their delivery costs. Moreover, drones can be used to access

hard-to-reach areas, such as remote or mountainous regions, where ground-based vehicles may be unable to deliver.

Finally, using drones for delivery is environmentally friendly, as they produce fewer carbon emissions compared to traditional delivery methods. This makes them a more sustainable option for companies looking to reduce their carbon footprint and contribute to environmental conservation efforts.

Much as these advantages, there are still several challenges associated with using drones for delivery, such as safety, regulations, and technical limitations. For example, drones must navigate around obstacles such as buildings and power lines, avoid collisions with other drones, and ensure the safety of people on the ground. Additionally, there are various regulations that govern the use of drones, such as flight restrictions, altitude limits, and licensing requirements.

Despite these challenges, the potential benefits are significant. The use of drones for delivery is becoming increasingly popular, and companies are investing heavily in drone technology to improve their delivery operations. The idea of using drones for delivery is still relatively new, and the technology is still in the early stages of development. As drone technology continues to evolve and become more advanced, it is expected that the use of drones for delivery will become more widespread and mainstream in the coming years.

1.2 Overview of Traveling Salesman Problem with Drone

In logistics, transportation, and delivery services, drones can be used to speed up the delivery process. Since the use of drones for delivery and transportation increases, optimizing the routing of deliveries and reducing transportation costs has become a critical concern for companies and organizations. Thus, this challenging idea in delivery has also come about a necessity to address a new routing problem. Consequently, this routing problem has made an attention to academicians in the fields of mathematics, computer science, and engineering.

The routing problem of drone-aided delivery has commonly referred to “*The Traveling Salesman Problem with Drone (TSP-D)*.” Murray & Chu (2015) introduced the first study on this concept sparked by such a drone delivery process which is The Flying Sidekick-Traveling Salesman Problem (FSTSP); whereas Agatz et al. (2018) presented TSP-D for the first time, which is very similar to FSTSP.

The TSP-D is a combinatorial optimization problem introducing a drone that can fly between customers, in addition to the ground vehicle (truck) which travels by road. The objective is to

find the shortest possible route for the truck and drone to collectively visit all the customers to deliver parcels and return to the starting point. The TSP-D offers a way to address optimizing routes both for truck and drone to minimize delivery times and reduce transportation costs.

The TSP-D problem is typically modeled as a graph, where nodes represent customers, and edges represent the distances between them. The drone can fly over a subset of the edges, skipping some of the customers, and reducing the distance traveled by the truck.

The TSP-D is a relatively new variant of the traditional Traveling Salesman Problem (TSP), which has emerged with the rise of unmanned aerial vehicles or drones. TSP-D extends traditional TSP through a drone is added to the problem setting.

The traditional TSP has been studied by mathematicians since the early 19th century, and it remains an important problem in combinatorial optimization. The problem was first introduced in a mathematical form in 1930 by the Irish mathematician W.R. Hamilton and the British mathematician Thomas Kirkman. The TSP has since been extensively studied, and numerous algorithms have been developed to solve it. In the traditional TSP, a salesman/vehicle needs to visit a set of cities to serve, each only once, and return to the starting city, while minimizing the total distance traveled. In TSP-D, the salesman has access to a drone that can carry some of the load and perform some of the deliveries.

Figure 1.1 compares a traditional TSP solution to a TSP-D solution. As demonstrated in the TSP-D tour representation, serving some customer locations with a drone rather than a truck can reduce the overall distance a truck must travel. The duration of the tour can be shortened with this parallelization of drone delivery tasks.

The TSP-D problem is also known to be NP-hard as like TSP. However, TSP-D also involves a decision-making task on which customer nodes will be served by which vehicle (truck or drone) in addition to optimizing subtours of drone and truck while optimizing duration of complete tour. That builds up the complexity of the problem.

Since TSP-D is NP-hard, finding the optimal solution for large problem instances is computationally infeasible. Therefore, researchers try to develop heuristics to find good solutions in a reasonable amount of time.

The TSP-D problem is a challenging optimization problem, as it may involve multiple objectives and constraints, such as limited drone battery life and the need to avoid no-fly zones. This has made it an attractive research topic for researchers looking to develop new algorithms and optimization techniques. The TSP-D problem also presents several research opportunities for academicians to explore different aspects of the problem, such as the impact of different drone characteristics on delivery times, the effect of different route planning strategies on transportation costs, and the trade-offs between different objectives in the problem.

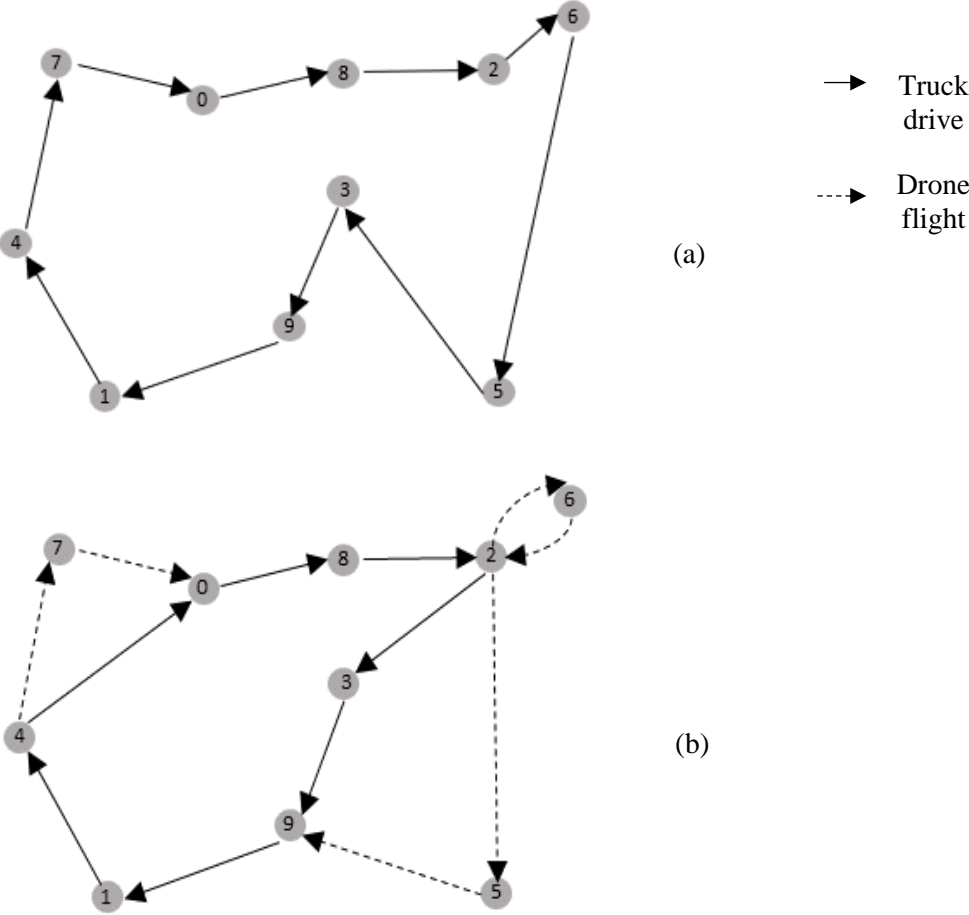


Figure 1.1: (a) a TSP tour (b) a TSP-D tour

Consequently, the practical importance of the TSP-D problem and its challenging nature makes it a popular research topic among academicians. As drone technology continues to advance, it is likely that research on the TSP-D problem will continue to grow and contribute to the development of more efficient and effective drone delivery system.

1.2.1 The problem description and mathematical formulation of TSP-D

In this section, thesis problem is defined, and an integer programming (IP) formulation of the problem as well is provided based on the original formulation in Agatz et al. (2018).

A graph $G = (V, A)$ can be used to model the TSP-D, with node v_0 indicates the depot and N number of nodes $v_1 \dots v_N$ correspond to customer's locations. Let $c^t(a) = c^t(v_i, v_j)$ is the travel time of truck, $c^d(a) = c^d(v_i, v_j)$ is representing flight time of drone between nodes v_i and v_j . There is a general assumption that drone is faster than truck as $c^d(a) \leq c^t(a)$. This is because the drone might be able to take shortcuts rather than following the road system. The objective of the problem is to get min-cost tour, whereas costs are specified in terms of travel times. Let $V_t \subseteq V$ be the set of nodes that must be delivered by a truck, since some of these cannot be delivered by drone.

TSP-D in its origin has the following assumptions:

1. After each delivery, the drone must return to the truck since it has a limited capacity. At each drone sortie consist of one-unit delivery to only one customer node.
2. Drone can only land on truck and take off from truck. Moreover, drone can only pick up parcels for its next delivery from the truck only if the truck is being parked at a customer location or the depot.
3. the problem assume that parcel pick-up and parcel delivery times are both neglected for both drone and truck. The drone's recharging time, for instance, switching out batteries is also assumed to be neglected.

Nodes in the problem should be first distinguished according to types as following:

- *Drone node*: is a node that delivered by drone alone.
- *Truck node*: is a node that delivered by truck alone.
- *Combined node*: is a node that visited by both vehicles.

A TSP-D solution contains a truck tour $T = (t_0 = v_0, t_1, \dots, t_n = v_0)$ with a drone tour $D = (d_0 = v_0, d_1, \dots, d_m = v_0)$ with every node $v \in V^t$, t is required to belong T .

To compute the completion time of a TSP-D tour is not only summation of truck and drone driving times in truck tour T and drone tour D , since each vehicle must be synchronized. Therefore, it is advantageous to introduce the idea of an *operation*.

A single drone node, at most one truck node, and two combined nodes, known as the beginning node and ending node, make up an operation k . If there is a drone node in the operation, the drone leaves the truck at the beginning node, serves the drone node, and then returns to the truck at the ending node. The truck can visit any number of truck nodes between the beginning node and the ending node, or it can travel directly between them. In the case of there is not a drone node in the operation k , there is only one arc connecting the start node and end node. Meanwhile, the drone remains parked on top of the truck as it travels from beginning node to ending node.

So that an operation is defined by subtours (T_k, D_k) . T_k is a subtour sequencing of truck tour T beginning from the k th combined node of T continues till the $(k + 1)$ th combined node of T . D_k is a subtour sequencing of drone tour D beginning from the k th combined node of D continues till the $(k + 1)$ th combined node of D .

$c^t(T_k) := \sum_{a \in T_k} c^t(a)$ is denoted as travel time of truck (cost of truck subtour) on the subtour T_k while $c^d(D_k) := \sum_{a \in D_k} c^d(a)$ is denoted to drone flight time (cost of truck subtour) on the subtour D_k . To compute the total time of a solution (T, S) is required to discuss (T, S) as a sequence of operations in the manner defined above. With the assumption of drone speed is at least equals to truck's speed, the completion time an operation o_k is calculated according to $t(o_k) = \max\{c^t(T_k), c^d(D_k)\}$.

Eventually, the time required to deliver all nodes in a TSP-D tour (T, D) is computed by Equation 1.1.

$$t(T, D) := \sum_{k \in K} t(o_k) \tag{1.1}$$

To exemplify that time computation, an operation is illustrated in Figure 1.2.

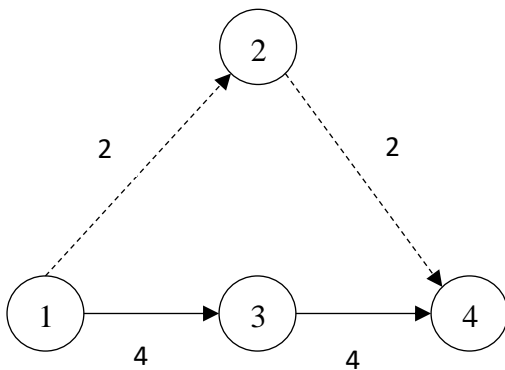


Figure 1.2: An operation example in TSP-D

The operation provided in Figure 1.2 contains node 1 as the beginning node as well as a combined node, node 2 as a drone node, node 3 as a truck node and node 4 as a combined node. Arcs showed with dashed lines represent drone flight path while with solid ones represent truck travel path. Each number given through the arcs indicates the required time to take the path. Consequently, to compute the total completion time of the illustrated operation is $\max(4, 8) = 8$.

Subsequently, an integer programming (IP) formulation of the TSP-D can be defined as follows. Let O represents the set of all possible operations with c_o designating the operation's costs for $o \in O$. As well as it can be easily considered some possible restrictions such as maximum flight time of drone, or set of nodes restricted to be served by drone V_t etc. The decision variable x is equal to 1 ($x_o = 1$) when the operation o is selected, otherwise is equal to 0 ($x_o = 0$).

Assume that $O^-(v) \subset O$ indicates operations set contains the beginning node v whereas $O^+(v) \subset O$ indicates the operations set contains the ending node v , so that $O(v) \subset O$ represents all operations set with node v . Assume an adjuvant variable y_v represents node v 's status chosen as a beginning node in at least one operation. So that, an IP formulation of the problem can be given as followingly:

$$\min \sum_{o \in O} c_o x_o \quad (1.2)$$

$$\text{such that } \sum_{o \in O^-(v)} x_o \geq 1 \quad \forall v \in V \quad (1.3)$$

$$\sum_{o \in O^+(v)} x_o \leq n \cdot y_v \quad \forall v \in V \quad (1.4)$$

$$\sum_{o \in O^+(v)} x_o = \sum_{o \in O^-(v)} x_o \quad \forall v \in V \quad (1.5)$$

$$\sum_{o \in O^+(S)} x_o \geq y_v \quad \forall S \subset V \setminus \{v_o\}, v \in S \quad (1.6)$$

$$\sum_{o \in O^+(v_o)} x_o \geq 1 \quad (1.7)$$

$$y_{v_o} = 1 \quad (1.8)$$

$$x_o \in \{0,1\} \quad \forall o \in O \quad (1.9)$$

$$y_v \in \{0,1\} \quad \forall v \in V \quad (1.10)$$

Equation (1.2) is the objective function of the problem that to minimize cost of the tour, can be computed as the sum of the selected operations' cost. Equation (1.3) provides the constraint of the requirement on every node to be covered. If at least one selected operation uses v as a beginning node, y_v should be set to 1 due to constraint in Equation (1.4). The left-hand side of Equation (1.4) has a maximum value of n since at least one node that is not visited before has to be contained in each operation.

Equation (1.5) -(1.7) provide the constraint of the selected operations $O' := \{o \in O: x_o = 1\}$ cross an Eulerian graph G' as a cycle of that graph indicates a feasible (T, D) route. A G' graph is defined as the graph (V', A') where $V'O := \{v \in V: y_v = 1\}$ and $a' = (v_i, v_j)$ that each operation $o \in O'$, beginning node v_i and ending node v_j . According to Equation (1.6) and (1.7), G' is connected and it can be seen that it is Eulerian according to Equation (1.5). By computing an Eulerian cycle in G' , a truck-drone route (T, D) can be obtained by IP solution. Equation (1.8) provides the beginning and ending of the route at the depot. The variables x_o and y_v are restricted to be binary by Equation (1.9) and (1.10).

1.3 Literature Review on TSP-D and Solution Methods

This section reviews the literature of the thesis problem with aiming to provide an overview of research conducted on TSP-D, highlighting the approaches, algorithms, and contributions made by researchers in this field. By examining the existing literature, this review aims to identify the current state of the art solution methodologies, key challenges, and potential future research directions for TSP-D.

In this study, the primary focus is on examining the cooperation between a single truck and a single drone, with the restriction of only one visit per flight. Consequently, an extensive review for the concerning papers in this particular area is conducted. Nonetheless, a brief overview of other variations of the problem is also provided.

TSP-D refers the general type of the cooperative truck-drone delivery problem which is first introduced by Agatz et al. (2018). However, this truck-drone tandem delivery problem

commonly varies in terms of the number of vehicles, besides assumptions and limitations.

Depending on the assumptions and limitations of the problem, it can be categorized as followingly. It is first introduced by Murray & Chu (2015) as a routing problem that incorporates both trucks and drones. They propose two novel variations called the Flying Sidekick Traveling Salesman Problem (FSTSP) and the Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP).

The primary distinction between TSP-D and FSTSP lies in TSP-D' s allowance for multiple visits to a customer node to ensure drone meeting with truck, whereas FSTSP restricts trucks and drones to visiting each customer node only once. So that, the only difference between these two variants is having different assumptions for customer visiting.

The PDSTSP, allows for customers to be served either by a single truck or by a fleet of identical drones. The main goal is to minimize the makespan. Unlike the first model, the PDSTSP does not involve cooperation or synchronization between the drones and the truck. Drones begin and end their routes at the depot, serving one customer at a time, and have the flexibility to make multiple trips, departing from the depot more than once if needed (Dell'Amico et al., 2020; Montemanni & Dell'Amico, 2023; Kim & Moon, 2019; Schermer et al., 2020; Nguyen et al., 2022; Mbiadou Saleu et al., 2018).

Depending on the number of vehicles involved in this collaborative delivery problem, it can be mainly categorized followingly. The problem where a single-truck and single-drone utilized in delivery is typically referred to TSP-D. On the other hand, when a single truck collaborates with multiple drones, it is known as Traveling Salesman Problem with Multiple Drones (TSP-mD) (Poikonen & Golden, 2020; Luo et al., 2021; Cavani et al., 2021; Karak & Abdelghany, 2019; Ferrandez et al., 2016; Moshref-Javadi et al., 2020; Phan et al., 2018; Salama & Srinivas, 2022; Li et al., 2020).

FSTSP also considers delivery with a single truck and single drone, however a variant of it presented as mFSTSP when multiple drones with a single truck involves the cooperation with the same assumption of the problem (Murray & Raj, 2020; Dell'Amico et al., 2021).

Another problem type, the Vehicle Routing Problem with Drones (VRP-D), occurs dealing with the routing problem when multiple trucks are employed alongside multiple drones (Poikonen et al., 2017; Ham, 2018; Schermer et al., 2019; Tamke & Buscher, 2021; Wang & Sheu, 2019; Euchli & Sadok, 2021; Sacramento et al., 2019).

For a comprehensive literature review on drone-aided routing problems and their diverse variants, interested readers are directed to the surveys conducted by Macrina et al., 2020; Otto et al., 2018; Khoufi et al., 2019 and Madani & Ndiaye, 2022.

Hereby, a review is conducted in the field of TSP-D, specifically, the solution approaches available in the literature. The existing literature on TSP-D solutions can be categorized into two main groups: exact methods and heuristic/metaheuristic methods.

Exact methods employ integer linear programming (ILP) formulations to solve the problem and generate optimal solutions, but they are only applicable to instances of limited size. Since the problem is known to be NP-hard, heuristic/metaheuristic methods have been introduced in recent literature to obtain reasonably good solutions for larger instances.

Two seminal works by Murray & Chu (2015) and Agatz et al. (2018) introduced the concepts of FSTSP and TSP-D laying the foundation for subsequent research on drone-assisted truck delivery problems. The authors also presented the mathematical formulations for these problems. In the related study, Murray & Chu (2015) proposed a mixed ILP formulation capable of solving problem instances with up to 10 customers, albeit requiring significant computational time. To improve solution quality, they also devised a heuristic approach and provided benchmark instances for evaluation. Furthermore, Agatz et al. (2018) developed an ILP formulation that achieved a solving capability of up to 10 customers within a one-hour timeframe. They additionally introduced two heuristics based on local search techniques and presented new benchmark instances for performance assessment.

Bouman et al. (2018a), the authors of first presented TSP-D, extended their work by presenting a dynamic programming model solvable for larger instances. Es Yurek & Ozmutlu (2018) proposed an iterative optimization algorithm based on decomposition giving a better computational time. The approach was evaluated on instances involving a maximum of 20 customers. They concluded that only instances with up to 12 customers are only solved optimally. Roberti & Ruthmair (2021) put forward a MILP approach along with a Branch-and-Price method. Additionally, they carried out a compelling computational analysis to compare how the side constraints impact the algorithm's overall performance and the resulting solutions. Vásquez et al. (2021) presented an alternative approach based on MILP. The authors leveraged the problem's inherent structure and divide it into two distinct decision stages: the construction of truck routes and the design of drone operations. Furthermore, they introduced a Benders-type decomposition algorithm which is able to solve instances involved up to 15 customers and were

subjected to a time limit of 2 hours. El-Adle et al. (2021) presented a mathematical formulation with showing that it is able to solve 16 customers optimally. Schermer et al. (2020) proposed two innovative ILP formulations and a third formulation that involves an exponential number of constraints. To solve this third formulation, they employed a Branch-and-Cut algorithm. The results indicated that the third formulation, outperforms the others. It successfully solves multiple instances with up to 19 customers within a time limit of 1 hour. The literature on exact solutions is relatively limited compared to the heuristics available. However, there are some notable studies (Boccia et al., 2021; Dell’Amico et al., 2021a; Cavani et al., 2021; Di Puglia Pugliese et al., 2020; Zhen et al., 2023) that can be cited as contributing to the advancement of exact solution approaches for the TSP-D.

In the majority of TSP-D studies, heuristic approaches take precedence. Due to the computational limitations of exact solutions, heuristic approaches are frequently employed to obtain feasible solutions for larger instances of TSP-D. Heuristics can be categorized into classical heuristics and metaheuristics. In almost the reviewed literature on TSP-D, classical heuristic algorithms have been widely utilized. The common solution approach is that they are two-staged algorithms involve route construction and route improvement stages. One notable example is the first proposed TSP-D heuristic by Agatz et al. (2018) which follows a route-first, cluster-second approach. Initially, a truck route is constructed without considering drones, often utilizing The Concorde solver to find a traditional TSP tour. Subsequently, drone nodes are determined using neighborhood search heuristics in most cases. Followingly, the other heuristic solutions proposed for TSP-D in literature will be presented.

In his thesis, Ponza (2016) utilized a heuristic method to tackle the FSTSP. This approach served as an extension to the research conducted by Murray and Chu (2015) and was based on the implementation of simulated annealing. Ha et al. (2018) focused on a distinct objective of the problem, which aimed at minimizing operational costs, including the waiting time of the truck for the drone. The mathematical formulation is first given to the introduced problem called min-cost TSP-D and they proposed two heuristic approaches: TSP-LS and GRASP. TSP-LS was adjusted from the method recommended in Murray and Chu (2015) that involved employing local search techniques after obtaining an optimal truck route as a TSP tour. The second one, GRASP based on a new split algorithm that selected drone nodes to establish a TSP-D solution during the construction phase, based on a TSP tour. In generating a giant TSP tour, three heuristics were used where k-nearest outperforms k-cheapest and random insertion. Through computational tests, they found that GRASP outperformed TSP-LS in terms of solution quality

within a reasonable run time. Marinelli et al. (2018) made modifications to the GRASP approach recommended by Ha et al. (2018), taking into account a new assumption that allowed the drone to launch and rendezvous with the truck along an arc rather than solely at a customer node.

Freitas & Penna (2020) introduced a heuristic method called HGVNS (Hybrid Greedy Variable Neighborhood Search). This approach employed a neighborhood search technique to incorporate drone deliveries into the solution after obtaining an optimal TSP tour using a MILP solution. The described algorithm includes an exact model for the initial solution and a heuristic algorithm for improvement. They utilized instances set adopted from well-known TSPLIB to evaluate their algorithm. Almuhaideb et al. (2021) adopted a similar route-first, partition-second procedure. They proposed a GRASP algorithm that incorporated two variants as local search procedures: hill-climbing (HCLS) and simulated annealing (SA). These variants were utilized to further enhance the quality of the solutions obtained by the GRASP approach. The proposed heuristic was compared with two state-of-art algorithms: LS (Agatz et al., 2018) and HGVNS (Freitas & Penna, 2020) and concluded by getting comparable performance to existing methods in terms of tour length. As well, they resulted from the HCLS variant performing better than the SA variant.

In their study, Gonzalez-R et al. (2020) presented a greedy heuristic that operated through an iterative procedure, utilizing simulated annealing. This heuristic allowed for multiple drone visits between consecutive rendezvous nodes, which was a novel aspect not previously considered. Additionally, they took into account battery limitations, which had not been formulated in previous approaches. They stated that the proposal applies to solving real-life problems for drone routing. Baniasadi et al. (2020) examined the clustered generalized TSP, an extended variation of the traditional TSP, and put forward a transformation model. They further adjusted their model to address the specific challenges of drone-assisted delivery problems. Their approach involved partitioning the problem nodes into clusters and subclusters. Within each subcluster, the truck would visit only one node, while drones would visit all the other nodes. This distinctive solution method presented a departure from conventional approaches to the problem.

Most TSP-D studies primarily focus on heuristic approaches, while the exploration of metaheuristics in the context of TSP-D has been relatively limited. The reviewed literature predominantly consists of heuristic methods, with fewer studies delving into the application of metaheuristic algorithms to address TSP-D. These currently existing are reviewed followingly.

In their work, Ha et al. (2020) presented a hybrid genetic algorithm called HGA (Hybrid Genetic

Algorithm) that combined the use of a genetic algorithm with a local search procedure to address the TSP-D. This algorithm aimed to optimize two objectives: minimizing cost and minimizing time. They incorporated an education step into the genetic algorithm to enhance solution quality by exploring different neighborhoods. The proposed HGA includes a set of 16 local search operators for that education step. They also presented a new crossover operator which also evaluated four well-known crossover operators frequently used in permutation problems. They conducted a benchmarking on instances of Murray & Chu (2015) and Ha et al. (2018). Through their experiments, they demonstrated that HGA outperformed their previously proposed GRASP algorithm as well as other existing methods in terms of solution quality. Ferrandez et al. (2016) also employed a genetic algorithm to address the TSP-D problem. In their work, the genetic algorithm was utilized to find an optimized TSP route for the truck, while the K-means algorithm was employed to determine the launch nodes for the drones. Lastly, Tong et al. (2022) also explored a metaheuristic approach by modifying the conventional tabu search algorithm. Their modification involved altering the neighborhood structure of the tabu search, allowing for an expanded search range. Furthermore, their model took into account the waiting time of trucks at the launch nodes, providing a comprehensive perspective on the TSP-D problem. A summary of the literature review, particularly by presenting the existing solution methodologies of the problem is given in Table 1.1.

Based on the findings from the reviewed literature, it is evident that heuristic approaches have garnered significant attention in TSP-D studies. This preference for heuristics can be attributed to the substantial computational time required by exact solutions, such as ILP formulations. Researchers aim to generate solutions for larger instances within acceptable computational limits, even if they may not be optimal. In contrast, the exploration of metaheuristic algorithms in TSP-D studies has been relatively limited compared to classical heuristics.

Another notable observation is that the majority of heuristic algorithms discussed in the TSP-D literature address the problem through a two-stage process: route construction and route improvement. These algorithms typically begin with the construction of a traditional TSP tour, achieved either using a Concorde solver or a heuristic algorithm. Subsequently, the route improvement stage involves determining the placement of drone nodes along the route to finalize it as a TSP-D tour. These enhancements commonly rely on neighborhood search procedures as a basis for improving the solution quality.

To the best of our knowledge, a method that simultaneously constructs truck and drone routes in a single stage has not been documented in the existing literature. However, adopting such an

Table 1.1: A summary of the literature review on TSP-D

<i>References</i>	<i>Problem class</i>	# <i>drone</i>	# <i>truck</i>	<i>of objective</i>		<i>Proposed Methodology</i>
				<i>makespan</i>	<i>op. cost</i>	
Murray & Chu (2015)	FSTSP/PDS	1 / n	1 / 1	x		Heuristic
Ponza (2016)	FSTSP	1	1	x		Simulated Annealing (SA)
Ferrandez et al. (2016)	TSP-mD	n	1	x		GA with K-means
Agatz et al. (2018)	TSP-D	1	1	x		Route first & cluster second heuristic
Ha et al. (2018)	TSP-D	1	1	x	x	GRASP (greedy randomized adaptive search procedure) & TSP-LS
Marinelli et al. (2018)	FSTSP	1	1	x		Modified GRASP
Freitas & Penna (2018)	FSTSP/TSP-	1 / 1	1 / 1	x		HGVNS (Hybrid general variable neighborhood search)
Phan et al. (2018)	TSP-mD	n	1	x		GRASP of HA et. al.(2018) & adaptive large neighborhood search (ALNS)
Mbiado et al. (2018)	PDSTSP	n	1	x		An iterative two-step heuristic
Poikonen et al. (2019)	mTSP-mD	n	1	x		3-phase heuristic: Route, transform, shortest path (RTS)
Sacramento et al. (2019)	VRP-D	n	m		x	Adaptive large neighborhood search (ALNS)
Karak & Abdelghany (2019)	TSP-mD	n	1		x	Extended C&W heuristic
Ha et al. (2020)	TSP-D	1	1	x		Hybrid GA with neighborhood search (swap and 2-opt)
Gonzalez et al. (2020)	mTSP-D	1	1	x		Greedy heuristic based on SA
Tong et al. (2022)	TSP-D	1	1	x		Tabu search
Dell' amico et al. (2020)	PDSTSP	n	1	x		Lin-Kernighan with LS
Salama & Srinivas (2020)	TSP-D	1	1	x		Machine-learning based heuristic
Murray & Raj (2020)	m-FSTSP	n	1	x		3-phase heuristic
Moshref-Javadi et al. (2020)	TSP-mD	n	1	x		Truck and drone routing algorithm (TDRA based on ALNS)
Wang et al. (2020)	TSP-D	1	1		x	Improved non-dominated sorting genetic algorithm
Almuhaideb et al. (2021)	TSP-D	1	1	x		GRASP with hill-climbing & GRASP with SA
Euchi & Sadok (2021)	VRP-D	n	m	x		GA with sweep local search
Luo et al. (2021)	mTSP-mD	n	1	x		Multi-start tabu search (MSTS)
Gomes-Lagos et al. (2021)	TSPDP	n	1	x		GRASP
Salama & Srinivas (2022)	m-FSTSP	n	1	x		Simulated Annealing (SA) & variable neighborhood search (VNS)
Arishi et al. (2022)	TSP-mD	n	1		x	Two-phase machine learning (ML): k-means clustering & deep reinforcement
Nguyen et al. (2022)	PDSTSP	n	m		x	A metaheuristic: Slack induction by string and sweep removals (SISSRs)
<i>this study</i>	TSP-D	1	1	x		Hybrid genetic algorithm with ant search-based solution (GA-AS)

algorithm would allow for optimizing the routes while determining the truck and drone nodes simultaneously. This approach, unlike staggered algorithms, would align more appropriately with the inherent characteristics of the problem.

1.4 Objective of the Thesis

The objective of this thesis is to provide a solution approach to solve the defined thesis problem and a novel contribution to the advancement of TSP-D solving methodologies. Besides, the outcome of this thesis aims to contribute to the field of optimization. The TSP-D is characterized by its complexity, as it involves both road and air transportation, and finding an optimal solution becomes increasingly difficult as the number of instances increases. By developing an efficient algorithm that can handle the complexity of solving the TSP-D, this thesis aims to contribute to the field of optimization and provide valuable insights for solving similar problems in domains of transportation and drone aided delivery processes in logistics.

To achieve this objective, the thesis will develop and implement the current state-of-the-art metaheuristic algorithms, GA and ACO, by hybridizing and customized them for TSP-D features. The thesis will present these well-known algorithms how to implement into a new optimization problem variant, TSP-D, as they are not presented before in a hybrid framework to solve the relevant problem.

The main motivation to chosen of the thesis methods is discussed followingly.

1. A consequence of the review of solution methods has been proposed to TSP-D shows that the problem is still addressed by a limited number of studies since it is presented very recently. That is results the need of develop new approaches.
2. TSP-D is considered to be NP-hard. Therefore, selection of heuristic methods as a solution approach are considered to be more appropriate for large-scale and complex TSP-D instances.
3. Heuristic methods proposed in TSP-D literature are distinguish by two classes: *classical heuristics* and *metaheuristics*. As a common approach in TSP-D classical heuristics is that consisting of two-staged solutions: Route construction and route improvement which are iteratively modify the route.
 - *Route construction stage* solves an optimal TSP tour for only truck delivery by regarding drone delivery.

- *Route improvement stage* applies various local search or neighborhood search techniques to add drone delivery nodes on the initially found TSP tour so that generates a final TSP-D tour.
4. As a consequence of literature review current algorithms for TSP-D involve these defined two-staged solution construction.
 5. Another consequence of literature review shows that metaheuristics are rarely studied so far to solve the problem when compared to classical ones. Moreover, the proposed ones also apply defined two-staged framework.
 6. That brings about a necessity on developing a heuristic approach whereas truck and drone nodes are determined simultaneously while minimizing the tour cost.

Thus, a metaheuristic approach is discussed in this thesis where routes are constructed and optimized in a single stage. Contrary to staggered algorithms, it may be more appropriate to the problem's characteristics.

7. GA and ACO are both population-based search metaheuristics whereas classical heuristics are single-solution based heuristics. It is preferred to discuss a population-based searching approach instead of existing heuristics using neighborhood search rules.
8. GA is a robust optimization technique that has been successfully applied to a wide range of combinatorial optimization problems with its ability to effectively explore a large search space and handle complex problem instances. However, a limitation of GA is that it may need to be combined with a local search method (Sivanandam & Deepa, 2008). As a result, it is discussed to present a hybridized application of GA in TSP-D.
9. The ACO algorithm is frequently employed in NP-hard problems as well, more specifically in conventional TSP. TSP is crucial in ACO, since ACO has been first presented with TSP implementation (Dorigo & Di Caro, 1999). Some of the factors that led to the chosen of ACO to be applied to TSP-D include the ones listed below:
 - ACO based approaches in traditional TSP having promising results (Dorigo et al., 2006) encourages applying it to TSP-D.
 - However, there is a lack of research on the implementation of ACO for the TSP-D in the existing literature.
 - Nonetheless, it is worth noting that the TSP-D as a routing problem can be easily adapted to the ant behavior metaphor and is well-suited for the characteristic of ACO.

The GA and ACO algorithms will be described in Section 2. Then, these selected algorithms will be further hybridized and customized to solving the TSP-D which is exhaustingly explained in Section 3. Followingly, the proposed hybrid algorithm will be evaluated on a set of benchmark instances through Section 4. Section 5 will discuss the results and findings of computational experiments. Section 6 will conclude the thesis.

2. OVERVIEW OF THESIS METHODS

This section introduces the methods applied in this thesis which are GA and ACO algorithms the ones from the state-of-art metaheuristics. This section provides the comprehensive overview of the algorithms and the fundamental principles underlying each algorithm with implementation frameworks in their origins.

2.1 Genetic Algorithm

Genetic Algorithm (GA) is a computational approach inspired by nature and falls within the category of evolutionary algorithms. Evolutionary algorithms are a group of search and optimization techniques that are intuitive, stochastic, population-based, and based on Mendelian genetics and Darwin's theory of evolution. The foundation of evolutionary algorithms lies in mimicking the evolutionary processes that occur in nature.

Genetic Algorithm, on the other hand, is a search and optimization method that works similarly to this observed evolutionary process in nature. GA is developed by taking inspiration from biological techniques such as genetic drift, natural selection, mutation, reproduction, and crossover, and is an approach used to find results in optimization problems with the principles of evolutionary biology.

GA was first introduced by John Holland in the 1970s through the imitation of evolutionary processes in a computer environment (Sampson, 1976).

At their core, GA is based on the principle of survival of the fittest. The algorithm starts with a population of potential solutions, each represented by a set of parameters or genes. In GA, all possible solutions are coded rather than using their parameter values. In Holland's GA, all possible solutions are coded as bit strings of the same length in a binary system. The population is then evaluated using a fitness function, which measures how well each solution performs the desired task. Solutions with higher fitness are more likely to survive and reproduce, passing on their genes to the next generation.

In each generation, the population undergoes selection, crossover, and mutation to create a new generation of solutions. During selection, solutions with higher fitness are more likely to be chosen as parents for the next generation. Similar to natural selection in nature, selection in GA ensures that better solutions are used more frequently during the search process. The selection operator chooses parents from the population to create children. The chance of being selected as

a parent is higher for individuals with a better fitness value, i.e., those who have a higher fitness function value as a solution to the problem. This allows the next generations to be made up of better solutions.

Genetic recombination and reproduction are achieved in GA through crossover and mutation mechanism. Crossover involves combining genes from two parents to create a new solution, while mutation involves randomly changing a small number of genes in a solution.

GA should have five essential components (Reeves & Rowe, 2002):

- Genetic representation of possible solutions to the problem.
- Determination of the initial population from these possible solutions.
- An evaluation function that ranks the solutions according to their "fitness" values.
- Genetic operators that create genetic diversity and generations.
- Values for various parameters used by the genetic algorithm, such as population size, probability of applying genetic operators, etc.

Through this iterative process, the population gradually evolves towards better and better solutions. The algorithm terminates when a stopping condition is met, such as a maximum number of generations or a satisfactory level of fitness.

One of the key advantages of GAs is their ability to explore a large search space efficiently. Traditional optimization methods may get stuck in local optima, while GAs can search a wide range of solutions and find a global optimum. GAs can also handle non-linear, non-continuous, and multi-objective problems, making them a versatile tool for many real-world applications.

GA can effectively handle large-scale problems. While GA does not guarantee finding the optimal solution to a problem, it generally provides acceptable solutions in an acceptable time frame.

The distinctive characteristics of GA from other heuristic approaches are as follows (Haupt & Haupt, 2004):

- GA works by encoding solutions rather than working with the solutions themselves. Therefore, solutions need to be effectively represented in chromosome form.
- Unlike other metaheuristics that move from one solution to another based on a single solution searching, such as simulated annealing and tabu search, GA starts searching from a set of solutions. In this respect, GA moves simultaneously in a versatile manner in the

solution space of the problem, reducing the probability of ending the search at local optima.

- GA only requires the objective function values. The problem does not have to be continuous or differentiable. Real-life problems generally have a discrete solution space.
- GA uses probability rules.
- It is a blind algorithm that does not contain information about what it does but knows how to do it.

There are several advantages of GA, including (Beasley et al., 1993):

- **Global Optimization:** GA can find the global optimal solution to a problem, instead of getting stuck in local optima like some other optimization methods.
- **Versatility:** GA can handle a wide range of problem types, including non-linear, non-continuous, and multi-objective problems.
- **No Derivative Required:** GA do not require derivative information, which is an advantage when dealing with complex problems where derivatives may not be easy to obtain.
- **Large Search Space:** GA can efficiently search through a large search space, making them useful for solving problems with many variables.
- **Multiple Solutions:** GA can produce a set of multiple different good solutions, rather than just a single solution to the problem.
- **Parallel Computing:** GA can be easily parallelized, allowing for faster optimization on multiple processors.

While GA have several advantages, they also have some limitations and disadvantages, including:

- **Computational Complexity:** GA can be computationally expensive, especially when dealing with large search spaces or complex fitness functions. The time required to find a solution can increase significantly as the number of variables or the complexity of the problem increases.
- **Parameter Tuning:** GA have several parameters that need to be tuned to achieve optimal performance, such as the population size, mutation rate, and crossover rate. Finding the right parameter settings can be time-consuming and requires expertise.

- **Premature Convergence:** GA can converge too quickly to a suboptimal solution if the population size is too small, or the mutation rate is too low. This can result in the algorithm getting stuck in a local optimum and not exploring the full search space.
- **Limited Accuracy:** GA may not always produce the most accurate results, especially when dealing with highly complex or noisy problems. Other optimization methods, such as gradient-based methods, may be more accurate in these cases.
- **Representation Bias:** The quality of the solutions found by GA can be influenced by the representation of the problem. If the representation is not well-suited to the problem, the solutions found by GA may not be optimal.
- **Difficulty Scaling:** GA may have difficulty scaling to very large problems, especially when dealing with combinatorial or discrete optimization problems.

Overall, while GAs have proven to be a useful optimization tool, their performance is highly dependent on the problem and the specific parameters used. It is important to carefully consider the strengths and limitations of GAs before using them for a particular application.

The working principle of GA can be described below in stages (Malhotra et al., 2011):

Stage 1 [start]: The objective function is defined, chromosomes are encoded, initial population consisting of n chromosomes (suitable solutions for the problem) is randomly determined, and genetic algorithm parameters such as crossover and mutation probabilities are set.

Stage 2 [fitness]: The fitness value of each chromosome in the population is determined.

Stage 3 [reproduction]: The following steps are repeated until the size of the new population is equal to the size of the initial population to create a new population.

Stage 3.1 [selection]: Two chromosomes are selected from the current population as parents based on their fitness values (those with better fitness values have a higher chance of being selected).

Stage 3.2 [crossover]: The selected chromosome pair is crossed to create offspring individuals with a probability of crossover. If the crossover does not occur, the offspring chromosomes are created as copies of the parents.

Stage 3.3 [mutation]: Offspring chromosomes created with a probability of mutation are subjected to mutation in their location.

Stage 3.4 [addition]: The resulting offspring chromosomes are added to the new population.

Stage 4 [replacement]: The new population consisting of offspring chromosomes is replaced with the current population consisting of parent chromosomes.

Stage 5 [termination]: If the stopping criterion is met, the algorithm is terminated, and the best solution found is returned. If the stopping criterion is not met, the next step is taken.

Stage 6 [loop]: Return to stage 2 to evaluate the fitness of the new population.

2.1.1 Encoding

One of the fundamental mechanisms used in GA is the encoding of individuals that represent possible solutions to the problem. The possible solutions to the problem are represented by a parameter set, and these parameters (known as genes) come together to form an array (corresponding to a chromosome). Encoding the parameters allows problem-specific information to be translated into a form that can be used by the GA. Before applying GA to a problem, the variables of the problem must be encoded as arrays of the same dimensions.

It is important to plan for each solution to be uniquely represented by an array. Each array should represent a possible random point in the problem's solution space. The encoding mechanism varies depending on the nature of the problem. Binary systems, integers, real numbers, letters, symbols, or many different objects can be used in the encoding of solutions.

Binary coding is the most commonly used method, where each chromosome is represented as a string of 1s and 0s. Each bit in the string corresponds to a particular gene in the chromosome, and the value of the bit indicates whether that gene is present or absent. This type of coding is easy to implement and computationally efficient, making it a popular choice for many optimization problems.

Real-valued coding, on the other hand, represents each gene as a real number. This type of coding is often used in optimization problems where the variables to be optimized can take on any real value within a certain range. Real-valued coding requires more computational resources compared to binary coding, but it provides more precision and accuracy in representing the variables.

The selection of the appropriate encoding greatly affects the performance of GA, and a coding that is applicable to other GA operators must also be chosen. Some genetic encoding formats are illustrated in Figure 2.1.

chromosome n	1100010110111000
chromosome m	1011001011001010

(a)

chromosome k	1.2445 5.3243 0.4856 2.3962 2.1345
chromosome n	ABEIFJDJDHDIERLDFLFEGJFDT
chromosome m	(back, (back), (left), (forward), (right))

(b)

chromosome n	1 5 3 2 6 4 7 9 8
chromosome m	8 5 6 7 2 3 1 4 9

(c)

Figure 2.1: Illustrations to some genetic encoding types in GA: (a) binary encoding, (b) value encoding, (c) permutation encoding

2.1.2 Genetic population

Once the coding method for chromosomes representing possible solutions to the problem is determined, the search begins with a randomly determined set of solutions consisting of these chromosomes. GA start searching for solutions not from a single point, but from a set of solution points defined as a population. As individuals comprising the population represent different points in the solution space of the problem, different points in the solution space are explored simultaneously. Population-based search reduces the likelihood of algorithm getting stuck at local optima. At this stage, it is important to determine how the initial population will be selected and the size of the population.

The initial population is usually determined randomly, but there are also approaches where individuals in the population are selected with other heuristic methods. The selection of individuals that will form the initial population can vary depending on the coding of the chromosomes or the type of the problem. For instance, if the solution of the problem is represented with binary coding, chromosomes are initially generated randomly. A random

number is generated, and if it is below a certain number, the first allele of the chromosome takes the value 0; otherwise, it takes the value 1. After all genes of the chromosome are created in this way, chromosomes are generated until the desired size of the initial population is reached. In a problem where permutation coding will be used, a number is first assigned to each possible solution point of the problem, and chromosomes are created by randomly selecting genes from a gene pool consisting of these values. For a different coding type, chromosomes are also generated first and the process continues until the population is filled.

The number of individuals (chromosomes) in the population is determined by the user and remains constant throughout the subsequent generations. A larger population size allows for searching a wider part of the solution space but will increase the time to reach the solution. This is because, as the number of individuals in the population increases, the search time in the solution space will increase logarithmically. A smaller population size increases the speed of the algorithm but reduces the probability of finding the best solution since a relatively smaller part of the solution space is explored.

2.1.3 Fitness evaluation

Each possible solution point of a problem encoded as sequences in the solution space and represented by chromosomes, has a fitness value. The fitness value that indicates the "goodness" of solutions must be determined for each individual (chromosome) in the population. The fitness value of a chromosome is the objective function value at the corresponding point in the solution space. In GA, an individual's fitness is the objective function value of its phenotype. In other words, a value is determined that shows the degree of fitness to the problem's objective by decoding the genotype created with the encoded individual. This fitness value indicates the extent to which individuals meet the problem's objective function value. As observed in nature, in GA, better solutions will have a greater chance of survival and to create the next generations if they have a higher fitness function value, and this will enable new generations to be composed of better solutions.

2.1.4 Reproduction

In the reproduction phase of GA, recombination is performed to generate offspring that will form the next generation from selected individuals in the population. In GA, selection, crossover, and mutation operators are used to determine which individuals will reproduce, how they will combine their genetic material, and how their offspring will be mutated to ensure genetic

diversity. These operators are used to ensure genetic diversity, which is essential for completing the evolution process. By inheriting the characteristics of parent individuals through crossover and mutation, offspring are produced, ensuring the continuation of genetic diversity. Crossover and mutation, also known as variation operators, provide the algorithm's search function. During the search for the optimal solution in the problem's search space, the algorithm's two essential functions, crossover, and mutation are performed.

2.1.4.1 Selection

Using the selection operator used in GA in combination with the natural selection model based on the survival of the fittest principle in nature, individuals that will play a role in the reproduction phase are selected by artificial means. Similar to natural selection observed in nature, selection in GA ensures that better solutions are used more in the search process. The individuals to be selected as parents to create offspring individuals from within the population pool are determined by the selection operator based on their fitness values. The selection operator ensures that individuals (chromosomes) with high fitness values, or relatively better candidate solutions, exist in subsequent iterations and move towards better solutions, similar to natural selection in nature.

Chromosomes that create better solutions for the problem according to the determined fitness values will be selected as parents more often than chromosomes with worse fitness values in the population. This way, the fitness of the solutions found in GA will increase from generation to generation. Since the reproduction chances of unsuccessful individuals, or individuals that do not provide a feasible solution for the problem or create relatively worse solutions, are reduced with the selection operator, the deteriorating trend in solving the problem becomes more difficult. However, it is necessary to allow bad solutions to also have a relatively lower chance of selection for reproduction. This is required to search different areas in the solution space beyond local optimal points and increase the chance of finding the optimum solution.

If the gene pool of parents is composed of individuals that are only similar to each other in terms of genotype, genetic diversity will not be ensured, and evolution will be completed quickly. The problem space will always be searched in the same places, and the algorithm will probably end up in a local optimum. Therefore, it is necessary to work with a selection method that can also provide population diversity for improving the search performance of GA. Below are some frequently used selection operators:

1. *Roulette wheel selection*: In this selection operator, the individuals in the population are assigned a probability of selection based on their fitness values. The individuals with higher fitness values have a higher probability of being selected for reproduction.
2. *Tournament selection*: In this operator, a subset of individuals is selected randomly from the population and the individual with the highest fitness value in the subset is chosen for reproduction.
3. *Rank selection*: In this operator, the individuals in the population are ranked based on their fitness values and the probability of selection is proportional to the rank.
4. *Boltzmann selection*: This operator uses a temperature parameter to choose individuals for reproduction. The probability of selection is determined by a Boltzmann distribution that takes into account the fitness values and the temperature parameter.
5. *Elitist selection*: In this operator, the best individuals from the current population are preserved in the next generation without any modifications. This helps to maintain the best individuals in the population and prevent the algorithm from losing good solutions.

2.1.4.2 Crossover

The crossover operator is used to create one or two offspring chromosomes from the parent chromosomes selected from the current population in the selection stage. Crossover allows the exploration of unexplored areas of the search space by providing genetic diversity and is a crucial process in the emergence of new genetic materials and the maintenance of genetic diversity.

Since different offspring chromosomes can be produced from the same chromosome pairs, the crossover operator is generally stochastic. Essentially, crossover is the exchange of genes between parent chromosomes, and new individuals are created by mixing the genetic materials of two individuals. Usually, the parent chromosomes are randomly separated, and the genes of the resulting offspring chromosome are randomly combined, with some genes coming from the mother and the rest coming from the father. The aim of crossover is to obtain new individuals with higher fitness values from parents with good genetic characteristics. In other words, it aims to achieve better solutions by starting from solutions that are suitable for the problem's goal. However, it is possible to obtain offspring with worse compatibility from parents with good genetic characteristics. Allowing individuals with worse fitness than their parents to be obtained and persist in the generations can prevent the algorithm from getting stuck at local optima by reaching different and unexplored areas of the search space.

Crossover is a recombination operator that is completed in three stages:

- Two individual strings are selected to mate for reproduction.
- A crossover point is randomly selected across the string length.
- The position values are exchanged between the two individual strings according to the crossover point.

Crossover is not applied to every mating individual. The proportion of individuals to which the crossover process will be applied is determined by the crossover rate, which is determined at the beginning of the algorithm, from the selected parents from the current population. If crossover is not applied, offspring chromosomes are created as copies of their parents. This situation allows individuals to directly carry their genetic characteristics to the next generation without being degraded by crossover. Various crossover has been discussed for crossover operation, but some of the main ones are: *single-point crossover*, *multi-point crossover*, *uniform crossover*; especially ones for permutation-based problems like TSP are *Ordered Crossover (OX)*, *Partially Mapped Crossover (PMX)*, *Precedence Preservative Crossover – PPX*, *Cycle Crossover*, *Greedy Subtour Crossover – GSX*.

Nevertheless, there are also many other variations or heuristic operators that have been developed for specific problems or applications.

2.1.4.3 Mutation

Mutation is one of the main genetic operators in GA, which introduces new genetic material into the population by randomly altering one or more genes in an individual's chromosome.

After crossover, changes are made to the child's genetic sequence using the mutation operator to ensure that it differs from its parents. In other words, the mutation operator randomly changes the values of certain genes in a chromosome to create a new solution that is different from the parent solutions. The mutation operator is necessary to create the genetic diversity required to prevent early convergence. By overcoming local optima with genetic diversity, unexplored areas of the search space can be reached.

The ratio of mutated individuals to individuals in the population is equal to the mutation rate. Some of the children generated after crossover are randomly selected and subjected to mutation based on the mutation rate set at the beginning of the algorithm, and they are added to the new population. Mutation is performed at a much lower rate compared to crossover, usually within

the range of 0.01 to 0.1. Some of the mutation operators are: *Bit-flip mutation*, *swap mutation*, *inversion mutation*, *interchanging mutation* etc.

2.2 Ant Colony Optimization Algorithm

Ant colony optimization (ACO) is a nature-inspired optimization algorithm based on the collective behavior of ants searching for food. ACO was first introduced by Dorigo et al. (1996) and has since gained significant attention in the field of optimization due to its ability to solve a variety of complex problems.

ACO is based on the observation that ants can efficiently find the shortest path between their nest and a food source by leaving a trail of pheromone behind them. This pheromone serves as a form of communication between the ants to guide other ants to the source so that allows them to find the shortest path by following the most pheromone. During food exploration, while ants search for food in a disorganized way, the ant that finds food carries it back to the nest. As ants

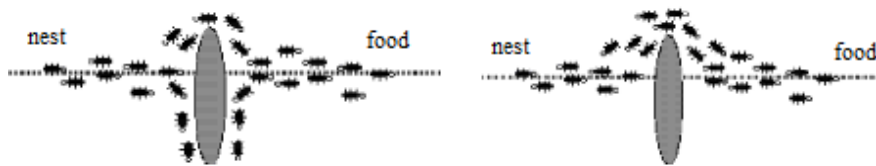


Figure 2.2: Illustration of real ants finding shortest path

move between the food and the nest, they leave a trail of pheromone secretion that helps other ants choose the path. The pheromone left by ants that reach the food source via a shorter path forms a stronger pheromone trail, as they pass through before it evaporates. After a while, all ants start following the stronger pheromone trail and form a loop, reaching the food source via the shortest path. Figure 2.2 shows an example of all ants determining the shortest path between the food and the nest after a while.

ACO algorithm mimics this behavior by simulating the movement of real ants in search of the best solution to a given optimization problem. The process of ACO involves creating a set of artificial ants, which move through the search space, depositing pheromone on the paths it takes and construct solutions using a probabilistic rule. The amount of pheromone deposited on a path is proportional to the quality of the solution found on that path. The pheromone information is then used by the ants to guide their search towards better solutions. As the ants move through the search space, the pheromone trail is reinforced by the ants that find good solutions, while they lead to poor solutions evaporate over time. These ants deposit pheromone along the path

they traverse, with the amount of pheromone proportional to the quality of the solution they construct. This process of pheromone deposition and evaporation allows the ants to concentrate their search on the most promising areas of the search space.

ACO has been successfully applied to a wide range of optimization problems, including the Traveling Salesman Problem, Vehicle Routing Problem, and Job Shop Scheduling Problem. It has also been used in various fields such as telecommunications, transportation, and robotics. The ACO algorithm has been shown to produce high-quality solutions in a reasonable amount of time and is particularly effective when dealing with complex, combinatorial optimization problems.

The steps of the ACO algorithm can be summarized as follows:

1. *Initialization:* Initialize the parameters of the algorithm, such as the number of ants, the pheromone evaporation rate, the initial pheromone values.
2. *Ant movement:* Each ant in the colony starts from a random position and moves through the solution space according to a set of probabilistic rules based on the pheromone trails. The ants build a solution by iteratively selecting the next component of the solution based on these probabilities.
3. *Global pheromone update:* After each ant completes its tour, it deposits pheromone on the edges of the solution components it used. The amount of pheromone deposited is proportional to the quality of the solution found by the ant. The idea is to reinforce the pheromone trails on the edges that are part of good solutions and reduce those that are part of bad solutions.
4. *Termination:* The algorithm terminates when a stopping criterion is met, such as reaching a maximum number of iterations or finding a satisfactory solution

2.2.1 Solution construction in ACO

Solution construction in ACO refers to the process of building a potential solution to the problem being optimized. In ACO, artificial ants are used to construct solutions by iteratively building paths between problem-specific elements or components. The ants probabilistically select the next component to add to the solution based on the amount of pheromone deposited on each possible component, as well as heuristic information about the quality of each component. This balance between exploiting the pheromone trail and exploring new possibilities helps the ants to find good solutions while avoiding getting stuck in local optima.

As each ant builds a solution, it updates the pheromone levels on the components it uses based on the quality of the solution constructed. This feedback loop helps to reinforce the paths taken by the ants that lead to better solutions, while allowing for some level of exploration to find new, potentially better solutions.

A TSP solution constructing in ACO can be define followingly. In the ant solution construction step, each ant constructs a solution by choosing the next city to visit based on a probability distribution. This distribution is determined by the pheromone level on the edges connecting the current city to the remaining unvisited cities, as well as an information about the distance between the cities. The probability of choosing a particular city is given in Equation (2.1) (Dorigo and Gambardella, 1997):

$$p(i, j) = \frac{[P_{ij}(t)]^\alpha \times [n_{ij}]^\beta}{\sum_{l \in N_i} [P_{il}(t)]^\alpha \times [n_{il}]^\beta}, \quad \forall j \in N_i \quad (2.1)$$

Where $p(i, j)$ is the probability of choosing ant k to move from city i to city j ; $P_{ij}(t)$ is the pheromone level on the edge connecting cities i and j at iteration t ; n_{ij} is the heuristic value of moving from city i to city j (usually $1/d_{ij}$ where d_{ij} is the distance between cities i and j); N_i is the set of all the remaining neighbor nodes of city i . α and β are parameters that control the relative importance of the pheromone and heuristic information.

In ACO, the α and β parameters affect the trade-off between exploration and exploitation in the search process. Specifically, alpha controls the relative importance of the pheromone trail, while beta controls the relative importance of the heuristic information (e.g., distance between two cities) in the ant's decision-making process. A higher alpha value puts more emphasis on the pheromone trail, which encourages exploration of new solutions, while a higher beta value puts more emphasis on the heuristic information, which encourages exploitation of the current best solution.

For instance, while solving TSP in ACO, the α parameter controls the importance of the pheromone trail. A higher value of alpha means that the ant is more likely to select the city with the higher pheromone level, indicating that it has been visited frequently in the past. This results in a more exploitative search, where the ants are more likely to converge to a single solution. On the other hand, the β parameter controls the importance of the heuristic information, which is the distance between cities. A higher value of beta means that the ant is more likely to select the city that is closer to the current city. This results in a more exploratory search.

If β is 0 in ACO, it means that only the pheromone trail information will be used for ant decision making, and the heuristic information will be ignored. In TSP, this means that ants will only follow the pheromone trails without considering the distance between the cities. This can result in ants getting stuck in a suboptimal solution and not exploring other possibilities.

On the other hand, if α is 0, it means that only the heuristic information will be used, and the pheromone trail information will be ignored. This approach is called "greedy" or "nearest neighbor" algorithm. In TSP, this means that ants will always choose the nearest city as the next destination, without considering the pheromone trails. This approach can lead to fast convergence to a local optimum solution, but it may not necessarily be the global optimum solution.

Therefore, the selection of α and β values in ACO depends on the characteristics of the problem being solved. Generally, a balance between exploration and exploitation is desired for TSP problems, as too much exploration may lead to slow convergence to a good solution, while too much exploitation may lead to getting stuck in a suboptimal solution. The values of alpha and beta are usually determined through experimentation and parameter tuning.

Overall, solution construction in ACO is a key component of the algorithm, as it allows for the search for good solutions to be guided by both the quality of the solutions found so far, as well as domain-specific knowledge about the problem being optimized.

2.2.2 Pheromone update in ACO

The pheromone update mechanism is a crucial component of ACO algorithm. Pheromones play a critical role in guiding the artificial ants towards the most promising regions of the solution space. The pheromone update is carried out after every iteration of the algorithm and is responsible for reinforcing or weakening the edges of the graph that correspond to the paths taken by the ants.

The pheromone update is based on two main processes: pheromone evaporation and pheromone deposition. Pheromone evaporation is the process of gradually reducing the amount of pheromone on all edges of the graph. This process is necessary to prevent the pheromone from accumulating on the edges indefinitely, which could lead to premature convergence of the algorithm to a suboptimal solution. Pheromone deposition, on the other hand, involves adding pheromone to the edges that were traversed by the ants during the current iteration. The amount of pheromone deposited on each edge is proportional to the quality of the solution found by the

ant that travelled along. In other words, the more optimal the solution, the more pheromone is deposited on the corresponding edge.

In ACO for TSP, the pheromone update rule is an important step to update the pheromone trail values of each ant's tour. This update process is based on the quality of the tour solutions obtained by the ants. The rule is designed to increase the pheromone levels on the edges that are included in good tours and decrease the pheromone levels on the edges that are not included in good tours. One common approach to update pheromones is the following formula.

Let $P_{ij}(t)$ is intensity of the current pheromone trail on edge ij at iteration t . Each ant selects its next city to visit at iteration $(t+1)$. Thus, an ACO iteration consist of m moves performed by m number of ants in the interval $(t, t+1)$. This means that in every n iteration of the algorithm each ant has finished a TSP tour. By the time an iteration is completed, the pheromone amount is updated by Equation (2.2):

$$P_{ij}(t + 1) = P_{ij}(t) e + \sum_{k=1}^m \Delta P_{ij}^k \quad (2.2)$$

The pheromone level on each edge is updated by adding ΔP_{ij}^k to the current pheromone level on the edge which is the pheromone amount laid on edge ij by the ant k between iteration t and $t+1$ and formulated as in Equation (2.3) (Dorigo and Gambardella, 1997):

$$\Delta P_{ij}^k = \begin{cases} Q/L_k, & \text{if edge } (ij) \text{ appears in the tour of ant } k \\ 0, & \text{otherwise} \end{cases} \quad (2.3)$$

L_k is the length of the tour constructed by ant k where Q is a constant. The pheromone level on each edge is also evaporated by a certain rate e after each iteration to prevent the pheromone level from becoming too high. The value of e can be chosen to control the rate of pheromone evaporation. A high value of e will cause the pheromone to evaporate quickly, which can prevent the algorithm from being stuck in local optima but can also lead to a lack of pheromone trails. A low value of e will cause the pheromone to evaporate slowly, which can help the algorithm to converge to a good solution but can also lead to the pheromone trails becoming too strong and causing premature convergence.

By using this pheromone update rule, ACO for TSP can gradually improve the quality of the solutions by exploiting the good solutions found by the ants and exploring new solution space.

Consequently, the pheromone update mechanism is a fundamental aspect of the ACO algorithm, allowing artificial ants to cooperate and communicate with each other, and guiding the search towards optimal solutions.

3. THE HYBRID METAHEURISTIC METHOD TO TRAVELING SALESMAN PROBLEM WITH DRONE

This section describes the hybrid metaheuristic method proposed to solve TSP-D. Proposed metaheuristic is based on the Genetic Algorithm (GA) that runs simultaneously with the Ant Colony Optimization Algorithm (ACO). The hybrid framework follows the traditional GA optimization steps, while a modified searching method based on conventional ACO has been developed. The proposed hybrid metaheuristic algorithm is called *The Genetic Algorithm with Ant Search-Based Solution Method (GA-AS)*. Following sections exhaustively presents framework of the GA-AS algorithm.

3.1 The Genetic Algorithm with Ant Search-Based Solution Method to TSP-D

Within a TSP-D solution, two separate optimization problems occur to be considered. In a TSP-D solution, given set of customer nodes to be served should be visited in a route with the lowest travelling cost which is referred to a classical route optimization. Meanwhile, the decision of which customer will be served by which vehicle -either truck or drone- should be determined. This decision brings additional optimization requisite about of determining min-cost truck routes and drone routes concurrently. In this thesis, a hybrid metaheuristic method is conducted to carry out these optimization tasks for solving a TSP-D problem, which is based on two state-of-arts algorithms, GA and ACO. GA is used to determine whether each customer node will be served by truck or by drone which are referred as truck delivery nodes and drone delivery nodes. Simultaneously, an Ant Search (AS) algorithm inspired from original ACO algorithm is used to generate lowest-cost route of these both truck and drone deliveries. Algorithm 1. provides the pseudo code of the GA-AS algorithm. In addition, Figure 3.1. displays a flow diagram of the GA-AS algorithm.

Algorithm 1: The GA-AS algorithm

```
1:   Initialize Population
2:   iter = 0
3:   for (iter < maxiter)
4:       gpop = 0;
5:       Generate truckphmtrx and dronephmtrx
6:       for (gpop=0, gpop < maximum number of population, gpop++)
7:           Select the chromosome (gpob)
8:           Apply AS algorithm (gpob, truckphmtrx and dronephmtrx)
9:           Generate the route and fitness of each chromosome
10:      end
```

```

11:         for (until the max number of crossover)
12:             Select the parents P1 and P2
13:             Generate offspring individual O from P1 and P2
14:             Replace unwilling chromosome with offspring O
15:         end
16:         for (until the max number of mutation)
17:             Select the random allele on the chromosome
18:             Mutate the selected allele
19:         end
20:         update truckphmtrx and dronemphmtrx
21:     end

```

In more detail of Algorithm 1, the genetic population is initialized at the beginning (line 1). In order to be constructed as a TSP-D route, each chromosome in the GA population undergoes the AS algorithm (line 8) which will be clearly revealed in Section 2.1.3, in addition pseudo code of the AS algorithm is provided in Appendix 1-Algorithm 2. In line 9, each chromosome is assessed to determine its fitness value, which corresponds to the cost of its tour. The roulette wheel method is used in line 12 to choose which chromosomes will serve as the parents during reproduction. Lines 13 to 18 represent the reproduction steps of GA consist of crossover and mutation; elitism is additionally applied to reproduction (line 14). Child chromosomes create the new population of GA following the reproduction phase, and the algorithm then turns back to

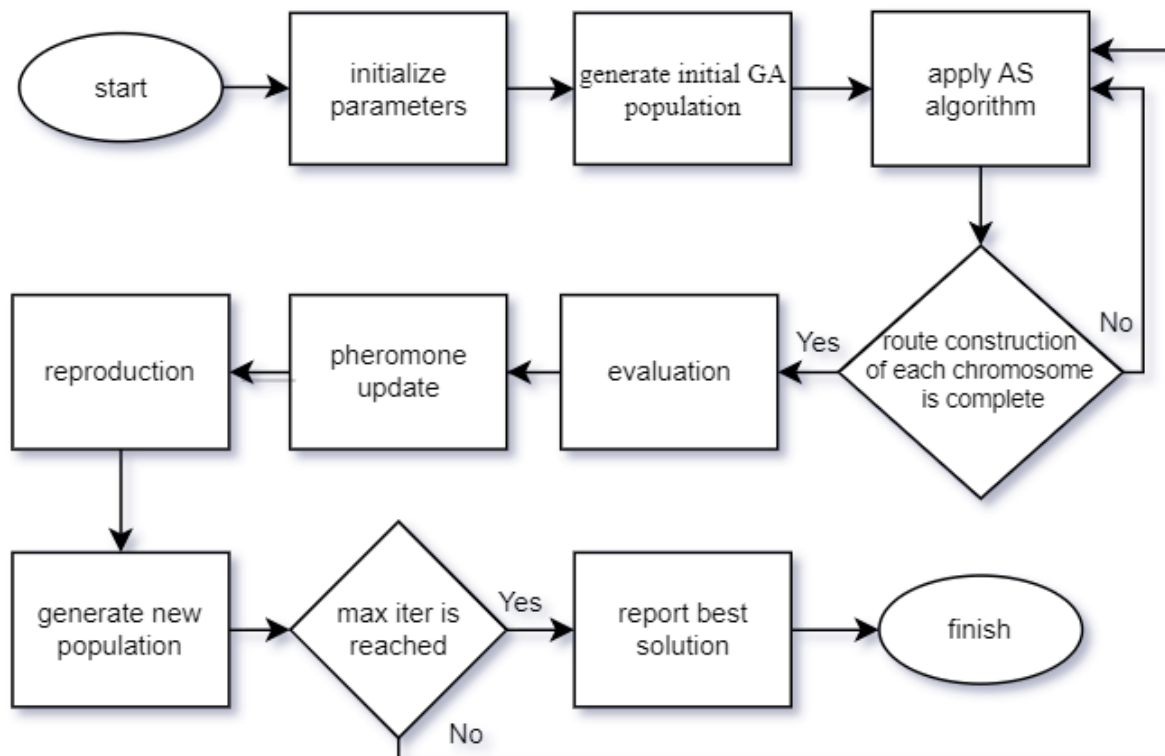


Figure 3.1: Flow diagram of the GA-AS algorithm

line 2. Up until the specified maximum number of iterations, the algorithm repeats. The algorithm finally returns the min-cost solution obtained. The following sections will provide a comprehensive explanation of each step of the proposed algorithm.

3.1.1 Supportive structure for algorithm

In a consequent TSP-D solution, it is seen that some customers are visited by the truck while some others visited by the drone to deliver parcels. So, a TSP-D tour facts that two separate subtours occur according to type of vehicles used to travel along customer paths. Thus, the primary decision is to distinguish transportation types in a TSP-D tour. A path among two customer nodes may be travelled in some possible ways separately: alone by truck, alone by drone or by truck while drone is standing on top of the truck. We first define the transportation types to be used in determining subtours which are essential to construct a final TSP-D route. Figure 3.2 illustrates these determined transportation types occurring in a TSP-D tour and explained as below:

- **Transportation Type 1:** The truck moves away from customer i alone and travels along path ij to serve customer j (simultaneously drone own a sortie to serve a different customer node). Then, the node j is called as Type 1.
- **Transportation Type 2:** The drone launches alone from customer i where it stands by on the truck at the beginning, has its own sortie along the path between customer i and j , serves customer j and then lands on top of the truck at a different node from beginning. Then, node j is called as Type 2. When drone meets with truck again at a customer node after its sortie, that customer node is defined as *rendezvous* node.
- **Transportation Type 3:** The path between customer i and j is travelled by a truck while drone is carried by the truck. Delivery is held by the truck while the drone is standing by on top of the truck. Then, the node j is called as Type 3.
- **Transportation Type 4:** The drone is located on top of the truck at the initial customer node i , where it launches from and then has a sortie to customer j . While customer j is being served by drone, truck waits at customer i . After delivery, drone returns to its launching customer node i and lands onto the waiting truck. In this type of transportation appeared with a drone loop indicates that the drone sortie takes $i-j-i$ path, so that customer node j is called as Type 4.

* A node's type will change to 'Type 3' if it serves as a rendezvous node in the algorithm.

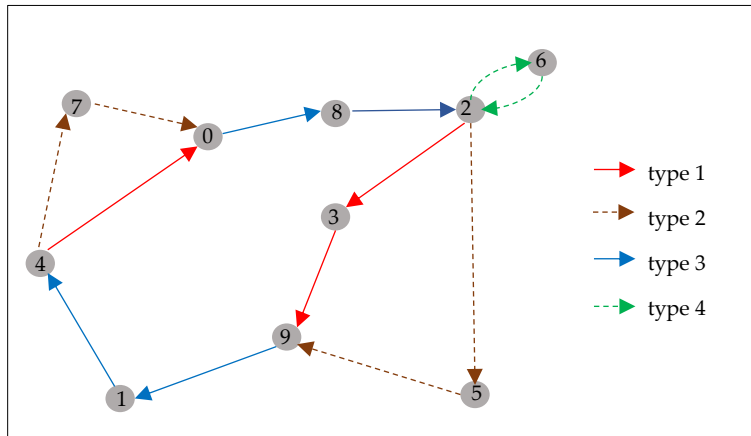


Figure 3.2: Transportation Types appeared in a TSP-D route

3.1.2 Solution representation in genetic algorithm

The initial step in GA is the encoding of individuals as to represent possible solutions of an optimization problem. Possible solutions to the problem are represented by a set of parameters - *genes*- which are come together to form a sequence -*chromosome*-. By encoding the parameters, the problem-specific information converted into a usable form in GA. Each solution should be encoded as a distinct and feasible random point in the problem's solution space. Since the appropriate encoding has a significant impact on the algorithm's performance, the solution representation is the primary decision stage in our GA.

In solution representation of described GA in this study, a binary encoding is applied where bits are used as gene parameters. Each gene of a chromosome takes the value of either 0 or 1. Thus, a large number of possible solution representations can be generated by a fewer algorithmic effort. Each parameter used in encoding, which are 0 and 1 in binary encoding, represents a problem specific information. In our binary represented chromosome, a customer node serves by a truck is encoded by 0 whereas a customer node serves by a drone encoded by 1. A randomly generated solution representation of a chromosome with binary encoding can be used in presented GA and how represents a TSP-D information are illustrated in Figure 3.3.

In the presented solution representation, each gene of a chromosome corresponds to customer nodes successively. So that the chromosome length is as much as the total number of customers (n) given in TSP-D. In a chromosome, each gen location i coincide with the customer node i ($i=1\dots n$). Contemporaneously, the algorithm generates two distinct arrays to present the information of whether a customer delivered by a truck or by a drone. These arrays are Truck

Delivery Nodes (*TDarray*) and Drone Delivery Nodes (*DDarray*). *TDarray* consists of customer nodes must be visited by the truck, in other words it determines the nodes to be in truck route. *DDarray* similarly consists of the determined customer nodes must be visited and served by the drone. Eventually, algorithm starts to generate random number for each gene correspondingly to determine whether it will take 0 or 1 value until the chromosome length n. If a gene takes 0, then corresponding customer node i determined as a truck node which means truck will serve i. customer. This customer i is directly located in *TDarray*. In contrary, when the gene takes value of 1, the customer node is added to *DDarray*. Algorithm generates chromosomes by this encoding up to a pre-given number (N). These N chromosomes forms a population of GA.

This solution representation does not present the delivery sequence (route) of customer nodes. The represented solution by binary encoded chromosome with *TDarray* and *DDarray* determines only the transportation type of customer nodes in this step of algorithm. These transportation types are identified in more detailed by the following section.

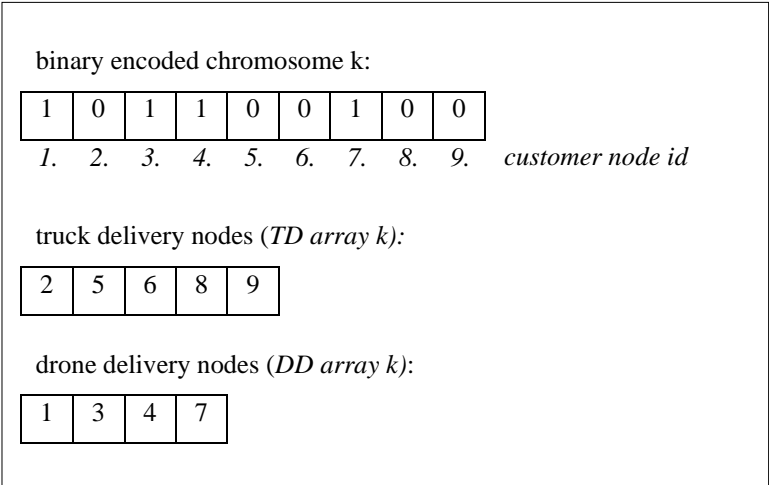


Figure 3.3: An illustration of solution representation with binary encoding for the TSP-D

3.1.3 Ant-Search algorithm

The AS algorithm of the GA-AS constructs TSP-D routes corresponding to determined delivery type of nodes by GA step. Since a chromosome represents only the information about delivery type of nodes, now each node should be sequenced as a feasible route. This sequencing is held by the AS algorithm by transforming each chromosome solution of the GA into a feasible TSP-D solution. AS algorithm is inspired from the traditional ACO algorithm. Ant behavior metaphor in ACO is imitated in AS algorithm. The procedure of artificial ants in that ACO algorithm constructing a TSP tour is implemented to the route construction process of TSP-D by the GA-AS with some modifications of problem specific features to the algorithm. Following parts

D solution, two types of vehicles (truck and drone) are used in delivery. Therefore, within an ant tour two types of sub-routes are appeared: truck route and drone route. To construct a truck route each truck path should have its own pheromone amount, similarly to construct a drone path each drone path should have its own pheromone amount. Owing to, it is decided truck and drone to have separate pheromone frameworks determined by separate formulations instead of a unique one. Consequently, a two-pheromone framework is presented in the proposed algorithm.

In the proposed AS algorithm, two separate pheromone matrices are generated: truck pheromone matrix (P_t) and drone pheromone matrix (P_d), each for $n \times n$ size corresponding to number of customer nodes. The current amount of pheromone on the truck path ij is indicated by $P_{t_{ij}}$ while the current amount of pheromone on the drone path ij is indicated by $P_{d_{ij}}$, where $i=1, \dots, n$ and $j=i, \dots, n$ for each. Initially, $1/n$ is designated to each $P_{t_{ij}}$ and $P_{d_{ij}}$ values. After an iteration ends each pheromone matrix is updated by a determined equation which will be explained furtherly.

3.1.4 Route construction in the GA-AS algorithm

A solution generated in GA step of algorithm which defines truck and drone delivery nodes by a chromosome goes through the AS step of algorithm to be construct as a complete route. Route construction bases on selecting customer nodes respectively to be sequenced as a TSP-D tour. The selection mechanism is a stochastic process occurs under the ant-search algorithm inspired from original ACO algorithm. Following sections defines this selection process exhaustingly.

3.1.4.1 Restrictions

At the beginning of the algorithm, an ant k initially located at the depot selects its subsequent visiting node by a probabilistic process until all customer nodes have been sequenced. But, this selection process also restricted according to the vehicle type used in the last delivery, that means the transportation type of customer nodes restricts the selection of a node to the route. When the transportation type of customer node is type 1 or type 3, that means delivery of the node is held by the truck. Otherwise, in case of type 2 and type 4 indicates that delivery is held by the drone. For instance, once a customer node is delivered by the truck alone (type 1), the only way of transportation to the next customer may held also by the truck (type1 or type 3). Or once the drone serves a customer node alone, then at the next node drone must land on the truck because of the assumption of drone may serve at most one customer at its each sortie. So that the next node cannot be a drone delivery node (type 2) only can be a truck delivery node (type3). Such

all restrictions based on the transportation types defined in section 3.1.1. are given below:

- If the last sequenced node i is *type 1*, then the next node j to be sequenced may only be one of:
 - *type 1*: where truck moves alone on path ij to serve j
 - *type 3*: where truck moves alone on path ij to serve j and also drone lands on the truck at node j ; makes node j a rendezvous node
- If the last sequenced node i is *type 2*, then the next node j to be sequenced may only be:
 - *type 3*: which is a rendezvous node
- If the last sequenced node i is *type 3*, then the next node j to be sequenced may only be one of:
 - *type 3*: where truck and drone move together along path ij
 - *type 2*: where drone alone has a sortie along ij to serve j and lands on the truck at another node.
 - *type 4*: where drone launches from i to j and returns to i
- If the last sequenced node i is *type 4*, then the next node j to be sequenced may only be one of:
 - *type 3*: which is a rendezvous node
 - *type 2*: where drone alone has a sortie between ij to serve j and lands on another node.

3.1.4.2 Selection process

In a traditional TSP route construction involves sequencing each customer node through an array with generally starting from depot and returning to depot. Assume that node i is the last node sequenced in the array and the algorithm will select a node j as to become next sequenced. This possible node j can be any node that is not sequenced before in the array. Possibility of being selected as the next visiting node in the route is only depends on the node j should be unsequenced before. In the presented algorithm for TSP-D, possibility of being selected is not only depends on being unsequenced, but also depends on some other restrictions about transportation types defined above.

Once ant k is located at node i , all possible paths from i to j first has to be determined. These possible paths are determined according to following requisites: If the transportation type of

node i is:

- type 2, type 3, or type 4 then node j may be either a truck node or a drone node. So that, node j may be any node selected from $TDarray$ and $DDarray$ which is not sequenced before. All unsequenced nodes are candidates to become j .
- type 1 then node j may be a truck node or rendezvous node. So that, node j may be any node selected only from $TDarray$ which is not sequenced before. All unsequenced nodes in $TDarray$ are candidates to become j .

Each candidate path i to j should then be evaluated to determine which will be selected. This evaluation depends on a probability of selection formulation proportional to the pheromone amount on the ij path and distance between ij path. First, a selection value v_{ij} is calculated for each candidate path ij as given in equation (3.1).

$$v_{ij} = \begin{cases} P_{t_{ij}}^{\alpha} \times 1/d_{ij}^{\beta}, & j \in TD \text{ array} \\ P_{d_{ij}}^{\alpha} \times 1/d_{ij}^{\beta}, & j \in DD \text{ array} \end{cases} \quad (3.1)$$

If candidate node j is an element of $TDarray$ then the pheromone amount $P_{t_{ij}}$ on truck path ij is considered in calculation, otherwise drone pheromone amount $P_{d_{ij}}$ on drone path ij is considered. Euclidian distance of ij is given by d_{ij} . α and β are parameters predetermined to regulate influences of pheromone amount and distance on the selection probability. These parameters can take a value between 0 and 1.

After calculation of v_{ij} for each possible path ij , the probability that node j will become the following customer node is calculated by a probability equation given in Equation (3.2), where each candidate node j will have a value between 0 and 1.

$$p(i, j) = \frac{v_{ij}}{\sum v_{ij}} \quad (3.2)$$

A random number r which can take a value between 0 and 1 is generated by the algorithm. After computing each $p(i, j)$ values for each possible path, algorithm starts to take the cumulative sum S of $p(i, j)$ values until $S \geq r$. At which $p(i, j)$ value S exceeds r , that j is selected to be next sequenced node into the solution array.

An example to route construction for a TSP-D problem with 9 customers (0 is depot) is illustrated

followingly. Consider that Figure 3.5 presents generated arrays for a solution of this problem in the GA-AS algorithm. Assume that chromosome k is randomly generated in GA as shown in the figure. Correspondingly TDarray k and DDarray k are generated to define which customer node will be served by which vehicle according to gene values of chromosome k . Simultaneously, solution array k and transportation array k are generated which will define a final route and type of each node in the route.

At the beginning first and last element of solution array k has directly “0” which represents depot node. Besides transportation type of depot can only be type 3; as the assumption of truck and drone must located together in the depot at the beginning and must return to the depot at the end of route. So that “3” is initially placed into first and last element of transportation type array k .

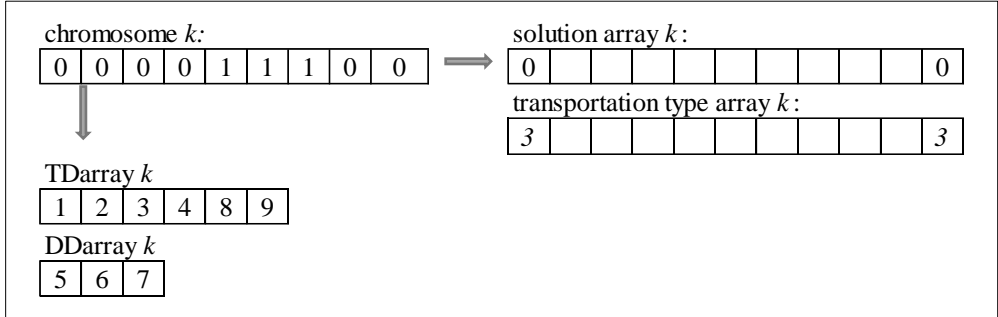


Figure 3.5: An example to solution construction in GA-AS: stage 1

After a random solution generated, next step in algorithm is to sequence these determined nodes as a TSP-D route. In Figure 3.6, initial solution array k is given where ant k is located currently at the depot. Following step is to determine candidate nodes for j which will be next visiting

	ant k location	possible node to be selected
customer node	i 0	j 1,2,3,4,5,6,7,8,9
transportation	type3	type2/type3/type4
↓ candidate ij paths		
unsequenced nodes for j from <i>TDarray</i>		0-1
		0-2
		0-3
		0-4
		0-8
		0-9
unsequenced nodes for j from <i>DDarray</i>		0-5
		0-6
		0-7

Figure 3.6: An example to solution construction in GA-AS: stage 2

node in the route. Since current node 0 is type 3, transportation type of candidate node j can be any of type 2, type 3 or type 4 which means any unsequenced node from TDarray k and DDarray k is possible to evaluate.

After determining candidate paths, each should be evaluated for selection. Firstly, selection values, then secondly probability of selection values are computed for each possible path as exemplified in Figure 3.7. Thus, each candidate node j gets a value between 0-1.

	ant k location	possible node to be selected	selection value	probability of selection
	i	j	v_{ij}	$p(i, j)$
customer node	0	1,2,3,4,5,6,7,8,9		
transportation	type3	type2/type3/type4		
	↓ candidate ij paths →		evaluating candidate ij paths	
unsequenced nodes for j from TDarray		0-1	$v_{01} = P_{t_{01}}^{\alpha} \times 1/d_{01}^{\beta}$	$p(0,1) = v_{01}/\Sigma v_{ij}$
		0-2	$v_{02} = P_{t_{02}}^{\alpha} \times 1/d_{02}^{\beta}$	$p(0,2) = v_{02}/\Sigma v_{ij}$
		0-3	$v_{03} = P_{t_{03}}^{\alpha} \times 1/d_{03}^{\beta}$	$p(0,3) = v_{03}/\Sigma v_{ij}$
		0-4	$v_{04} = P_{t_{04}}^{\alpha} \times 1/d_{04}^{\beta}$	$p(0,4) = v_{04}/\Sigma v_{ij}$
		0-8	$v_{08} = P_{t_{08}}^{\alpha} \times 1/d_{08}^{\beta}$	$p(0,8) = v_{08}/\Sigma v_{ij}$
		0-9	$v_{09} = P_{t_{09}}^{\alpha} \times 1/d_{09}^{\beta}$	$p(0,9) = v_{09}/\Sigma v_{ij}$
unsequenced nodes for j from DDarray		0-5	$v_{05} = P_{d_{05}}^{\alpha} \times 1/d_{05}^{\beta}$	$p(0,5) = v_{05}/\Sigma v_{ij}$
		0-6	$v_{06} = P_{d_{06}}^{\alpha} \times 1/d_{06}^{\beta}$	$p(0,6) = v_{06}/\Sigma v_{ij}$
		0-7	$v_{07} = P_{d_{07}}^{\alpha} \times 1/d_{07}^{\beta}$	$p(0,7) = v_{07}/\Sigma v_{ij}$

Figure 3.7: An example to solution construction in GA-AS: stage 3

Consider that $p(0,8)$ value where S exceeds r , is found by the algorithm; then $j=8$ will be selected to be the next node in sequence. Thus, second location of solution array k is placed by “8”. Figure 3.8 illustrated current solution array k and corresponding TSP-D graph after determining first assignment to route construction. Since node 8 belongs to TDarray k , actually indicates the 8.

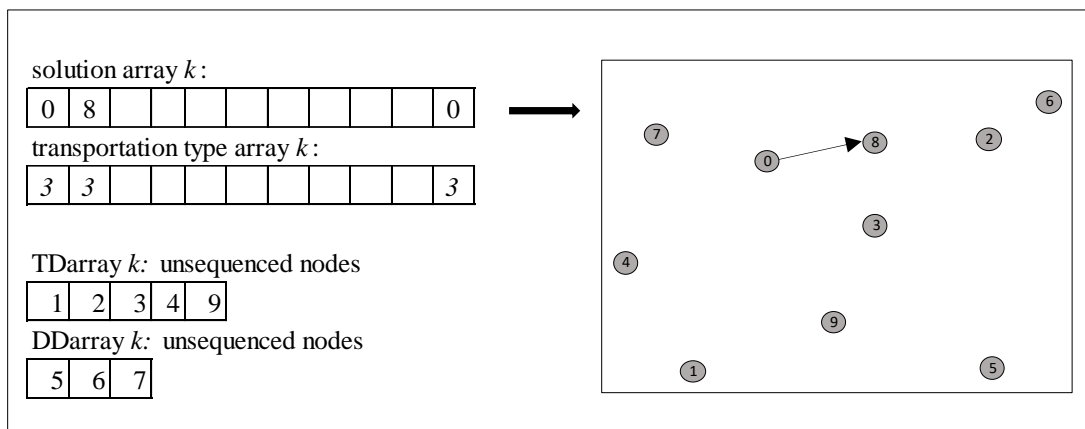


Figure 3.8: An example to solution construction in GA-AS: stage 4

customer will be visited and delivered by truck. Moreover, drone is still carried by truck owing to any drone delivery node has not been selected yet. These concludes transportation type of node 8 is type 3 so, “3” is placed in second location of transportation type array k corresponding to its node’s sequence. Unsequenced nodes are also updated.

Consider the algorithm selects next visiting nodes as $j=2$ and $j=6$ respectively into the sequence by following same process defined. Since node 6 belongs to DDarray k which indicates the 6. customer is visited and delivered by drone. Once a drone delivery node is found, transportation type has to be distinguished as type 2 or type 4. Current route construction is illustrated in Figure 3.9.

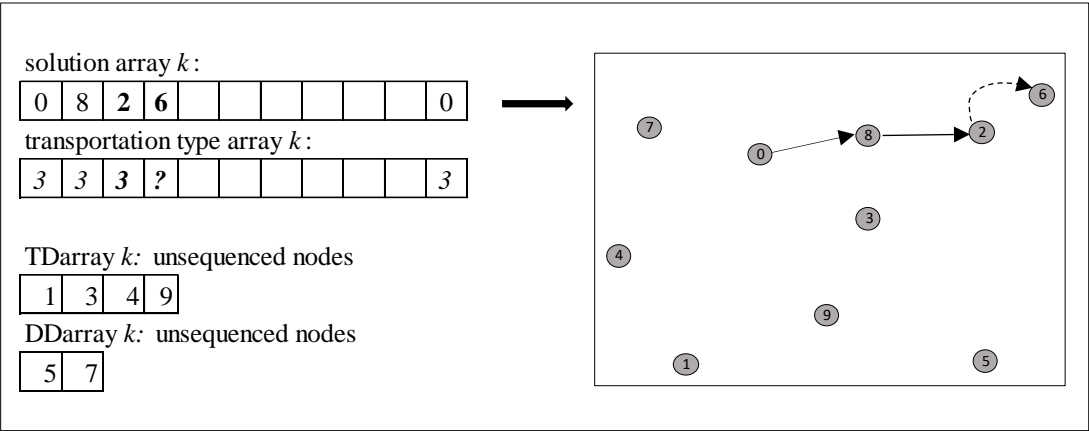


Figure 3.9: An example to solution construction in GA-AS: stage 5

There is an assumption of problem that drone cannot serve more than one customer at each time it has a delivery. Once drone has delivered a customer it must return to truck to be recharged and get a new parcel for its next delivery. After drone has its visit to node 6, now the algorithm should try to find a rendezvous node for drone to meet with truck (see Figure 3.10). So that

	ant k location	possible node to be selected
	i	j
customer node	6	1,3,4,9,2
transportation	type2/type 4	type3
↓ candidate ij paths		
unsequenced nodes for j from <i>TDarray</i>		6-1 6-3 6-4 6-9
last i node (launching node)		6-2

Figure 3.10: An example to solution construction in GA-AS: stage 6

selection process continues searching on current node 6. All truck delivery nodes from unsequenced TDarray k are candidate for next j . In addition, a candidate node will also be drone's last launch node which is node 2. If a truck delivery node is selected, then node 6 will be type 2. In the case of any truck delivery node is not selected, drone returns to its launch node i which will make node 6 as type 4.

Consider that $p(6,2)$ value where S exceeds r , is found by the algorithm; then $j=2$ will be selected which determines accurately node 6 as type 4. The current sequence and corresponding route on a graph is updated as illustrated in Figure 3.11.

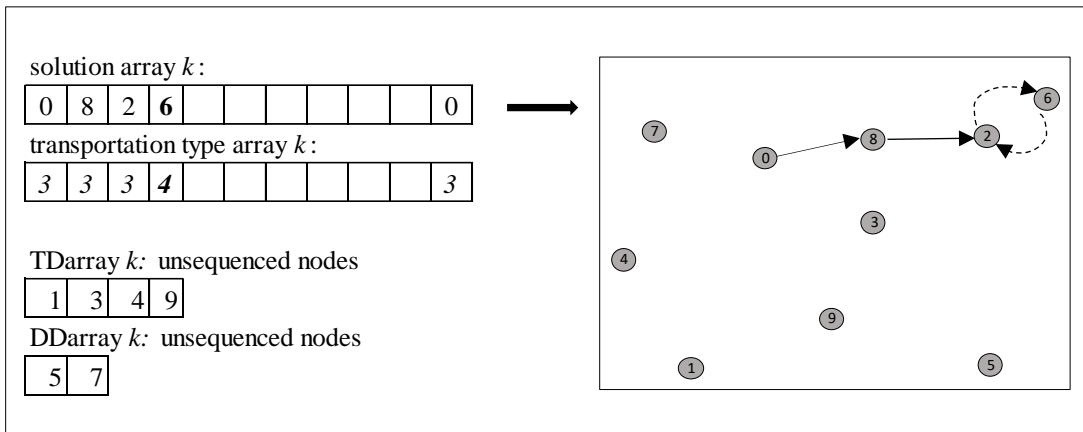


Figure 3.11: An example to solution construction in GA-AS: stage 7

Algorithm always continues on searching from node i where truck is last located, after a drone delivery is finished. Currently on the given example, truck and drone located together at node 2 which is type 3. So that candidate nodes for next j can be again any unsequenced node both from TDarray and DDarray. Candidate paths are given in Figure 3.12.

	ant k location	possible node to be selected
	i	j
customer node	2	1,3,4,9,5,7
transportation	type3	type2/type3/type4
↓ candidate ij paths		
unsequenced nodes for j from <i>TDarray</i>		2-1 2-3 2-4 2-9
unsequenced nodes for j from <i>DDarray</i>		2-5 2-7

Figure 3.12: An example to solution construction in GA-AS: stage 8

After evaluating candidate paths through probability of selection process, consider that $p(2,5)$ value where S exceeds r , is found by the algorithm; then $j=5$ will be selected to be next node into the sequence. Updated route construction is illustrated in Figure 3.13.

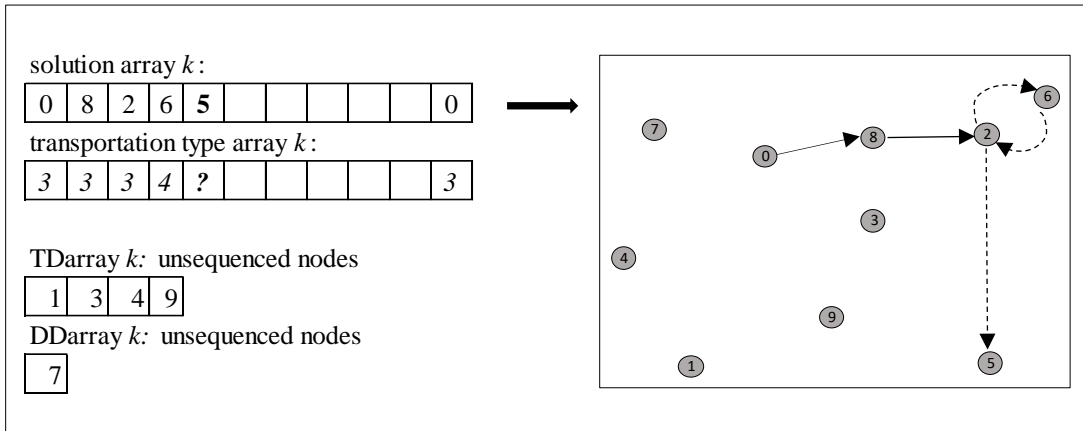


Figure 3.13: An example to solution construction in GA-AS: stage 9

Once again, a drone delivery node is selected, algorithm searches to figure out transportation type 2 or type 4 (see Figure 3.14) by following the same process defined above.

	ant k location	possible node to be selected
customer node	i 5	j 1,3,4,9,2
transportation	type2/type4	type3
↓ candidate ij paths		
unsequenced nodes for j from $TDarray$		5-1 5-3 5-4 5-9
last i node (launching node)		5-2

Figure 3.14: An example to solution construction in GA-AS: stage 10

Considering this once, selection process come out with $p(5,9)$ value where S exceeds r that makes $j=9$ as the next node into the sequence. Since node 9 is from $TDarray k$, it is a truck delivery node. That indicates drone serves node 5 than will have a sortie to meet with truck at node 9. Thus, node 9 is defined as a rendezvous node in the algorithm. Finding a rendezvous node also determines transportation type of node 5 as type 2. (See updated route construction in Figure 3.15).

When a rendezvous node has been found, drone continues its sortie or waits at rendezvous node

until the same node is found for the truck. Algorithm continues on searching at node i where truck is located. When the rendezvous node 9 is found also for truck, then it will be sequenced in the solution array k .

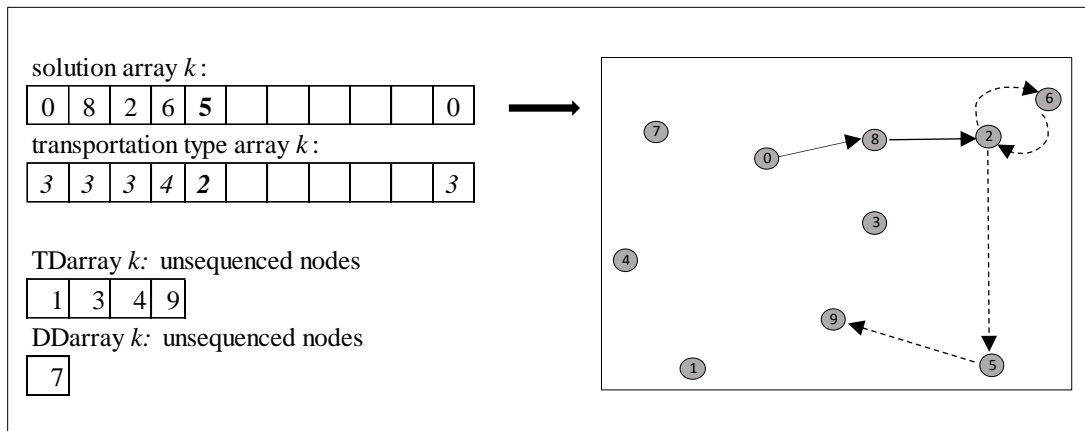


Figure 3.15: An example to solution construction in GA-AS: stage 11

The algorithm always follows last locations of truck and drone during route construction. As be seen by TSP-D graph in Figure 3.15, drone is waiting to meet with truck at its last location node 9 while truck is having its own delivery at node 2. To be next step, truck's new location has to be found in the current route as drone cannot have any new delivery until meeting with truck again. Thus, algorithm continues on searching from node 2. When last locations of drone and truck at different nodes in the algorithm, last truck location behaves as type 1 in searching (whether it is type 3 actually). As defined in restrictions; when a node i is type 1, node j may only be a truck delivery node (type 1 or type 3). Thus, candidate nodes are only searched through TDarray k (see Figure 3.16).

	ant k location	possible node to be selected
	i	j
customer node	2	1,3,4,9
transportation	type3	type1/type3
↓ candidate ij paths		
unsequenced		2-1
nodes for j from		2-3
$TDarray$		2-4
		2-9

Figure 3.16: An example to solution construction in GA-AS: stage 12

Consider node 3 selecting this once to be next visiting node. That means rendezvous node has still not selected. Thus, truck travels and serve alone to node 3 which makes it as type 1. Current

route after this selection is seen in Figure 3.17.

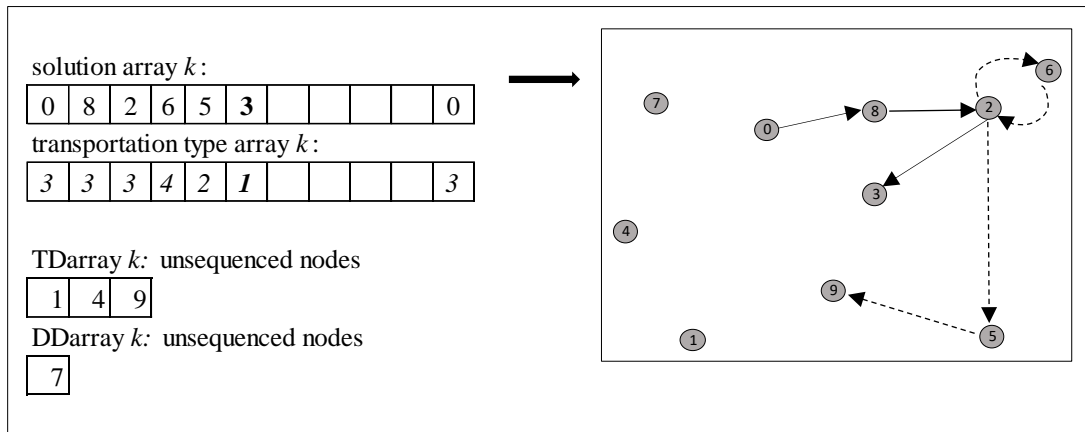


Figure 3.17: An example to solution construction in GA-AS: stage 13

The next node selection continues from last node 3 as given in Figure 3.18.

	ant k location	possible node to be selected
customer node	i 3	j 1,4,9
transportation	type1	type1/type3
↓ candidate ij paths		
unsequenced nodes for j from $TDarray$		3-1 3-4 3-9

Figure 3.18: An example to solution construction in GA-AS: stage 14

Considering this once the rendezvous node is found to sequence, then algorithm updates last location of both truck and drone as node 9 which they meet at (see Figure 3.19).

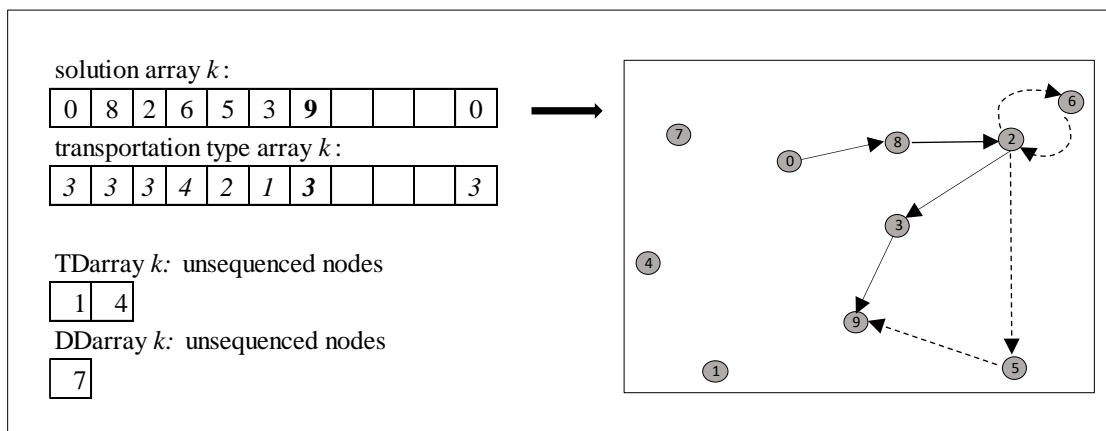


Figure 3.19: An example to solution construction in GA-AS: stage 15

Algorithm repeats this framework until all customer nodes will have been ordered in solution array k . Remaining route construction illustration can be followed in Figure 3.20.

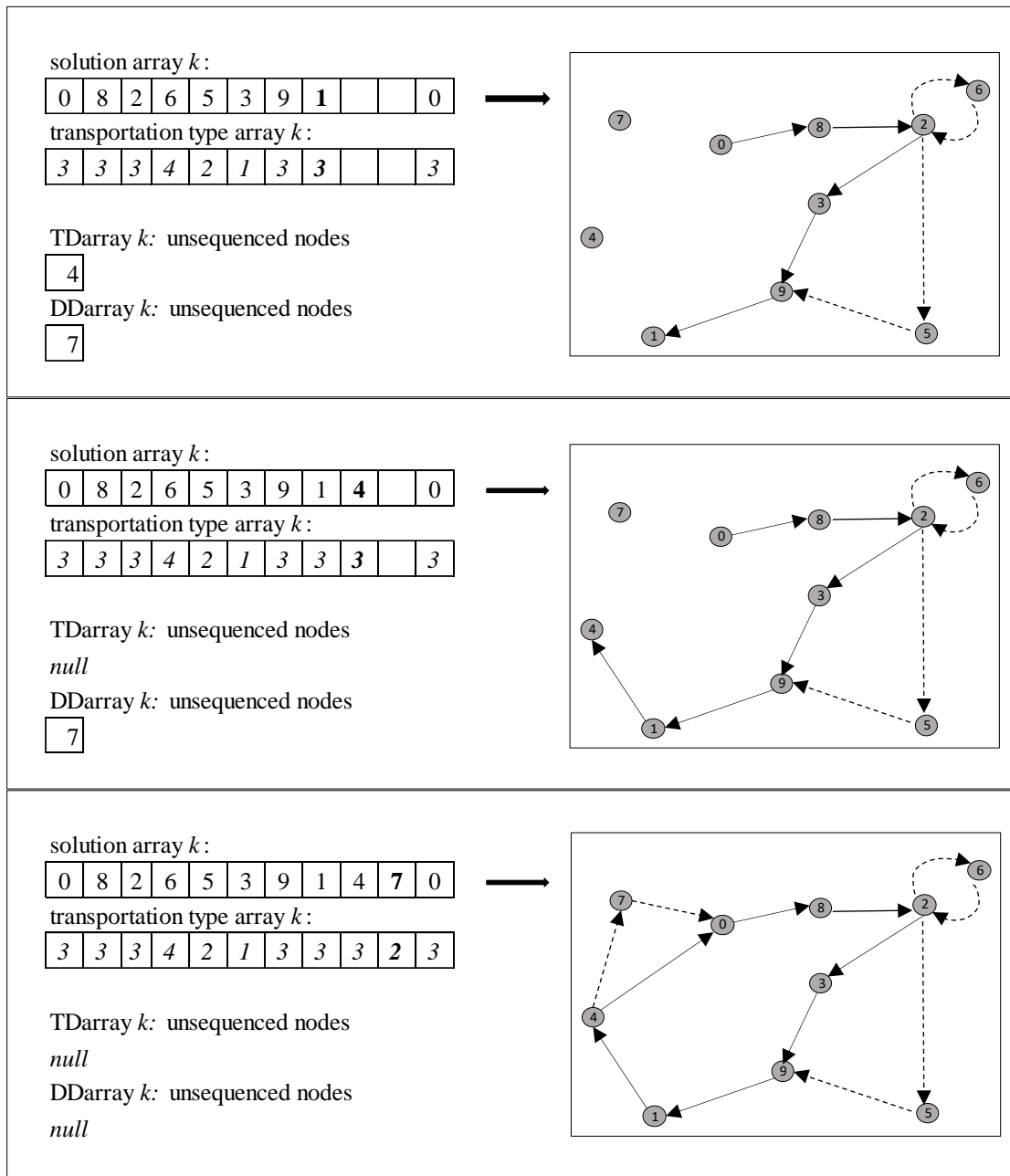


Figure 3.20: An example to solution construction in GA-AS: stage 16, 17, 18

Each solution array k presents a TSP-D route, which corresponds to chromosome k . As the N numbers of chromosome in a GA population, N numbers of solution array constructs N numbers of TSP-D routes. This is formed in a single iteration of algorithm.

On account of defined route construction framework of proposed algorithm, *both truck and drone routes are generated simultaneously*. In proposed hybrid metaheuristic, GA algorithm

generates many separate node assignments to truck and drone while AS algorithm constructs feasible routes for these nodes of each GA individual. And the optimization eventualize through generations of the GA-AS algorithm.

3.1.5 Evaluation

After a TSP-D route is constructed of the nodes in chromosome k , now that the route cost has to be found. This cost calculation is held by the individual evaluation step as called fitness measurement of GA algorithm. Prior to evaluation, to calculate a route's cost some data has to be provided to algorithm. Since the objective function of the problem is to minimize total completion time of the tour, then primarily the time required to travel between the nodes should be known both for truck and drone. The required data is distances between nodes to each other and the speed of vehicles. To compute distances an input data matrix consists of x- and y-coordinates of each customer node is provided to algorithm. The distance matrix d is generated by transforming the coordinate matrix, where each d_{ij} represents Euclidean distance between customer nodes i and j . Since truck speed S_t and drone speed S_d can be different, costs are computed over time traveled instead of distance traveled. So that, two cost matrices are generated in the algorithm: *truck cost matrix* T_C and *drone cost matrix* D_C which respectively consists of time taken to travel of truck and time taken to travel of drone between nodes. Each $T_{C_{ij}}$ value of T_C matrix presents the total delivery time of truck departs from customer i and arrives to customer j . Each $D_{C_{ij}}$ value of D_C matrix presents the total sortie time of drone launches from customer i and lands to customer j . Computation of these values are given in Equation (3.3) and (3.4) respectively.

$$T_{C_{ij}} = \frac{d_{ij}}{S_t} \quad (3.3)$$

$$D_{C_{ij}} = \frac{d_{ij}}{S_d} \quad (3.4)$$

Each chromosome k in a GA population gets a fitness measurement value f_k during the evaluation step of algorithm. To measure the fitness of a chromosome k in a GA population, the total cost ($TCost_k$) of the corresponding TSP-D tour is found in *solution array* k should be computed. Total cost of a tour is computed followingly. Let customer i is the launch node while customer j is the subsequent visiting node. If node j is a type 1 or type 3 node, then $T_{C_{ij}}$ value is

added to $TCost_k$. If node j is a type 4 node, then $D_{C_{ij}}$ value is multiplied by two and added to $TCost_k$. If node j is a rendezvous node, then the maximum one of $T_{C_{ij}}$ and $D_{C_{ij}}$ value is added to $TCost_k$. After the algorithm computes total cost, the fitness value of each chromosome k is calculated by the Equation (3.5).

$$f_k = 1/TCost_k \quad (3.5)$$

3.1.6 Pheromone update

After each AS iteration ends, the pheromone amounts along ij paths are changed as called pheromone update. This may occur in two ways: pheromone deposit and pheromone evaporation. The amount of pheromone will be deposited proportionally to the frequency with which path ij appears as an edge of the tour found in the *solution array* k . The amount of pheromone that is left on a path ij is determined by how much of that path ij is taken by ant k .

On the contrary, evaporation is reducing an amount of pheromone on a path ij at each iteration ends. By using an evaporation constant e where $0 < e < 1$, the pheromone amount on path ij at iteration t is decreased in pheromone update of iteration $t+1$.

The pheromone update process has to be occurred in two cases, for both truck paths and drone paths independently, since we have established a two-pheromone framework in the algorithm.

The pheromone update process for truck can be specified as followingly. Let a path ij appears in *solution array* k , and that node j is included in *TDarray* k , indicates that the path ij is traveled by the truck. So, it is required to update the pheromone amount P_{tij} on that truck path ij . Equation (3.6) gives the formula to update the pheromone amount on a truck path at iteration $(t+1)$.

$$P_{tij}(t+1) = P_{tij}(t) e + \sum_{k=1}^N \Delta P_{tij}^k \quad (3.6)$$

Equation (3.7) computes ΔP_{tij}^k which is the amount of pheromone change on truck path ij . That is inversely proportional to the total cost of the route which is defined previously as fitness value and formulated as $f_k = 1/TCost_{P_k}$.

$$\Delta P_{tij}^k = \begin{cases} f_k, & j \in TDarray \text{ and path } ij \in solutionarray_k \\ 0, & otherwise \end{cases} \quad (3.7)$$

The pheromone update process for drone can be specified as followingly. Let a path ij appears in *solution array* k , and that node j is included in *DDarray* k , indicates that the path ij is taken by the drone. So, it is required to update the pheromone amount $P_{d_{ij}}$ on that drone path ij . Equation (3.8) gives the formula to update the pheromone amount on a drone path at iteration $(t+1)$.

$$P_{d_{ij}}(t+1) = P_{d_{ij}}(t) e + \sum_{k=1}^N \Delta P_{d_{ij}}^k \quad (3.8)$$

Equation (3.9) computes $\Delta P_{d_{ij}}^k$ which is the amount of pheromone change on drone path ij . That is inversely proportional to the total cost of the route which is defined previously as fitness value and formulated as $f_k = 1/TCost_{P_k}$.

$$\Delta P_{d_{ij}}^k = \begin{cases} f_k, & j \in DDarray \text{ and path } ij \in solutionarray_k \\ 0, & otherwise \end{cases} \quad (3.9)$$

3.1.7 Parent selection

In each GA-AS iteration, subsequently the evaluation of chromosomes, a selection mechanism is driven to determine the ones to become parents and to go through reproduction. Individuals in a GA population are chosen by using the *roulette wheel method*, which is a fitness-proportionate selection, to join the mating pool and become parents. Roulette wheel selection is a commonly used method for selecting parent solutions to the reproduction step of a GA. The method is based on the metaphor of a roulette wheel, where each slice of the wheel corresponds to a solution in the population, and the size of each slice is proportional to the fitness of the corresponding solution. Such that higher-fitness solutions have larger slices on the wheel. In this selection method, the least-fit individuals that has a worse fitness value, also have a possibility to be selected as parents, resulting in genotypic diversity that prevents algorithm convergence.

The GA-AS algorithm applies *roulette wheel method* as followingly defined. Each chromosome k in N numbered genetic population, is assigned by a *fitness probability* value $p(f_k)$ which defines the probability of selection as become a parent. $p(f_k)$ is proportional to chromosome k 's fitness relative to the population's overall fitness. That normalizes the fitness values, so they sum to 1. Equation (3.10) is applied to each chromosome k to assign a fitness probability.

$$p(f_k) = \frac{f_k}{\sum_{k=1}^N f_k} \quad (3.10)$$

Subsequently assigning each $p(f_k)$ value to the population, to select a parent solution, the algorithm then spins the roulette wheel by generating a random number r between 0 and 1. Then, algorithm takes cumulative sum S of $p(f_k)$ values until $S \geq r$. At which $p(f_k)$ value S exceeds r , that chromosome k is selected into the mating pool to become parent. Which imitates the solution corresponding to the segment that the random number falls into is selected as the parent. Roulette wheel is spined N times as the population size. Once a chromosome is selected into the mating pool, it has a probability to be selected again. In this case, individuals with higher fitness values will have larger segments on the roulette wheel compared to those with lower fitness values, which increases their chances of being selected when the wheel stops. Over N spins, individuals with higher fitness values will be more numerous in the parent pool compared to others and will generate more offspring individuals. However, it still allows lower-fitness solutions to have a chance of being selected. This helps to maintain diversity in the population.

3.1.8 Crossover and mutation

The GA-AS algorithm applies crossover and mutation operators to randomly chosen two parent chromosomes from mating pool which are determined through the selection process. The method to be used as crossover and mutation operator varies depending on the problem type. As it is customary for most of combinatorial optimization problems, solutions of TSP/TSP-D by GA also need to be represented by permutation encoding.

Permutation encoding represents each potential solution as a sequence of values. For instance, in the TSP, a solution is represented as a sequence of cities that defines the order in which they are visited. Thus, this encoding also results in a necessity on development of appropriate crossover and mutation operators to ensure the permutation.

However, chromosomes of proposed GA-AS algorithm represent solutions with the unusual use of binary coding in permutation problems, which results in eliminating that requirement of using permutation crossover operators while also reducing computational effort. So that a common method, *single-point crossover* for binary GAs is applied as the crossover operator.

The single-point crossover operator works by selecting a random crossover point along the length of two parent chromosomes and swapping the gene sequence of two parents to create two offspring solutions. According to selected crossover point position in the chromosomes, the

offspring will inherit the genes before this point from one parent and the remaining genes from the other parent. So that, two offspring chromosomes are generated from two parent chromosome. *TDarray* and *DDarray* are also recomposed due to new gene sequence of offspring chromosomes. Figure 3.21 exemplifies a crossover process used in the GA-AS.

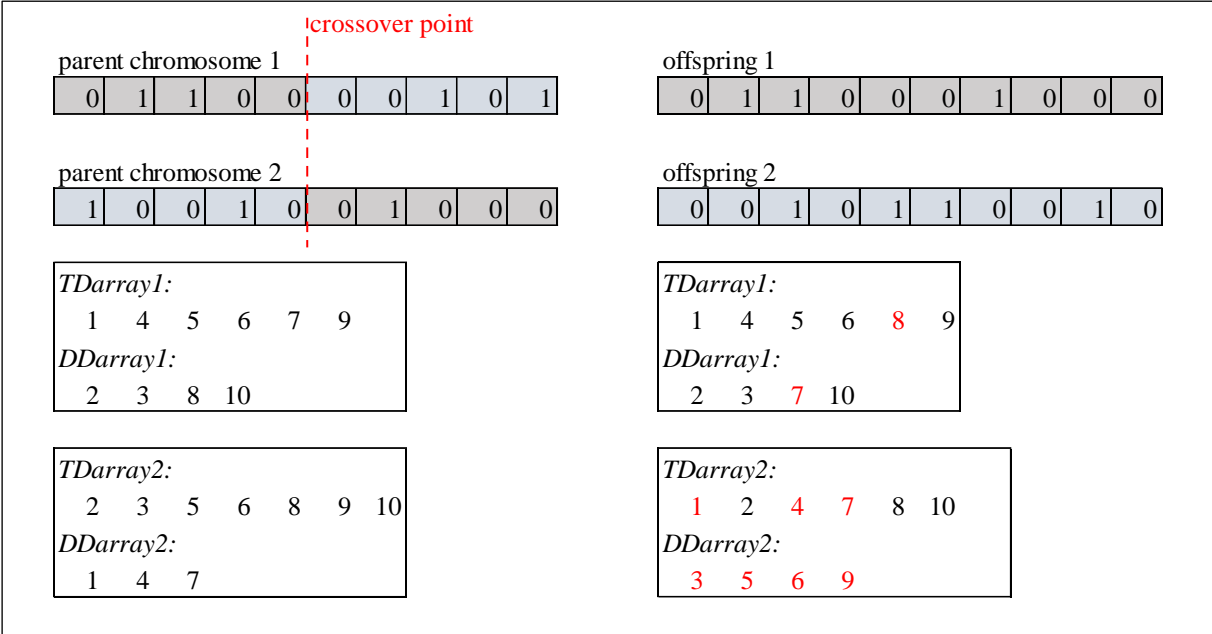


Figure 3.21: Crossover operator in GA-AS

After generating offsprings through crossover mutation is applied to some of them due to the predetermined probability. The purpose of this mutation operator is to explore different regions of the search space to potentially discover better solutions. In the GA-AS a single-bit mutation is used which is a simple and commonly used operator for introducing genetic diversity into a population. It involves flipping the value of a randomly selected bit within an individual's binary string representation. *TDarray* and *DDarray* are also recomposed due to changed gene of offspring chromosome. Figure 3.22 exemplifies a mutation process used in the GA-AS.

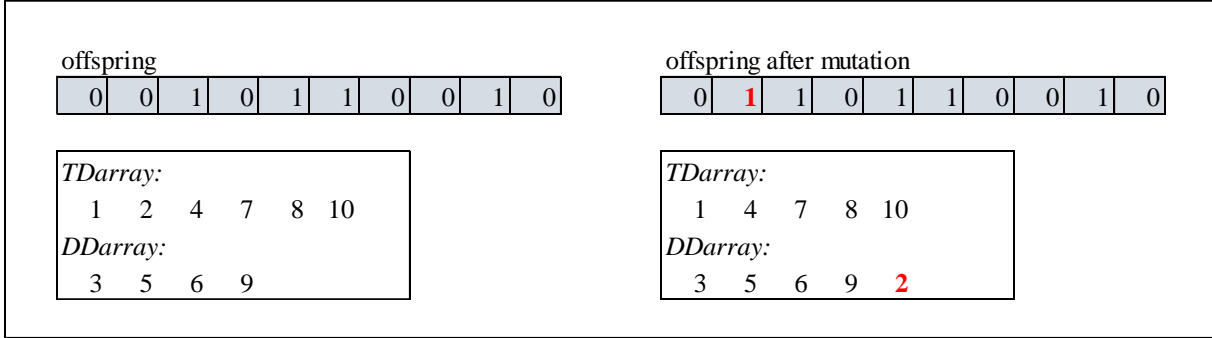


Figure 3.22: Mutation operator in GA-AS

4. COMPUTATIONAL EXPERIMENTS

The computational experiments focus on the evaluation of the proposed algorithm, GA-AS, for solving the TSP-D. Through a comprehensive set of computational experiments and analysis, this chapter seeks to provide insights into the capabilities and limitations of GA-AS, contributing to the advancement of solution methods for TSP-D. It is aimed to assess the algorithm's effectiveness in finding solutions and to compare its performance against existing state-of-the-art approaches.

In the following sections, primarily experimental factors will be systematically determined, later on the performance of GA-AS will be investigated on benchmark instances, numerical results and statistical analysis will be presented.

4.1 Parameter Setting

In order to evaluate the performance of the proposed GA-AS algorithm, it is crucial to determine primarily appropriate parameter settings for both the GA and the ACO components. The parameter settings play a vital role in achieving better results and ensuring the effectiveness of the algorithms. This section describes the parameter setting process for GA-AS explaining the rationale behind the choices. The fundamental parameters of GA-AS are population size, crossover and mutation rate, evaporation rate, α and β values.

- **Population Size:** determines the number of individuals in each generation of GA. Different population sizes are experimented ranging from small to large, to assess their influence on the algorithm's performance. The population sized set to 100 for GA-AS algorithm since experimental trials have been showed that any more increase in size incurs only with additional computational time while does not improve solution quality.
- **Crossover Rate:** determines the probability of performing crossover in GA-AS, which combines genetic information from two parent individuals to create offspring. Different crossover rates have been investigated to identify a value that balances exploration and exploitation.
- **Mutation Rate:** controls the probability of introducing random changes to individual solutions, maintaining diversity within the population. Different mutation rates have been examined to ensure a proper balance between exploration and exploitation, preventing premature convergence.

- **Pheromone Evaporation Rate:** controls the rate at which pheromone trails decay over time. It helps to balance the influence of the previous iterations' pheromone trails with the newly discovered solutions. Different evaporation rates have been tested to find the optimal balance between exploitation and exploration.
- α : controls the importance of the pheromone trail. A higher value of alpha means that the ant is more likely to select the customer node with the higher pheromone level, indicating that it has been visited frequently in the past. This results in a more exploitative search, where the ants are more likely to converge to a single solution.
- β : controls the importance of the heuristic information, which is the distance between customer nodes. A higher value of beta means that the ant is more likely to select the node that is closer to the current node. This results in a more exploratory search. Thus, it is aimed to determine suitable values for these parameters, ensuring that the ants can effectively exploit the available information to guide their search.

The selected values to be experimentally tested of these GA-AS parameters are given in Table 4.1 as parameter levels. Previous studies in literature have been lead to select these parameter values in the table (Dorigo et al., 1996; Srinivas & Patnaik, 1994).

Table 4.1: Determined parameter levels for design of experiment

Parameter	Level 1	Level 2	Level 3
crossover rate	0.6	0.7	0.8
mutation rate	0.1	0.2	0.3
evaporation rate	0.5	0.7	0.9
α	1	2	3
β	3	4	5

After determining parameter values for experiment, the Taguchi method, a statistical design of experiments technique, has been employed to ensure a robust parameter setting. The Taguchi method allows for a systematic and efficient exploration of the parameter space while minimizing the number of experiments required. By applying this method, it is aimed to identify the optimal parameter settings that would enhance the performance of the proposed algorithm.

A test problem “singlecenter-62-n20” is selected from Bouman et al. (2018b) to be used in parameter setting experiments. Taguchi method is applied over a statistical software package, Minitab. Taguchi method selected L27 orthogonal order to represent the combinations of

parameter settings to be tested, due to defined parameter levels in Table 4.1. The results of experimental design were then subjected to the signal-to-noise (S/N) ratio analysis. S/N values and graphically effects of parameter levels for the conducted experiment by Taguchi is given in Figure 4.1.

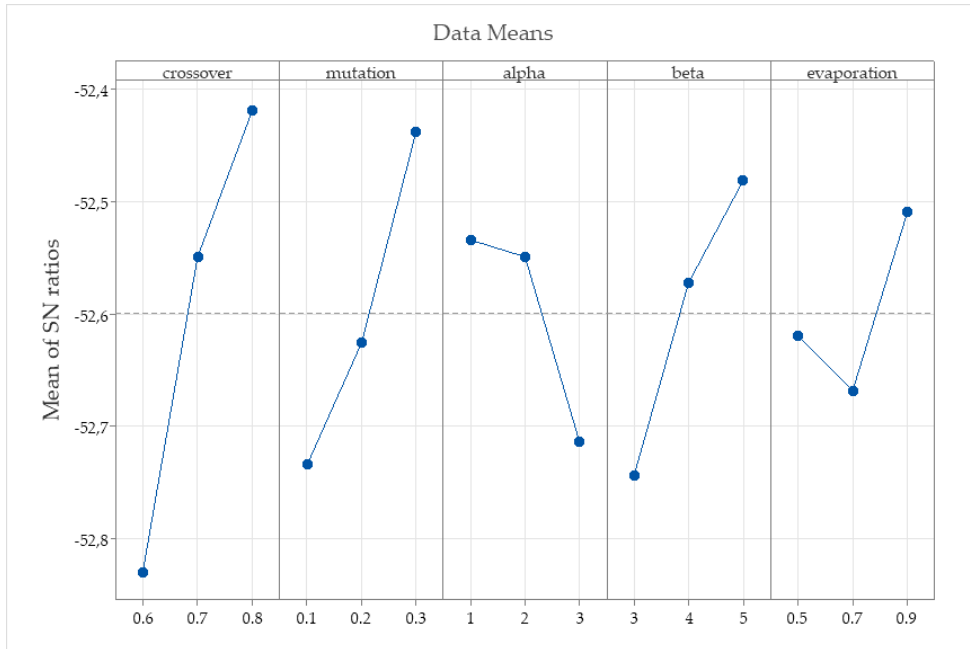


Figure 4.1: S/N analysis results

Maximizing the S/N value helps to identify the most influential parameters and their optimal levels. The highest S/N values in the analysis results indicate the best parameter combinations. Once the parameter setting process using the Taguchi method was completed, the finalized parameter values of GA-AS were obtained and set as given in Table 4.2.

Table 4.2: Parameter Setting in GA-AS

Parameter	Value
Crossover rate	0.8
Mutation Rate	0.3
Evaporation constant	0.9
α	1
β	5

Through extensive experimentation, the parameter settings were established for GA-AS that provided the best performance on the problem domain. These settings were then utilized in the subsequent computational experiments to evaluate the effectiveness and efficiency of the

proposed GA-AS algorithm.

4.2 Computational Experiments on TSP-D Instances

In this section, benchmark instances specifically designed for TSP-D will be utilized to rigorously evaluate the performance of the proposed GA-AS algorithm. Commonly used benchmarking problems with their known optimal solutions are merely available since the TSP-D is a problem that has been very recently studied in the literature and continues to be researched. Yet, a set of benchmark instances designed for the TSP-D are provided by the work of Bouman et al. (2018b). These instances provide a standardized platform to assess algorithm performance across a range of problem scenarios. They enable researchers to compare the performance of different algorithms under controlled conditions and facilitate the advancement of solution approaches for the TSP-D.

This conducted experiment uses test problems of Bouman et al. (2018b) where n is from 5 to 75 and $\alpha=2$. Each n -sized data set contains 10 different sub-problem instances.

Bouman et al. (2018b)'s instances following the TSP-D assumptions in its origin which are defined before in problem description by Section 1.2.1. By respecifying;

- each customer node is eligible to delivery by both truck and drone,
- service time of both truck and drone are neglected and set to 0,
- drone launch time is also set to 0,
- drone's endurance is thought to be infinite.

The conducted experiments also include these additional assumptions:

- relative drone/truck speed is equals to 2, which means the drone is twice as fast as the truck.
- distances between a pair of customer nodes follow the Euclidean metric.

Following sections represents results of the computational evaluation which is conducted in two separate test categories. Primarily, in Section 4.2.1, the algorithm is evaluated through instances of which optimal solutions are known. In Section 4.2.2, the proposed algorithm is compared against the existing algorithms which have solutions on the same benchmarking instances for larger-sized instances.

4.2.1 Results of GA-AS algorithm for optimal TSP-D solutions

In their study, Bouman et al. (2018b) introduced a set of instances along with optimal solutions. These optimal solutions serve as reference points and provide the best-known solutions for those particular instances. It is worth mentioning that these optimal solutions were derived using the IP formulation of the problem. Therefore, they provide optimal solutions for those benchmarking instances; only for problems involving a maximum of 9 customers ($n=9$). The dataset consists of 10 distinct instances for each problem size n . The evaluation of the GA-AS algorithm in this section, a total of 50 “single-center” instances were selected from data set, using $\alpha=2$ indicates the relative drone/truck speed. The results of the computational experiments are presented in Table 4.3.

The first column of Table 4.3 provides the instance IDs used in the experiments, while the "n" column represents the size of each instance. The optimal solutions for each problem instance obtained from Bouman et al. (2018b), are presented in the "opt" column. The "GA-AS" column reports the best solution cost in terms of time achieved by the GA-AS algorithm for each instance. “SR%” indicates the success rate of the algorithm which is a metric that measures the proportion of successful outcomes. In the context of this experiment, it indicates how often the algorithm obtained optimal solutions over 10 runs. The computation time of algorithm in terms of seconds is also reported in “time” column. To assess the performance of the GA-AS algorithm, the "gap%" column computes the percentage difference between the best solution found by GA-AS and the optimal solution for each problem instance. This comparison allows for an evaluation of how close GA-AS comes to the optimal solutions for the instances under consideration.

In terms of numerical results, the GA-AS algorithm has exhibited success in finding optimal solutions, particularly for the instances where optimal solutions are available. Out of the 50 problem instances tested, GA-AS successfully identified the optimal solutions in 38 cases, while achieving results that were closely comparable to the optimal solutions in the remaining instances. The experimental findings revealed that GA-AS was able to generate TSP-D tours that were optimally close to the IP solutions, with an average gap of only 0.80%. This demonstrates the effectiveness of GA-AS in solving the TSP-D, showcasing its capability in producing optimal solutions for almost of the instances tested. Nonetheless, the results suggests that it consistently achieved a success rate of 100% in the majority of instances. This level of accuracy and reliability demonstrates the robustness of the GA-AS.

Table 4.3: Numerical results of GA-AS on instances from Bouman et al. (2018b) with optimal solutions.

Ins.	n	opt	GA-AS	SR%	gap %	time	Ins.	n	opt	GA-AS	SR%	gap %	time
1	5	154.26	154.26	100	0.00	0.45	26	7	98.71	98.77	100	0.06	0.91
2	5	140.54	140.54	100	0.00	0.45	27	7	177.86	177.86	100	0.00	0.91
3	5	52.67	52.92	100	0.46	0.45	28	7	169.77	170.97	100	0.71	0.91
4	5	108.94	108.94	100	0.00	0.45	29	7	193.34	193.34	100	0.00	0.91
5	5	122.15	122.15	100	0.00	0.45	30	7	177.10	177.10	100	0.00	0.91
6	5	162.22	162.22	100	0.00	0.45	31	8	155.20	155.48	100	0.18	1.09
7	5	133.95	133.95	100	0.00	0.45	32	8	107.20	107.20	90	0.00	1.09
8	5	81.50	81.50	100	0.00	0.45	33	8	172.85	177.06	100	2.44	1.09
9	5	143.15	143.15	100	0.00	0.45	34	8	226.91	226.91	100	0.00	1.09
10	5	140.40	140.40	100	0.00	0.45	35	8	188.46	188.46	100	0.00	1.09
11	6	128.01	128.01	100	0.00	0.61	36	8	181.37	181.37	100	0.00	1.09
12	6	125.94	125.94	100	0.00	0.61	37	8	134.62	134.62	100	0.00	1.09
13	6	215.91	215.91	100	0.00	0.61	38	8	286.71	286.71	100	0.00	1.09
14	6	119.24	147.71	100	23.87	0.61	39	8	181.07	181.07	100	0.00	1.09
15	6	169.62	169.62	100	0.00	0.61	40	8	214.90	217.51	100	1.21	1.09
16	6	117.84	117.84	100	0.00	0.61	41	9	116.93	116.93	100	0.00	1.36
17	6	263.03	263.03	100	0.00	0.61	42	9	316.25	318.84	100	0.82	1.36
18	6	258.65	258.65	100	0.00	0.61	43	9	226.28	237.68	100	5.04	1.36
19	6	188.80	188.80	100	0.00	0.61	44	9	228.28	233.03	100	2.08	1.36
20	6	122.17	122.17	100	0.00	0.61	45	9	279.66	279.66	80	0.00	1.36
21	7	208.34	208.34	90	0.00	0.91	46	9	214.02	216.19	100	1.01	1.36
22	7	178.38	178.38	100	0.00	0.91	47	9	277.73	282.98	100	1.89	1.36
23	7	116.24	116.24	100	0.00	0.91	48	9	200.95	200.96	100	0.00	1.36
24	7	152.59	152.59	100	0.00	0.91	49	9	349.49	349.49	100	0.00	1.36
25	7	130.50	130.50	100	0.00	0.91	50	9	196.84	196.84	100	0.00	1.36

4.2.2 Comparison results of GA-AS algorithm with rival algorithms

This section presents an assessment of the performance of the GA-AS algorithm on larger-sized problem instances provided in Bouman et al. (2018b). For the experiment, 40 instances from “single-center” problem category and 40 instances from “uniform” problem category were

utilized; overall a total of 80 problem instances are tested. These instances had varying sizes, with n values of 10, 20, 50, and 75 customers. Each n size category consists of 10 distinct instance sets.

It should be mentioned that optimal solutions for instances larger than $n=9$ are not available. Therefore, the GA-AS algorithm was compared against the best-found solutions in literature achieved by three rival algorithms: LS algorithm presented by Agatz et al. (2018), HGVNS algorithm proposed by Freitas & Penna (2020) and two variants of the GRASP algorithm proposed as GRASP-SA and GRASP-HCLS in the work of Almuhaideb et al. (2021).

These four algorithms follow the classical heuristic approaches involve two-staged solution construction: route construction and route improvement which are iteratively modify the route. These rival algorithms first solve an optimal TSP tour for only truck, then applies some local search or neighborhood search techniques to add drone delivery nodes on the initially found TSP tour so that generates a final TSP-D tour.

The numerical results of the experiment display the best-found solution values achieved by each referent algorithm. Table 4.4 provides a summary of these values. "Instance" column presents the problem IDs for instances categorized as 'single-center' and 'uniform' problems tested from Bouman et al. (2018b). "n" is the size of problem; each set consist of 10 distinct problems set. Under average cost row, best-found solutions of each referent algorithm are found by algorithm's name. These present relevant problem's solution cost in terms of time. Because of each *n-sized* problem type involves 10 different problems set, the values reported in the table represent the mean value of the best-found solutions obtained from 10 problem instances for each specific size category. It is important to note that, each 10 instance's solution have not been reported separately, instead results are reported as mean value of those. Therefore, in this evaluation it was not possible to make a comparison among them one-by-one.

To assess the performance of the GA-AS algorithm, the "gap %" column was computed for each to report percentage difference between the solutions of GA-AS algorithm and referent algorithm.

In the experimental evaluation, the proposed GA-AS algorithm demonstrated better performance compared to four rival algorithms in instances with a size of $n=10$. Consequently, GA-AS improved the best-known solution of the problem instances where $n=10$. The GA-AS algorithm consistently outperformed the GRASP-SA variant across all instances by getting an average gap of -10.46 %. GA-AS was also exhibited nearly better performance than the GRASP-HCLS

variant with an average gap of -1.04 %. For the instances bigger than n=10, numerical results indicates that GA-AS had a near performance compared to LS and HGVNS algorithms with an average gap of 7.72% and 6.98% respectively.

Table 4.4: Comparison results of GA-AS algorithm with rival algorithms on Bouman et al. (2018b) instances.

<i>instance</i>	<i>n</i>	<i>average cost</i>								
		GA-AS	LS	gap %	HGVNS	gap %	GRASP-HCLS	gap %	GRASP-SA	gap %
single-center instances										
51 to 60	10	263.50	278.22	-5.29	291.36	-9.56	287.13	-8.23	295.49	-10.83
61 to 70	20	399.73	384.87	3.86	364.08	9.79	416.47	-4.02	428.09	-6.62
71 to 80	50	649.61	554.58	17.14	593.54	9.45	617.43	5.21	723.88	-10.26
81 to 90	75	978.25	741.38	31.95	754.43	29.67	861.21	13.59	1030.92	-5.11
<i>average</i>		572.77	489.76	11.91	500.85	9.84	545.56	1.64	619.60	-8.20
uniform instances										
51 to 60	10	226.02	231.29	-2.28	233.20	-3.08	230.75	-2.05	242.09	-6.64
61 to 70	20	298.70	293.59	1.74	293.60	1.74	312.82	-4.51	321.88	-7.20
71 to 80	50	466.47	428.63	8.83	420.80	10.85	478.31	-2.47	568.37	-17.93
81 to 90	75	524.65	495.90	5.80	490.40	6.98	557.15	-5.83	648.30	-19.07
<i>average</i>		378.96	362.35	3.52	359.50	4.12	394.76	-3.72	445.16	-12.71
<i>average</i>				7.72		6.98		-1.04		-10.46

To evaluate performances of these five different algorithms in solving same instances, a statistical analysis was performed using the Friedman test. The Friedman test is a non-parametric statistical test used to determine if there are significant differences among multiple dependent groups. It is commonly employed when the data cannot be assumed to follow a normal distribution. Hypothesis of the test is conducted as below:

H_0 : all algorithms perform equally.

H_1 : at least one algorithm performs differently.

The obtained p-value was found to be significant at 0.000, indicating a difference in the performance of the algorithms as $p < 0.05$. This result suggests that there is a statistically

significant variation in the performance across the algorithms tested. The null hypothesis, which assumes that there is no difference in the performances of the algorithms, can be rejected in favor of the alternative hypothesis, which states that at least one algorithm performs differently.

The Friedman test does not provide information on which specific algorithms perform differently. To gain further insights and identify any pairwise differences between the algorithms, a Wilcoxon Signed Rank test was conducted. The Wilcoxon Signed Rank test is a non-parametric statistical test used to compare two samples. The purpose of this test was to determine whether there is a significant difference between the results obtained from two algorithms by comparing GA-AS with each algorithm one by one.

The test calculates the signed ranks of the differences between the paired observations and compares them to the expected distribution under the null hypothesis. The null hypothesis (H_0) considered in this analysis was that there is no significant difference between the performance of the two algorithms compared. Wilcoxon test is performed for each pair of algorithms and p -values obtained for each comparison is given by Table 4.5.

Table 4.5: p -values obtained from Wilcoxon Signed Rank test

<i>p value</i>	GA-AS
LS	0.08
HGVNS	0.726
GRASP-HCLS	0.624
GRASP-SA	0.014

Analyzing the results suggests that there is no significant difference between the performances of algorithm pairs GA-AS and LS, GA-AS and HGVNS, GA-AS and GRASP-HCLS in solving TSP-D since null hypothesis is accepted for $p > 0.05$. However, the algorithms GA-AS and GRASP-SA indicates significant difference in performance, as the obtained p -value of 0.014 is below the conventional significance level of 0.05, indicating strong evidence to reject the null hypothesis. It can be concluded that the two algorithms perform significantly differently in terms of their effectiveness in solving the TSP-D problem. As to the numerical results, GA-AS consistently improved the solution of GRASP-SA algorithm over each instance.

It can be concluded that the GA-AS algorithm performs equal performance with three algorithm (LS, HGVNS and GRASP-HCLS) and better performance than GRASP-SA algorithm in terms of their effectiveness in solving the TSP-D problem.

4.3 Computational Experiments on TSP Instances

TSP-D is thought to improve overall efficiency with reducing travel time compared to traditional TSP, by allowing for the integration of drones into the problem. To evaluate this consideration, this section provides an evaluation TSP-D along with traditional TSP.

A computational experiment is conducted on TSP instances from the data set of TSPLIB (Reinhelt, 2014). The TSPLIB dataset is a widely recognized and publicly available problem dataset specifically designed for the TSP. On instances selected from the TSPLIB dataset, the proposed GA-AS algorithm is executed to obtain solutions corresponding to TSP-D.

Optimal solutions of TSP are in terms of distance unit, so solution cost of the problem is considered as distance of route. However, cost function in TSP-D is in terms of time unit. To convert these into required time travel for both vehicles, costs are converted to time unit with dividing distances by the unit speed of drone and truck. Under the assumption of drone is faster than truck in TSP-D, drone speed considered as two times faster than truck speed. Euclidian metrics represent the distances between customer nodes.

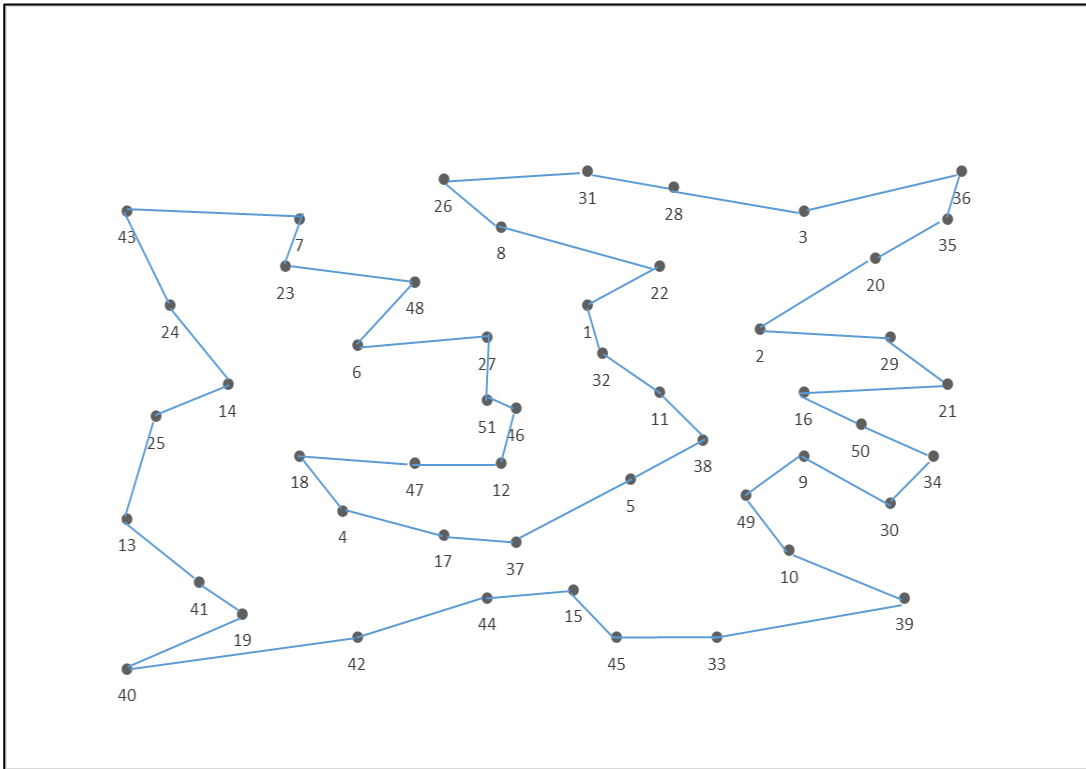
A total of 20 TSP problems ranging between 51 and 225 customer nodes are tested. For each instance from the TSPLIB dataset, the GA-AS algorithm is executed to obtain solutions for the corresponding TSP-D problem. Optimal TSP solutions of these selected problem instances are compared with the TSP-D results of the GA-AS algorithm on the same instances, in order to determine if any improvement is being obtained where a drone is integrated to the traditional truck travel. The GA-AS algorithm run for 10 times for each instance. Table 4.6 reports the numerical results of evaluation. Instance column gives the name of each problem instance executed from TSPLIB where n is the number of customers in corresponding instance. “ S_{TSP} ” is the optimal solution of TSP tour reported in TSPLIB. “ S_{TSP-D} ” is the best solution of TSP-D tour obtained through ten runs of GA-AS. “ $\overline{S_{TSP-D}}$ ” is the mean value of TSP-D solutions obtained through ten runs of GA-AS. “St.D” presents the standard deviation of the results obtained through ten runs of GA-AS. “ $gap\%$ ” is the difference between optimal TSP solution and best found TSP-D solution by GA-AS where “ $\overline{gap\%}$ ” is the difference between optimal TSP solution and average solution value of GA-AS. A negative (–) gap indicates the improvement. In addition, average computation time of GA-AS on each instance is reported in *time* column.

Furthermore, Figure 4.2 illustrated to provide an example for the representation of two tours

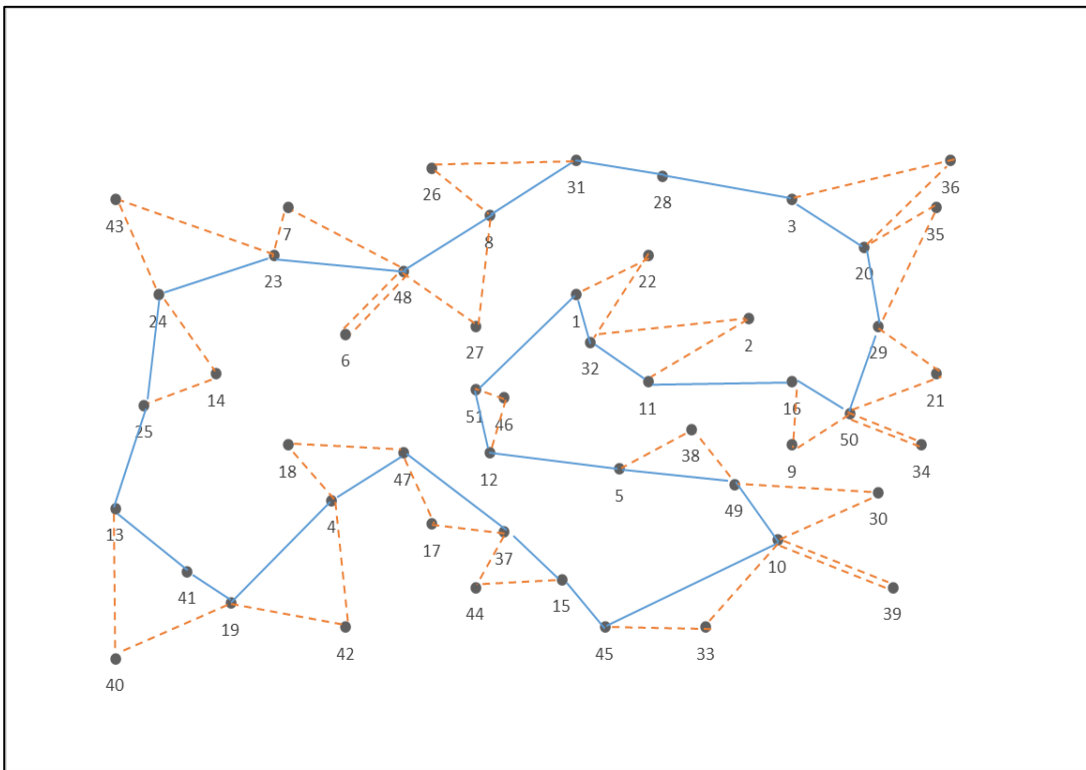
occurred on the randomly selected “eil51” problem instance: (a) is the illustrated graph of the optimal TSP tour, (b) is the illustrated graph of TSP-D tour obtained by GA-AS. By this figure, it can be seen the conversion of a TSP tour into a TSP-D tour once a drone is integrated into the delivery along with truck for an instance.

Table 4.6: Results of GA-AS algorithm on TSP instances

		<i>GA-AS</i>						
<i>instance</i>	<i>n</i>	S_{TSP}	S_{TSP-D}	<i>gap %</i>	$\overline{S_{TSP-D}}$	<i>St.D</i>	$\overline{gap \%}$	<i>time (min)</i>
berlin52	52	7542	6109.11	-19.00	6131.65	12.08	-18.70	1.17
bier127	127	118282	92259.96	-22.00	93040.62	188.80	-21.34	6.08
ch130	130	6110	5326.09	-12.83	5594.93	101.43	-8.43	6.16
ch150	150	6528	5896.74	-9.67	6086.05	91.58	-6.77	7.31
eil51	51	426	329.18	-22.73	348.43	19.16	-18.21	1.21
eil76	76	538	411.89	-23.44	423.73	45.75	-21.24	2.26
eil101	101	629	477.85	-24.03	490.56	28.06	-22.01	4.45
kroA100	100	21282	19375.85	-8.96	19885.90	110.24	-6.56	4.32
kroA150	150	26524	23906.08	-9.87	24616.92	146.61	-7.19	7.45
kroB100	100	22141	19763.06	-10.74	20241.30	117.19	-8.58	4.03
kroB150	150	26130	24057.89	-7.93	25147.51	218.90	-3.76	7.54
lin105	105	14379	13624.94	-5.24	13963.45	177.39	-2.89	4.20
pr76	76	108159	95402.55	-11.79	96207.43	125.31	-11.05	2.36
pr107	107	44303	38206.91	-13.76	38853.73	157.81	-12.30	4.48
pr124	124	59030	52513.09	-11.04	53634.66	195.89	-9.14	6.07
pr144	144	58537	52355.49	-10.56	54755.51	166.69	-6.46	6.49
pr152	152	73638	66664.48	-9.47	69418.54	126.86	-5.73	8.08
rat99	99	1211	1154.44	-4.67	1201.55	49.96	-0.78	4.03
rd100	100	7910	6230.71	-21.23	6375.46	68.48	-19.40	4.52
tsp225	225	3919	3567.86	-8.96	3744.60	55.81	-4.45	11.29
<i>average</i>				-13.40			-10.75	



(a) optimal TSP tour of “eil51” problem



(b) TSP-D tour of “eil51” problem obtained by GA-AS algorithm

Figure 4.2: illustrated TSP and TSP-D graphs of “eil51” problem

Across the tested instances from the TSPLIB dataset, the GA-AS algorithm consistently produced improved solutions at best found ones with roughly %24 deviations at the highest, %13 deviations at average over the optimal solutions of TSP. This indicates that a drone integrated delivery improves the tour costs whereas the delivery is handled by only a truck. These results of the computational experiment demonstrated the effectiveness of the TSP-D over TSP. Moreover, the computational time analysis revealed that the algorithm executed within reasonable time limits even for larger-scale instances.

5. DISCUSSION

The GA-AS algorithm proposed to solve TSP-D, was evaluated through a series of computational experiments. The experiments were divided into three parts: (1) evaluating the algorithm on TSP-D instances with known optimal solutions, (2) comparing it against rival algorithms, (3) testing it on optimal TSP solutions. Based on the numerical results, findings can be summarized as follows.

1. The GA-AS algorithm has shown promising performance in finding optimal solutions for the TSP-D problem particularly in instances where the optimal solutions are known. Out of the 50 problem instances tested, GA-AS successfully identified the optimal solutions in 38 cases, showcasing its ability to effectively generate optimal TSP-D routes.

Moreover, even in cases where the optimal solutions were not reached, GA-AS still produced results that were very close to the optimal solutions. The experimental findings indicate that the average gap between the solutions generated by GA-AS and the optimal solutions was only 0.80%. This small gap demonstrates the high accuracy and efficiency of the GA-AS algorithm in generating TSP-D tours optimally.

2. The performance of the proposed GA-AS algorithm was compared to four rival algorithms. The results demonstrated that GA-AS outperformed the rival algorithms and improved upon the best-known solutions for instances of size $n=10$.

Specifically, GA-AS consistently outperformed compared to the GRASP-SA algorithm across all instances, with an average gap of -10.46%. This negative gap indicates that GA-AS achieved solutions that were, on average, 10.46% better than the best-known solutions obtained by GRASP-SA. Similarly, GA-AS showed nearly better performance than the GRASP-HCLS variant, with an average gap of -1.04%.

For instances larger than $n=10$, the numerical results indicated that GA-AS had comparable performance to the LS and HGVNS algorithms, with average gaps of 7.72% and 6.98% respectively. Based on the solutions of 80 instances tested, the Wilcoxon Signed Rank test suggests that there is no significant difference in performance between the algorithm pairs GA-AS and LS, GA-AS and HGVNS, and GA-AS and GRASP-HCLS in solving the TSP-D problem. This conclusion is supported by accepting the null hypothesis for p-values above 0.05, indicating these algorithm pairs perform equally.

However, the comparison between GA-AS and GRASP-SA revealed a significant

difference in performance. The obtained p-value of 0.014 rejects the null hypothesis. This indicates that GA-AS and GRASP-SA perform significantly differently in terms of their effectiveness in solving the TSP-D problem. Notably, GA-AS consistently improved upon the solutions generated by the GRASP-SA algorithm across each instance.

The results highlight the competitive performance of the GA-AS algorithm compared to the rival algorithms. It can be concluded that the GA-AS algorithm performs equal or better for some instances in solving TSP-D. These findings support the proposed hybrid metaheuristic approach as a novel contribution to solving TSP-D.

3. The GA-AS algorithm was applied to a set of TSP instances. The objective was to assess the impact of integrating a drone into traditional truck travel of TSP which converts it into a TSP-D. To evaluate the effectiveness of this integration, the optimal TSP solutions from the TSPLIB dataset were compared with the TSP-D solutions obtained by the GA-AS algorithm on the same instances.

The experimental results indicate that the GA-AS algorithm consistently produced improved solutions for the TSP-D problems compared to the optimal TSP solutions. The deviations from the optimal TSP solutions ranged on average from of 13% to 24% at the highest. These deviations indicate that integrating a drone into the delivery process leads to significant improvements in tour costs compared to relying solely on truck travel.

These findings demonstrate the effectiveness of the TSP-D formulation over the traditional TSP, as the integration of a drone enhances the overall solution quality. The inclusion of a drone as a complementary delivery mechanism allows for more efficient and cost-effective routing, resulting in better tour costs and improved overall performance.

4. Furthermore, the computation time of the algorithm revealed that the GA-AS algorithm executed within reasonable time limits, even for larger-scale instances. This indicates that the proposed algorithm is computationally efficient and can handle TSP-D instances of practical significance.

Nonetheless, while the GA-AS algorithm has shown promising results in solving the TSP-D problem, it is important to acknowledge and discuss the limitations.

1. A limitation worth considering is the relatively new nature of the TSP-D problem in the literature. As a relatively recent research topic, there is a lack of various instances and benchmarking sets specifically designed for TSP-D and particularly with known optimal

solutions. The current best-found solutions of instances are still not known whether they are near-optimal solutions. This can pose challenges when conducting comprehensive analysis or experiments.

The lack of well-established benchmark instances for TSP-D makes it difficult to compare and evaluate the performance of different algorithms consistently across studies. Researchers often resort to creating their own problem instances or adapting existing TSP instances to incorporate the drone delivery aspect as like it is undertaken in this study. This can make direct comparisons between algorithms less straightforward. Researchers might have to focus on a smaller number of problem instances or specific problem settings, which may not fully represent the diversity and complexity of real-world scenarios. This limitation restricts the generalizability of the findings and hampers the ability to draw robust conclusions about the algorithm's performance across different problem instances or settings.

2. Particularly, TSP-D instances utilized in this thesis, consist of different n-sized problem set each have 10 different problem instances in itself. However, they are evaluated through their mean solution in each referent study. With access to each individual instance solutions, it would have been able to offer more detailed results and conduct a comprehensive analysis.
3. Another significant limitation in the context of TSP-D is the lack of real-life cases and data specifically related to drone-integrated route planning. While computational experiments and simulations provide valuable insights, they often rely on synthetic or hypothetical scenarios that may not fully capture the complexity and nuances of real-world situations. In real-life scenarios, there are numerous factors that can impact the effectiveness and feasibility of drone-integrated route planning. These factors include airspace regulations, weather conditions, infrastructure limitations, payload capacities, battery life, and operational constraints. Incorporating these real-world considerations into algorithm design and evaluation is crucial for developing practical and applicable solutions.

Consequently, the results from the computational experiments provide empirical evidence supporting the advantages of the TSP-D formulation and the capability of the GA-AS algorithm to effectively tackle the TSP-D. The integration of a drone into the delivery process offers potential benefits in terms of cost reduction and improved efficiency. These findings contribute to the existing literature on TSP-D and highlight the potential of proposed approach.

6. CONCLUSION

Traveling salesman problem with drone is considered in this thesis which is the routing problem of drone-aided delivery. It is a relatively new concept that has gained popularity over recent years due to the rapid advancements in drone technology. The idea of using drones for delivery can be traced back to the early 2010s when drone technology began to advance rapidly, what have in fact led to increased use of drones in numerous business area. One of those areas is last-mile delivery systems in logistics. Drones offer several advantages over traditional delivery methods, what the most significant one is using drones in delivery for its speed. Drones can cover long distances quickly and efficiently, bypassing traffic and other obstacles that can slow down ground-based vehicles.

The use of drones for delivery is becoming increasingly popular, and companies are investing heavily in drone technology to improve their delivery operations. Drones for delivery is still relatively new, and the technology is still in the early stages of development. As drone technology continues to evolve and become more advanced, it is expected that the use of drones for delivery will become more widespread and mainstream in the coming years.

Since the use of drones for delivery and transportation increases, optimizing the routing of deliveries and reducing transportation costs has become a critical concern for companies and organizations. Thus, this challenging idea in delivery has also come about a necessity to address this new routing problem, TSP-D. The TSP-D is a relatively new variant of the traditional TSP, which has emerged with the rise of drones. TSP-D extends traditional TSP through a drone is added to the problem setting. Consequently, TSP-D has made an attention to academicians in the fields of mathematics, computer science, and engineering. As drone technology continues to advance, it is likely that research on the TSP-D problem will continue to grow and contribute to the development of more efficient and effective drone delivery system.

The TSP-D is a combinatorial optimization problem introducing a drone that can fly between certain cities, in addition to the ground vehicle (truck) which travels by road. The objective is to find the shortest possible route for the truck and drone to collectively visit all the customers to deliver parcels and return to the starting point. The TSP-D offers a way to address optimizing routes both for truck and drone to minimize delivery times and reduce transportation costs.

The objective of this thesis is to provide a solution approach to solve the defined thesis problem TSP-D. Since TSP-D is NP-hard, developing of heuristic method as a solution approach are

considered in this thesis. The TSP-D is characterized by its complexity, as it involves both road and air transportation, and finding an optimal solution becomes increasingly difficult as the number of instances increases. By developing an algorithm that can handle the complexity of solving the TSP-D, this thesis aims to contribute to the field of optimization and provide valuable insights for solving similar problems in domains of transportation and drone aided delivery processes in logistics.

To achieve this objective, the thesis proposes a novel hybrid metaheuristic algorithm, GA-AS. The GA-AS algorithm is based on the implementation of the current state-of-the-art metaheuristic algorithms, GA and ACO, by hybridizing and customized them for TSP-D features.

The GA-AS algorithm is having its novelty through some perspectives.

- A consequence of literature review shows that metaheuristics are rarely studied so far to solve TSP-D when compared to classical heuristics. The thesis presents well-known algorithms GA and ACO as how to implement into the TSP-D, as they are not presented before in a hybrid framework to solve the relevant problem.
- As a consequence from literature review, TSP-D algorithms have precedence on classical heuristics that commonly consisting of two-staged solution approach: route construction and route improvement which are iteratively modify the route. Route construction stage: solves an optimal TSP tour for only truck delivery by regarding drone delivery. Route improvement stage: applies various local search or neighborhood search techniques to add drone delivery nodes on the initially found TSP tour so that generates a final TSP-D tour. Contrary to two-staggered algorithms in the literature, the GA-AS determines truck and drone nodes simultaneously that routes are constructed and optimized in a single stage.
- GA-AS is a population-based search metaheuristics whereas classical heuristics are single-solution based heuristics. To present a novelty, it is discussed to implement a population-based searching approach instead of already existing neighborhood search rules.

Consequently, a novel approach is proposed in this thesis that does not involve neighborhood search and two-staged solution construction as already existing in literature.

- Another motivation is that there is a lack of research on the implementation of ACO for the TSP-D in the existing literature. To the best of our knowledge, an ACO based

algorithm is the first time implemented and presented to solve TSP-D in the literature.

Moreover, a novel binary-pheromone structure for ACO algorithm is presented that can lead to implementation of ACO for relevant problems.

The proposed GA-AS algorithm to solve TSP-D was evaluated through a series of computational experiments. Based on the numerical results, findings can be summarized as follows.

- The GA-AS algorithm has shown promising performance in finding optimal solutions for the TSP-D, particularly in instances where the optimal solutions are known. GA-AS successfully identified the optimal solutions in most of the cases, showcasing its ability to effectively generate optimal TSP-D routes.
- Computational experiments also highlighted the competitive performance of the GA-AS algorithm compared to the rival algorithms. The proposed GA-AS algorithm improved upon the best-known solutions for instances of size $n=10$. Numerical results and statistical analysis concluded that the GA-AS algorithm performs equal or better for remaining instances in solving TSP-D. These findings support the proposed hybrid metaheuristic approach as a novel contribution to solving TSP-D.
- The GA-AS algorithm was also applied to a set of TSP instances. The experimental results indicate that the GA-AS algorithm consistently produced improved solutions for the TSP-D problems compared to the optimal TSP solutions. These findings demonstrate the effectiveness of the TSP-D formulation over the traditional TSP, as the integration of a drone enhances the overall solution quality. The inclusion of a drone as a complementary delivery mechanism allows for more efficient and cost-effective routing, resulting in better tour costs and improved overall performance.

In conclusion, the results from the computational experiments provide empirical evidence supporting the advantages of the TSP-D formulation and the capability of the GA-AS algorithm to effectively tackle the TSP-D. The integration of a drone into the delivery process offers potential benefits in terms of cost reduction and improved efficiency. These findings contribute to the existing literature on TSP-D and highlight the potential of hybrid metaheuristic approach, GA-AS.

In addition to the findings of this study, some further investigations can be recommended as follow:

- This study has focused on the basic assumptions of the TSP-D with a single truck and a

single drone. However, there are some other variants of the problem that could be explored in future research, such as the multi-drone and multi-vehicle scenarios.

- Another limitation of this study is that it assumes negligible battery recharge time and infinite drone endurance. It is recommended that future research consider the remaining battery charge and drone endurance time. Incorporating these considerations into the model can provide insights into the operational feasibility of the proposed solution in real-world delivery scenarios.
- Lastly, it is worth considering the potential for further improvements on the GA-AS algorithm. The current implementation utilizes an ACO based search mechanism, future research could explore the possibility of using this mechanism by hybridizing ACO component itself with GA.

REFERENCES

- Agatz, N., Bouman, P., & Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4), 965–981. <https://doi.org/10.1287/TRSC.2017.0791>
- Almuhaideb, S., Alhussan, T., Alamri, S., Altwaijry, Y., Aljarbou, L., & Alrayes, H. (2021). Optimization of truck-drone parcel delivery using metaheuristics. *Applied Sciences (Switzerland)*, 11(14). <https://doi.org/10.3390/app11146443>
- Arishi, A., Krishnan, K., & Arishi, M. (2022). Machine learning approach for truck-drones based last-mile delivery in the era of industry 4.0. *Engineering Applications of Artificial Intelligence*, 116. <https://doi.org/10.1016/j.engappai.2022.105439>
- Baniasadi, P., Foumani, M., Smith-Miles, K., & Ejov, V. (2020). A transformation technique for the clustered generalized traveling salesman problem with applications to logistics. *European Journal of Operational Research*, 285(2), 444–457. <https://doi.org/10.1016/j.ejor.2020.01.053>
- Beasley, D., Bull, D. R., & Martin, R. R. (1993). *An Overview of Genetic Algorithms: Part 1, Fundamentals* (Vol. 15, Issue 2).
- Boccia, M., Masone, A., Sforza, A., & Sterle, C. (2021). A column-and-row generation approach for the flying sidekick travelling salesman problem. *Transportation Research Part C: Emerging Technologies*, 124. <https://doi.org/10.1016/j.trc.2020.102913>
- Bouman, P., Agatz, N., & Schmidt, M. (2018a). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4), 528–542. <https://doi.org/10.1002/net.21864>
- Bouman, P., Agatz, N., & Schmidt, M. (2018b). *Instances for the TSP with Drone (and some solutions) (v1.2)*. Zenodo. <https://doi.org/10.5281/zenodo.1204676>.
- Cavani, S., Iori, M., & Roberti, R. (2021). Exact methods for the traveling salesman problem with multiple drones. *Transportation Research Part C: Emerging Technologies*, 130. <https://doi.org/10.1016/j.trc.2021.103280>
- de Freitas, J. C., & Penna, P. H. V. (2020). A variable neighborhood search for flying sidekick traveling salesman problem. *International Transactions in Operational Research*, 27(1), 267–290. <https://doi.org/10.1111/itor.12671>
- Dell’Amico, M., Montemanni, R., & Novellani, S. (2020). Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. *Annals of Operations Research*, 289(2), 211–226. <https://doi.org/10.1007/s10479-020-03562-3>
- Dell’Amico, M., Montemanni, R., & Novellani, S. (2021a). Algorithms based on branch and bound for the flying sidekick traveling salesman problem. *Omega (United Kingdom)*, 104. <https://doi.org/10.1016/J.OMEGA.2021.102493>
- Dell’Amico, M., Montemanni, R., & Novellani, S. (2021b). Modeling the flying sidekick traveling salesman problem with multiple drones. *Networks*, 78(3), 303–327. <https://doi.org/10.1002/NET.22022>
- Di Puglia Pugliese, L., Macrina, G., & Guerriero, F. (2020). Trucks and drones cooperation in the last-mile delivery process. *Networks*. <https://doi.org/10.1002/NET.22015>

- Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, 1(4), 28–39. <https://doi.org/10.1109/MCI.2006.329691>
- Dorigo, M., & Di Caro, G. (1999). Ant colony optimization: A new meta-heuristic. *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 1999*, 2, 1470–1477. <https://doi.org/10.1109/CEC.1999.782657>
- Dorigo, M., & Gambardella, L. M. (1997). Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. In *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION* (Vol. 1, Issue 1).
- Dorigo, M., Maniezzo, V., & Colomi, A. (1996). Ant system: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 26(1), 29–41. <https://doi.org/10.1109/3477.484436>
- El-Adle, A. M., Ghoniem, A., & Haouari, M. (2021). Parcel delivery by vehicle and drone. *Journal of the Operational Research Society*, 72(2), 398–416. <https://doi.org/10.1080/01605682.2019.1671156>
- Es Yurek, E., & Ozmutlu, H. C. (2018). A decomposition-based iterative optimization algorithm for traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 91, 249–262. <https://doi.org/10.1016/j.trc.2018.04.009>
- Euchi, J., & Sadok, A. (2021). Hybrid genetic-sweep algorithm to solve the vehicle routing problem with drones. *Physical Communication*, 44. <https://doi.org/10.1016/j.phycom.2020.101236>
- Ferrandez, S. M., Harbison, T., Weber, T., Sturges, R., & Rich, R. (2016). Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management*, 9(2), 374–388. <https://doi.org/10.3926/jiem.1929>
- Gonzalez-R, P. L., Canca, D., Andrade-Pineda, J. L., Calle, M., & Leon-Blanco, J. M. (2020). Truck-drone team logistics: A heuristic approach to multi-drop route planning. *Transportation Research Part C: Emerging Technologies*, 114, 657–680. <https://doi.org/10.1016/j.trc.2020.02.030>
- Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2018). On the min-cost Traveling Salesman Problem with Drone. *Transportation Research Part C: Emerging Technologies*, 86, 597–621.
- Ha, Q. M., Deville, Y., Pham, Q. D., & Hà, M. H. (2020). A hybrid genetic algorithm for the traveling salesman problem with drone. *Journal of Heuristics*, 26(2), 219–247. <https://doi.org/10.1007/s10732-019-09431-y>
- Ham, A. M. (2018). Integrated scheduling of m-truck, m-drone, and m-depot constrained by time-window, drop-pickup, and m-visit using constraint programming. *Transportation Research Part C: Emerging Technologies*, 91, 1–14. <https://doi.org/10.1016/j.trc.2018.03.025>
- Haupt, R. L., & Haupt, S. E. (2004). *Practical genetic algorithms* (2nd ed.). John Wiley & Sons.
- Jeffrey R. Sampson. (1976). Adaptation in Natural and Artificial Systems (John H. Holland). *SIAM Review*, 18, 529–530. <https://doi.org/10.1137/1018105>. (n.d.).
- Karak, A., & Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-

- up and delivery services. *Transportation Research Part C: Emerging Technologies*, 102, 427–449. <https://doi.org/10.1016/j.trc.2019.03.021>
- Khoufi, I., Laouiti, A., & Adjih, C. (2019). A survey of recent extended variants of the traveling salesman and vehicle routing problems for unmanned aerial vehicles. In *Drones* (Vol. 3, Issue 3, pp. 1–30). MDPI AG. <https://doi.org/10.3390/drones3030066>
- Kim, S., & Moon, I. (2019). Traveling salesman problem with a drone station. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 49(1), 42–52. <https://doi.org/10.1109/TSMC.2018.2867496>
- Li, M., Zhao, Y., Xiong, X., & Ma, Y. (2020). Comprehensive optimization of the synchronous delivery network in the model of OTMD for traveling salesman problem with drone. *Journal of Intelligent and Fuzzy Systems*, 39(5), 7505–7519. <https://doi.org/10.3233/JIFS-200818>
- Luo, Z., Poon, M., Zhang, Z., Liu, Z., & Lim, A. (2021). The Multi-visit Traveling Salesman Problem with Multi-Drones. *Transportation Research Part C: Emerging Technologies*, 128. <https://doi.org/10.1016/j.trc.2021.103172>
- Macrina, G., Di Puglia Pugliese, L., Guerriero, F., & Laporte, G. (2020). Drone-aided routing: A literature review. *Transportation Research Part C: Emerging Technologies*, 120. <https://doi.org/10.1016/j.trc.2020.102762>
- Madani, B., & Ndiaye, M. (2022). Hybrid Truck-Drone Delivery Systems: A systematic literature review. *IEEE Access*. <https://doi.org/10.1109/ACCESS.2022.3202895>
- Malhotra, R., Singh, N., & Singh, Y. (2011). Genetic Algorithms: Concepts, Design for Optimization of Process Controllers. In *Computer and Information Science* (Vol. 4, Issue 2). www.ccsenet.org/cis
- Marinelli, M., Caggiani, L., Ottomanelli, M., & Dell’Orco, M. (2018). En route truck-drone parcel delivery for optimal vehicle routing strategies. *IET Intelligent Transport Systems*, 12(4), 253–261. <https://doi.org/10.1049/iet-its.2017.0227>
- Mbiadou Saleu, R. G., Deroussi, L., Feillet, D., Grangeon, N., & Quilliot, A. (2018). An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks*, 72(4), 459–474. <https://doi.org/10.1002/net.21846>
- Montemanni, R., & Dell’Amico, M. (2023). Solving the Parallel Drone Scheduling Traveling Salesman Problem via Constraint Programming. *Algorithms*, 16(1). <https://doi.org/10.3390/a16010040>
- Moshref-Javadi, M., Hemmati, A., & Winkenbach, M. (2020). A truck and drones model for last-mile delivery: A mathematical model and heuristic approach. *Applied Mathematical Modelling*, 80, 290–318. <https://doi.org/10.1016/j.apm.2019.11.020>
- Murray, C. C., & Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54, 86–109. <https://doi.org/10.1016/j.trc.2015.03.005>
- Murray, C. C., & Raj, R. (2020). The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones. *Transportation Research Part C: Emerging Technologies*, 110, 368–398. <https://doi.org/10.1016/j.trc.2019.11.003>
- Nguyen, M. A., Dang, G. T. H., Hà, M. H., & Pham, M. T. (2022). The min-cost parallel drone scheduling vehicle routing problem. *European Journal of Operational*

Research, 299(3), 910–930. <https://doi.org/10.1016/j.ejor.2021.07.008>

- Otto, A., Agatz, N., Campbell, J., Golden, B., & Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (UAVs) or aerial drones: A survey. In *Networks* (Vol. 72, Issue 4, pp. 411–458). Wiley-Liss Inc. <https://doi.org/10.1002/net.21818>
- Phan, A. T., Nguyen, T. D., & Pham, Q. D. (2018). Traveling salesman problem with multiple drones. *ACM International Conference Proceeding Series*, 46–53. <https://doi.org/10.1145/3287921.3287932>
- Poikonen, S., & Golden, B. (2020). Multi-visit drone routing problem. *Computers and Operations Research*, 113. <https://doi.org/10.1016/j.cor.2019.104802>
- Poikonen, S., Wang, X., & Golden, B. (2017). The vehicle routing problem with drones: Extended models and connections. *Networks*, 70(1), 34–43. <https://doi.org/10.1002/net.21746>
- Ponza, A. (2015). Optimization Of Drone-Assisted Parcel Delivery. Master 's Thesis, Universita Degli Studi Di Padova, Italy.
- Reeves, C., & Rowe, J. E. (2002). Genetic algorithms: principles and perspectives: a guide to GA theory (Vol. 20). Springer Science & Business Media.
- Reinhelt, G. (2014). {TSPLIB}: a library of sample instances for the TSP (and related problems) from various sources and of various types.
- Roberti, R., & Ruthmair, M. (2021). Exact methods for the traveling salesman problem with drone. *Transportation Science*, 55(2), 315–335. <https://doi.org/10.1287/TRSC.2020.1017>
- Sacramento, D., Pisinger, D., & Ropke, S. (2019). An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. *Transportation Research Part C: Emerging Technologies*, 102, 289–315. <https://doi.org/10.1016/j.trc.2019.02.018>
- Salama, M. R., & Srinivas, S. (2022). Collaborative truck multi-drone routing and scheduling problem: Package delivery with flexible launch and recovery sites. *Transportation Research Part E: Logistics and Transportation Review*, 164, 102788. <https://doi.org/10.1016/j.tre.2022.102788>
- Schermer, D., Moeini, M., & Wendt, O. (2019). A matheuristic for the vehicle routing problem with drones and its variants. *Transportation Research Part C: Emerging Technologies*, 106, 166–204. <https://doi.org/10.1016/j.trc.2019.06.016>
- Schermer, D., Moeini, M., & Wendt, O. (2020). A branch-and-cut approach and alternative formulations for the traveling salesman problem with drone. *Networks*, 76(2), 164–186. <https://doi.org/10.1002/net.21958>
- Sivanandam, S. N., & Deepa, S. N. (2008). Genetic algorithms. Springer, Berlin Heidelberg.
- Srinivas, M., & Patnaik, L. M. (1994). Genetic algorithms: A survey. *Computer*, 27(6), 17–26.
- Tamke, F., & Buscher, U. (2021). A branch-and-cut algorithm for the vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 144, 174–203. <https://doi.org/10.1016/j.trb.2020.11.011>

- Tong, B., Wang, J., Wang, X., Zhou, F., Mao, X., & Zheng, W. (2022). Optimal Route Planning for Truck–Drone Delivery Using Variable Neighborhood Tabu Search Algorithm. *Applied Sciences*, *12*(1), 529. <https://doi.org/10.3390/app12010529>
- Vásquez, S. A., Angulo, G., & Klapp, M. A. (2021). An exact solution method for the TSP with Drone based on decomposition. *Computers and Operations Research*, *127*. <https://doi.org/10.1016/j.cor.2020.105127>
- Wang, Z., & Sheu, J. B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, *122*, 350–364. <https://doi.org/10.1016/j.trb.2019.03.005>
- Zhen, L., Gao, J., Tan, Z., Wang, S., & Baldacci, R. (2023). Branch-price-and-cut for trucks and drones cooperative delivery. *IIE Transactions*, *55*(3), 271–287. <https://doi.org/10.1080/24725854.2022.2060535>

APPENDIX

APPENDIX 1-Pseudo code of the AS algorithm

Algorithm AS

```
1: Generate the truckdeliverynodes and dronedeliverynodes for selected chromosome
2: Initialize ant for selected chromosome
3: Generate the solutionarray, restnodes and transportypearray
4: solutionarray.add (0) (Depot)
5: Apply firstcustomerassignment algorithm
6: int meetingcustomernode = 0
7: while (until all customers are assigned to the route) do
8: int transtype = transportypearray [transportypearray.count -1]
9: pre = solutionarray[transportypearray.count -1]
10:      Switch (transtype)
11:      case (1):
12:          totalpheromone = 0
13:          Generate randomnumber (0,1)
14:          selectpheromone = 0
15:          for (i = 0, i < truckdeliverynodes.count, I++ )
16:          totalpheromone += phcal (truckphmtrx, pre,i)
17:          for (i = 0, i < truckdeliverynodes.count, I++ )
18:              If (selectpheromone < randomnumber)
19:                  selectpheromone += phcal (truckphmtrx, pre,i) /
totalpheromon
20:          else
21:              if (meetingcustomernode == i)
22:                  solutionarray.add (i)
23:                  truckdeliverynodes.remove (i)
24:                  restnodes.remove(i)
25:                  transportypearray.add (3)
26:                  pre = i
27:
28:                  meetingcustomernode = null
29:              else
30:                  solutionarray.add (i)
31:                  restnodes.remove(i)
32:                  truckdeliverynodes.remove (i)
33:                  transportypearray.add (1)
34:                  pre = i
35:          break
36:      case (2):
37:          totalpheromone = 0
38:          Generate randomnumber
39:          selectpheromone = 0
40:          Bool truck = false
41:          if (dronedeliverynodes.count ==0)
42:              meetingcustomernode = 0
43:          for (i = 0, i < truckdeliverynodes.count, i++ )
```

```

44:             totalpheromone += phcal (truckphmtrx, pre,i)
45:         for (i = 0, i < truckdeliverynodes.count, i++ )
46:             If (selectpheromone < randomnumber)
47:                 selectpheromone += phcal (dronephmtrx, pre,i) /
totalpheromone
48:             else
49:                 meetingcustomernode = i
50:                 truck = true
51:                 break
52:         if (truck ==false)
53:             transportypearray[transportypearray.count -1] = 4
54:             goto exit1
55:     case (3):
56:         totalpheromone = 0
57:         Generate randomnumber
58:         selectpheromone = 0
59:         for (i = 0, i < truckdeliverynodes.count, i++ )
60:             totalpheromone += phcal (truckphmtrx, pre,i)
61:         for (i = 0, i < dronedeliverynodes.count, i++ )
62:             totalpheromone += phcal (dronephmtrx, pre,i)
63:         for (i = 0, i < truckdeliverynodes.count, i++ )
64:             If (selectpheromone < randomnumber)
65:                 selectpheromone += phcal (truckphmtrx, pre,I) /
totalpheromon
66:             else
67:                 solutionarray.add (i)
68:                 truckdeliverynodes.remove (i)
69:                 restnodes.remove(i)
70:                 transportypearray.add (3)
71:                 pre = i
72:         for (i = 0, i < dronedeliverynodes.count, i++ )
73:             If (selectpheromone < randomnumber)
74:                 selectpheromone += phcal (dronephmtrx, pre,i)/
totalpheromon
75:             else
76:                 solutionarray.add (i)
77:                 dronedeliverynodes.remove (i)
78:                 restnodes.remove(i)
79:                 transportypearray.add (2)
80:                 pre = i
81:     case (4):
82:         exit1:
83:         totalpheromone = 0
84:         Generate randomnumber
85:         selectpheromone = 0
86:         for (i = 0, i < truckdeliverynodes.count, i++ )
87:             totalpheromone += phcal (truckphmtrx, pre,i)
88:         for (i = 0, i < dronedeliverynodes.count, i++ )
89:             totalpheromone += phcal (dronephmtrx, pre,i)
90:         for (i = 0, i < truckdeliverynodes.count, i++ )

```

```

91:                If (selectpheromone < randomnumber)
92:                    selectpheromone += phcal (truckphmtrx, pre,i)/
totalpheromon
93:                else
94:                    solutionarray.add (i)
95:                    truckdeliverynodes.remove (i)
96:                    restnodes.remove(i)
97:                    transportypearray.add (3)
98:                for (i = 0, i < dronedeliverynodes.count, i++ )
99:                    If (selectpheromone < randomnumber)
100:                       selectpheromone += phcal (dronephmtrx, pre,i)/
totalpheromon
101:                else
102:                    solutionarray.add (i)
103:                    restnodes.remove(i)
104:                    dronedeliverynodes.remove (i)
105:                    transportypearray.add (2)
106:                end while

```

Algorithm *firstcustomerassignment*

```

1: firstpheromone = 0, selectpheromone = 0
2: Generate randomnumber
3: for (i = 0, i < truckdeliverynodes.count, i++ )
4:     firstpheromone += phcal (truckphmtrx, 0,i)
5: for (i = 0, i < dronedeliverynodes.count, i++ )
6:     firstpheromone += phcal (dronephmtrx, 0,i)
7: for (i = 0, i < truckdeliverynodes.count, i++ )
8:     If (selectpheromone < randomnumber)
9:         selectpheromone += phcal (truckphmtrx, 0,i)/ firstpheromone
10:    else
11:        solutionarray.add (i)
12:        truckdeliverynodes.remove (i)
13:        restnodes.remove(i)
14:        transportypearray.add (3)
15:    for (i = 0, i < dronedeliverynodes.count, i++ )
16:        If (selectpheromone < randomnumber)
17:            selectpheromone += phcal (dronephmtrx, 0,i)/ firstpheromone
18:        else
19:            solutionarray.add (i)
20:            dronedeliverynodes.remove (i)
21:            restnodes.remove(i)
22:            transportypearray.add (2)

```

Algorithm *phcal (matrix, first,last)*

```

1: Assign alpha and beta
2: total = Math.Pow(matrix [first, last], alpha) * Math.Pow(1 / distance[first, last], beta)
3: return total

```

CURRICULUM VITAE

Education

PhD	Marmara University	Industrial Engineering	2023
Master's	Istanbul University	Industrial Engineering	2013
Bachelor's	Istanbul Okan University	Industrial Engineering	2008

Employment

Lecturer	Istanbul Okan University	Industrial Engineering Department	2019-2022
Quality Manager	Istanbul Okan University	Rectorate	2012-2015
Research Assistant	Istanbul Okan University	Industrial Engineering Department	2008-2019

Scientific Research

Gunay-Sezer, N. S., Cakmak, E., Bulkan, S. (2023). A Hybrid Metaheuristic Solution Method to Traveling Salesman Problem with Drone. *Systems*, 11(5), 259. doi.org/10.3390/systems11050259

Simsit, Z.T., Gunay, N.S., Vayvay, O. (2015). Organizational Learning to Managing Change: Key Player of Continuous Improvement of 21st Century. Göksoy, A. (Ed.), *Organizational Change Management Strategies in Modern Business*, 95-109, IGI global, (ISBN: 1466695331).

Simsit, Z.T., Gunay N.S., Vayvay, O. (2014). Theory of Constraints: A Literature Review. *Procedia - Social and Behavioral Sciences*, 150, 930-936. doi:10.1016/j.sbspro.2014.09.104

Cakmak, E., Gunay, N.S., Aybakan, G., Tanyas, M. (2012). Determining the Size and Design of Flow Type and U-Type Warehouses. *Procedia-Social and Behavioral Sciences*, 58, 1425-1433. doi:10.1016/j.sbspro.2012.09.1127

Simsit, Z.T., Gunay, N.S., Vayvay, O. (2014). Theory of Constraints: A Literature review. 10th International Strategic Management Conference, Rome, Italy, 19-21 June.

Cakmak E., Gunay N.S., Aybakan G., Tanyas M. (2012). Determining the Size and Design of Flow Type and U-Type Warehouses. 8th International Strategic Management Conference, Barcelona, Spain, 21-23 June.

Gunay, N.S., Cakmak, E., Tanyas, M. (2010). Modeling with Simulation for the Synchronization of Manufacturing and Warehousing Activities in a Paint Production Company. EUROXXIV 24th European Conference on Operational Research, Universidade de Lisboa, Portugal, 11-14 July.

Cakmak E., Gunay N.S., Tanyas, M. (2009). Determination of Warehouse Storage Area Size with Pareto Analysis. 7th International Logistics & Supply Chain Congress, Istanbul, 5-6 November.

Simsit, Z.T., Oktay-Fırat, S.Ü., Es, H.A., Erdem, M., Topgul, M., Gunay, N.S. (2014). Bilgi ve İletişim Teknolojileri Çerçevesinden Küresel İnovasyon Endeksinin Analizi ve Veri Madenciliği Kullanılarak Ülkelerin Kümelenmesi. Yönetim Bilişim Sistemleri Kongresi, Boğaziçi University, Istanbul, Turkey, 16-17 October.

Cakmak E., Gunay N.S., Tanyas, M. (2010). Üretim ve Depo Faaliyetleri Yönetiminde Simülasyon ile Modelleme. 10. Ulusal Üretim Araştırmaları Sempozyumu, Girne Amerikan University, Cyprus, 16-18 September.

Gunay N.S., Cakmak E., Tanyas, M. (2009). Akış Tipi Depo için Kapı Sayısı Gözönünde Bulundurarak Depo Eni, Boyu ve Raf Sayısı Belirleme. 9. Ulusal Üretim Araştırmaları Sempozyumu, Eskişehir Osmangazi University, Turkey, 15-17 October.