A heterogeneous vehicle routing problem with drones and multiple depots

Panagiotis Zachos

June 2024

1 INTRODUCTION

2 Related Work

Introductory statement on why routing problems (TSP, VRP) are fundamental in the field of operations research and logistics, why they receive much attention from the academic community, in what practical applications are they used in

2.1 The Traveling Salesman Problem with UAVs

The Traveling Salesman Problem (TSP) is a classic, widely researched optimization problem where the objective is to determine the shortest possible route that allows a salesman, starting from an origin city, to visit a set of cities exactly once and then end the tour by returning to the origin city. The TSP is known for its computational complexity, being an NP-hard problem, which means that the time required to solve it increases exponentially with the number of cities. (**Make the point that a truck with unlimited capacity can take on the role of the salesman (refer to Dell'Amico 2021)**) Murray and Chu were the first researchers who formalized and extensively studied the concept of UAVs working alongside trucks in routing problems in 2015. In their seminal paper **{ADD REFERENCE}** the authors introduced the Flying Sidekick Traveling Salesman Problem (FSTSP) and the Parallel Drone Scheduling Traveling Salesman Problem (PDSTSP). The two problems are similar in their use of a single truck alongside one or more UAVs but there are key differences between the two.

2.1.1 The Flying Sidekick Traveling Salesman Problem

The FSTSP involves a truck which takes on the role of the salesman, and is equipped with a single UAV (drone) which the driver can utilize by dispatching it to serve drone-eligible customers. The objective is to minimize the total time in which all customers are serviced. {**This is general and concerns every problem so it could be placed in the introduction (2.1)**{The underlying principle is that by making use of a lower-cost vehicle which is also not bound by road conditions (drone), the total time of the operation can be reduced substantially, leading to increased efficiency and thus profitability for the distributor but also increased customer satisfaction and reduced environmental footprint. By adding a drone to the operation, the distance that the truck needs to travel is reduced, leading to overall cost reduction of the operation (fuel consumption, working hours) while also limiting CO2 emissions.}} In the FSTSP, a truck and a drone are called to begin from a depot, although not necessarily in tandem, visit a set of customers exactly once, and then end their tour by returning to the depot. The driver is considered to load the UAV with the parcel of the intended customer inbetween traveling i.e when the truck has stopped at a customer location to deliver a parcel. Then, the driver dispatches the UAV and immediately continues with the truck onto the next customer(s), thus creating a time window in which both the truck and UAV are traveling simultaneously. After the UAV has delivered its parcel, it must return to the truck to be retrieved by the driver, or if flight endurance limitations allow, it can return to the depot, thus ending its tour. The retrieval of the UAV by the truck must be performed at a customer location assigned to the truck, which must be different from the launch location i.e the truck cannot remain stationary after launching the drone. The vehicle which arrives first at the determined retrieval node has to wait for the other vehicle, therefore synchronization between the truck and drone is required. Furthermore, the drone is associated with a launch setup and a retrieval time cost which must be considered in the synchronization and a maximum flight endurance value (battery capacity) which must be respected when assigning customers to the drone.

The authors used MILP to model the problem and noted that, deriving from the NP-hard nature of the problem, MILP solvers in certain cases required several hours even for instances with as low as 10 customers. Consequently, they proposed a *route and re-assign* heuristic. The heuristic begins by solving a TSP that assigns the truck to visit all customers. This TSP subproblem was solved using various techniques, namely *Integer Programming* (IP), and the *Clarke-Wright Savings*, *Nearest Neighbour*, *Sweep* heuristics. Once the truck route is established, the heuristic attempts to improve the solution by re-assigning customers to the drone. The algorithm iterates through each pair of consecutive stops in the truck route, and for each pair, it considers the possibility of sending the drone from the first stop to the second stop. It calculates the savings in travel time that would result from making this change, while considering the drone's and truck's travel time as well as the time required to retrieve the drone. If the resulting savings are greater than zero, the customer at the second stop is reassigned to the drone and the truck route is updated accordingly. The heuristic continues to iterate through all pairs of stops in the truck route, re-assigning customers to the drone whenever possible, until no further improvements can be made.

The authors' findings show that this *route and re-assign* heuristic provides a framework for solving the FSTSP in a reasonable amount of time, while still achieving good solutions. Especially in the version where the initial truck route is solved with Integer Programming (IP), the proposed heuristic, on average, outperformed the MILP solver. Based on these results, the authors highlight the impact of the initial TSP tour towards the final solution and how effective TSP solution approaches show promise in positively impacting the performance of their algorithm. Regarding the

remaining approaches, the use of the *Clarke-Wright Savings* heuristic yielded good results with minimal computation time whereas the use of the *Nearest Neighbour* and *Sweep* heuristics did not provide competitive results.

Ha et. al (2015) researched the FSTSP, although under a different name; the "Traveling Salesman Problem with Drone". They proposed two heuristics using two opposite concepts; (i) *cluster first - route second* and (ii) *route first - cluster second*. The clustering step consists of determining the set of drone routes that should be in the final solution. To solve this, the authors propose a Mixed Integer Linear Programming (MILP) model where the objective function is to maximize the profit generated by selecting specific drone routes. In the first approach, the MILP model is initially solved, to identify the optimal set of drone routes. Once those are determined, the heuristic then constructs the truck's route, using the Concorde TSP solver, by integrating the drone's routes with the remaining customer locations, producing the final solution. In the second *route first - cluster second* approach, the truck route is first solved (Concorde), visiting all customers. The truck's route is then used as the basis to identify potential drone routes. The MILP model used in the clustering step is adapted to include constraints that ensure that any drone route selected is part of the truck's predetermined tour and that the nodes that form the drone *sortie* are arranged in the same order as they appear in the truck's route. **Results** Experiments compared the two heuristics to the FSTSP model and the standard TSP without drones, using 2 newly generated datasets with 10 and 100 customers. The results show that the proposed heuristics are competitive compared to the FSTSP model, especially as customer size increases and indicate that drones servicing more distant customers from the TSP route is more beneficial.

Freitas & Penna (2018) **{Add reference}** extended the FSTSP research and employed a hybrid heuristic termed *Randomized Variable Neighborhood Descent Framework* (RVNDF) to solve the problem. This heuristic makes use of a three-step approach. Similar to the previous heuristic by Murray & Chu, the RVNDF begins by generating the optimal solution for the TSP where all of the customers are serviced by the truck only. To achieve this, they use the widely known in the field Concorde TSP solver **{ADD REFERENCE}**. The second phase consists of creating an initial solution which makes use of the drone, by assigning it to random drone-eligible customers while respecting the drone's flight endurance. Finally, in the third step, the *Randomized Variable Neighborhoods Descent* (RVND) algorithm is called, which is the core of the heuristic and as the name implies, a randomized variant of the VND **{ADD REFERENCE}** algorithm, where the order of the neighborhood visits is not predetermined. RVND aims to improve the initial solution by systematically exploring different neighboring solutions. It does this by applying five different "neighborhoods" or local search operations, each designed to modify the current solution in a specific way. These neighborhoods consist of the following 5 operations: (i) *Reinsertion*: Moving a truck customer to a different position on its route, (ii) *Exchange*: Swap the positions of two truck customers, (iii) *Exchange(2,1)*: Swap the positions of two consecutive customers and an other one, (iv) *2-Opt*: Remove two non-adjacent arcs from the truck's route and insert two new arcs such as to create a new route, (v) *RelocateCustomer*: Transfer a customer from the truck's route to the drone's route. The RVNDF evaluates all possible moves within each neighborhood, ensuring that the drone's flight range and customer accessibility constraints are respected. After each operation, the resulting solution is either accepted and used to update the current solution if the overall delivery time improves, or discarded otherwise. **Report results or not?** {To evaluate their algorithm's quality, they solved 11 instances ranging from 50 to 99 customers, running the RVNDF ten times for each one. The authors then compared the FSTSP solutions to the TSP solution where only a truck is used. Results varied, improving by as high as 19.28% on a certain instance, while not being able to improve in the case where the distances between customers exceed the drone's flight endurance.}

Dell'Amico et al. (2019) **{ADD REF}** proposed a modified formulation of Murray & Chu's FSTSP, aiming to improve it and achieve a more realistic interpretation of the problem. They considered two different versions, one where the drone is allowed to land and wait at customers to conserve battery and another where waiting is only allowed in flight. The authors also introduce a three-indexed formulation termed *DMN* and a two-indexed formulation *DMN2*. The objective function aims to minimize the overall completion time, accounting for the truck's travel time, drone launch and return times, and waiting time at customers (depending on the version).

The *DMN* formulation leverages three-indexed boolean variables to represent *sorties*; flights undertaken by the drone. This formulation includes various improvements which are focused on eliminating infeasible drone sorties. To achieve this, a new objective function and a set of inequalities are presented. The two-indexed formulation, *DMN2*, simplifies the representation of sorties by using two-indexed arc variables. These variables represent the drone's entry and exit points from customer nodes. This approach significantly reduces the number of variables needed, leading to a more efficient and faster solution process. *DMN2* is built upon the *DMN* formulation, adapting all constraints involving the three-indexed variables to use the new two-indexed variables. The objective function in *DMN2* is also modified to reflect the use of two-indexed variables.

The authors conducted comprehensive computational experiments to evaluate the performance of their proposed formulations - $\bar{M}C$, *DMN* and *DMN2*. While the modified Murray & Chu ($\bar{M}C$) model could be solved directly by CPLEX, the *DMN*, *DMN2* models required a branch-and-cut implementation due to their exponentially large number of constraints. The authors tested their methods on the benchmark instances provided by Murray & Chu, for both the "wait" and "no-wait" versions of the problem, and with varying characteristics, including different drone endurance levels, depot locations, and speeds and compared their results to those of existing methods. The results showed that DMN2, the two-indexed formulation, significantly outperformed previous methods, solving 59 out of 72 benchmark instances to optimality, with an average percentage gap between the best lower and upper bound of 1.70% and an average computing time of less than 20 minutes, demonstrating the effectiveness of their newly proposed formulations in solving the FSTSP.

In a second paper, Dell'Amico et al. **{ADD REFERENCE}** then proposed two algorithms to solve the FSTSP; a Branch and Bound (BB) algorithm for small instances and a heuristic based on the former for larger instances. The BB algorithm utilizes the concept of "missions," which can be either truck movements or

phases where the truck and drone operate in parallel. It works by incrementally expanding partial solutions by adding new missions at each level of the search tree until the final depot is reached. Customers are assigned to drones only in the final stage by solving an Assignment Problem, a strategy aimed at limiting the search tree size. The heuristic, heavily based on BB, starts with an initial feasible TSP solution and repeatedly applies the BB method as a subroutine to improve the solution. It is designed to iteratively refine the solution by adding drone services to optimize the overall delivery time. The authors report that the BB algorithm performs well on small instances, showing competitive computation times compared to other approaches in the literature. The heuristic algorithm also demonstrates promising results on medium and large instances, providing state-of-the-art solutions.

Boccia et al. (2021) proposed compact integer linear programming formulations for the FSTSP, built on a novel extended graph representation. This representation avoids the use of big-M constraints, a common drawback of existing models that hinders their solvability. The authors' approach leverages a Branch-and-Cut (B&C) algorithm coupled with a column generation procedure, further strengthened by variable fixing strategies and valid inequalities specifically designed for the FSTSP. The B&C approach begins by considering a subset of variables and constraints and progressively adds more complex ones as needed. The column generation procedure avoids a full enumeration of variables, leading to more efficient computation. Their experimental results on the set of instances from Murray & Chu demonstrate the effectiveness of their approach by optimally solving instances up to 20 customers, exceeding the performance of prior exact approaches.

Genetic algorithms also have been proposed in recent years to solve the FSTSP. Kuroswiski et al. (2023) presented an improved MILP formulation and a Hybrid Genetic Algorithm (HGenFS). The formulation, based on existing work, features thirteen constraints, significantly reducing the complexity compared to previous ones with 28 restrictions. This proposed model is shown to be effective for solving instances with up to ten customers, proving its feasibility within seconds. Recognizing that the exact solution may not be efficient for larger instances, the authors introduce a hybrid Genetic Algorithm to tackle the problem. The HGenFS algorithm utilizes a chromosome representation that incorporates both the sequence of deliveries and the delivery method (truck or drone) for each customer. The algorithm employs various optimization techniques, including: (i) *Crossover*; two crossover methods, SWAP and DX2, are proposed for combining information from parent chromosomes to generate offspring, (ii) *Mutation*; two mutation methods to diversify the population, including swapping customer positions and alternating the delivery method, (iii) *Local Search*; incorporating a local search phase to further refine the solution, exploring alternative drone routes and delivery methods. The authors conduct extensive tests with nine real-world instances involving ten customers. Their results demonstrate that the HGenFS algorithm is capable of finding optimal solutions for the FSTSP in an average time of 6.27 seconds (with the DX2 crossover model). This significantly outperforms the exact solution, especially for more complex instances, and positions HGenFS as a viable alternative for solving the FSTSP.

Pilcher (2023) proposed a novel self-adaptive genetic algorithm and introduced a novel two-stage mutation process designed to address the complexities of the FSTSP. This GA distinguishes itself by co-evolving a memeplex alongside the population, where the memeplex comprises eight options representing crossover and mutation operators and their probabilities of being applied. This dynamic adaptation enables the algorithm to learn and adjust its evolutionary strategy over generations. Additionally, the GA utilizes a two-stage mutation process, allowing for simultaneous modifications to the tour sequence and node types within a single generation, thus offering greater flexibility in exploring the search space. The algorithm also incorporates a range of problem-specific crossover and mutation operators, further enhancing its effectiveness in addressing the complexities of the FSTSP. Furthermore, the initial population generation leverages a scoring system based on potential makespan savings, encouraging exploration of non-obvious drone node configurations. The author conducted a series of experiments to evaluate their proposed Genetic Algorithm, showing its effectiveness by finding the optimal solution for a large number of smaller instances with up to 9 customers. Furthermore, it outperformed four other rival algorithms for larger instances with up to 75 customers, while achieving a 5% improvement in the best-found solution for one of them.

2.1.2 The Parallel Drone Scheduling Traveling Salesman Problem

The PDSTSP, introduced in the same paper by Murray & Chu **{ADD REF}**, extends the concept of the FSTSP by involving multiple drones and optimizing the delivery process through parallel drone operations. In contrast to the FSTSP, the drones act independently from the truck and launch from and return to the depot. It was designed for scenarios where a significant portion of the customers are located within the UAV's flight range from the depot. This problem can be reduced to two classic, widely researched operations research problems, the TSP; to find the optimal truck route, and the Parallel identical-Machines Scheduling problem (PMS); to find the optimal assignment of UAVs (machines) to customers (jobs), such that the total completion time (makespan) of the UAV fleet is minimized. The authors proposed a MILP formulation and employed heuristic algorithms to solve larger instances of the PDSTSP. The heuristic proposed by the authors is a greedy, iterative approach that aims to partition customers between the truck and the UAV fleet. The heuristic begins by assuming all UAV-eligible customers are assigned to the UAV fleet and the remaining customers are assigned to the truck. Regarding the latter, to solve the truck's TSP route, three methods were tested by the authors; *(i) MILP*, *(ii) Savings heuristic*, *(iii) Nearest Neighbour heuristic*. To solve the PMS and optimally assign the UAV fleet to customers, a binary integer programming (IP) formulation and the *Longest Processing Time first* (LPT) heuristic were used. This initial assignment is then refined iteratively by re-assigning individual customers between the truck and UAV partitions. The heuristic calculates the total completion time for the UAV fleet (makespan) and the truck route based on their respective assignments. It then aims to balance the total makespan by re-assigning customers. If the UAV makespan exceeds the truck route's completion time, the heuristic seeks to reduce the UAV makespan by transferring a customer from the UAV partition to the truck partition. The algorithm chooses the move that offers the greatest net savings in overall makespan (LPT). Otherwise, if the truck makespan exceeds the UAV makespan, the heuristic attempts to find the best swap of a customer between the

truck and UAV partitions. This process of reallocating customers continues until no further improvements can be made. **RESULTS** To evaluate the performance of their proposed methods, the authors generated test problems with 10 or 20 customers, where 80-90% of the customers were UAV-eligible. To assess the quality of the heuristic solutions, the authors compared them to solutions obtained from a mixed-integer linear programming (MILP) formulation using Gurobi, with a 3-minute time limit per problem. The authors observed that using exact approaches to solve the underlying TSP and PMS subproblems yielded near-optimal results but required long runtimes, particularly for the 20-customer instances. However, employing the *Savings* heuristic for TSP subproblems resulted in significantly shorter runtimes and competitive solutions in terms of optimality gaps. This was true regardless of whether the PMS subproblems were solved using IP or LPT. Further analysis indicated that using the LPT heuristic performed nearly as well as IP but was generally faster. These experiments suggest that the proposed PDSTSP solution framework is effective for solving larger instances of the PDSTSP problem, particularly when combined with the *Savings* heuristic for the TSP subproblems and the LPT heuristic for the PMS subproblems.

Mbiadou Saleu et al. (2018) proposed an MILP formulation along with an iterative two-step heuristic, which consists of a coding step and a decoding step. The coding step transforms a given PDSTSP solution into a customer sequence, representing the order in which customers are visited, while the decoding step decomposes the customer sequence into a tour for the vehicle and trips for the drones. The decoding step is formulated as a bicriteria shortest path problem and solved using dynamic programming. This DP approach involves progressively associating a list of labels with each node in the graph representing the customer sequence, where each label represents the cost vector of a partial path, calculated by summing the costs of the crossed arcs. The algorithm iteratively builds these label lists, employing bounding mechanisms and a dominance rule to reduce the computational complexity. Furthermore, the authors address the challenge of handling multiple drones (M > 1) by adapting the decoding procedure, introducing an approximation where the drone delivery time is shared equally among the drones, allowing them to utilize the same dynamic programming scheme as in the single-drone case. To evaluate their approach, the authors conducted experiments on two sets of instances: first, they tested their method on benchmark instances generated by Murray and Chu (2015), demonstrating its superior performance compared to existing methods in terms of optimality gaps and computation time, and then generated new, larger instances based on TSPLIB datasets, further investigating the behavior of their heuristic and the benefits of using drones.

Dinh et al. (2021) **{ADD REF}** propose a Hybrid Ant Colony Optimization (HACO) algorithm to solve the problem. The HACO algorithm builds upon the idea of representing the PDSTSP as a giant TSP tour, which is then decomposed into a truck tour and drone trips using a new dynamic programming approach. This dynamic programming approach considers the cost of both the truck and the drones, aiming to balance the workload and minimize the overall completion time. To further enhance the efficiency and quality of the algorithm, HACO incorporates several components adapted from other metaheuristics, such as a drone node selection step based on Large Neighborhood Search, and a 3-opt local search for improving the TSP tour. The authors validate their algorithm on benchmark instances, demonstrating its superiority in terms of solution quality and computational time over existing methods, while finding new best known solutions (BKS) on 23 out of the 90 instances used.

Nguyen et al. (2022) presented a novel branch-and-cut algorithm to solve the PDSTSP. The authors begin by formulating the PDSTSP as a Mixed Integer Linear Programming (MILP) model. Their formulation utilizes truck-flow variables defined on undirected edges, reducing the number of variables compared to previous models that used directed arcs. To enhance the efficiency of the algorithm, they incorporate several valid inequalities, including connectivity constraints and 2-matching constraints. The branch-and-cut algorithm itself involves iteratively solving a linear programming relaxation of the MILP, identifying violated constraints, adding them to the LP, and re-optimizing. The process continues until all constraints are satisfied. If fractional variables exist, branching is performed. The authors provide detailed descriptions of the separation procedures for the various constraints, emphasizing the efficiency of their approach. **{RESULTS}** The authors report computational experiments conducted on a set of benchmark instances, including both existing and new, breaking new ground by achieving optimal solutions for instances up to 783 customers for the first time. The authors also analyze the performance of state-of-the-art metaheuristics, concluding that new metaheuristics are necessary for efficiently solving larger instances.

In **{ADD REF}** Mbiadou Saleu at al. introduce the Parallel Drone Scheduling Multiple Traveling Salesman Problem (PDSMTSP), where a fleet of both trucks and drones operate independently from a depot to deliver packages to customers. The objective remains that same as the PDSTSP; minimize the overall delivery completion time. The authors propose a hybrid metaheuristic algorithm to solve the PDSMTSP. The algorithm begins by constructing a giant tour that visits all customers using a *Nearest Neighbor* construction procedure. Then, a dynamic programming procedure, named "split", decomposes the giant tour into K complementary subsequences, $\tau 1$ to τK, for customers assigned to the K vehicles and a set π drones for customers assigned to the drones. This step ensures that the vehicle tours follow the order defined by the giant tour. Each vehicle route is then reoptimized using the TSP Lin-Kernighan heuristic to obtain optimal or near-optimal routes for the vehicles. Customers assigned to drones are allocated to individual drones using the *Longest Processing Time first* (LPT) heuristic, solving a Parallel Machine Scheduling (PMS) problem. Finally, local search moves are applied to improve the current solution by swapping customers between the trucks or between trucks and drones.

Constraint programming (CP) models have also been effectively utilized to handle the scheduling complexities of drone-assisted TSP variants. Montemanni and Dell'Amico (2023) proposed a novel CP model for the PDSTSP, which leverages recent developments in black-box solvers capable of exploiting parallel computation. This approach provides a fresh perspective for optimization on modern personal computers, offering a compelling alternative to traditional heuristic methods that often struggle with large instances. The model utilizes a set of binary variables (yij and xdi) to represent the truck's tour and the drone assignments, respectively. Additionally, a variable α captures the objective function value, representing the time taken by the last vehicle to return to the depot. The CP model incorporates constraints that

ensure a feasible truck tour, drone assignments, and minimize the overall delivery time. The authors conducted extensive computational experiments to validate the effectiveness of their CP approach. Their results demonstrate that the CP model significantly outperforms existing exact and heuristic methods, achieving optimal solutions for all the instances tested, including those previously unsolved. The research also highlights the substantial improvements in exact methods for the PDSTSP over the years, and the advantage of using parallel computation in modern CP solvers.

Nguyen et al. (2023) introduced a new variant of the parallel drone scheduling traveling salesman problem that aims to increase the utilization of drones, particularly for heavy item deliveries, which they termed the PDSTSP with collective drones (PDSTSP-c). The authors developed a two-index MILP formulation to solve small-size instances to optimality and proposed a ruin-and-recreate metaheuristic to efficiently handle larger problem instances. Montemanni et al. (2024) expanded on the research for the PDSTSP-c and introduced a MILP and a constraint programming approach.

2.1.3 TSP-D

Agatz et al. (2018) introduced the Travelling Salesman Problem with Drones (TSP-D). While the TSP-D shares mostly the same characteristics as the FSTSP, the authors decide to neglect the delivery time cost of both vehicles and the recharging time cost of the drone. The authors model the problem and propose an IP formulation with which they manage to solve instances with up to 12 customers to optimality. As exact methods become computationally intractable for larger instances, the authors propose several fast *route first-cluster second* heuristics based on local search and dynamic programming. These heuristics first construct a truck-only TSP tour, using either an optimal TSP solution (Concorde TSP Solver) or a faster *minimum spanning tree* (MST) heuristic. Next, they assign drone nodes to this tour using either a greedy partitioning algorithm or an exact partitioning algorithm where they employ a dynamic programming algorithm which finds the optimal partitioning of the TSP tour into truck and drone subtours. To refine the solution, they apply an iterative improvement procedure that modifies the initial TSP tour and reevaluates the drone node assignment using the chosen partitioning method. The authors provide a theoretical analysis of their heuristics, establishing a worst-case approximation guarantee for their heuristics, and demonstrating their effectiveness in providing near-optimal solutions. They also prove a lower bound on the optimal solution value of the TSP-D, showcasing that TSP algorithms can be used as approximation algorithms for the TSP-D. Furthermore, experiments were conducted to assess the performance of a drone delivery system under various conditions. They examine three key aspects: customer density, geographic distribution, and drone speed. Their experiments demonstrate that the relative savings of using a drone remain consistent regardless of the number of customer locations (n = 10, 50, 100). However, savings are more pronounced when customers are clustered geographically, especially around one or two centers. This aligns with the intuitive notion that drones can efficiently serve locations outside cluster centers, reducing overall delivery time. Further, the benefits of a combined truck-and-drone delivery system increase with the drone's speed, as a faster drone can more effectively parallelize deliveries and minimize waiting times.

Yurek et al. (2018) contributed to the TSP-D research by proposing a decomposition-based, two-stage iterative optimization algorithm. In the first stage, the truck route is determined. This determination also dictates which customers are assigned to the drone, as those not served by the truck must be visited by the drone. The second stage, employing an MIP formulation, then optimally determines the drone route, taking into account the fixed truck route and drone customer assignments established in the first stage. To improve the efficiency of the algorithm, the authors propose a heuristic that significantly reduces the number of truck routes generated in the first stage. This heuristic utilizes the nearest neighborhood approach for each possible drone node assignment, generating a single truck route and eliminating other routes involving the same customer nodes in different sequences. The algorithm's performance is tested on randomly generated instances with varying problem sizes and customer distribution patterns, demonstrating its ability to efficiently solve instances with up to 12 customers within a reasonable time, compared to previous studies that could only solve instances up to 10 customers within the same timeframe.

Ha et al. (2018) introduced the min-cost TSP-D, a novel variant of the problem with the objective of minimizing the operational costs, which include transportation costs and waiting penalties for both the truck and drone. The authors proposed a MILP formulation and two heuristic methods to solve the problem. The authors propose a MILP formulation for the problem along with two heuristics. The TSP-LS heuristic, adapted from the heuristic by Murray and Chu (2015), is designed to solve the min-cost TSP-D problem by converting an optimal TSP solution into a feasible TSP-D solution through a series of local searches. It begins by calculating cost savings for relocating a truck node to a different position or inserting a drone node between two existing nodes in the truck tour. Based on these cost savings, the algorithm then iteratively relocates truck nodes and inserts drone nodes, seeking to improve the solution, and terminates when no further positive cost savings can be achieved. The second proposed heuristic is a *Greedy Randomized Adaptive Search Procedure* (GRASP) that aims to solve the problem. It operates in two phases: construction and local search. The construction phase involves a split algorithm that converts a TSP solution into a feasible TSP-D solution. This is achieved by building an auxiliary graph where each subsequence of nodes represents a potential drone delivery. The algorithm then extracts the shortest path in the auxiliary graph, which corresponds to the best TSP-D solution derived from the initial TSP tour. The local search phase further improves the constructed solution through a set of operators, including relocation, drone relocation, and drone removal. The authors also adapt the GRASP algorithm to address the min-time TSP-D problem by adjusting the cost computation in the auxiliary graph to prioritize time over cost. The authors conducted several experiments to assess the performance of their proposed heuristics for both min-cost and min-time TSP-D problems. They evaluated GRASP's performance using different TSP tour construction heuristics, finding that k-nearest neighbor yielded the best results. They also compared GRASP and TSP-LS with optimal solutions for 10-customer instances, demonstrating that GRASP consistently found optimal solutions while being significantly faster than MILP. Further analysis on larger instances showed that GRASP outperformed TSP-LS in terms of solution quality, even though it

required slightly more computation time.

2.1.4 Other

Schermer et al. (2019) introduced the Traveling Salesman Drone Station Location Problem (TSDSLP), combining TSP, facility location, and parallel machine scheduling problems to minimize operational costs. Kim & Moon (2019) introduced a mixed integer programming model for the TSP with a drone station (TSP-DS), which extends the operational range of drones.

Numerical results indicate that optimizing battery weight and reusing drones are important considerations for drone delivery (Dorling 2016).

We thank Dr. Eddie Cheng for asking about faster drones versus more drones.

At the same time, it is important that we address flight range limitations. Such limitations have important roles in much of the existing drone routing research. However, even he first drones developed by Amazon and JD.com ahad a flight range of 15 to 20 miles [33, 56]. Such a range is suitable to allow out and back travel in the medium-sized cities in which the authors live, Braunschweig, Germany, and Iowa City, United States. In fact, it is suitable for many larger cities such as Hamburg, Munich and Paris. Thus, in contrast to other drone applications, in this work, we do not consider flight range as a limiting factor. (Ulmer, 2018)

2.2 The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is a cornerstone in the field of combinatorial optimization and logistics, focusing on the optimal routing of a fleet of vehicles to service a set of customers. Since its introduction by Dantzig & Ramser (1959) **{ADD REF}**, numerous variants and extensions of the VRP have been developed to address practical and complex logistics challenges. The Capacitated VRP (CVRP), seeks to determine the optimal set of routes for a fleet of vehicles located in a single depot node to deliver goods to a set of customers, minimizing the total cost while satisfying constraints such as vehicle capacity and customer demand. The cost is typically defined as either the total distance, time, fuel consumption, pollution or a combination of those. The primary distinction between the VRP and the TSP lies in the capacity constraint of the vehicles; the vehicles return to the depot when they can no longer service any customer, to either reload or end their workday. When the vehicles have unlimited or sufficiently large capacity as to not restrict their movement, the CVRP effectively reduces to a multiple Traveling Salesman Problem (mTSP). Therefore, in this paper, we will refer to the CVRP simply as the VRP.

2.2.1 Drone-related VRP Research

The rapid advancement of drone technology has spurred significant interest in its application to logistics and transportation, particularly in addressing the complexities of last-mile delivery. Vehicle Routing Problems with Drones (VRP-D) represent an innovative extension of traditional VRPs, where drones are deployed alongside traditional delivery vehicles to enhance efficiency and reduce operational costs.

This integration leverages the agility and speed of drones to complement the capacity and range of trucks, presenting new challenges and opportunities in route optimization. The VRP-D and its variants focus on optimizing the coordinated routes of both vehicles and drones to ensure timely and cost-effective deliveries, considering constraints such as drone battery life, payload capacity, and synchronization requirements. This literature review explores the foundational concepts of VRP with Drones, the evolution of algorithmic approaches, and the diverse applications and implications of integrating drones into vehicle routing problems.

Wang et al. **{ADD REFERENCE}** introduced the Vehicle Routing Problem with Drones (VRPD), where a fleet of m trucks, each capable of carrying a number k of drones, deliver packages to customers with a demand of 1 parcel each. The primary objective is to minimize the total completion time in which all of the customers are serviced. Each drone must be launched and recovered from the same truck and synchronization between the two is needed, but unlike the FSTSP and TSP-D, the authors allow the drone to launch from and return to its truck at the same node location. Furthermore, this paper assumes a drone battery large/efficient enough such that it does not limit the drone's flight endurance and the drone capacity to be equal to 1 parcel i.e drones can service one customer per sortie, while trucks have a capacity C, and both the truck and drone follow the same road network. To address the VRPD the authors employ a worst-case analysis approach to evaluate the potential benefits of integrating drones into the vehicle routing problem. This analysis is structured around comparing the traditional VRP, which utilizes trucks only, and the VRPD. The primary objective is to establish theoretical bounds on the time savings achievable by using drones. The authors formulate several theorems that derive upper bounds on the ratio of the completion times of these two problems, aiming to demonstrate that the use of drones can lead to significant reductions in delivery times. The analysis of specific cases, such as when drones travel faster than trucks, allow the authors to illustrate the worst-case scenarios where the completion time with trucks alone is significantly longer than with the combined use of trucks and drones. This structured approach not only highlights the theoretical underpinnings of the VRPD but also sets the stage for future research that could explore more complex scenarios, such as limited battery life and varying distance metrics for drones and trucks.

Schermer et al. (2018) **{ADD REF}** address a modified version of the above VRPD. In this paper, the authors assume that the drones have a limited flight range due to battery capacity and that they cannot be recovered at the same node location where they were launched from. The authors introduce the *Two-Phase Heuristic* (TPH) and the *Single-Phase Heuristic* (SPH) to solve the problem. The TPH begins by constructing a good VRP tour using the *Nearest Neighbor* heuristic. This initial tour includes all customer vertices and is then split into segments corresponding to the number of available m trucks. In the first phase, the algorithm focuses on improving the VRP solution by employing improvement techniques such as *2-opt* local search and *String Mix* (SM), a combination of *String Exchange* and *String Relocation* **{EXPLAIN}**. After a predetermined amount of time has passed, the algorithm enters the second phase, where drones are integrated into the constructed routes. This involves two procedures: *Drone Insertion* (DI); drones are inserted into the existing tours based on the first feasible sortie that reduces the overall completion time and *Delivery Exchange*

(DE); attempts to change the delivery method of one tour between the truck and the drone, ensuring that the time to complete the tour is minimized. Similarly, the SPH starts with the NN heuristic to create an initial tour that includes all vertices. Unlike the former, the SPH inserts drones into the routing solution right after the initial tour is created. This means that the drone operations are considered throughout the entire optimization process along with the truck operations, and the algorithm iteratively calls the aforementioned optimization procedures (2-opt, SM, DI, DE) until a stopping criterion is met. **{RESULTS}** The authors conducted experiments to evaluate the performance of their proposed heuristics utilizing 3 trucks with unlimited capacity and each equipped with 1 drone with a capacity of 1 parcel. The results indicate that the TPH generally outperforms the SPH across a range of scenarios, particularly in larger instances. However, the SPH demonstrates competitive performance in specific cases, suggesting that the choice of heuristic may depend on the problem's constraints and parameters.

The same authors in **{ADD REF}** 2019 introduced for the first time a MILP formulation for the VRPD which supports multiple drones per truck and cyclic drone operations (drones may return to the same node they were launched from). To enhance the performance of MILP solvers, the authors propose several sets of valid inequalities that significantly improve the solver's ability to find optimal solutions for small instances. However, they acknowledge the limitations of MILP formulations for larger instances, prompting the development of a matheuristic approach. This matheuristic effectively exploits the problem structure of the VRPD and incorporates the Drone Assignment and Scheduling Problem (DASP) as a key component, which optimizes the assignment and scheduling of drones based on existing truck routes. The authors conduct extensive computational experiments to evaluate the performance of their proposed methods, with the results highlighting the value of drone usage in last-mile delivery scenarios. They also conduct a sensitivity analysis on relevant drone parameters, providing insights into how variations in drone capabilities like speed and flight range affect overall performance.

Sacramento et al. (2019) expanded the research by introducing additional constraints to further simulate real-world challenges. The authors take into account time constraints associated with the launch and recovery of the drone, service times for customer deliveries, limited truck capacity and a truck maximum route duration. In the VRP-D proposed, a fleet of homogenous trucks, each equipped with a single drone must service a set of customers with known demands while minimizing the total operational cost. This cost is defined as the sum of the transportation costs incurred by both the trucks and the drones. Specifically, the cost for the truck is related to the fuel price and consumption rate, while the cost for the drone is set as a factor of the truck's cost, reflecting that using the UAV is considerably cheaper than using the truck. The authors propose a mathematical model that extends the Mixed Integer Programming (MIP) formulation of the FSTSP, incorporating multiple trucks and capacity and time constraints. To solve the problem, the authors employ an Adaptive Large Neighborhood Search (ALNS) metaheuristic. This algorithm iteratively enhances an initial solution by utilizing a systematic approach that combines destruction and repair methods. The initial solution is generated through a three-phase process. First, the *Nearest Neighbor* heuristic is used to obtain a truck-only solution. Then, a *String Relocation* local search algorithm is employed to refine the truck-only solution. Finally, a drone addition algorithm is called which considers all drone-eligible customers to be swapped from the truck to the drone, and iteratively makes swaps that provide the maximum possible savings until no improvement is possible. The ALNS operates by first destroying a portion of the current solution, allowing for exploration of new solution spaces, with the destruction controlled by parameters that determine the number of customers to remove. The algorithm adapts the choice of destruction and repair methods based on their historical performance, facilitating dynamic search processes that can escape local optima. The ALNS incorporates an acceptance criterion inspired by Simulated Annealing, permitting the acceptance of worse solutions with a certain probability, particularly in the early search stages. **{RESULTS}** Extensive experiments demonstrate the algorithm's effectiveness in significantly improving initial solutions and achieving substantial cost savings compared to traditional truck-only delivery approaches, highlighting its potential for practical applications in logistics and delivery operations.

Wang & Sheu (2019) **{ADD REF}** propose a novel arc-based MIP model along with a branch-and-price algorithm to solve the VRPD. Due to the complexity and large number of constraints in this model, the authors reformulate the problem into a path-based model, focusing on feasible paths for both trucks and drones, allowing for a more efficient solution process. Then, the authors introduce a branch-and-price algorithm which consists of three key components. First, a pricing sub-problem which aims to find the minimum reduced cost paths for both trucks and drones. The authors design a specialized network that distinguishes between different types of paths and nodes, facilitating the identification of optimal routes. Next, the algorithm employs column generation techniques to manage the large number of potential paths efficiently. This involves generating new columns (paths) iteratively based on variables from the restricted master problem. Finally, to mitigate slow convergence issues, the authors implement stabilization techniques, including the computation of alternative Lagrangian lower bounds and the application of column generation stabilization methods. Extensive computational experiments demonstrate the algorithm's effectiveness, achieving optimal solutions significantly faster than traditional methods, while also revealing substantial cost savings and reduced delivery times when utilizing drones in urban logistics.

Lei et al. (2022) **{ADD REF}** address the VRPD by proposing a Dynamic Artificial Bee Colony (DABC) algorithm, which solves the problem by dynamically managing two swarms of artificial bees - employed bees and onlooker bees - based on the quality of solutions. Initially, the algorithm generates two swarms and identifies the best solution among them. The algorithm iteratively evaluates the swarms to determine which solutions belong to the employed bee group (EB) and the onlooker bee group (OB). In the employed bee phase, each solution in EB undergoes a Variable Neighborhood Descent (VND) process, which utilizes 15 distinct neighborhood structures to explore and refine solutions. In the onlooker bee phase, solutions in OB are assessed against the average objective value of the group, and if a solution is below this average, it is also subjected to VND; otherwise, a probabilistic decision is made to either perform VND or replace the solution with one from EB through a binary tournament. The algorithm incorporates a scout phase, where solutions that fail to improve after a set number of trials are replaced with new random solutions. This dynamic adjustment of the bee groups

and the application of VND in different manners across the two phases enhance the exploration and exploitation capabilities of the algorithm, ultimately leading to an improved solution quality. The authors compare their DABC algorithm against several comparative algorithms, including the ALNS proposed by Sacramento **{ADD REF}**, showing promising results.

Stodola (2024) **{ADD REF}**

3 The Novel Problem

The MD-mfcmTSP can be modeled as an undirected weighted graph $G = \{V, E\}$ where $V = \{C, D\}$ is the set of vertices of both customers and depots, and E is the set of edges. Vehicles starting from each depot must visit every customer in C exactly once and then finish their tour(s) at their depot, in the minimum time. Considering 3 different vehicle types (Truck, Motorbike, Drone), customer restrictions (trucks cannot visit some customers due to alleys, drones cannot visit some customers due to customer preference, safe landing space limitations) and the fact that vehicles cannot visit depots other than their own, there are several induced subgraphs $G_v = \{G_v^i \cup G_v^{i+1} \cup ... \cup G_v^m\}$ produced for each vehicle type v and each depot $d^i \in D$. For each vehicle type v, G_v contains all customer nodes that are accessible by v and all depot nodes which are equipped with a vehicle v. The weight of each edge e_{ij} is calculated as the euclidean distance d_{ij} divided by the velocity of vehicle v, s_v; ($e_{ij} = d_{ij}/s_v$). In the case of vehicles with capacity > 1 these graphs are fully connected, allowing travel between customers. In the case of drones travel between customers is not possible due to capacity = 1, thus each G_D^i, $d^i \in D$ is a star graph. The objective is to minimize the total completion time in which all of the customers are serviced i.e $M_{total} = max(M_T, M_M, M_D)$.

3.1 Makespan Calculations

In the case where a depot is equipped with more than 1 vehicle of any type (excluding vehicles with capacity = 1 i.e drones), and two or more routes need to be assigned to them, a *job scheduling problem* arises, where the routes are the jobs that need to be assigned to machines (vehicles). Since the vehicles are identical, it's more specifically an *identical-machines scheduling* problem. Minimizing the maximum completion time (makespan) is NP-hard. Many exact and approximation algorithms are known. We use the *Longest Processing Time first* (LPT) greedy algorithm, which orders the jobs by descending order of their cost and then schedules each job in this sequence into the machine of which the current load is smallest.

4 Solving the MD-mfcmTSP

In this section, we introduce the two algorithmic approaches to address the MD-mfcmTSP. The first approach which is used to tackle the problem comprises of a clustering phase, which transforms the multi-depot problem into multiple single-depot mfcmTSP problems, and a routing phase which calls the mfcmTSP heuristic for each depot. The second approach leverages a hybrid metaheuristic, combining Ant Colony Optimization (ACO) with the mfcmTSP heuristic.

4.1 MD-mfcmTSP heuristic

To manage the complexity of multi-depot routing problems, a widely used approach is their transformation to multiple single-depot routing problems. Although a naive approach, it provides feasible solutions of which the results are used as a baseline for comparison with other solution methods.

For this purpose, we use a straightforward constrained proximity clustering to assign customers to depots, and then run the mfcmTSP heuristic for each depot. Each customer is assigned to the closest *possible* depot, creating clusters where each depot serves the customers nearest to it. Specifically, vehicle availability at each depot may limit the customers that are assigned to its cluster. The reason is that the initialization phase of the heuristic, which is based on these clusters, must form routes that visit every node $c_i \in C$. Therefore, while the assignment is based on proximity, we must adjust the clustering to account for these constraints. This means that some customers may need to be assigned to a more distant depot if the nearest depot lacks the vehicle that can serve them. For example, a customer only accessible by motorbike will be assigned to a more distant depot when the closest to them doesn't have a motorbike available in its fleet.

4.2 Pseudocode for the MD-mfcmTSP Heuristic

Algorithm 1: MD-mfcmTSP heuristic

Input: $G_T, G_M, ..., G_D$
Output: M_{total}, $Sol = \{Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}, Sol^{i+1}, ..., Sol^m\}$ for each $i \in D$

Create clusters K^i of customer nodes for each depot $d^i \in D$
by assigning each customer to the closest possible depot
for *each* $d^i \in D$ **do**
  Call $Initialization(d^i, K^i)$
  while $(M_T^i > M_M^i \parallel M_T^i > M_D^i)$ && $stop \neq true$ **do**
    $diff_M = M_T^i - M_M^i$
    $diff_D = M_T^i - M_D^i$
    if $diff_M \geq diff_D$ **then**
      $vt = M$
      $cap =$ Motorbike's capacity
    else
      $vt = D$
      $cap = 1$
    end if
    $M_{dep} = M_T^i$
    $r_{best} = \emptyset$
    for $j = 1$ *to* $|R_T^i| - cap$ **do**
      $successive\_nodes = \emptyset$
      $load = 0$
      while $load + v_j^{demand} \leq cap$ && $v_j \in G_{vt}$ **do**
        $successive\_nodes \mathrel{+}= v_j$
      end while
      if $|successive\_nodes| == cap$ **then**
        $r_{new} = R_T^i[0] + \{successive\_nodes\} + R_T^i[0]$
        $R'^i_{vt} = R_{vt}^i + r_{new}$
        $M'_{vt} = R'^i_{vt}$'s makespan
        $R'^i_T = R_T^i - \{successive\_nodes\}$
        $M'_T = R'^i_T$'s makespan
        $M_{new} = MAX(M'_T, M'_{vt})$
        if $M_{new} < M_{dep}$ **then**
          $M_{dep} = M_{new}$
          $r_{best} = r_{new}$
        end if
        $r_{new} = \emptyset$
      end if
      $j \mathrel{+}= 1$
    end for
    if $r_{best} \neq \emptyset$ **then**
      $R_T^i = R_T^i - \{r_{best}^{customers}\}$
      $M_T = R_T^i$'s makespan
      $R_{vt}^i \mathrel{+}= r_{best}$
      $M_{vt} = R_{vt}^i$'s makespan
      (+Swap)
      Call $local\_optimization(R_T^i, n_{max})$
      Call $local\_optimization(R_{vt}^i, n_{max})$
    else
      $stop = true$
    end if
  end while
  $Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}$
end for
$M_T = MAX(M_T^i, M_T^{i+1}, ..., M_T^m)$
$M_M = MAX(M_M^i, M_M^{i+1}, ..., M_M^m)$
$M_D = MAX(M_D^i, M_D^{i+1}, ..., M_D^m)$
$M_{total} = MAX(M_T, M_M, ..., M_D)$
Call $optimization\_full(Sol, n_{max})$

Algorithm 2: Initialization(d^i, K^i)

while $\{K^i\} \cap \{G_T\} \neq \emptyset$ **do**
  $R_T^i \mathrel{+}= NearestNeighbour(\{K^i\} \cap \{G_T\})$
end while
$M_T^i = R_T^i$'s makespan
$v_{free} = \{K^i\} - \{G_T\}$
if $v_{free} = \emptyset$ **then**
  return R_T^i
else
  while $v_{free} \neq \emptyset$ **do**
    if $M_T - M_M \geq M_T - M_D \parallel G_D = \emptyset$ **then**
      $R_M^i \mathrel{+}= NearestNeighbour(\{K^i\} \cap \{G_M\})$
      $v_{free} = v_{free} - \{R_M^i\}$
      $M_M^i = R_M^i$'s makespan
    else
      $R_D^i \mathrel{+}= closest(\{K^i\} \cap \{G_D\})$
      $M_D^i = R_D^i$'s makespan
    end if
  end while
end if
return Sol^i

Algorithm 3: $optimization\_full(Sol, n_{max})$

do
  Call $vt\_optimization(Sol, n_{max} = 2)$
  for *each* vt **do**
    for *each* $i \in D$ **do**
      Call $local\_optimization(R_{vt}^i, n_{max} = 2)$
    end for
    Call $mutual\_depot\_optimization(R_{vt}, n_{max} = 2)$
  end for
while *any route improves*

Algorithm 4: $local\_optimization(r, n_{max})$

for $n = 1$ *to* n_{max} **do**
  for each *combination of n successive nodes on the route*
    move the node(s) to a different place on the same route
    evaluate the new route
    if *this route is better than the original and all constraints are satisfied* **then**
      replace the original route with the new one
    continue in point 3 unless all possible places in the route have been evaluated
  end for
end for
return r

Algorithm 5: $mutual\_depot\_optimization(R_{vt}, n_{max})$

for $n = 1$ *to* n_{max} **do**
  for each *possible pair of depots* $c1$ *and* $c2$
    for each *combination of* n *successive nodes in the route of* $c1$
      remove the nodes from the route of c1 and insert them into c2
      evaluate the newly-created routes
      if $MAX(|R'^{c1}_{vt}|, |R'^{c2}_{vt}|) < MAX(|R^{c1}_{vt}|, |R^{c2}_{vt}|)$ *and all constraints are satisfied* **then**
        replace the original routes with the new ones
      continue in point 4 unless all possible places in c2 have been evaluated
    end for
  end for
end for
return R_{VT}

Algorithm 6: $vt\_optimization(Sol, n_{max})$

for $n = 1$ *to* n_{max} **do**
  for each *depot* $i \in D$
    for each *possible pair of vehicle types* $t1, t2 \in VT$
      for each *combination of* n *successive nodes in* R^i_{t1}
        remove the nodes from R^i_{t1} and insert them in R^i_{t2}
        if $MAX(|R'^i_{t1}|, |R'^i_{t2}|) < MAX(|R^i_{t1}|, |R^i_{t2}|)$ *and all constraints are satisfied* **then**
          replace the original routes with the new ones
        continue in point 5 unless all possible places in R^i_{t2} have been evaluated
      end for
    end for
  end for
end for
return Sol

4.3 Hybrid Ant Colony Optimization

For the second approach to solve the MD-mfcmTSP we introduce a novel hybrid metaheuristic based on the principles of Ant Colony Optimization combined with the mfcmTSP heuristic. Specifically, we employ a modified version of the Adaptive Ant Colony Optimization with Node Clustering (AACONC) (Stodola et al. 2022) which was developed for the Multi-Depot Vehicle Routing Problem and showed promising results. From now on, we refer to this algorithm as the H-AACONC.

4.3.1 H-AACONC Algorithm

The Ant Colony part of the algorithm disregards accessibility constraints and assumes that all customers must be served by trucks such as to minimize the total time (makespan). As such, the pheromone matrix, node clustering and the AntSolution function remain the same as in AACONC and are used to create an initial trucks-only solution which is stored in Rt. Then, with frequency n_{freq}, which is one of the AACONC control parameters, the single and mutual colony local optimization procedures may be called. In the resulting solution Rt, customers that are not accessible by trucks are then offloaded to the motorbike(s) of the same depot if possible, else to the closest depot with atleast one motorbike in its fleet. The resulting routes are inserted into R_{best} and the MD-mfcmTSP heuristic is called, while skipping the clustering and initialization phase.

4.3.2 Pheromone matrix update

One key distinction between the AACONC and the H-AACONC is the pheromone matrix update procedure. The pheromone matrix update procedure uses the same logic as the AACONC. The Simulated Annealing principle in combination with the Metropolis criterion probabilistically decide which solution will be used for the update. Although in the H-AACONC, while solutions R and R_{best} are used to make this decision, it is then truck solutions Rt and Rt_{best} respectively that are used to update the pheromone matrix.

$$p(R^{best}) = 1 - p(R) = \begin{cases} e^{-\frac{(|R^{best}|-|R|)/|R|}{T_{update}}} & \text{for } |R^{best}| > |R|, \\ 1 & \text{otherwise} \end{cases} \quad (1)$$

$$T_{\text{update}}(iter+1) = \alpha_{\text{update}} \cdot T_{\text{update}}(iter) \quad (2)$$

The solution for updating the pheromone matrix R^{update} is selected based on the calculated probabilities: $R^{update} = R^{best}$ and $Rt^{update} = Rt^{best}$ with $p(R^{best})$ or $R^{update} = R$ and $Rt^{update} = Rt$ with $p(R) = 1 - p(R^{best})$.The update itself is then conducted using (3); the pheromone trails lying on the routes are increased in proportion to the pheromone updating coefficient δ and the quality of the updating route (a ratio of R to R^{update}).

$$\tau_{ij}^{k} = \tau_{ij}^{k} + \chi_{ij} \cdot \delta \cdot \frac{|R|}{|R^{update}|} \text{ for all } v_i, v_j \in V \text{ and } d_k \in D \quad (3)$$

$$\chi_{ij} = \begin{cases} 1 & \text{if there is an edge between } v_i \text{ and } v_j \text{ in } Rt^{update}, \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

Algorithm 7: Hybrid AACONC

Input: $G_T, G_M, ..., G_D$
Output: M_{total}, $Sol = \{Sol^i = \{R_T^i, R_M^i, ..., R_D^i\}, Sol^{i+1}, ..., Sol^m\}$ for each $i \in D$

$iter = 0$
Initialize pheromone matrices τ
for each $v_i \in V$
  $K^{(v_i)}$ = Call CreateClusters
end for
$Rt = \infty, R = \infty$
while *NOT TERMINATED* **do**
  $Rt_{best} = \infty, R_{best} = \infty$
  for $\alpha = 1$ *to* n_{ants} **do**
    R_a = **Call AntSolution**
    //which construct routes, using Trucks, for every $c \in C$
    if $|R_a| < |Rt_{best}|$ **then**
      $Rt_{best} = R_a$
    end if
  end for
  if $iter \ mod \ n_{freq}$=0 **then**
    Rt_{best} = Call $local\_optimization$
    Rt_{best} = Call $mutual\_depot\_optimization$
  end if
  Assign customers in Rt_{best} which do not belong to G_T to R_{best}^M, R_{best}^D
  $R_{best}^T = Rt_{best} - \{R_{best}^M, R_{best}^D\}$
  R_{best} = Call MD-mfcmTSP heuristic
  if $|R_{best}| < |R|$ **then**
    $R = R_{best}$
    $Rt = Rt_{best}$
  end if
  Update pheromones
  Evaporate pheromones
  $iter$ += 1
end while