A Learning Based Algorithm for Drone Routing

by

Umut Ermağan

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Industrial Engineering

KOÇ ÜNİVERSİTESİ

August 5, 2020

A Learning Based Algorithm for Drone Routing

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Umut Ermağan

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Sibel Salman

_____

Assist. Prof. Amine Gizem Tiniç

_____

Assist. Prof. Barış Yıldız

Date: _____

ABSTRACT

In this thesis, we optimize the route of a drone used in applications such as precision agriculture, search and rescue operations and military surveillance. Due to the drone's battery limit and its energy consumption rate, the drone needs to stop at recharge stations before its charge depletes. It follows a route between the recharge stations, avoiding restricted regions, to cover all of the identified points to be processed in minimum total time. To update the route dynamically as environmental conditions change, the route should be constructed in a short run time. Therefore, we propose a fast algorithm. The algorithm learns from a set of good feasible solutions obtained from a column generation procedure. Before running our algorithm, two alternative classification algorithms in machine learning, i.e., k-Nearest Neighbors and Decision Tree are used to predict the circumstances under which the drone should return to a recharge station. The algorithm that we propose utilizes the output of the classification algorithm to decide whether the drone should visit the next node in a pre-generated giant walk obtained from the optimal traveling salesman problem solution, or a recharge station. We show that learning is beneficial for obtaining feasible solutions with small optimality gaps in very short time by computational tests on the data sets that are generated randomly and also derived from a traveling salesman problem benchmark instance in the literature.

ÖZETÇE

Bu çalışmada, askeri keşifler, arama-kurtarma operasyonları ve tarım gibi birçok alanda kullanılan dronların rotalaması üzerine odaklanılmıştır. Sınırlı batarya kapasitesi ve enerji tüketim oranından dolayı dronlar ihtiyaç duyulduğu zamanlarda şarj istasyonlarına iniş yapmak zorundadırlar. Bu nedenle şarj istasyonları arasında gerçekleşen rotalamada, yasaklı bölgelere girmeden, önceden belirlenmiş tüm hedef noktalara minimum toplam sürede gitmek amaçlanmaktadır. Oluşturulan rotayı çevresel koşullar değiştikçe dinamik olarak güncelleyebilmek için rotalamanın kısa sürede yapılması gerekmektedir. Bu nedenle, öğrenme temelli hızlı bir algoritma geliştirilmiştir. Geliştirilen bu algoritma kolon türetme algoritması ile elde edilen iyi olurlu çözümlerin özelliklerini öğrenmektedir. Öğrenme temelli algoritma uygulanmadan önce, dronun hangi şartlar altında şarj istasyonuna iniş yaptığını tahmin edebilmek için iki farklı makine öğrenmesi sınıflandırma algoritması olan k-en yakın komşu ve karar ağacı algoritmaları kullanılmaktadır. Öğrenme temelli algoritma bu iki faklı sınıflandırma algoritmalarının sonuçlarından yararlanmaktadır. Bu aşamada dronun gezgin postacı problemi çözüm yöntemleri ile elde edilen optimal tur içinde bir sonraki hedef noktaya mı, yoksa şarj istasyonuna mı gideceğine karar verilir. Literatürde bulunan bir gezgin postacı problemi karşılaştırma örneğinde elde edilen ve rastgele oluşturulan veri setleri üzerinde yapılan testlerde, öğrenme temelli algoritmanın çok kısa süre içerisinde optimal sonuçlara yakın uygulanabilir çözümler bulduğu gösterilmiştir.

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor, Prof. F. Sibel Salman, for her incredible and precious guidance during my Master's program. She generously shared her knowledge, passion, time and experience with me with great kindness. My sincere thanks must also go to Assist. Prof. Barış Yıldız for giving his time to offer me valuable comments toward improving my work. I also need to thank Assist. Prof. Amine Gizem Tiniç for being in my thesis defense committee and for her feedback on my thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

UAV	Unmanned Aerial Vehicle
RS	Recharge Station
L&F	Learn and Fly
DRP	Drone Routing Problem
IP	Integer Programming
TSP	Travelling Salesman Problem
VRP	Vehicle Routing Problem
VRP-TW	Vehicle Routing Problem with Time Windows
VRP-STW	Vehicle Routing Problem with Soft Time Windows
ARP	Arc Routing Problem
MIP	Mixed Integer Programming
EVRP	Electric Vehicle Routing Problem
ALNS	Adaptive Large Neighborhood Search
G-VRP	Green Vehicle Routing Problem

Chapter 1

INTRODUCTION

The use of unmanned aerial vehicles (UAVs), commonly known as drones, is on the rise in many fields such as small-parcel delivery, humanitarian logistics, surveillance and environmental monitoring, in addition to military applications. For delivery services, drones have a significant potential to reduce the cost and delivery time. Apart from product delivery, they can be used to transport humanitarian aid over hazardous areas where the traditional vehicles cannot travel (Dorling et al., 2016). In disaster-relief operations, drones gain more importance because they do not need to use the road infrastructure, which may be damaged and non-operational (Chowdhury et al., 2017). In military applications, reconnaissance and surveillance missions can be carried out by drones. For example, the troops that need to be rescued from a dangerous area or battlefield can be supported by these aerial vehicles (Kopfstedt et al., 2008). Another interesting field for drones is precision agriculture, which inspires us to study the optimal routing of drones. Precision agriculture can be described briefly as the usage of information technologies for agricultural practices to improve profitability and sustainability in production (Gebbers and Adamchuk, 2010). As environmental conditions, such as rainfall, humidity and temperature, can affect agricultural products easily, farmers need to get reliable information on soil and crops to be able to take precautionary actions (Swain et al., 2007). Drones can be utilized to capture aerial images to be processed digitally to extract valuable information about the status of the crops, such as vegetation density, crop stress and water deficit, or measure various key parameters via remote sensing. Precision agricultural systems provide a scientific and data-driven support system for crop management. Even though satellites and planes are two other ways apart from the

drone to obtain images, some reasons such as high cost, low availability, and weather conditions cause the use of drones to gain more importance (Barrientos et al., 2011). In systems that utilize images captured by drones, the drone is equipped with a camera, a positioning system, storage memory, a wireless transceiver and a battery for the system to work autonomously. In order to minimize human intervention, a fast-running routing algorithm is needed to guide the drone over the fields. An optimal altitude is selected for the drone in the allowed flight range, considering the trade-off between image resolution and battery consumption. Points at which images should be taken are decided upon such that there will be sufficient overlaps in the pictures and the inspected area is fully covered. The route of the drone consists of flight legs visiting a subset of the points between recharge stations (RSs) as the battery depletes frequently.

The objective of this study is to develop an efficient algorithm to determine the route of the drone over the selected image capturing/processing points, while taking into consideration the conditions related to the limited battery capacity, restricted areas, multiple route legs, and the need to stop at recharge stations. In particular, the algorithm should have sufficiently small running times to allow rerouting when conditions change. The characteristics of the route optimization problem addressed in our study can be listed as below.

- Battery Limit: The drone has a battery consumption limit which leads to an allowed flight range.

- Recharge Stations: When the drone comes to the RS, the battery is either recharged or swapped with a fully-charged one by the help of a person. There can be multiple RSs in the network. For the sake of simplicity, we assume that the drone has to visit each RS at least one time. This assumption is based on the fact that RS locations and their number is determined for the best system efficiency and it is unlikely to have a RS which is not utilized.

- Restricted Areas: No fly over zones exist. The drone can pass through their boundary if necessary.

- Velocity of the Drone: We assume that the drone can fly at an average constant speed so that the travel time between two nodes can be calculated.

A network is constructed as an input to the problem by defining nodes for both points required to be covered in the area and the RS locations. Distances are calculated such that the restricted areas will be avoided. The drone should visit one of the RSs before it runs out of battery charge and then takes off again to continue its route until all required nodes are covered in minimum time. We develop a learning based fast algorithm that we name Learn and Fly (L&F), which can solve the Drone Routing Problem (DRP) in almost real-time. The routing decisions are updated each time operational conditions (i.e., weather conditions such as wind speed and direction) change. A standard arc based integer programming (IP) formulation of the routing problem falls short of obtaining high quality solutions to implement or strong bounds to help with measuring the quality of a given feasible solution. Therefore, we present a novel path-segment based formulation to model the routing problem, motivated by the fact that such formulations have been successfully implemented in recent years for other problems in other forms (Yıldız and Karaşan, 2017). Using this formulation, we develop a column generation algorithm to obtain high quality solutions and strong bounds on the total route time. Furthermore, we use high quality solutions obtained by the column generation algorithm to train the learning algorithm so that it can solve the routing problem in milliseconds. Thus, the algorithm can be utilized in real time for reoptimization as the conditions change dynamically. We test the performance of our algorithm on several data sets of varying sizes. We observe that the algorithm gives near-optimal solutions in very short running times on data sets with up to 40 nodes. We think that it is not viable for a single drone to handle a larger number of points in its mission and multiple drones can operate in their dedicated areas, if the need arises.

This thesis is organized as follows. In Chapter 2, we provide a literature review on static and dynamic routing problems related to UAVs/drones. In Chapter 3 we present the problem description, the path and cycle based model, and the column generation procedure. In Chapter 4, we propose our learning based algorithm. In

Chapter 5, the computational results, instance generation and the performance analysis of the learning based algorithm are presented. Finally, in Chapter 6 we provide our concluding remarks and discuss possible extensions of the study.

Chapter 2

LITERATURE REVIEW

As drone usage is rapidly expanding and future usage is expected to increase further due to anticipated regulations, recent years have seen an exponential increase in the number of studies on optimization related to drones. Otto et al. (2018) provide a comprehensive review of studies addressing different application areas of drones, as well as related optimization problems including drone routing studies. Studies have focused on applications such as disaster response (Chowdhury et al. (2017), Oruc and Kara (2018), and Rabta et al. (2018)), real-time traffic monitoring (Li et al., 2018), accident awareness in the oil and gas industry (Cho et al., 2015), and extensively in recent years on drone-assisted parcel delivery (Murray and Chu (2015), Song et al. (2018) and Wang and Sheu (2019), to name a few). Mostly routing of multiple drones have been studied, together with lately the cooperation of trucks and drones for parcel delivery (Murray and Chu (2015), Savuran and Karakaya (2016), Luo et al. (2017), Mbiadou et al. (2018), Ulmer and Thomas (2018), Boysen et al. (2018), Chang and Lee (2018), Ha et al. (2018), Jeong et al. (2019) and Kitjacharoenchai et al. (2019)). Solution methods comprise of mainly heuristic algorithms due to the NP-hardness of the routing problem and its variants (e.g., Levy et al. (2014), Sundar and Rathinam (2014), Song et al. (2018), Ha et al. (2018) and Zhen et al. (2019)) and exact algorithms that solve small-sized instances (e.g. Stump and Michael (2011), Guerriero et al. (2014), Alotaibi et al. (2018) and Wang and Sheu (2019)).

We next focus on recent studies aiming to provide solutions that are applicable in a dynamic environment. We then review studies on static drone and electric vehicle routing problems, and we finally discuss our contributions at the end of this chapter.

2.1 Dynamic drone routing problems

The use of drones, with obvious advantages of the technology, attracts significant attention from both the industry and academia. However, there are not enough studies in the literature to deal with the novel algorithmic challenges to realize this potential, mostly due to the complexity of the underlying routing problem which limits the use of existing methods to guide online decision making (Crainic et al., 2009).

Modaresi et al. (2020) benefit from learning to solve combinatorial optimization problems with instances that are unknown in advance. They study a decision-making problem whose single-period decision problem is a combinatorial optimization problem. Apart from answering the question of *when* in the traditional bandit setting, they propose two policies to answer the key questions of *what* and *how* in combinatorial settings. The main idea in the learning approach is based on the feedback obtained from the previous solutions. On the other hand, learning is carried out by machine learning algorithms in our study. Cui et al. (2018) address a path planning problem in which the future actions of a UAV can be decided. In the study, because of the unstructured environment, Markov decision process is used to handle some dynamic uncertainties such as control strategies, flight environments, and any other bursting-out threats. An optimal strategy by which the UAV path problem can be solved by means of Q-learning is developed and the best action of the UAV experiencing the moving threats in the unstructured environment is obtained. In our study, we use supervised learning algorithms instead of the reinforcement learning algorithms like Q-learning. The most important reason is that we can address the drone routing problem as a classification problem in which the output variables, i.e. target labels, can be categorized. The supervised machine learning algorithms identify a function mapping the input variables, namely the features, to the target labels. After mapping to the target labels consisting of visiting a RS and not visiting a RS in our problem with the training data set, these algorithms can produce correct labels for the new samples. Hence, we avoid extensive data and computation required for Q-learning.

Several studies consider uncertain elements. Chow (2016) deal with a UAV routing problem in which the UAVs monitor traffic under uncertainty using a stochastic policy. The study, which aims to minimize the routing cost of a certain number of UAVs, provides a new policy by reformulating the Bellman equation, as well as an approximate algorithm. The results obtained from the developed algorithm are compared against static and myopic policies. Kim et al. (2018) take the drone routing problem into account under uncertainty in the battery capacity subject to air temperature changes. A robust optimization approach utilizing uncertainty sets is proposed. Al-Sabban et al. (2013) modify Markov Decision Processes to identify an optimal energy-based path of single UAV while minimizing total travel time and power consumption. A hybrid Gaussian distribution is used to find the direction and magnitude of the wind. Hence, the next action of the UAV can be decided by considering the wind effect. Similarly, Thibbotuwawa et al. (2020) consider the weather conditions during planning flight missions of three UAVs. The goal of the study is to meet the demand of each customer within a time horizon. The solution approach is based on finding the flying mission at two levels, one of them is the mission planning level and the other is sub-mission planning level. In the sub-mission planning level, the mission area is subdivided according to UAVs' relative capacities. The solution is obtained via the proposed model for each sub-divided area. The combined solution of the sub-mission solutions represents the mission plan of UAVs.

In addition to methods based on probabilistic policies (e.g., Chow (2016), Kim et al. (2018), Al-Sabban et al. (2013)), some deterministic algorithms, by means of which dynamic drone routing problems can be solved quickly, have also been developed. Li et al. (2018) propose a heuristic algorithm to solve the UAVs' scheduling problem for traffic monitoring considering the uncertain monitoring demands. They formulate a mixed integer linear programming model by combining the capacitated arc routing problem (ARP) with the inventory routing problem to be able to monitor the traffic with the time-dependent inventory-based method. With the local branching based method, the best objective value is chosen among the solution spaces found with the branching cuts until the maximum iteration number is reached or there is

no improvement.

2.2 Static drone routing problems

The routing of a single drone or multiple drones have been studied as variants of the travelling salesman problem (TSP) and the vehicle routing problem (VRP). As one of the earliest studies, Stump and Michael (2011) address a multi-UAV routing problem over a continuous time horizon with differing visitation frequencies to required points. They propose to solve this problem at discrete time points as a vehicle routing problem with time windows (VRP-TW) using exact methods. They modify the VRP-TW by adding repeat visits of the same site as new sites with different time windows. The battery life constraint is handled by requiring the UAVs to return to recharge at a known base location. They demonstrate their approach on the task of surveying a building using multiple UAVs. Guerriero et al. (2014) consider multiple UAVs covering events that occur over time in a finite area. Each event randomly occurs at a location during a time window, such as the events during a sports action. They develop a multi-criteria optimization model, in which the criteria are to minimize the total distances traveled by the UAVs, maximize the customer satisfaction by arriving on time to capture each event and minimize the number of used UAVs. They pose the problem as an instance of the vehicle routing problem with soft time windows (VRP-STW) and test both an epsilon-constraint and a rolling horizon approach such that when the problem is solved at a time point, only the known events are considered and no information is given on future events. (Stump and Michael, 2011) and (Guerriero et al., 2014) studies differ from our work in that a continuous time window is considered in these studies while we minimize the time to cover an identified set of locations by flying over them. In (Stump and Michael, 2011), multiple visits are required, whereas in our problem we need to visit each point exactly once.

Levy et al. (2014) develop heuristics to find tours for multiple heterogenous UAVs with differing fuel tank capacities such that all the specified targets are visited at least once by some vehicle, the tours satisfy the fuel constraints, and the total travel

cost of the vehicles is at minimum. The travel cost is the total fuel consumed by all the vehicles, which is assumed to be directly proportional to the distance traveled by the vehicles. The vehicles visit the fuel stations as they run out of fuel, hence in each leg the initial and final stations may differ as in our problem. In their computational tests, the number of visit points vary from 50 to 360 while 2 to 9 vehicles exist, and the run times are at most 600 seconds. Sundar and Rathinam (2014) consider the same problem with a single drone. They develop an approximation algorithm and fast construction and improvement heuristics. Their test data involve five depots and 25 target points. Our problem is similar to the one addressed there but we are able to solve larger instances in very short running times, hence enabling responding to real-time changes. Avellar et al. (2015) optimize the routes of multiple-drones with image sensors to cover an area from which they select visiting points. They propose a mixed integer programming (MIP) formulation and provide experimental results with two drones. Zhen et al. (2019) study a variant of the VRP, where one must determine not only the order in which to visit a set of nodes located in the plane, but also the height at which to visit them, which impacts the accuracy level and the service time. They provide an integer programming model and a tabu search heuristic to solve this multi-drone routing problem.

Campbell et al. (2018) present several drone arc routing problems and study their relations with well-known arc routing problems. A set of required edges are served in these problems. Therefore, an algorithm for the rural postman problem is utilized. Song et al. (2018) study a multi-drone problem in which the flight time of a drone between certain points is a function of the amount of loaded product carried during the flight. They propose a MIP formulation for the routing problem with charging stations as well as a heuristic algorithm. Alotaibi et al. (2018) study a multi-drone routing problem that maximizes the total number of visited points in limited route travel time while also limiting total threat exposure level. They generate visit points so that the routes can avoid high-risk edges in the network representation. They develop a branch and cut and price algorithm to solve a set covering formulation.

In our routing problem, the drone starts its route at the home base and each

consecutive leg of the route may start and end at any one of the RSs. Each leg is subject to a length limit due to energy consumption. Crevier et al. (2007) study a variant of the multi-depot VRP where depots can act as intermediate replenishment facilities along the route of a vehicle. For each vehicle, the total length of the route possibly consisting of several legs is limited, as opposed to the time limit on each leg in our problem. The authors develop a Tabu Search heuristic by dividing the problem into three types of subproblems. The heuristic is tested on data with 48 to 288 customers and up to 7 depots and the solution times are reasonably short while the optimality gaps are satisfactory.

Multiple objectives have been defined for DRP by Ramirez-Atencia et al. (2017) and Coelho et al. (2017). Ramirez-Atencia et al. (2017) consider the makespan, fuel consumption, and distance minimization for a problem with multiple UAVs and a set of ground control stations under several constraints representing the system requirements. They present a multi-objective genetic algorithm with a hybrid fitness function using a constraint satisfaction problem to check whether solutions are valid. Coelho et al. (2017) study the routing of a heterogeneous fleet of drones with multiple charging stations under operational constraints. The drones having weight capacity limitation can collect and deliver packages. Seven different objective functions are considered and minimized using a MIP formulation, which is solved by a matheuristic algorithm.

2.3 Routing in truck-drone delivery systems

In recent years, providing delivery services with both drones and trucks simultaneously has been proposed. Correspondingly, the academic community has been studying routing problems for such systems at increasing pace. In routing problems with such two modes, drones take-off from and land on trucks that visit customer locations as well. The challenge in these problems is that the routes of trucks and drones that should collectively cover the customers need to be synchronized. The advantages of using drones as a complimentary mode have been shown in these studies.

Considering a system in which a drone works together with a delivery truck to distribute parcels, Murray and Chu (2015) introduce two variants of the TSP that they name the Flying Sidekick TSP and Parallel Drone Scheduling Travelling Salesman Problem. In the first problem, the drone can take-off and land at certain visits on the truck route, making it possible for both vehicles to deliver goods to customers in parallel while the truck serves a set of customers. On the other hand, the drone has to return to the depot before serving the next customer in the second problem. The authors also consider the multi-drone version and present MIP formulations for the two problems, along with two simple heuristic solution approaches to solve problems of practical size. Effectively combining a truck and a drone gives rise to the problem that is also called the TSP with drone. For Parallel Drone Scheduling Travelling Salesman Problem, Dell'Amico et al. (2020) provide a MIP formulation and a set of matheuristic algorithms. They show that their algorithms obtain high quality solutions in very short time for the instances in the literature. Mbiadou et al. (2018) also consider a setting in which deliveries are performed by either a ground vehicle or a drone. In their problem, the vehicles can serve multiple customers on a delivery tour, while the drones are constrained to perform back and forth trips. Mbiadou et al. (2018) present a heuristic for this problem, where the objective is to minimize the completion time. Experiments conducted on benchmark instances show the efficiency of the approach. Bouman et al. (2018) present exact solution approaches for the TSP with drone based on dynamic programming and compare them computationally. Agatz et al. (2018) model the TSP with drone as an integer program and develop several fast route-first, cluster-second heuristics based on local search and dynamic programming. They prove worst case approximation ratios for the heuristics and test them computationally as well. Savuran and Karakaya (2016) propose a genetic algorithm to maximize the number of points visited by the UAV which takes off from a point on the given path of the carrier vehicle, and compare its performance to that of two simple heuristics. Luo et al. (2017) study a problem with similar characteristics except that they minimize the time of the UAV route and propose two heuristics that can solve the problem with up to 200 nodes. Boysen et al. (2018) first study a problem where the truck route is fixed and then turn this

into a decision. They show complexity results before providing MIP formulations, and compare the potential savings in different settings. Ha et al. (2018) consider a variant of the TSP with drone where the objective is to minimize operational costs including total transportation cost and cost of waste time occurring when a vehicle has to wait for the other. A MIP formulation and two heuristic algorithms are proposed for the solution. The heuristics are tested on randomly generated instances with up to 100 nodes and it is shown that one of the algorithms finds near-optimal solutions in short times. Poikonen et al. (2019) also take the TSP with drone into account as a hybrid truck and drone model of delivery. A heuristic solution approach is developed using a branch-and-bound algorithm in which each node of the branch-and-bound tree corresponds to a potential order to deliver a subset of packages. A dynamic program is solved to get a so called "approximate lower bound" at each node of the branch-and-bound tree, where the insertion of an additional location onto the TSP with drone route occasionally decreases the objective value while it always increases the objective value of a TSP route. A constant called "TER" is used to evaluate the assumed lower bounds which can yield better results. As the value of "TER" approaches infinity, the solution also approaches the global optimal. In addition, two variants of the heuristic approach that depend on the ability to prune large portions of the decision tree are provided in order to overcome the long computation times of the branch and bound algorithm.

Wang and Sheu (2019) study the VRP with drones, which is a generalization of TSP with drone to the multiple truck case and also an extension of the classic capacitated VRP. They consider two cases with dependent and independent movement of trucks and drones. The authors propose a path-based model and develop a branch-and-price algorithm. They evaluate the performance of the branch-and-price algorithm on randomly generated small-sized instances. Sacramento et al. (2019) also study the VRP with drones and propose an Adaptive Large Neighborhood Search metaheuristic for its solution. They are able to solve instances with up to 200 customers. They also show the savings with respect to the truck-only approach. In another recent study on truck-drone routing, Karak and Abdelghany (2019) propose a variant of the VRP where pick-ups and deliveries can be per-

formed simultaneously by means of drones and a single truck for their launching. They extend the classic Clarke and Wright algorithm and compare it with two other heuristics. Jeong et al. (2019) consider the effect of parcel weight on drone energy consumption in a truck-drone hybrid delivery system, as in other similar studies. However, they also consider restricted flying areas as in our study. Dorling et al. (2017) propose a multi-trip VRP with drone delivery having cost minimization and delivery time minimization objectives that are handled in two single objective problems with a constraint on the other objective. They propose mathematical models and a simulated annealing algorithm.

Ulmer and Thomas (2018) study a dynamic VRP with heterogeneous fleets, where customers order goods over the course of the day. These goods are delivered either by a drone or by a ground vehicle within a delivery deadline. It has to be decided which customers to serve with which type of vehicle. The objective is to maximize the expected number of served customers. The authors present a Markov decision process model and a policy function approximation based on geographical districting and show its satisfactory performance. Liu (2019), in their study, take the dynamic vehicle routing problem with drone into account for delivery services. Static and dynamic MIP models are developed, and an online dispatch algorithm is proposed to carry out online operations. In the algorithm, a first-come first-serve principle is applied to decide which customer will be served.

Chang and Lee (2018) also study the case of a truck carrying drones. They first cluster customer locations to identify the points at which the drones will take-off and then optimize the truck's route. Kitjacharoenchai et al. (2019) study the multiple truck and multiple drone case and propose a heuristic based on insertion algorithms to solve instances with up to a 100 delivery locations. As in several other studies mentioned above, they also show the potential benefit of the truck-drone system.

2.4 Electric vehicle routing problems

Electric vehicles have similar features with drones in terms of the battery limit and the need for recharging, albeit they need to follow a fixed infrastructure. Motivated

by the benefits of green logistics, we see that studies on the routing of electric vehicles have also been growing in number.

In one of the earlier studies, Schneider et al. (2014) study the electric vehicle routing problem (EVRP) with time windows and RSs. They present a hybrid heuristic that combines a variable neighborhood search algorithm with a tabu search heuristic and allows infeasible solutions during the search. They test the heuristic on a set of small-sized instances, with at most 15 customers, that can be solved exactly by a MIP model, and a set of larger instances with up to 100 customers. The heuristic is compared to existing ones on the latter data set. However, the run times surpass 500 minutes on large instances, making the method not viable for use in drone routing. Goeke and Schneider (2015) study the same problem with a mixed fleet of electric and diesel vehicles. They provide a realistic energy consumption model based on speed, gradient and cargo load factors. They develop an Adaptive Large Neighborhood Search (ALNS) heuristic and perform several analyses. Sassi and Oulamara (2017) also consider a mixed fleet of electric and regular vehicles but assign the vehicles to a given set of tours to minimize the charging cost of electric vehicles subject to energy requirement and driving ranges of the electric vehicles. They divide the time horizon into equidistant intervals between the charger locations and consider the power requirements and the electricity grid capacity during each interval. They develop a MIP model to solve this NP-hard problem on real instances with up to 46 tours and 26 vehicles, and random instances with up to 200 nodes. Near-optimal solutions can be obtained for fleets of up to 80 vehicles. They propose a heuristic that first selects feasible tours for each vehicle sequentially and then optimizes the charging to minimize cost. They also propose a simultaneous approach to assign tours to all vehicles in the first step. The heuristic algorithms provide fast solutions and satisfactory performance on some of the large instances.

Desaulniers et al. (2016) propose four variants of the EVRP with time windows and for each variant, they present exact branch-price-and-cut algorithms. Their computational tests show that all four variants are solvable for instances with up to 100 customers and 21 RSs. Liao et al. (2016) present an exact polynomial time dynamic programming algorithm for the EVRP. They also consider the fixed tour

version of the problem and develop exact algorithms. They provide approximation algorithms for two other variations of the problem. Hiermann et al. (2016) address the electric fleet size and utilize VRP with time windows and RSs to decide on the fleet composition and the vehicle routes that include the recharging locations along the way. The vehicle types differ in their transport capacity, battery size and acquisition cost. The authors solve this problem by means of branch-and-price and a hybrid heuristic, which combines an ALNS with an embedded local search and labeling procedure for intensification. They test their algorithms on small and large instances and report that their heuristic is able to find solutions within a gap of around one percent to the best known solution in around 25 minutes on the average. Similarly, Yıldız et al. (2016) consider the locations of recharge stations with the routing of the electric vehicles in their study. They propose a branch-and-price algorithm for the problem in which the non-simple paths are considered unlike other studies.

With emergence of alternative fuel vehicles, green logistics has attracted the attention of governments and companies. Hence, a variant of the VRP named as the Green Vehicle Routing Problem (G-VRP) (Erdoğan and Miller-Hooks, 2012) that considers minimizing the energy consumption in transportation has been studied in the literature. In one of the earlier studies, Kara et al. (2007) present a linear integer formulation including a new cost function to reduce energy consumption. Felipe et al. (2014) model G-VRP with different types of charging technologies and allow partial charging. They develop constructive and local search heuristics. Wen et al. (2016) also consider the problem where vehicles can be recharged fully or partially at any of the given RSs. The objective is to minimize the number of vehicles needed to cover all the trips and to minimize the total traveling distance. A MIP formulation and an ALNS heuristic are developed. Keskin and Catay (2016) propose an ALNS algorithm to solve the EVRP with time windows and partial charging. Macrina et al. (2019), in their study which is one of the recent studies on G-VRP, consider the problem with a fleet of vehicles that consists of not only electric vehicles but also traditional vehicles. In the study, apart from partial recharging, they develop a comprehensive energy consumption model in which speed, acceleration, deceleration,

loaded cargo and road gradients are taken into account. A matheuristic, which is a hybrid version of large neighborhood search algorithm (Shaw, 1998), is presented. In the algorithm, customers are inserted or removed from the routes in the solution iteratively.

Recently, different variations of the EVRP have emerged. Breunig et al. (2019) provide an exact method based on decompostion approaches and a large neighborhood search algorithm for the electric two-echelon VRP. In the two-echelon problem, large trucks are used to deliver goods to intermediate facilities in accessible locations, whereas smaller vehicles reach the final customers. The EVRP with a nonlinear charging function has been studied by Froger et al. (2019), while Keskin et al. (2019) introduce the EVRP with time windows involving queueing times at recharging stations and propose a matheuristic that combines ALNS with an exact method. Hiermann et al. (2019) study an EVRP that involves a mix of conventional, plug-in hybrid, and electric vehicles and propose a hybrid genetic algorithm. Taş (2020) take into account the EVRP with flexible time windows that allows vehicles to serve customers outside their time windows with some penalty costs. A column generation procedure is applied to find feasible routes, and then integer solutions are obtained with the routes at hand. To evaluate the qualities of the solutions, Schneider's problem instances (Schneider et al., 2014) are used. By comparison to the case with hard time windows, they present the advantages of flexible time windows. Because the battery consumption rate can change with the effects of uncertain conditions as in drones, Soysal et al. (2020) address stochastic battery depletion in their study. They provide a model for the Pickup and Delivery Problem with Electric Vehicles and Stochastic Battery Depletion. The benefits of respecting stochasticity in energy consumption are shown via a comparison between the solutions obtained by stochastic and deterministic settings.

While electric vehicles are subject to a road infrastructure, this is not the case for drones. Direct flights take place between the points to be covered unless restricted areas are on the way. We propose a method to avoid the restricted areas by generating the distance matrix accordingly. We also discuss how to incorporate the external factors into the input travel times. Although EVRPs tolerate moderate

run times, drone routing requires particularly fast solution times due to the need to rerun the solution algorithm in real-time when the current conditions, such as the wind effect, change.

Our problem focuses on finding the route of a single vehicle such as the TSP but it differs from it as follows. Due to the battery consumption and the limited capacity, we cannot visit all the given points in one tour. Therefore, the route should consist of sequential flight legs such that the drone should start its next leg from the RS it landed to in the previous leg. Moreover, our capacity constraint is in terms of the battery level and differs from the standard capacitated VRP models as follows. The remaining battery should be updated after each visit of the drone to the points to be covered and the stops at RSs differently since the remaining capacity decreases and increases along the route correspondingly.

2.5 Our contributions

With respect to the existing studies, the contribution of this study is threefold.

- We propose a new integer programming formulation for DRP, coupled with a column generation approach. Not only the linear programming (LP) relaxation of the formulation yields strong lower bounds, but also the columns generated at the time the LP relaxation is solved provide high quality solutions for DRP instances to train our learning algorithm.

- We propose an innovative approach that "learns" from the features of high quality solutions obtained via the column generation algorithm and guides the heuristic on recharging decisions. Building on a TSP solution that disregards the recharging needs of the drone, the solution is rendered energy feasible according to the learned recharging actions. Thereby, near optimal solutions can be obtained in a fraction of a second, starting from the TSP solution. To the best of our knowledge, this study is the first one to develop a learning based approach, utilizing a column generation procedure, to solve the DRP.

- We conduct numerical experiments to test the effectiveness of the suggested

method and gain important insights about algorithm design to solve DRP instances and its extensions. In particular our results indicate that:

- The learning based algorithm L&F can generate solutions with 5% optimality gap on the average, for the instances we consider in our experiments.
- Trained on the solutions of small-sized problem instances, L&F can still successfully find high quality solutions for the large size problem instances.
- We are able to identify which features contribute to the learning accuracy the most and hence these findings provide hints to guide the development of new approaches for drone routing and its extensions.

Chapter 3

PROBLEM DEFINITION AND MODEL FORMULATION

In this chapter, we provide the formal definition of the DRP, introduce our integer programming (IP) formulation and present the column generation (CG) procedure to obtain high quality feasible solutions and strong upper bounds.

3.1 Notation and Problem Definition

Let N be the set of nodes to be processed by the drone and R be the set of recharging stations, where the drone can swap its battery with a fully charged one. DRP is defined on a complete directed graph $G = (V, A)$, with the node set $V = (N \cup R)$ and the arc set $A = \{(i, j) : i \in V, j \in V \setminus \{i\}\}$. An arc $(i, j) \in A$ is called a flight arc if $j \in N$ and it is called a recharging arc, otherwise. The travel time associated with an arc $(i, j) \in A$ is denoted by t_{ij} and it includes the time spend to process the node j, if the arc is a flight arc, and the time spend to swap batteries if $j \in R$. The range of the drone, i.e., total flight time with fully charged batteries, is denoted by τ.

The DRP is defined as finding the minimum length walk in G, that starts from an arbitrary recharging station $o_w \in R$, visits all the nodes in N and ends in an arbitrary recharging station $d_w \in R$ such that the length of the walk segments between any two recharging node visits is less than the range of the drone, i.e., it can fly without running out of battery charge.

3.2 Path-segment and Cycle Formulation

The idea to build routes by concatenating path-segments, each representing the route segments between replenishment stops, have been successfully used to model and solve the routing problems with replenishment that arise in many transportation

and telecommunications applications (Yıldız et al., 2016, Yıldız and Karaşan, 2017, Yildiz and Savelsbergh, 2019). Extending this idea, here we consider building the route (walk) of the drone by concatenating simple paths and cycles, where the drone stops at a RS at the end of each path/cycle to replenish its batteries to start a new path/cycle.

Our formulation is built with path-segments which are directed simple paths in G, where the start and destination nodes o_p and d_p of path p are recharging stations. A path-segment p is called feasible, if its length $t_p = \sum_{(i,j)\in p} t_{ij}$ is at most the range of the drone, τ. The set of nodes in path-segment p is denoted by $N(p)$ and the set of all feasible path-segments in G is denoted by P. For notational convenience, we denote the set of all feasible path-segments that visit a node $i \in N$ with $P(i)$. For a set $S \subset R$, we define $\delta^-(S) = \{p \in P : o_p \in S, d_p \in R \setminus S\}$.

Sets	**Description**
V	Set of nodes consisting of nodes that require to be visitied and RSs.
N	Set of nodes that require to be visited
R	Set of RSs.
S	Subsets of RSs.
A	Set of arcs
P	Set of simple paths
C	Set of simple cycles
$P(i)$	Set of simple paths that include node i.
$C(i)$	Set of simple cycles that include node i.
$\delta^-(S)$	Set of simple paths whose starting RSs (s) are in the subset S, but ending RSs (e) are not in the subset S.
Parameters	**Description**
t_{ij}	Travel time associated with arc (i, j)
d_{ij}	Eucledian distance associated with arc (i, j)
t_p	Travel time of simple-path p
t_c	Travel time of simple-cycle c
τ	Total flight time of the drone with fully charged batteries.
Decision Variables	**Description**
x_p	Binary variable that indicates whether a path-segment $p \in P$ is used.
y_c	Binary variable that indicates whether a simple-cycle $c \in C$ is used.
z_r	Binary variable that indicates whether an RS $r \in R$ is visited.
s_r	Binary variable that indicates whether the drone start its tour at $r \in R$.
e_r	Binary variable that indicates whether the drone end its tour at $r \in R$.

Table 3.1: Outline of the Notation

In addition, let us consider a directed simple cycle, where only one node called

the origin can be repeated. In our formulation, we consider only simple cycles where the origin is a recharging station. A simple cycle c is called feasible if its length $t_c = \sum_{(i,j)\in c} t_{ij}$ is at most the drone's range, τ. The origin of a cycle c is denoted by o_c and the set of all nodes visited by the cycle is denoted by $N(c)$. The set all feasible simple cycles in G are denoted by C and the set of all feasible simple cycles that visit a node $i \in N$ is shown by $C(i)$.

With the sets, parameters and decision variables defined in Table 3.1, we develop our path-segment and cycle formulation $\mathcal{P}_{DRP}$ as follows.

$$\min \sum_{p\in P} t_p x_p + \sum_{c\in C} t_c y_c \tag{3.1}$$

s.t.

$$\sum_{r\in R} s_r = 1 \tag{3.2}$$

$$\sum_{r\in R} e_r = 1 \tag{3.3}$$

$$\sum_{p\in P(i)} x_p + \sum_{c\in C(i)} y_c = 1 \qquad \forall i \in N \tag{3.4}$$

$$e_r - s_r + \sum_{p\in P:o_p=r} x_p - \sum_{p\in P:d_p=r} x_p = 0 \qquad \forall r \in R \tag{3.5}$$

$$x_p \le z_{o_p} \qquad \forall p \in P \tag{3.6}$$

$$y_c \le z_{o_c} \qquad \forall c \in C \tag{3.7}$$

$$\sum_{p\in\delta^-(S)} x_p \ge z_r - \sum_{\bar{r}\in S} e_{\bar{r}} \qquad \forall S \subset R, r \in S \tag{3.8}$$

$$x_p \in \{0,1\} \qquad \forall p \in P \tag{3.9}$$

$$y_c \in \{0,1\} \qquad \forall c \in C \tag{3.10}$$

$$z_r \in \{0,1\} \qquad \forall r \in R \tag{3.11}$$

$$s_r \in \{0,1\} \qquad \forall r \in R \tag{3.12}$$

$$e_r \in \{0,1\} \qquad \forall r \in R \tag{3.13}$$

The objective is to minimize the total travel time. Constraints (3.2) and (3.3) determine the start and end nodes of the drone's walk. Constraints (3.4) ensure

that each node is visited by the drone exactly once. Constraints (3.5) are flow balance constraints to provide the concatenation of the simple paths to form the route of the drone between recharging stations. Constraints (3.6) and (3.7) ensure that a path/cycle can only be used if the origin is an activated recharging point. Constraints (3.8) eliminate the sub-tours formed by path-segments. To achieve this, for any set of recharging stations that contains at least one activated recharging station and does not include the recharging station in which the drone completes its walk (i.e., $\sum_{\bar{r}\in S} e_{\bar{r}} = 0$), Constraints (3.8) ensure that there is at least one path-segment going out of S (i.e., $\sum_{p\in\delta^-(S)} x_p \geq 1$). Finally, (3.9)- (3.13) are the variable domain restrictions.

Note that formulation $\mathcal{P}_{DRP}$ determines the recharge points to activate, along with the route of the drone. However, due to the related cost of establishing a recharging location, it is not very likely to have a large number of them in real world DRP instances. In fact, what basically necessitate a special treatment of DRP, instead of using a basic TSP, is the scarcity of those recharging locations. Therefore, in the optimal solutions for those realistic problem instances with a small yet strategically located recharging points, it is quite likely that the drone would visit all of the established recharging stations, as we also observe in our numerical experiments. With this observation, one can simplify $\mathcal{P}_{DRP}$ by focusing on those solutions where $z_r = 1$ for all $r \in R$. Note that with this modification constraints (3.6) and (3.7) become redundant, and constraints (3.8) simplify to:

$$\sum_{p\in\delta^-(S)} x_p + \sum_{\bar{r}\in S} e_{\bar{r}} \geq 1 \qquad \forall S \subset R, \tag{3.14}$$

Obviously, a similar simplification would be possible if some of the variables z are set to zero, meaning that those stations will not be visited by the drone. Considering the significant computational advantages of such a simplification, solving $\mathcal{P}_{DRP}$ for each active recharge station combinations and then choosing the one with the best result becomes a computationally viable option when the number or recharging stations are small, as one would expect in realistic problem instances. Although the column generation approach we describe next can be used for the original formulation

(3.1)-(3.13), for the sake of simplicity, from here on, when we mention $\mathcal{P}_{DRP}$ we refer to the simplified formulation: (3.1) -(3.5), (3.9), (3.10) and (3.12)-(3.14).

Clearly, the number of feasible path-segments $|P|$ and the number of feasible simple-cycles $|C|$ grow exponentially with the number of nodes to be visited by the drone and it is not possible to solve $\mathcal{P}_{DRP}$ directly for realistic size problem instances. However, contrary to compact "arc-based" formulation alternatives in which big M type constraints are needed to keep track of the energy level of the drone and eliminate subtours, $\mathcal{P}_{DRP}$ can provide strong linear relaxation bounds and solving it with a "good" subset of path and cycle variables can provide high quality solutions with small optimality gaps. Both of these outputs are critical to develop (train and test) the learning based solution method we propose in this study and can be achieved by solving the linear relaxation LP_{DRP} with a column generation approach we present in the next section.

3.3 Column Generation Algorithm

We start the column generation algorithm (CG) with a restricted formulation that contains a subset of path and cycle variables and extend it iteratively by adding new path-segments and simple-cycles as needed. More precisely, at the k_{th} iteration of the CG we solve the restricted formulation LP^k_{DRP}, which contains the path and cycle variables added in the previous iterations, and then look for the paths and the cycles variables that are not in the restricted master problem but have favorable reduced costs to include in them in the model. To identify such variables we solve two *pricing problems* (one for path variables and one for cycle variables) and terminate the CG when solving the pricing problems do not return path and cycle variables with negative reduced costs. Here we want to note that the number of subtour elimination constraints (3.8) is a function of the number of recharging stations (not the number of nodes to visit), and thus for the problem instances we consider in this study (which have no more than five recharging stations) it is possible to include all of those constraints in the model a priori. However, from a technical perspective one can consider a cutting plane approach to include them in the model in an iterative

way by using the efficient separation approaches that exists in the literature (Lee et al., 1996)

Pricing problem for the path variables: Let α, β and λ be the dual variables associated with the constraints (3.4), (3.5) and (3.14), respectively. For a path-segment $p \in P$ the reduced cost π_p of the variable x_p can be calculated as follows.

$$\pi_p = \sum_{(i,j)\in p} t_{ij} - \sum_{i\in p} \alpha_i + \beta_{o_p} - \beta_{d_p} + \sum_{S\subset R: d_p\in S, o_p\notin S} \lambda_S \tag{3.15}$$

Let $P(o,d)$ be the set of the feasible path-segments in G that start from the recharging station o and end at the recharging station d, i.e., $P(o,d) = \{p \in P : o_p = o, d_p = d\}$. Note that, for all $p \in P(o,d)$ we can write the reduced cost as $\pi_p = F_{od} + \sum_{(i,j)\in p} \bar{t}_{ij}$, where $F_{od} = \beta_o - \beta_d + \sum_{S\subset R: d\in S, o\notin S} \lambda_S$ and $\bar{t}_{ij} = t_{ij} - \alpha_i$, for all $(i,j) \in A$. So it is easy to see that for an origin destination pair (o,d), the path-segment with the smallest reduced cost can be found by solving a resource constrained shortest path problem (RCSP; Beasley and Christofides, 1989) on the graph G considering the arc costs $\bar{t}_{ij}$ and resource usage t_{ij} for all $(i,j) \in A$. Let $p^*_{od} \in P(o,d)$ be the simple-path with the shortest length $\bar{t}_{p^*} = \sum_{(i,j)\in p^*} \bar{t}_{ij}$. Clearly, if $F_{od} + \bar{t}_{p^*} < 0$ we can conclude that the variable x_{p^*} needs to be added to the restricted formulation in the next iteration of the CG, and $F_{od} + \bar{t}_{p^*} \geq 0$ implies that $\pi_p \geq 0$ for all $p \in P(o,d)$. The MIP model we use to solve RCSP instances are presented in Appendix 7.1. We note that solving RCSP by MIP suffices for our purposes as the run times are reasonable for the size of the problems that we solve. For larger instances, specialized RCSP algorithms may be utilized instead of MIP.

Pricing problem for the cycle variables: Let α be the unrestricted dual variable associated with the constraints (3.4). Then the reduced cost π_c of a cycle variable $y_c, c \in C$ can be calculated as follows.

$$\pi_c = \sum_{(i,j)\in c} t_{ij} - \sum_{i\in c} \alpha_i \tag{3.16}$$

Note that using a similar cost transformation for the arcs we used in the previous case, i.e., setting $\hat{t}_{ij} = t_{ij} - \alpha_i$, for all $(i,j) \in A$, we can solve the pricing problem to detect cycle variables with negative reduced costs by solving RCSP problems on the *pricing* graphs $G_o = (\hat{V}, \hat{A}), o \in R$, which we define as follows.

For each pricing graph $G_o = (\hat{V}, \hat{A})$, we create a copy of the recharging station o, denoted as $\hat{o}$, and define $\hat{V} = V \cup \{\hat{o}\}$. Similarly, we define the arc set as $\hat{A} = A \cup \{(\hat{o}, i) : i \in V \setminus \{o\}\} \cup \{(i, \hat{o}) : i \in V \setminus \{o\}\}$. For the "original" arcs $(i,j) \in A$, $\hat{t}_{ij}$ is the cost and t_{ij} is the resource usage. For the "artificial" arcs the costs and the resource useges are the same with the associated original arcs (i.e., $\hat{t}_{\hat{o},i} = \hat{t}_{o,i}$, $\hat{t}_{i,\hat{o}} = \hat{t}_{i,o}$, $t_{\hat{o}i} = t_{oi}$ and $t_{i\hat{o}} = t_{io}$).

Clearly, if the solution of the RCSP problem from o to $\hat{o}$ in G_o, with the arc costs $\bar{t}$ and resource usage t, has a non-negative solution value we can conclude that there is not any cycle that starts from o, for which the variable y_c has a negative reduced cost. Otherwise, the solution of the RCSP problem returns a cycle c^* such that y_{c^*} has a negative reduced cost and needs to be added to the restricted formulation in the next iteration of the CG algorithm.

As explained above, we solve the pricing problem at each iteration k of the CG algorithm by solving a $|R| \times (|R| - 1) + R$ RCSP problems (one for each origin destination pair $(o,d) \in R \times R$ for the path-segment variables, and for each recharging station $o \in R$ for the cycle variables). If any of those RCSP solutions produce a path or cycle variable with negative reduced costs we add the associated variables to the restricted model LP^k_{DRP} and move to the next iteration. Otherwise, we terminate the CG algorithm declaring the solution of LP^k_{DRP} as the optimal solution of LP_{DRP}.

Once LP_{DRP} is solved, we then apply the well-known heuristic to find a high quality solution for the $\mathcal{P}_{DRP}$ by solving it by considering only those path and cycle variables generated throughout the CG iterations. As we discuss in more detail when we present the results of our numerical experiments, such a procedure can provide high quality solutions that can be used to train our learning algorithm as we explain next.

Chapter 4

LEARNING BASED ALGORITHM

This chapter presents the learning based algorithm L&F to solve the DRP. The main idea behind the learning based algorithm L&F is to first construct a *giant walk* that visits all the nodes without considering the recharging needs of the drone and then insert recharging stops by learning how to do so from the high quality solutions. With the help of the learning outputs, at any point during the giant walk, it is decided whether to visit an RS or not before moving to the next node. Because the prediction takes very short time, L&F essentially reduces DRP to solving merely a TSP for which there exists several approaches that can produce good solutions in very short run times (Dorigo et al. (1996), Focacci et al. (2002)). As we discuss in detail in Section 5, where we present the results of our numerical experiments, such a two step approach, namely solving the TSP without considering the recharging needs and then inserting the recharging stops on the resulting walk can indeed provide good solutions (i.e., solutions with around 5% optimality gaps on the average for the large instances).

Next, we discuss the features we consider to learn the timing and the locations of recharging stops and their relative importance to make accurate predictions as well as the alternative supervised machine learning algorithms that can be used for the task.

4.1 Describing the Features and Learning

Clearly, the most important step of devising a learning algorithm is determining the feature set (variables) to base the predictions. During the feature selection process, one decides which features are to be measured in light of learning targets. Clearly, in our case the target labels are visiting or not visiting an RS and our features

should be closely related to the circumstances that affect the replenishment time and place. Along this goal, we selected a diverse set of features that can be divided into three groups. The first group consists of the features that affect the current battery capacity of the drone, i.e. features F1, F2, F3, and F4 in Table 4.1. The second group is related to the factors that affect the battery capacity in the next step determined by the decision whether to visit an RS or not. Features F5, F6, F7, and F8 constitute this group. The third group with only one member, feature F9, helps L&F to make recharging decisions by considering how many nodes are left to complete the tour. As we discuss more in detail in Chapter 5 when we present the results of our numerical experiments, all these features collectively play important roles to provide high accuracy predictions, even though the battery level (F1) and extra distance (F4) have the highest impact on the predictions.

ID	**Feature**	**Definition**
1	Battery Level	The current battery level
2	Current-Next	The distance between the current node and the next node in the giant walk
3	Current-Nearest	The distance between the current node and the nearest node to it
4	Extra Distance	The extra distance covered by the drone if it visited the recharge station instead of visiting the next node in the giant walk
5	Next-RS	The distance between the next node in the giant walk and the nearest recharge station to it
6	Nearest-RS	The distance between the nearest node to the current node and the nearest recharge station to it
7	Next-Second	The distance between the two successive nodes after the current node in the giant walk
8	Second-RS	The distance between the second node after the current node in the giant walk and the nearest recharge station to it
9	Unvisited Nodes	The number of nodes that the drone has not visited yet

Table 4.1: Definitions of the Features

4.2 The Choice of the Supervised Machine Learning Algorithm

One of the critical steps is to determine which supervised machine algorithm is best able to characterize the training set. The algorithms that are judged to be satisfactory with the preliminary tests can be used as classifiers (Kotsiantis et al., 2007). Choosing a classifier from the learning algorithms and predicting a classifier's accuracy rate in the real world data sets are strongly related to the final accuracy of a classifier. Therefore, estimating a consistent classification accuracy is significant for the success of learning (Kohavi (1995)). As in most studies on learning, we also prefer to evaluate the learning model via the k-fold cross validation method. In this method, the data set is split into k subsets such that each of them has almost the same size. While k-1 subsets constitute the training set, the remaining one is used as the validation set, and this process is repeated k times for each subset to reduce bias and variance.

Among supervised machine algorithms (e.g., Support Vector Machine, Linear Regression, Naive Bayes and Neural Networks), we preferred to use the Decision Tree and k-Nearest Neighbors (kNN) algorithms as two different approaches for the learning phase in the L&F, as they resulted in the highest classification accuracy in almost every tested instance of the DRP.

We next explain the implementation of the L&F.

4.3 Implementation Procedure of Learn and Fly

For the learning phase, we utilize the feasible integer solutions obtained from the CG procedure as the training set to understand when and where the drone is replenished. After obtaining the IP solutions for the training set, the values of the features listed in Table 4.1 are calculated. Then, as shown in Figure 4.1, the two selected supervised machine learning algorithms as two alternative ways are used to conduct the learning phase.

A decision tree is a tree-like structure constructed with data partitioning in a recursive manner called "recursive partitioning". In each iteration, each child node, i.e. each leaf, represents the outcome of the test while the branches represent the

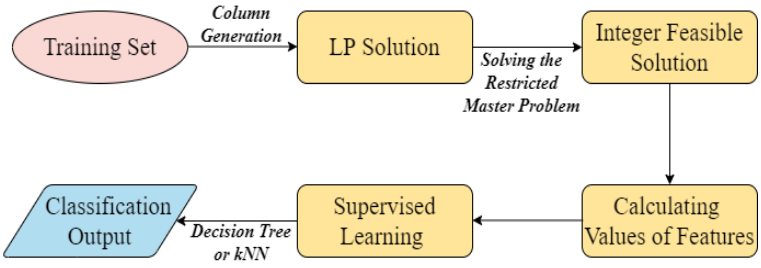

Figure 4.1: Steps of Learning

decision rules. In the decision tree learning, each internal node is split into two child nodes via the chosen split criterion named as "Attribute (Feature) Selection Measures (ASM)" (Aruna, 2013, Nefeslioglu et al., 2010, Shaikhina et al., 2019). There is no strict rule for ASM and it has been the subject of many articles (Patidar et al. (2015), Shaheen et al. (2020) and Sagoolmuang and Sinapiromsaran (2020), to name a few). In our study, we prefer to use the "Gini Index" measure (Aruna, 2013) because it is not only easy to implement but also provides a high accuracy score (94%) for the DRP. On the other hand, the performance of the kNN algorithm depends on the value of "k". Therefore, some studies (e.g., Guo et al. (2003) and Song et al. (2007)) consider the problem of choosing the parameter "k". Because the number of features utilized in our training data is rather small, i.e. 9, the value of "k" is chosen empirically according to the problem at hand (Hassanat et al., 2014); thus, we determine the value of "k" as 5 with the best accuracy scores.

After we obtain a high classification accuracy (e.g. 94% for Decision Tree, and 95% for kNN), we split our training data into two parts (i.e. the training set and the testing set) to get a good generalization for future predictions. Because the training data is randomly split, different classification outputs can diversify the solutions obtained at the end of the learning algorithm. Therefore, we repeat the supervised learning phase 10 times to obtain 10 different classification outputs before implementing our algorithm. Although there are small differences between the objective values of the solutions obtained through each output (e.g., the average value of the solutions for the instances with 30 nodes decreases around 2 percent), we solve the

same instance repetitively with these classification outputs to get the best solution on the grounds that this approach does not increase the computational time too much (e.g., the average time for the instances with 30 nodes increases from 0.48s to 3.95s).

The integer solutions that we obtain from the the column generation procedure for the training data set show that the order of the nodes in the optimal solutions is almost the same as the order in the TSP solutions. Therefore, we find the optimal TSP solution for each instance. For each TSP solution, we decide which node will be the starting point in the L&F by excluding the arc with the largest distance between two nodes in the related TSP solution. The nodes that have the biggest distance between them give us two starting points. After we decide the starting points, we obtain the giant walks. The L&F, whose overview is presented in Figure 4.2, predicts whether the drone should visit an RS or not whenever the drone is at a node along the giant walk. When it is decided that the drone should visit an RS, the RS causing the minimum increase in the distance of the tour, according to the parameter “Extra Distance", is preferred firstly. If visiting this RS causes infeasibility (i.e., the drone runs out of battery on the way), then the drone visits the nearest RS to it. We start and finish the algorithm at an RS while not forcing the drone to visit all the existing RSs. Once the drone visits all the required nodes, we look for an improvement in the solution. For each simple path which has only one processing node, the algorithm implements a local improvement procedure by adding the node in the simple path to the previous path or the next path in the solution. When the move provides an improvement in the solution, the simple path with one node is excluded from the solution. The algorithm ends after the improvements have been completed. Algorithm 1 presents the pseudo-code of the L&F. Lines 1 to 6 repeat the L&F procedure by the number of classification outputs and giant walks. Line 8 runs the L&F until the all nodes are processed. Lines 9 to 14 set the values of the parameters used in the L&F. Lines 15 to 39 determine the next behavior of the drone by updating the battery charge. Lines 40 to 54 carry out the improvement step. Line 55 returns the best solution found.

Algorithm 1 Pseudo-code of the Learn and Fly algorithm

Input: V: Set of nodes, N: Set of nodes that should be processed, R: Set of RSs, τ: Battery Limit, giant walks, classification outputs

Output: A feasible route in which all nodes in N are processed

Begin
$i \leftarrow 1$ to 10 *(the number of the classification outputs)*
for each classification output **do**
  $k \leftarrow 1$ to 2 *(the number of the giant walks)*
  $\tau^- = \tau$
  for each giant walk **do**
    Find the nearest RS to the first node of the giant walk and start routing
    while the number of nodes processed $< |N|$ **do**
      Calculate the values of the features for the current node
      $RS_e \leftarrow$ RS found via feature "Extra Distance"
      $RS_c \leftarrow$ the nearest RS to the current node
      $RS_n \leftarrow$ the nearest RS to the next node in the giant walk
      $d_n \leftarrow$ distance to the next node in the giant walk
      $d_{rs} \leftarrow$ distance between the next node in the giant walk and RS_n
      Predict whether the drone should visit an RS according to the classification output
      if the drone visits an RS **then**
        if distance to $RS_e \leq \tau^-$ **then** *(feasibility check)*
          the current node $\leftarrow RS_e$
          $\tau^- = \tau$
        else
          the current node $\leftarrow RS_c$
          $\tau^- = \tau$
        end if
      else
        if $d_n + d_{rs} \leq \tau^-$ **then** *(feasibility check)*
          the current node $\leftarrow$ the next node in the giant walk
          $\tau^- = \tau^- - d_n$
        else
          if distance to $RS_e \leq \tau^-$ **then** *(feasibility check)*
            the current node $\leftarrow RS_e$
            $\tau^- = \tau$
          else
            the current node $\leftarrow RS_c$
            $\tau^- = \tau$
          end if
        end if
      end if
    end while
    $GiantSol^k \leftarrow$ solution
    Define each group of the nodes from an RS to another RS in the $GiantSol^k$ as a path
    Start improvement steps
    for each path in $GiantSol^k$ **do**
      if there is only one node processed in the path **then**
        if it is feasible and provides an improvement **then**
          Add the node of the previous path or the next path in $GiantSol^k$ whichever provides the best improvement
          $GiantSol^k \leftarrow$ new solution
        end if
      end if
    end for

50: $k \leftarrow k + 1$
51: **end for**
52: $BestGiantSol^i \leftarrow BestGiantSol^k$
53: $i \leftarrow i + 1$
54: **end for**
55: $BestSolution \leftarrow BestGiantSol^i$
56: **End**

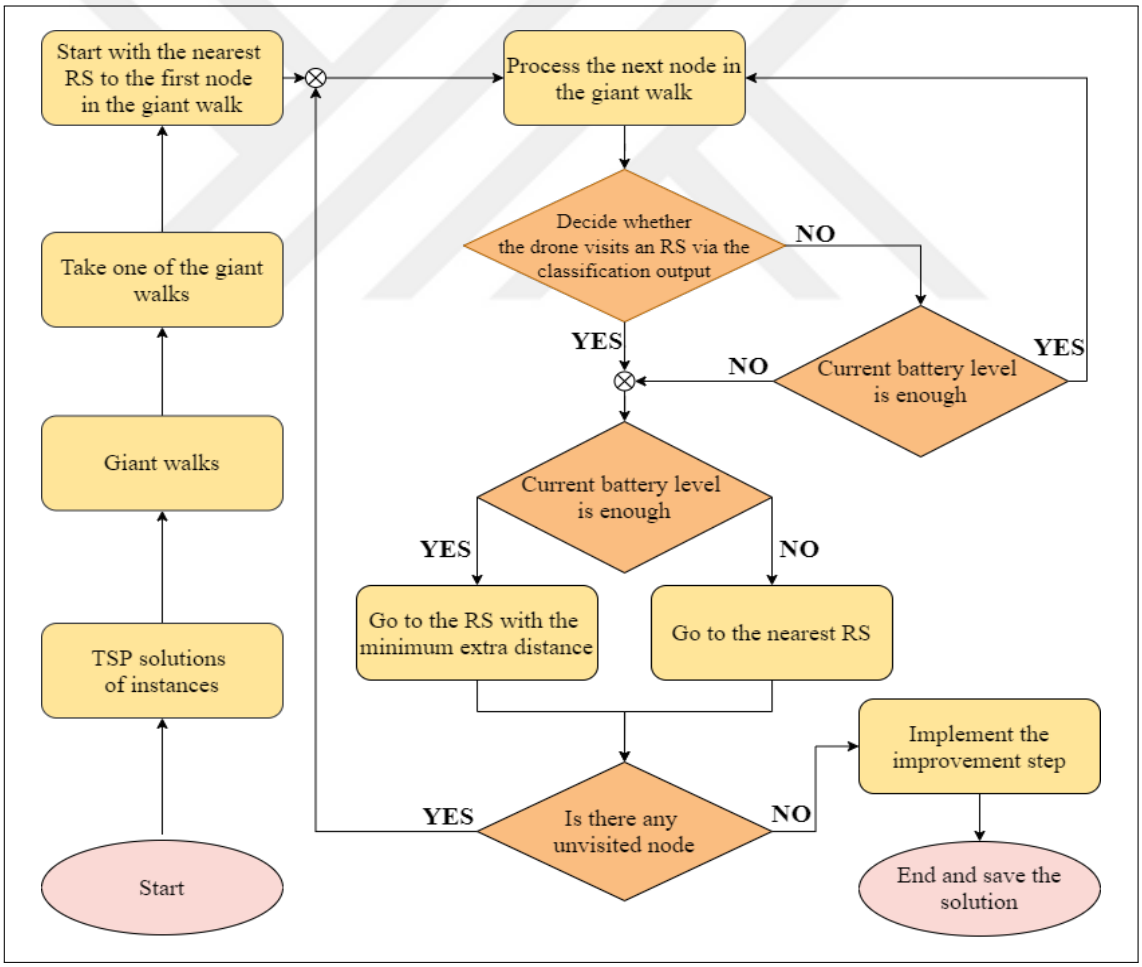

Figure 4.2: An Overview of the Learn and Fly Algorithm

Chapter 5

COMPUTATIONAL RESULTS

In this section, we carried out extensive experiments to analyze the performance of L&F. Our computational experiments are performed on a computer with Intel(R) Core(TM) i7-7600U CPU @ 2.80 GHz processor with a memory of 8 GB. The mathematical models are implemented using Java while Python 3.8.1 is used for the L&F. The mathematical models are solved using ILOG CPLEX 12.8.

5.1 Instance Generation

We test our proposed algorithm on two groups of instances (Ermağan et al. (2020)). The first group is targeted towards the precision agriculture application. For each instance, a number of nodes are selected randomly by choosing coordinates in a plane containing some restricted areas. Then, recharge stations are selected as cluster centers of the nodes. The second group of instances are obtained from the "kroA100.tsp" TSP benchmark instance (Krolak et al. (1971)), with an aim to investigate whether the learning outputs generated from a specific network topology is still useful for solving instances coming from a different one. In both groups it is assumed that the battery of the drone allows a 30 minute flight time. Accordingly, to ensure that each instance is feasible, we select the nodes in an area with dimensions MxM, where "M" represents the maximum distance that can be covered by the drone with fully charged battery. The first group consists of 60 problem instances, having 20, 30, and 40 nodes, where 20 instances are generated for each fixed number of nodes. In all instances, the nodes are selected within an area having the same size so that different node densities can be tested.

In addition to the first instance group, we obtain 20 more instances, each having 40 randomly chosen nodes from the ones in "kroA100.tsp". Since the area in which

the nodes are distributed is generally bigger than the MxM area in these instances, we reduce the indicated distances to get feasible solutions with the same battery range as we use in the first group. Because the dimensions of the biggest area that covers all nodes in the instances obtained from 'kroA100.tsp" is almost twice M, we scale the distances to 1/2. Here we want to note that processing a large number of nodes (i.e., more than 40) with a single drone is impractical as it would take too much time and hence multiple drones will need to be used in such cases. Therefore, we limited the size of our test instances by 40 nodes.

To determine the number of RSs, we performed preliminary tests. When we implemented L&F on several instances with different number of RSs, we observed that the drone does not visit one or more RSs at all when the number of RSs is more than five. On the other hand, when the number of the RSs is set between one and four, it causes infeasible solutions for the instances in the MxM area. Therefore, we locate five RSs whose locations are selected as the cluster centers by dividing the nodes into five groups (i.e. the number of RSs) using the K-means++ clustering algorithm (Arthur and Vassilvitskii (2007)) that is solved by Python 3.8.1. After determining the locations of the nodes and the RSs, we calculate the pairwise Euclidean distances and obtain the distance matrix for each instance.

5.1.1 Avoiding Restricted Areas

Flying over some regions such as military bases and private property may not be allowed. When there is a restricted area between two nodes, the drone has to go around it. Therefore, we update our distance matrices according to the locations of the restricted areas.In order to detect which arcs in the complete directed graph intersect with restricted areas, we divide the lines between each pair of nodes into multiple points. Then, we check whether the line passes through the restricted area or not by using two equations from analytical geometry that are specified in the following.

We define all the restricted areas as a polygon where A represents its area and (x, y) represents the coordinates of the corners. The formula of the area is as follows:

$$A = \frac{1}{2}|(x_1y_2 - y_1x_2) + (x_2y_3 - y_2x_3).....(x_ny_1 - y_nx_1)|$$

With the points which we get from the lines, we draw four triangles whose corners consist of the point, (x_t, y_t), we found and two corners of the polygon as shown in Figure 5.1a and Figure 5.1b. Each triangle's area can be found as follows:

$$A_{triangle} = \frac{1}{2}|x_ty_1 + x_1y_2 + x_2y_t - (x_1y_t + x_2y_1 + x_ty_2)|$$

By using these two formulas, we check all edges and determine which edges intersect with the restricted areas. For any point on the arc, if the sum of the areas of the four triangles equals to the area of the quadrilateral, this implies that the point is inside the quadrilateral as in the Figure 5.1a. On the other hand, if the sum of the areas of the four triangles is greater than the area of the quadrilateral for every point on the arc, we can deduce that the arc does not intersect with the restricted area. After detecting the arcs which intersect with the restricted area, the

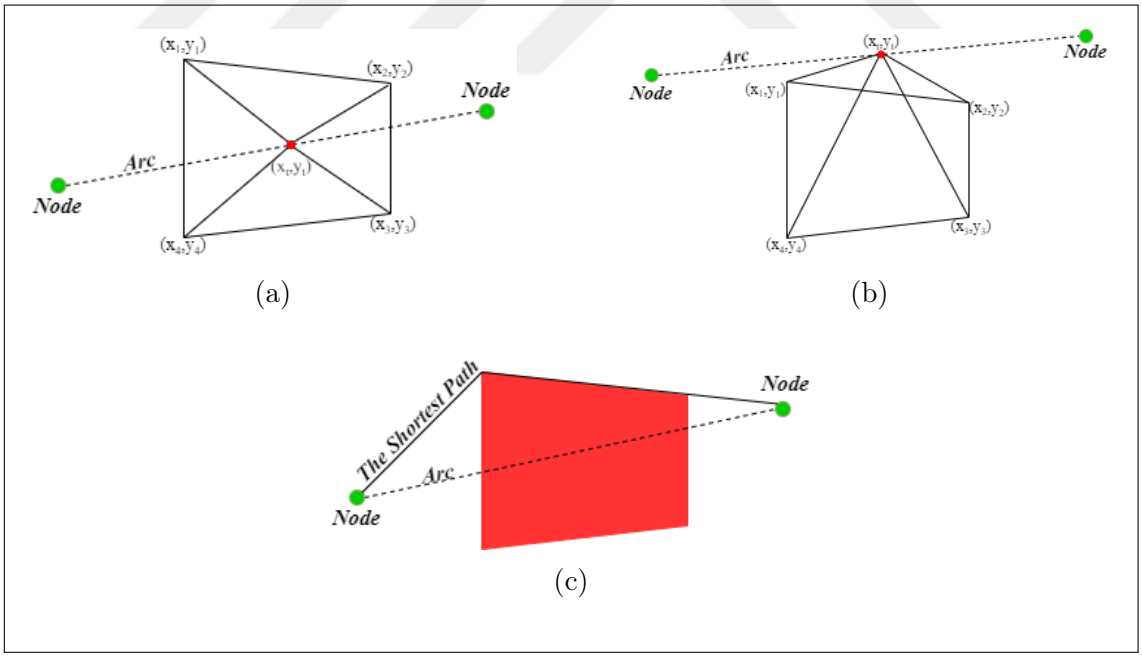

Figure 5.1: Avoiding Restricted Areas

distances of the shortest paths between all pairs of the nodes associated with these arcs are calculated and the distance matrix is updated accordingly. An example

of an arc that intersects with a restricted area and the shortest path between the related nodes is provided in Figure 5.1c.

5.2 Performance Analysis

This section presents the performance of the L&F with the quality of the integer solutions obtained from the CG procedure.

5.2.1 Quality of the Integer Solutions Obtained from the Column Generation Method.

In Table 5.1, we present the optimality gaps of the integer solutions obtained from CG with respect to the LP relaxation bounds and the computational times for the first group of instances. The average optimality gap obtained from CG is less than 2% for these instances of various sizes. LP solutions are generated in an average of 8.33 seconds for the instances with 20 nodes. On the other hand, the average computational times are 37.43 seconds and 170.14 seconds for the instances with 30 nodes and 40 nodes, respectively. The integer solutions obtained from CG for the “kroA100.tsp" instances also provide us low optimality gaps. In these instances, while the minimum and maximum optimality gaps are 0% and 4.73%, respectively, the average optimality gap is less than 2% as in the first group of instances. Among the 20 instances obtained from "kroA100.tsp", the CG finds the optimal solution for the 8 instances. As a result, we conclude that the integer solutions generated by CG provide us a good training set for classification as they are very close to the optimal solutions. On a different note, the computational times of the “kroA100.tsp" instances are well above those of the first group of instances (e.g., the average computational time is 863.51s). The high computational times show that obtaining integer feasible solutions from CG is not a viable method for dynamic environments, where re-optimization should be achieved in very short time.

Instance	20 Nodes			30 Nodes			40 Nodes		
	LP Time(s)	**IP Time(s)**	**GAP (%)**	**LP Time(s)**	**IP Time(s)**	**GAP (%)**	**LP Time(s)**	**IP Time(s)**	**GAP (%)**
1	16.72	16.80	2.80	38.68	38.88	1.03	136.15	136.31	2.88
2	5.30	5.33	0.00	37.00	37.04	0.00	165.29	165.59	2.44
3	7.25	7.28	0.00	32.60	32.64	0.00	180.38	180.48	1.80
4	5.80	5.88	1.20	35.08	35.42	0.33	132.60	133.01	5.31
5	7.47	7.61	0.40	29.48	29.64	0.18	190.06	190.22	1.35
6	6.92	6.97	1.80	26.81	26.93	1.57	137.94	137.98	0.00
7	9.27	9.29	0.00	37.33	37.46	1.24	205.93	206.15	3.20
8	5.88	5.91	0.00	31.28	31.31	0.00	149.39	149.55	1.67
9	8.72	8.87	0.50	21.89	21.92	0.00	114.22	114.39	0.96
10	5.75	5.79	0.00	32.35	32.40	1.67	147.55	147.92	6.65
11	4.79	5.04	0.90	37.10	37.16	0.00	173.39	173.51	1.68
12	12.27	12.30	0.00	33.85	34.02	1.66	340.42	340.63	1.99
13	11.21	11.36	0.20	40.45	40.54	0.73	78.10	78.24	2.43
14	9.38	9.41	0.00	48.27	48.42	1.03	320.02	320.18	0.09
15	9.32	9.40	1.30	29.19	29.34	1.78	130.33	130.44	2.63
16	7.71	7.87	3.40	85.55	85.71	0.78	207.64	207.77	0.57
17	9.56	9.70	1.00	44.38	44.53	1.17	250.35	250.39	0.00
18	9.54	9.57	0.00	30.68	30.83	1.12	137.63	137.73	0.50
19	5.07	5.21	0.00	31.70	31.86	0.19	126.73	126.81	4.35
20	8.61	8.64	0.00	44.87	44.95	0.00	78.59	78.74	0.45
Min:	4.79	5.04	0.00	21.89	21.92	0.00	78.10	78.24	0.00
Max:	16.72	16.80	3.40	85.55	85.71	1.78	340.42	340.63	6.65
Avg.:	8.33	8.41	0.68	37.43	37.55	0.72	170.14	170.30	2.05

Table 5.1: LP-IP Solutions

5.2.2 Performance Analysis of the Classification

In Figure 5.2a, the features' impacts on classification accuracy (Nembrini et al. (2018), Ronaghan (2018)) are demonstrated. Because the feature "Extra Distance" gives the drone a hint about the extra time spent in case an RS is visited, it has the highest importance score compared to others. On the other hand, even though some features do not seem to have an important role in the learning, as seen in Figure 5.2a, they contribute to the accuracy collectively. The accuracy of learning with only two features leading to the highest impacts, namely, "Battery Level" and "Extra Distance", decreases around 12% (e.g. the accuracy decreases from 94% to 83% for Decision Tree with 30 nodes learning). This result drives us to use more features instead of two, even if the two are the features with by far the highest importance scores.

The final accuracy rates of the classification algorithms are calculated via k-fold cross validation method which is explained in Section 4.2. As seen in Figure 5.2b, the accuracy scores of the Decision Tree and KNN are around 94% and 95%,

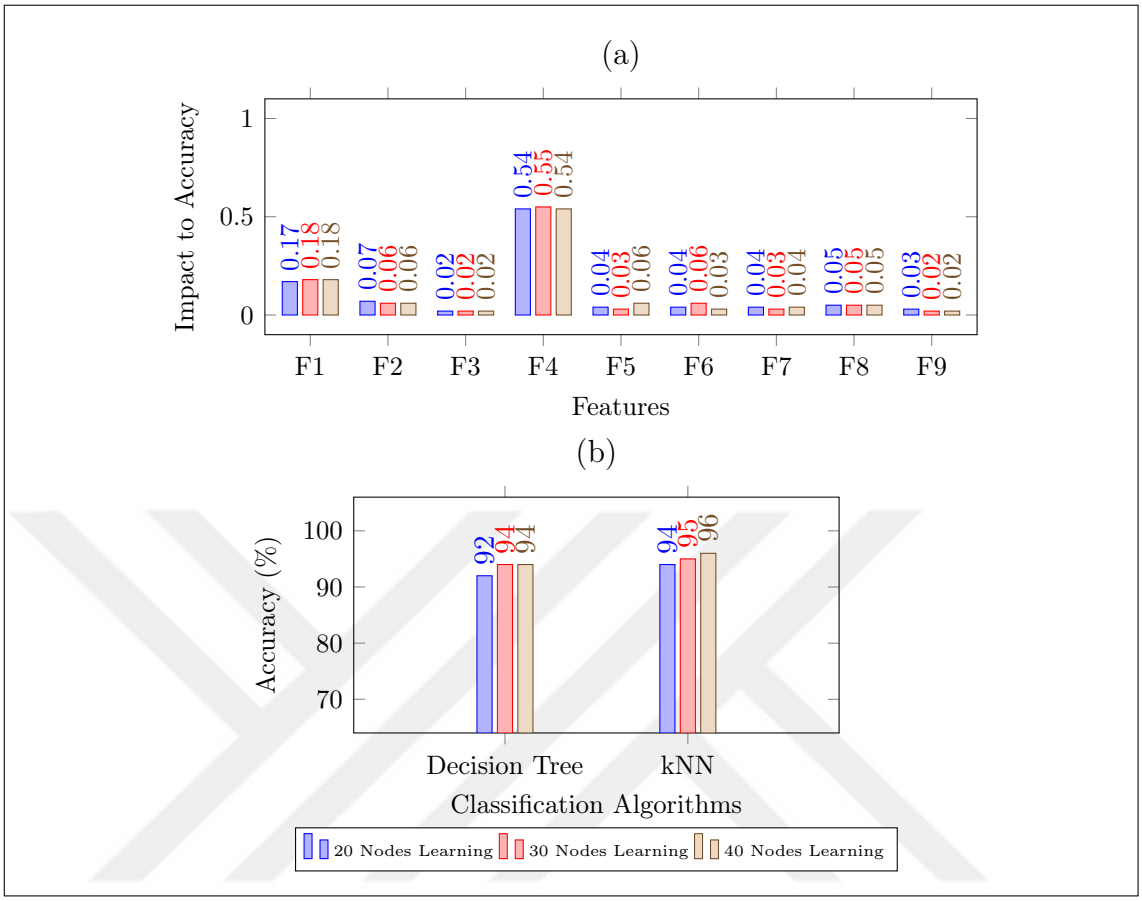

Figure 5.2: The Analysis of Accuracy Scores

respectively. The effect of the number of classification outputs is shown in Figure 5.3. As explained in Section 4.3, the solution improves as the number of classification outputs increases while the increase in the number of the classification outputs does not bring about too much increase in the computational time.

5.2.3 Performance of the Learn and Fly Algorithm

We report the results of our experiments with the L&F in this subsection. We perform tests on a total of 80 instances, which are generated as explained in Subsection 5.1. We compare the objective value of the solutions found by L&F with the LP relaxation lower bound obtained from the IP model using CG as presented in Section 3.

The L&F algorithm starts by solving a TSP on the set of nodes N. Algorithms

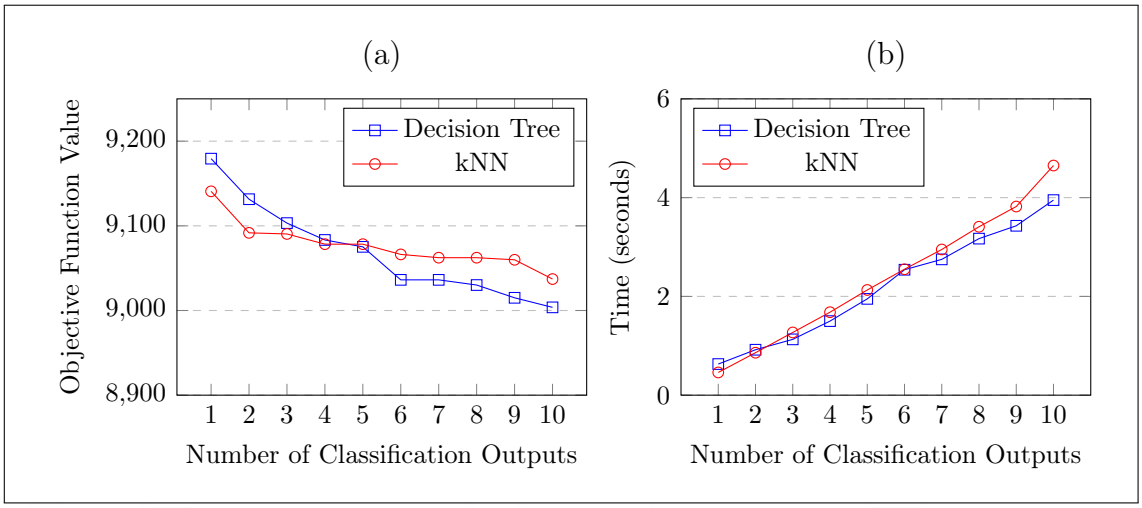

Figure 5.3: Results with Different Number of Classification Outputs for Instances with 30 Nodes

developed to solve the TSP are abundant. Laporte (2010) review the exact and heuristic solution methods. For instance, Grötschel and Holland (1991) propose a cutting plane procedure that can solve the symmetric TSP with up to 1000 nodes optimally. Also, Applegate et al. (2009) develop a TSP solver application named as Concorde. With this application, the symmetric TSP can be solved with up to tens of thousand nodes optimally. When we solve our instances with 40 nodes via Concorde (the distance matrices in this study are symmetric), we observe that it obtains the optimal solution in 0.05 seconds for each one. Furthermore, when dynamic environments are considered for the DRP, the solution methods for asymmetric TSP gain importance. (For instance, the wind factor can make the distance matrix asymmetric.) For the asymmetric TSP, Carpaneto et al. (1995) propose a branch and bound algorithm based on the assignment relaxation. In their study, they present the solutions of the instances with up to 2000 vertices that are found in less than three minutes. Hence, if the conditions change while the drone hovers, a new TSP solution that is used at the beginning of the L&F can be found using these algorithms in a short time.

In Tables 5.2-5.5, the results of our experiments with L&F are reported. In these tables, the columns “Decision Tree Solution" and “kNN Solution" represent

the objective values of the L&F solutions obtained when the classifier is Decision Tree and kNN, respectively. The column “Time" shows the execution time of L&F that differs slightly according to the supervised machine learning algorithm used. The column “LP Relaxation Solution" is used to calculate the values of the columns “Decision Tree GAP(%)" and “kNN GAP(%)" that are the percentage gaps between the L&F and the LP relaxation solutions.

In Table 5.2, for the 20 instances having 20 nodes, the average execution time of L&F is shown as 1.67 seconds for the Decision Tree and as 2.03 seconds for the kNN versions. Besides running fast, the L&F tends to find the solutions very close to the optimal objective value of the model $\mathcal{P}_{\mathcal{DRP}}$ presented in Section 3. The average percentage optimality gaps are 5.78 and 5.76 for the two versions of L&F. L&F finds the optimal solutions for instances numbered as 8, 14, and 20.

Instance	**Decision Tree Solution**	**Time (s)**	**kNN Solution**	**Time (s)**	**LP Relaxation Solution**	**Decision Tree GAP (%)**	**kNN GAP (%)**
1	6775.67	1.54	6775.67	2.11	6356.46	6.60	6.60
2	8107.05	1.58	8088.51	2.14	7569.80	7.10	6.85
3	5909.07	1.96	5909.07	1.93	5819.40	1.54	1.54
4	7651.92	1.97	7554.51	1.97	7361.60	3.94	2.62
5	7485.16	1.87	7501.35	1.95	6616.87	13.12	13.37
6	6540.45	1.49	6540.45	2.00	6340.27	3.16	3.16
7	6891.93	1.49	6891.11	2.15	6295.31	9.48	9.46
8	6464.16	1.46	6464.16	1.99	6464.16	0.00	0.00
9	6921.31	1.94	7024.81	2.04	6504.87	6.40	7.99
10	6088.54	1.65	6176.37	1.94	5890.98	3.35	4.84
11	7864.03	1.78	7560.66	2.05	7163.68	9.78	5.54
12	7142.53	1.67	7142.53	1.91	6488.03	10.09	10.09
13	6273.01	1.45	6273.01	2.10	6262.93	0.16	0.16
14	5596.23	1.68	5596.23	1.99	5596.23	0.00	0.00
15	7318.09	1.76	7448.91	2.22	6711.58	9.04	10.99
16	6147.00	1.64	6140.31	1.97	5775.19	6.44	6.32
17	6717.00	1.79	6717.00	2.13	6362.75	5.57	5.57
18	7019.27	1.48	7019.54	2.06	6463.32	8.60	8.61
19	8409.07	1.61	8433.12	1.98	7566.23	11.14	11.46
20	6529.58	1.50	6529.58	1.98	6529.58	0.00	0.00
Min:		1.45		1.91		0.00	0.00
Max:		1.97		2.22		13.12	13.37
Avg:		1.67		2.03		5.78	5.76

Table 5.2: Results for 20 Nodes

When we examine the results of the instances with 30 nodes in Table 5.3, we see that the average execution time of the L&F is still very low (using the Decision Tree, it is 3.95 seconds; using kNN, it is 4.65 seconds). The average gaps are 5.04% and 5.42% for Decision Tree and kNN versions, respectively, which are slightly lower than those values for the 20 node instances. Similarly, in Table 5.4 in which the results of the instances with 40 nodes are shown, the average execution time for the Decision Tree version of L&F is 5.23 seconds, while it is 6.40 seconds for the kNN version. The average gap values increase by around 1%, along with the increase in the percentage optimality gaps of the integer solutions obtained from CG, that are used to train the classification algorithms.

Instance	**Decision Tree Solution**	**Time (s)**	**kNN Solution**	**Time (s)**	**LP Relaxation Solution**	**Decision Tree GAP (%)**	**kNN GAP (%)**
1	8728.81	3.96	8732.50	4.60	8426.53	3.59	3.63
2	8393.88	4.04	8481.64	5.13	8178.18	2.64	3.71
3	9382.27	3.91	9416.26	4.53	8884.83	5.60	5.98
4	9920.40	3.99	9936.79	4.80	9068.23	9.40	9.58
5	9357.51	4.17	9481.22	4.87	8972.68	4.29	5.67
6	8650.90	3.87	8646.01	4.95	8421.15	2.73	2.67
7	9166.98	3.97	9166.98	4.84	8877.82	3.26	3.26
8	9745.99	3.94	9745.99	4.85	9161.35	6.38	6.38
9	9702.83	4.08	9526.65	4.62	8830.52	9.88	7.88
10	8951.49	3.90	9213.71	4.69	8661.48	3.35	6.38
11	9548.58	3.92	9548.58	4.50	8553.15	11.64	11.64
12	7832.98	3.86	7832.98	4.36	7391.22	5.98	5.98
13	8901.86	4.07	8890.67	4.60	8608.82	3.40	3.27
14	10027.19	3.86	10131.85	4.65	9271.73	8.15	9.28
15	9950.94	3.92	10034.50	4.52	9609.29	3.56	4.42
16	8171.58	3.82	8171.58	4.58	7977.91	2.43	2.43
17	7821.38	3.92	7885.10	4.35	7420.97	5.40	6.25
18	8634.67	3.83	8634.67	4.63	8290.79	4.15	4.15
19	8921.25	4.03	8993.07	4.47	8645.94	3.18	4.01
20	8262.94	4.00	8272.25	4.47	8125.88	1.69	1.80
Min:		3.82		4.35		1.69	1.80
Max:		4.17		5.13		11.64	11.64
Avg:		3.95		4.65		5.04	5.42

Table 5.3: Results for 30 Nodes

Next we test L&F on the instances obtained from "kroA100.tsp" benchmark TSP instance and report the results in Table 5.5. Importantly, to solve these instances

Instance	**Decision Tree Solution**	**Time (s)**	**kNN Solution**	**Time (s)**	**LP Relaxation Solution**	**Decision Tree GAP (%)**	**kNN GAP (%)**
1	10341.69	5.20	10402.52	6.55	9923.85	4.21	4.82
2	10619.56	5.14	10681.03	6.29	10047.38	5.69	6.31
3	11411.97	5.28	11411.97	6.65	10778.16	5.88	5.88
4	11007.66	4.99	11007.66	6.44	10155.77	8.39	8.39
5	10527.50	5.17	10511.15	6.46	10019.35	5.07	4.91
6	10850.51	5.06	11100.78	6.12	10484.35	3.49	5.88
7	9968.22	5.16	9876.44	6.59	9225.15	8.05	7.06
8	10186.18	5.60	10156.79	6.30	9089.24	12.07	11.75
9	9975.37	5.10	10309.88	6.46	9272.06	7.59	11.19
10	10537.71	5.23	10573.52	6.43	10147.30	3.85	4.20
11	10950.36	4.94	10962.03	6.44	10260.86	6.72	6.83
12	10645.97	5.15	10864.64	6.44	9702.62	9.72	11.98
13	11383.12	5.22	11430.49	6.43	10965.81	3.81	4.24
14	9164.47	5.30	9173.45	6.24	8714.25	5.17	5.27
15	10974.73	5.27	11014.47	6.31	9774.68	12.28	12.68
16	10851.90	5.16	10851.90	6.33	10041.09	8.07	8.07
17	10427.37	5.27	10484.45	6.43	9925.46	5.06	5.63
18	11069.05	5.38	11069.05	6.36	10197.18	8.55	8.55
19	9516.11	5.57	9516.11	6.31	8916.56	6.72	6.72
20	10053.25	5.46	10053.25	6.41	9750.66	3.10	3.10
Min:		4.94		6.12		3.10	3.10
Max:		5.60		6.65		12.28	12.68
Avg:		5.23		6.40		6.67	7.17

Table 5.4: Results for 40 Nodes

with L&F, we use the learning outputs used for instances in Table 5.4. As seen in Table 5.5, the average optimality gaps are less than 5%. Interestingly, L&F can still find high quality solutions with the learning outputs generated using the instances coming from a different network topology. In addition, L&F runs in a few seconds, while the average execution time of CG is 863.51s. Hence, we see that obtaining integer solutions from the CG is not a viable method in a dynamic environment because of the excessive run time.

Apart from these experiments, we compare L&F with two rule-based heuristic approaches to observe the advantages of learning. These algorithms also utilize an initial TSP tour excluding the RSs, and decide on when to recharge at which RS. The parameters used in the decision rules of the two heuristics are derived from the most influential features of the classification algorithms. In the first algorithm

Instance	**Decision Tree Solution**	**Time (s)**	**kNN Solution**	**Time (s)**	**LP Relaxation Solution**	**Time (s)**	**Decision Tree GAP (%)**	**kNN GAP (%)**
1	7854.57	3.60	7854.57	4.99	7607.15	963.91	3.25	3.25
2	8032.88	5.55	8188.60	4.62	7703.13	1036.67	4.28	6.30
3	7682.10	3.58	7682.10	4.27	7396.83	570.91	3.86	3.86
4	7914.71	3.67	7914.71	4.36	7421.44	931.88	6.65	6.65
5	8069.92	3.46	8108.87	4.13	7709.72	706.50	4.67	5.18
6	8202.41	3.24	8204.40	3.90	8010.76	779.18	2.39	2.42
7	7930.33	3.18	8071.04	4.01	7512.94	785.18	5.56	7.43
8	7984.66	3.14	7910.9	4.21	7411.9	585.30	7.73	6.73
9	8110.56	3.12	8110.56	4.42	7850.18	1044.71	3.32	3.32
10	7410.12	3.17	7326.26	4.51	7024.62	658.61	5.49	4.29
11	7671.26	3.21	7617.61	4.03	7379.13	768.55	3.96	3.23
12	8091.78	3.17	8100.59	4.06	7929.77	818.31	2.04	2.15
13	7496.17	3.23	7507.35	4.11	7158.65	726.11	4.71	4.87
14	6952.72	3.18	6943.05	3.81	6797.03	1260.66	2.29	2.15
15	7636.20	3.13	7636.20	3.97	7381.16	1088.11	3.46	3.46
16	8002.84	3.44	7891.47	4.06	7522.98	802.06	6.38	4.90
17	7465.97	3.52	7439.54	4.27	6834.42	1125.58	9.24	8.85
18	7670.84	3.57	7788.55	3.92	7552.96	925.78	1.56	3.12
19	8427.64	4.08	8344.93	4.09	7861.88	1058.10	7.20	6.14
20	8135.38	3.33	8135.38	4.30	7557.32	634.10	7.65	7.65
Min:		3.12		3.81		570.91	1.56	2.15
Max:		5.55		4.99		1260.66	9.24	8.85
Avg:		3.48		4.20		863.51	4.78	4.79

Table 5.5: L&F Results for “kroA100" Instances

called the Insufficient Battery Heuristic (IBH), the drone visits an RS when the current battery charge is not enough to process the next node in the giant walk. The RS that it visits is chosen according to the parameter "Extra Distance" defined in Table 4.1. If the RS chosen according to "Extra Distance" causes infeasibility, the drone visits the nearest RS to it as in L&F. The pseudo-code of IBH is presented in Algorithm 2 in Appendix 7.2. The results associated with IBH are reported in the column "Insufficient Battery Heuristic" in Table 5.6. The second algorithm, whose results are reported in the "Threshold Heuristic" column of Table 5.6 is named the Threshold Heuristic (TH). In this method, first the "Battery Level" and "Extra Distance" feature values are found every time the drone visits an RS in the training set solutions. The corresponding averages of these values are set as two thresholds. When the giant tour is processed in the algorithm, if the current battery level and

the extra distance that the drone will take by visiting an RS are less than the corresponding threshold values at the same time, the drone visits an RS. The RS that the drone visits is chosen in the same manner as in IBH. The pseudo-code of TH is given in Algorithm 3 in Appendix 7.3. In Table 5.6 in which the results of the two heuristics for 30 node instances are shown, the average gap of the Insufficient Battery Heuristic is 10.36 percent. Similarly, the average gap of the Threshold Heuristic is 11.87 percent. In comparison, the average gap of L&F for instances with 30 nodes is around 5 percent. The large difference between the performance of L&F and the rule-based heuristics show the importance of using the classification algorithms. Also, it can be observed one more time that the learning accuracy does not depend on only the most important features.

Instance	**Insufficient Battery Heuristic**	**Simple Threshold Heuristic**	**LP Relaxation Solutions**	**Insufficient Battery Heuristic GAP (%)**	**Simple Threshold Heuristic GAP (%)**
1	9404.42	9288.72	8426.53	11.60	10.23
2	8589.37	8634.92	8178.18	5.03	5.58
3	10099.30	10099.30	8884.83	13.67	13.67
4	10316.11	10335.74	9068.23	13.76	13.98
5	10025.14	10307.53	8972.68	11.73	14.88
6	9716.17	9943.22	8421.15	15.38	18.07
7	9442.13	9441.98	8877.82	6.36	6.35
8	10147.85	10309.65	9161.35	10.77	12.53
9	10269.66	10179.28	8830.52	16.30	15.27
10	9506.89	10025.24	8661.48	9.76	15.75
11	9986.34	10308.05	8553.15	16.76	20.52
12	8481.70	8477.21	7391.22	14.75	14.69
13	9123.64	9594.30	8608.82	5.98	11.45
14	10683.53	10486.34	9271.73	15.23	13.10
15	10030.22	11162.75	9609.29	4.38	16.17
16	8318.88	8274.59	7977.91	4.27	3.72
17	7982.48	7982.48	7420.97	7.57	7.57
18	9033.27	9033.27	8290.79	8.96	8.96
19	9205.30	9205.30	8645.94	6.47	6.47
20	8812.02	8812.02	8125.88	8.44	8.44
			Min:	4.27	3.72
			Max:	16.76	20.52
			Avg:	10.36	11.87

Table 5.6: Results of Insufficient Battery and Threshold Heuristics for 30 Nodes

The learning procedure, which is a pre-processing step for the L&F algorithm,

takes considerably longer time as the size of the instances grows. However, this can be overcome by using the classification outputs obtained from a smaller-sized set of test instances to solve larger instances. To test the viability of this approach, we utilize the classification outputs from the 20 node instances to solve the instances having 30 and 40 nodes. The results are reported in Table 5.7. When learning from the 20 node instances, the average gap for the instances with 30 nodes is almost the same as those reported in Table 5.3. Although the execution time of L&F increases compared with the results in Table 5.3, it can still be said that the L&F runs very fast because the increase in the execution time is about 2.5 seconds. Similarly, the results of the instances with 40 nodes, where L&F makes its decisions based on the learning outputs from 20 node instances, show that we can obtain almost the same quality results as in Table 5.4. Another observation here is that the execution time does not increase for instances with 40 nodes compared with the results in Table 5.4. As a result, we conclude that learning from instances of smaller size can still lead to high quality solutions in short run times.

	30 NODES							40 NODES						
Instance	**Decision Tree Solution**	**Time (s)**	**kNN Solution**	**Time (s)**	**LP Relaxation Solution**	**Decision Tree GAP (%)**	**kNN GAP (%)**	**Decision Tree Solution**	**Time (s)**	**kNN Solution**	**Time (s)**	**LP Relaxation Solution**	**Decision Tree GAP (%)**	**kNN GAP (%)**
1	8732.50	6.61	8796.67	7.70	8426.53	3.63	4.39	10402.52	5.27	10341.69	6.77	9923.85	4.82	4.21
2	8543.22	6.09	8545.38	7.59	8178.18	4.46	4.49	10514.29	5.60	10835.28	6.49	10047.38	4.65	7.84
3	9374.36	6.15	9416.26	7.28	8884.83	5.51	5.98	11411.97	5.87	11411.97	6.60	10778.16	5.88	5.88
4	9936.79	6.52	9999.88	7.03	9068.23	9.58	10.27	10995.43	5.39	10995.43	6.39	10155.77	8.27	8.27
5	9442.14	6.57	9667.28	7.36	8972.68	5.23	7.74	10630.99	5.61	10574.71	6.92	10019.35	6.10	5.54
6	8650.90	6.46	8650.90	7.35	8421.15	2.73	2.73	10982.09	5.06	10714.27	6.95	10484.35	4.75	2.19
7	9166.98	6.38	9166.98	7.34	8877.82	3.26	3.26	10010.96	5.22	10311.27	6.45	9225.15	8.52	11.77
8	9719.83	6.54	9745.99	7.56	9161.35	6.10	6.38	10107.39	5.85	10199.08	6.58	9089.24	11.20	12.21
9	9677.84	6.56	9702.83	7.20	8830.52	9.60	9.88	9997.83	5.56	10219.13	6.50	9272.06	7.83	10.21
10	9121.96	6.42	9383.28	7.16	8661.48	5.32	8.33	10573.52	5.24	10578.86	6.53	10147.30	4.20	4.25
11	9655.94	6.23	9660.34	7.33	8553.15	12.89	12.94	10910.05	5.59	10962.03	6.38	10260.86	6.33	6.83
12	7832.98	6.18	7833.63	7.14	7391.22	5.98	5.99	10877.60	5.45	10911.23	6.24	9702.62	12.11	12.46
13	8901.86	6.41	8852.64	7.35	8608.82	3.40	2.83	11434.24	5.16	11441.96	6.60	10965.81	4.27	4.34
14	10349.47	6.43	10436.16	7.38	9271.73	11.62	12.56	9153.59	4.92	9133.73	6.52	8714.25	5.04	4.81
15	9950.94	6.12	10034.50	7.45	9609.29	3.56	4.42	11146.88	5.49	11202.39	6.65	9774.68	14.04	14.61
16	8173.40	6.44	8171.58	7.57	7977.91	2.45	2.43	10892.05	5.24	10851.90	6.59	10041.09	8.47	8.07
17	7821.38	6.34	7884.36	9.56	7420.97	5.40	6.24	10457.22	5.50	10722.72	6.88	9925.46	5.36	8.03
18	8634.67	6.32	8634.67	7.78	8290.79	4.15	4.15	10992.31	5.38	11069.05	6.50	10197.18	7.80	8.55
19	8921.25	6.63	8993.07	7.49	8645.94	3.18	4.01	9724.25	5.02	9516.11	6.25	8916.56	9.06	6.72
20	8272.25	6.51	8262.94	7.14	8125.88	1.80	1.69	10053.25	5.16	10053.25	6.37	9750.66	3.10	3.10
Min.:		**6.09**		**7.03**		**1.80**	**1.69**	**Min.:**	**4.92**		**6.24**		**3.10**	**2.19**
Max.:		**6.63**		**9.56**		**12.89**	**12.94**	**Max.:**	**5.87**		**6.95**		**14.04**	**14.61**
Avg.:		**6.40**		**7.49**		**5.49**	**6.04**	**Avg.:**	**5.38**		**6.56**		**7.09**	**7.49**

Table 5.7: Results with 20 Nodes Learning

Chapter 6

CONCLUSION

In our study we propose a learning based fast algorithm for the DRP that can be utilized in dynamic environments, as well as static ones, by reruns with updated inputs. Our algorithm benefits from machine learning outputs in making key decisions. The supervised machine learning algorithms are appropriate for the DRP because we can classify the target labels as visiting an RS or not visiting an RS. Because we do not have to keep the training data in the memory for solving new instances, the learning phase does not cause an increase in computation time each time the algorithm is rerun. The learning output mapping our features to predict the labels is enough to execute the further steps of our learning based algorithm, which we name Learn and Fly (L&F).

A critical issue in the learning phase is which features are used. We define 9 features according to the parameters playing a foremost role in the timing of recharging. Clearly, the training data set affects the quality of the resulting solutions. In our study, we apply a column generation procedure on a novel integer program with path-segment and cycle variables, so as to obtain integer feasible solutions for the DRP that are very close to the optimal solutions (within 2 percent) as the training set. L&F takes advantage of two supervised learning algorithms, as alternatives to each other, for classification.

We show that L&F provides near-optimal solutions in our computational tests with planar data having 20, 30 and 40 nodes, as well as other 40 node instances derived from a TSP benchmark instance. The successful performance of L&F is confirmed, even if the training solutions come from a smaller sized data set or instances having different topology. Benchmarking with two rule-based algorithms also verified the good performance of L&F. Since the L&F algorithm runs in a few seconds, it can be used in dynamic environments by rerunning the algorithm as the

input parameters of DRP change.

For future research, different approaches may be utilized for the learning phase to improve the optimality gaps further. The features that we proposed and their impact on the classification accuracy rate can provide hints for future algorithms. Also, it remains a challenge to apply the ideas presented in this study to a multi-drone routing problem that can be solved in dynamic environments.

BIBLIOGRAPHY

Agatz, N., Bouman, P., and Schmidt, M. (2018). Optimization approaches for the traveling salesman problem with drone. *Transportation Science*, 52(4):965–981.

Al-Sabban, W. H., Gonzalez, L. F., and Smith, R. N. (2013). Wind-energy based path planning for unmanned aerial vehicles using markov decision processes. In *2013 IEEE International Conference on Robotics and Automation*, pages 784–789. IEEE.

Alotaibi, K. A., Rosenberger, J. M., Mattingly, S. P., Punugu, R. K., and Visoldilokpun, S. (2018). Unmanned aerial vehicle routing in the presence of threats. *Computers & Industrial Engineering*, 115:190 – 205.

Applegate, D. L., Bixby, R. E., Chvátal, V., Cook, W., Espinoza, D. G., Goycoolea, M., and Helsgaun, K. (2009). Certification of an optimal tsp tour through 85,900 cities. *Operations Research Letters*, 37(1):11–15.

Arthur, D. and Vassilvitskii, S. (2007). k-means++: the advantages of careful seeding. In *SODA'07: proceedings of the eighteenth annual ACM-SIAM symposium on discrete algorithms. Society for Industrial and Applied Mathematics, Philadelphia, PA*, pages 1027–1035.

Aruna, R. (2013). Construction of decision tree : Attribute selection measures. *International Journal of Advancements in Research Technology*, 2(4):343–346.

Avellar, G. S. C., Pereira, G. A. S., Pimenta, L. C. A., and Iscold, P. (2015). Multi-uav routing for area coverage and remote sensing with minimum time. *Sensors*, 15(11):27783–27803.

Barrientos, A., Colorado, J., Cerro, J. d., Martinez, A., Rossi, C., Sanz, D., and Valente, J. (2011). Aerial remote sensing in agriculture: A practical approach to

area coverage and path planning for fleets of mini aerial robots. *Journal of Field Robotics*, 28(5):667–689.

Beasley, J. E. and Christofides, N. (1989). An algorithm for the resource constrained shortest path problem. *Networks*, 19(4):379–394.

Bouman, P., Agatz, N., and Schmidt, M. (2018). Dynamic programming approaches for the traveling salesman problem with drone. *Networks*, 72(4):528–542.

Boysen, N., Briskorn, D., Fedtke, S., and Schwerdfeger, S. (2018). Drone delivery from trucks: Drone scheduling for given truck routes. *Networks*, 72(4):506–527.

Breunig, U., Baldacci, R., Hartl, R., and Vidal, T. (2019). The electric two-echelon vehicle routing problem. *Computers & Operations Research*, 103:198 – 210.

Campbell, J. F., Corberan, A., Plana, I., and Sanchis, J. M. (2018). Drone arc routing problems. *Networks*, 72(4):543–559.

Carpaneto, G., Dell'Amico, M., and Toth, P. (1995). Exact solution of large-scale, asymmetric traveling salesman problems. *ACM Transactions on Mathematical Software (TOMS)*, 21(4):394–409.

Chang, Y. S. and Lee, H. J. (2018). Optimal delivery routing with wider drone-delivery areas along a shorter truck-route. *Expert Systems with Applications*, 104:307 – 317.

Cho, J., Lim, G., Biobaku, T., Kim, S., and Parsaei, H. (2015). Safety and security management with unmanned aerial vehicle (uav) in oil and gas industry. *Procedia Manufacturing*, 3:1343 – 1349. 6th International Conference on Applied Human Factors and Ergonomics (AHFE 2015) and the Affiliated Conferences, AHFE 2015.

Chow, J. Y. (2016). Dynamic uav-based traffic monitoring under uncertainty as a stochastic arc-inventory routing policy. *International Journal of Transportation Science and Technology*, 5(3):167–185.

Chowdhury, S., Emelogu, A., Marufuzzaman, M., Nurre, S. G., and Bian, L. (2017). Drones for disaster response and relief operations: A continuous approximation model. *International Journal of Production Economics*, 188:167 – 184.

Coelho, B. N., Coelho, V. N., Coelho, I. M., Ochi, L. S., K., R. H., Zuidema, D., Lima, M. S., and da Costa, A. R. (2017). A multi-objective green uav routing problem. *Computers & Operations Research*, 88:306 – 315.

Crainic, T. G., Gendreau, M., and Potvin, J.-Y. (2009). Intelligent freight-transportation systems: Assessment and the contribution of operations research. *Transportation Research Part C: Emerging Technologies*, 17(6):541–557.

Crevier, B., Cordeau, J. F., and Laporte, G. (2007). The multi-depot vehicle routing problem with inter-depot routes. *European Journal of Operational Research*, 176(2):756 – 773.

Cui, J.-H., Wei, R.-X., Liu, Z.-C., and Zhou, K. (2018). Uav motion strategies in uncertain dynamic environments: A path planning method based on q-learning strategy. *Applied Sciences*, 8(11):2169.

Dell'Amico, M., Montemanni, R., and Novellani, S. (2020). Matheuristic algorithms for the parallel drone scheduling traveling salesman problem. *Annals of Operations Research*, pages 1–16.

Desaulniers, G., Errico, F., Irnich, S., and Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, 64(6):1388–1405.

Dorigo, M., Maniezzo, V., and Colorni, 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.

Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S. (2016). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85.

Dorling, K., Heinrichs, J., Messier, G. G., and Magierowski, S. (2017). Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85.

Erdoğan, S. and Miller-Hooks, E. (2012). A green vehicle routing problem. *Transportation research part E: logistics and transportation review*, 48(1):100–114.

Ermağan, U., Yıldız, B., and Salman, S. (2020). Datasets for drone routing problem, v2. *Mendeley Data*.

Felipe, A., Ortuno, M. T., Righini, G., and Tirado, G. (2014). A heuristic approach for the green vehicle routing problem with multiple technologies and partial recharges. *Transportation Research Part E: Logistics and Transportation Review*, 71:111 – 128.

Focacci, F., Lodi, A., and Milano, M. (2002). A hybrid exact algorithm for the tsptw. *INFORMS Journal on Computing*, 14(4):403–417.

Froger, A., Mendoza, J. E., Jabali, O., and Laporte, G. (2019). Improved formulations and algorithmic components for the electric vehicle routing problem with nonlinear charging functions. *Computers & Operations Research*, 104:256 – 294.

Gebbers, R. and Adamchuk, V. I. (2010). Precision agriculture and food security. *Science*, 327(5967):828–831.

Goeke, D. and Schneider, M. (2015). Routing a mixed fleet of electric and conventional vehicles. *European Journal of Operational Research*, 245(1):81 – 99.

Grötschel, M. and Holland, O. (1991). Solution of large-scale symmetric travelling salesman problems. *Mathematical Programming*, 51(1-3):141–202.

Guerriero, F., Surace, R., Loscra, V., and Natalizio, E. (2014). A multi-objective approach for unmanned aerial vehicle routing problem with soft time windows constraints. *Applied Mathematical Modelling*, 38(3):839 – 852.

Guo, G., Wang, H., Bell, D., Bi, Y., and Greer, K. (2003). Knn model-based approach in classification. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 986–996. Springer.

Ha, Q. M., Deville, Y., Pham, Q. D., and Ha, M. H. (2018). On the min-cost traveling salesman problem with drone. *Transportation Research Part C: Emerging Technologies*, 86:597 – 621.

Hassanat, A. B., Abbadi, M. A., Altarawneh, G. A., and Alhasanat, A. A. (2014). Solving the problem of the k parameter in the knn classifier using an ensemble learning approach. *International Journal of Computer Science and Information Security*, 12(8):33–39.

Hiermann, G., Hartl, R. F., Puchinger, J., and Vidal, T. (2019). Routing a mix of conventional, plug-in hybrid, and electric vehicles. *European Journal of Operational Research*, 272(1):235 – 248.

Hiermann, G., Puchinger, J., Ropke, S., and Hartl, R. F. (2016). The electric fleet size and mix vehicle routing problem with time windows and recharging stations. *European Journal of Operational Research*, 252(3):995 – 1018.

Jeong, H. Y., Lee, S., and Song, B. D. (2019). Truck-drone hybrid delivery routing: Payload-energy dependency and no-fly zones. *International Journal of Production Economics*. forthcoming.

Kara, I., Kara, B. Y., and Yetis, M. K. (2007). Energy minimizing vehicle routing problem. In *International Conference on Combinatorial Optimization and Applications*, pages 62–71. Springer.

Karak, A. and Abdelghany, K. (2019). The hybrid vehicle-drone routing problem for pick-up and delivery services. *Transportation Research Part C: Emerging Technologies*, 102:427 – 449.

Keskin, M. and Catay, B. (2016). Partial recharge strategies for the electric vehicle routing problem with time windows. *Transportation Research Part C: Emerging Technologies*, 65:111 – 127.

Keskin, M., Laporte, G., and Catay, B. (2019). Electric vehicle routing problem with time-dependent waiting times at recharging stations. *Computers & Operations Research*, 107:77 – 94.

Kim, S. J., Lim, G. J., and Cho, J. (2018). Drone flight scheduling under uncertainty on battery duration and air temperature. *Computers & Industrial Engineering*, 117:291–302.

Kitjacharoenchai, P., Ventresca, M., Moshref-Javadi, M., Lee, S., Tanchoco, J. M., and Brunese, P. A. (2019). Multiple traveling salesman problem with drones: Mathematical model and heuristic approach. *Computers & Industrial Engineering*, 129:14 – 30.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada.

Kopfstedt, T., Mukai, M., Fujita, M., and Ament, C. (2008). Control of formations of uavs for surveillance and reconnaissance missions. *IFAC Proceedings Volumes*, 41(2):5161–5166.

Kotsiantis, S. B., Zaharakis, I., and Pintelas, P. (2007). Supervised machine learning: A review of classification techniques. *Emerging Artificial Intelligence Applications in Computer Engineering*, 160(1):3–24.

Krolak, P., Felts, W., and Marble, G. (1971). A man-machine approach toward solving the traveling salesman problem. *Communications of the ACM*, 14(5):327–334.

Laporte, G. (2010). A concise guide to the traveling salesman problem. *Journal of the Operational Research Society*, 61(1):35–40.

Lee, Y., Chiu, S. Y., and Ryan, J. (1996). A branch and cut algorithm for a steiner tree-star problem. *INFORMS Journal on Computing*, 8(3):194–201.

Levy, D., Sundar, K., and Rathinam, S. (2014). Heuristics for routing heterogeneous unmanned vehicles with fuel constraints. *Mathematical Problems in Engineering*, 2014.

Li, M., Zhen, L., Wang, S., Lv, W., and Qu, X. (2018). Unmanned aerial vehicle scheduling problem for traffic monitoring. *Computers & Industrial Engineering*, 122:15 – 23.

Liao, C.-S., Lu, S.-H., and Shen, Z.-J. (2016). The electric vehicle touring problem. *Transportation Research Part B: Methodological*, 86:163 – 180.

Liu, Y. (2019). An optimization-driven dynamic vehicle routing algorithm for on-demand meal delivery using drones. *Computers & Operations Research*, 111:1–20.

Luo, Z., Liu, Z., and Shi, J. (2017). A two-echelon cooperated routing problem for a ground vehicle and its carried unmanned aerial vehicle. *Sensors*, 17(5).

Macrina, G., Laporte, G., Guerriero, F., and Pugliese, L. D. P. (2019). An energy-efficient green-vehicle routing problem with mixed vehicle fleet, partial battery recharging and time windows. *European Journal of Operational Research*, 276(3):971–982.

Mbiadou, S., G., R., Deroussi, L., Feillet, D., Grangeon, N., and Quilliot, A. (2018). An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem. *Networks*, 72(4):459–474.

Modaresi, S., Sauré, D., and Vielma, J. P. (2020). Learning in combinatorial optimization: What and how to explore. *Operations Research*, to appear.

Murray, C. C. and Chu, A. G. (2015). The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86 – 109.

Nefeslioglu, H., Sezer, E., Gokceoglu, C., Bozkir, A., and Duman, T. (2010). Assessment of landslide susceptibility by decision trees in the metropolitan area of istanbul, turkey. *Mathematical Problems in Engineering*, 2010.

Nembrini, S., König, I. R., and Wright, M. N. (2018). The revival of the gini importance? *Bioinformatics*, 34(21):3711–3718.

Oruc, B. E. and Kara, B. Y. (2018). Post-disaster assessment routing problem. *Transportation Research Part B: Methodological*, 116:76 – 102.

Otto, A., Agatz, N., Campbell, J., Golden, B., and Pesch, E. (2018). Optimization approaches for civil applications of unmanned aerial vehicles (uavs) or aerial drones: A survey. *Networks*, 72(4):411–458.

Patidar, P., Dangra, J., and Rawar, M. (2015). Decision tree c4. 5 algorithm and its enhanced approach for educational data mining. *Engineering Universe for Scientific Research and Management*, 7(2):1–14.

Poikonen, S., Golden, B., and Wasil, E. A. (2019). A branch-and-bound approach to the traveling salesman problem with a drone. *INFORMS Journal on Computing*, 31(2):335–346.

Rabta, B., Wankmuller, C., and Reiner, G. (2018). A drone fleet model for last-mile distribution in disaster relief operations. *International Journal of Disaster Risk Reduction*, 28:107 – 112.

Ramirez-Atencia, C., Bello-Orgaz, G., R-Moreno, M. D., and Camacho, D. (2017). Solving complex multi-uav mission planning problems using multi-objective genetic algorithms. *Soft Computing*, 21(17):4883–4900.

Ronaghan, S. (2018). The mathematics of decision trees, random forest and feature importance in scikit-learn and spark. *Medium*, May, 11.

Sacramento, D., Pisinger, D., and 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.

Sagoolmuang, A. and Sinapiromsaran, K. (2020). Decision tree algorithm with class overlapping-balancing entropy for class imbalanced problem. *International Journal of Machine Learning and Computing*, 10(3):444–451.

Sassi, O. and Oulamara, A. (2017). Electric vehicle scheduling and optimal charging problem: complexity, exact and heuristic approaches. *International Journal of Production Research*, 55(2):519–535.

Savuran, H. and Karakaya, M. (2016). Efficient route planning for an unmanned air vehicle deployed on a moving carrier. *Soft Computing*, 20(7):2905–2920.

Schneider, M., Stenger, A., and Goeke, D. (2014). The electric vehicle-routing problem with time windows and recharging stations. *Transportation Science*, 48(4):500 – 520.

Shaheen, M., Zafar, T., and Ali Khan, S. (2020). Decision tree classification: Ranking journals using igidi. *Journal of Information Science*, 46(3):325–339.

Shaikhina, T., Lowe, D., Daga, S., Briggs, D., Higgins, R., and Khovanova, N. (2019). Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation. *Biomedical Signal Processing and Control*, 52:456–462.

Shaw, P. (1998). Using constraint programming and local search methods to solve vehicle routing problems. In *International conference on principles and practice of constraint programming*, pages 417–431. Springer.

Song, B. D., Park, K., and Kim, J. (2018). Persistent uav delivery logistics: Milp formulation and efficient heuristic. *Computers & Industrial Engineering*, 120:418 – 428.

Song, Y., Huang, J., Zhou, D., Zha, H., and Giles, C. L. (2007). Iknn: Informative k-nearest neighbor pattern classification. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 248–264. Springer.

Soysal, M., Çimen, M., and Belbağ, S. (2020). Pickup and delivery with electric vehicles under stochastic battery depletion. *Computers & Industrial Engineering*, page 106512.

Stump, E. and Michael, N. (2011). Multi-robot persistent surveillance planning as a vehicle routing problem. *In proceedings of the IEEE Conference on Automation Science and Engineering (CASE)*, pages 569 – 575.

Sundar, K. and Rathinam, S. (2014). Algorithms for routing an unmanned aerial vehicle in the presence of refueling depots. *IEEE Transactions on Automation Science and Engineering*, 11(1):287 – 294.

Swain, K. C., Jayasuriya, H. P., and Salokhe, V. M. (2007). Suitability of low-altitude remote sensing images for estimating nitrogen treatment variations in rice cropping for precision agriculture adoption. *Journal of Applied Remote Sensing*, 1(1):013547.

Taş, D. (2020). Electric vehicle routing with flexible time windows: a column generation solution approach. *Transportation Letters*, pages 1–7.

Thibbotuwawa, A., Bocewicz, G., Radzki, G., Nielsen, P., and Banaszak, Z. (2020). Uav mission planning resistant to weather uncertainty. *Sensors*, 20(2):515.

Ulmer, M. W. and Thomas, B. W. (2018). Same-day delivery with heterogeneous fleets of drones and vehicles. *Networks*, 72(4):475–505.

Wang, Z. and Sheu, J.-B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 122:350 – 364.

Wen, M., Linde, E., Ropke, S., Mirchandani, P., and Larsen, A. (2016). An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. *Computers & Operations Research*, 76:73 – 83.

Yıldız, B., Arslan, O., and Karaşan, O. E. (2016). A branch and price approach for routing and refueling station location model. *European Journal of Operational Research*, 248(3):815–826.

Yıldız, B. and Karaşan, O. E. (2017). Regenerator location problem in flexible optical networks. *Operations Research*, 65(3):595–620.

Yildiz, B. and Savelsbergh, M. (2019). Provably high-quality solutions for the meal delivery routing problem. *Transportation Science*, 53(5):1372–1388.

Zhen, L., Li, M., Laporte, G., and Wang, W. (2019). A vehicle routing problem arising in unmanned aerial monitoring. *Computers & Operations Research*, 105:1 – 11.

Chapter 7

APPENDIX

7.1 Formulations of the Pricing Problems

With the parameters and the decision variables summarized in Table 7.1, we present the formulations of pricing problems.

Sets	**Description**
N_{od}	$N \cup \{o, d\}$ for all $o \in R, d \in R \setminus \{o\}$
N_o	N $\cup$ $\{o\}$ for all $o \in R$
A_{od}	Set of arcs in A for all $o \in R, d \in R \setminus \{o\}$
A_o	Set of arcs in A for all $o \in R$
$\bar{t}_{ij}/\hat{t}_{ij}$	Cost of arc (i, j)
t_{ij}	Resource usage of arc (i, j)
τ	Total flight time of the drone with fully charged batteries.
Decision Variables	**Description**
u_{ij}	Binary variable that indicates whether arc (i, j) is used.
v_i	Binary variable that indicates whether node i is in the path/cycle

Table 7.1: Nomenclature

Pricing Problem for the Path Variables: To find the simple paths with the negative reduced costs, for each pair $(o, d) \in R \times R$ where $o \neq d$ we solve the pricing problem formulated as follows:

$$\min \sum_{(i,j)\in A_{od}} \bar{t}_{ij} u_{ij} \tag{7.1}$$

s.t.

$$\sum_{(ij)\in A_{od}} u_{ij} - \sum_{(ji)\in A_{od}} u_{ji} = \begin{cases} 1 & \text{if } i = o \\ -1 & \text{if } i = d \\ 0 & \text{ow.} \end{cases} \qquad \forall i \in N_{od} \tag{7.2}$$

$$\sum_{(ij)\in A_{od}} u_{ij} = v_i \qquad \forall i \in N_{od} \tag{7.3}$$

$$\sum_{(ij)\in A_{od}} u_{ij} = \sum_{i\in N_{od}} v_i - 1 \tag{7.4}$$

$$\sum_{(ij)\in A_{od}} t_{ij} u_{ij} \leq \tau \tag{7.5}$$

$$\sum_{(ij)\in S(A_{od})} u_{ij} \leq \sum_{i\in S\setminus\{k\}} v_i \qquad \forall S \subset N_{od}, k \in S \tag{7.6}$$

$$u_{ij} \in \{0,1\} \qquad \forall (ij) \in A_{od} \tag{7.7}$$

$$v_i \in \{0,1\} \qquad \forall i \in N_{od} \tag{7.8}$$

Pricing Problem for the Cycle Variables: Similarly, to find the simple cycles with the negative reduced costs, for each station $o \in R$ we solve the pricing problem formulated as follows:

$$\min \sum_{(i,j)\in A_o} \hat{t}_{ij} u_{ij} \tag{7.9}$$

s.t.

$$\sum_{(ij)\in A_o} u_{ij} - \sum_{(ji)\in A_o} u_{ji} = \begin{cases} 1 & \text{if } i = o \\ -1 & \text{if } i = \hat{o} \\ 0 & \text{ow.} \end{cases} \qquad \forall i \in N_o \tag{7.10}$$

$$\sum_{(ij)\in A_o} u_{ij} = v_i \qquad \forall i \in N_o \tag{7.11}$$

$$\sum_{(ij)\in A_o} u_{ij} = \sum_{i\in N_o} v_i - 1 \tag{7.12}$$

$$\sum_{(ij)\in A_o} t_{ij} u_{ij} \leq \tau \tag{7.13}$$

$$\sum_{(ij)\in S(A_o)} u_{ij} \leq \sum_{i\in S\setminus\{k\}} v_i \qquad \forall S \subset N_o, k \in S \tag{7.14}$$

$$u_{ij} \in \{0,1\} \qquad \forall (ij) \in A_o \tag{7.15}$$

$$v_i \in \{0,1\} \qquad \forall i \in N_o \tag{7.16}$$

The objective functions (7.1) and (7.9) minimize the total arc costs of the path and the cycle, respectively. If a node is the origin or destination point in the path/ cycle, it should have degree one. Otherwise, its degree should be two. This is forced by the flow balance constraints (7.2) and (7.10). Constraints (7.3), (7.4) for the paths and (7.11), (7.12) for the cycles ensure that the solution is a simple path. Constraints (7.6) and (7.14) eliminate the sub-tours formed by arcs. Constraints (7.5)/(7.13) prevent the paths/cycles whose costs are larger than the battery limit from adding to the restricted master problem.

Branch and Cut Algorithm for the Pricing Problem: We use the branch and cut algorithm to solve the pricing problems because they may be comprised of exponential numbers of subtour elimination constraints. In the branch and cut tree, the LP relaxations of pricing problems are solved at each node. If the solution that is found is integer and it is not feasible at any node, i.e. there are sub-tours, we implement the lazy constraints call-back to eliminate the sub-tours. Otherwise, we branch. In the algorithm, the branching strategies of CPLEX are used to explore the branch and cut tree.

7.2 Insufficient Battery Heuristic

Algorithm 2 Pseudo-code of the Insufficient Battery Algorithm

Input: V: Set of nodes, N: Set of nodes that should be processed, R: Set of RSs, τ: Battery Limit, giant walks

Output: A feasible route in which all nodes in N are processed

Begin
$k \leftarrow 1$ to 2 *(the number of the giant walks)*
$\tau^- = \tau$
for each giant walk **do**
    Find the nearest RS to the first node of the giant walk, and start routing
    while the number of nodes processed $< |N|$ **do**
        Calculate the values of the parameters for the current node
        $RS_e \leftarrow$ RS found via the feature "Extra Distance" for the current node
        $RS_n \leftarrow$ the RS found via the feature "Extra Distance" for the next node in the giant walk
        $d_n \leftarrow$ distance to the next node in the giant walk
        $d_{rs} \leftarrow$ distance between the next node in the giant walk and RS_n
        if $d_n + d_{rs} \leq \tau^-$ **then** *(feasibility check)*
            the current node $\leftarrow$ the next node in the giant walks
            $\tau^- = \tau^- - d_n$
        else
            the current node $\leftarrow RS_e$
            $\tau^- = \tau$
        end if
    end while
    $GiantSol^k \leftarrow$ solution
    Define each group of the nodes from a RS to another RS in the $GiantSol^k$ as a path
    Start improvement steps
    for each path in $GiantSol^k$ **do**
        if there is only one node processed in the path **then**
            if it is feasible and provides an improvement **then**
                Add the node the previous path or the next path in $GiantSol^k$ whichever provides the best improvement
                $GiantSol^k \leftarrow$ new solution
            end if
        end if
    end for
    $k \leftarrow k + 1$
end for
$BestSolution \leftarrow BestGiantSol^k$
End

7.3 The Threshold Heuristic

Algorithm 3 Pseudo-code of the Threshold Heuristic

Input: V: Set of nodes, N: Set of nodes that should be processed, R: Set of RSs, τ: Battery Limit, giant walks, the threshold parameters (B_{TH}: Battery Level and D_{TH}:Extra Distance)

Output: A feasible route in which all nodes in N are processed

Begin
$k \leftarrow 1$ to 2 *(the number of the giant walks)*
$\tau^- = \tau$
for each giant walk **do**
    Find the nearest RS to the first node of the giant walk, and start routing
    while the number of nodes processed $< |N|$ **do**
        Calculate the values of the parameters for the current node
        $RS_e \leftarrow$ RS found via feature "Extra Distance"
        $RS_c \leftarrow$ the nearest RS to the current node
        $RS_n \leftarrow$ the nearest RS to the next node in the giant walk
        $d_e \leftarrow$ the extra distance
        $d_n \leftarrow$ distance to the next node in the giant walk
        $d_{rs} \leftarrow$ distance between the next node in the giant walk and RS_n
        Decide whether the drone should visit an RS according to the threshold parameters
        if $\tau^- \leq B_{TH}$ and $d_e \leq D_{TH}$ **then**
            The drone should visit an RS
            if distance to $RS_e \leq \tau^-$ **then** *(feasibility check)*
                the current node $\leftarrow RS_e$
                $\tau^- = \tau$
            else
                the current node $\leftarrow RS_c$
                $\tau^- = \tau$
            end if
        else
            if $d_n + d_{rs} \leq \tau^-$ **then** *(feasibility check)*
                the current node $\leftarrow$ the next node in the giant walk
                $\tau^- = \tau^- - d_n$
            else
                if distance to $RS_e \leq \tau^-$ **then** *(feasibility check)*
                    the current node $\leftarrow RS_e$
                    $\tau^- = \tau$
                else
                    current node $\leftarrow RS_c$
                    $\tau^- = \tau$
                end if
            end if
        end if
    end while

$GiantSol^k \leftarrow$ solution
Define each group of the nodes from a RS to another RS in the $GiantSol^k$ as a path
Start improvement steps
for each path in $GiantSol^k$ **do**
    if there is only one node processed in the path **then**
        if it is feasible and provides an improvement **then**
            Add the node the previous path or the next path in $GiantSol^k$ whichever provides the best improvement
            $GiantSol^k \leftarrow$ new solution
        end if
    end if
end for
$k \leftarrow k + 1$
end for
$BestSolution \leftarrow BestGiantSol^k$
End