Optimizing Traveling Salesman Problem in Multi-Agent Systems with Practical Constraints

by

Ruixiao Yang

Submitted to the Department of Aeronautics and Astronautics

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: Ruixiao Yang
Department of Aeronautics and Astronautics
May 2, 2024

Certified by: Chuchu Fan
Assitant Professor of Aeronautics and Astronautics, Thesis Supervisor

Accepted by: Jonathan P. How
R.C. Maclaurin Professor of Aeronautics and Astronautics
Chair, Graduate Program Committees

Optimizing Traveling Salesman Problem in Multi-Agent Systems with Practical Constraints

by

Ruixiao Yang

Submitted to the Department of Aeronautics and Astronautics
on May 2, 2024 in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN AERONAUTICS AND ASTRONAUTICS

ABSTRACT

The Traveling Salesman Problem (TSP) is a fundamental challenge in multi-agent systems, particularly in task allocation scenarios. Traditional models considering the unconstrained multi-agent TSP, which require multiple salesmen to visit all customers collectively, often fail to produce feasible solutions for real-world applications due to practical constraints. To address this gap, we explore two prevalent constraints: energy limitations and aerial robot collaboration. We introduce two novel formulations: the Multi-Agent Energy-Constrained TSP (MA-ECTSP) and the Multi-Agent Flying Sidekick TSP (MA-FSTSP). The MA-ECTSP considers constraints such as limited battery levels and inter-agent conflicts at replenishment sites, while the MA-FSTSP models scenarios where multiple trucks, each equipped with several drones, collaborate to visit all customers, with trucks restricted to roads and drones having greater freedom in their flight paths. We propose a three-phase framework that first deconstructs these complex problems into more manageable single-agent versions, then optimizes them separately without constraints as heuristics, and finally integrates the heuristics and optimizes under the practice constraints. For the MA-ECTSP, we decompose the instance into smaller sub-problems by splitting the minimum spanning

tree (MST), solve each using a combination of TSP solvers and heuristic searches, and then aggregate the tours into a feasible solution using a Mixed-Integer Linear Program (MILP) with significantly few variables and constraints. For the MA-FSTSP, we initially decompose the problem into subproblems of one truck with multiple drones, compute routes for trucks without drones, and use these in the final phase as heuristics to optimize both drone and truck routes concurrently. Our approach demonstrates significant effectiveness and scalability compared to existing baselines, as validated on real-world road networks.

Thesis supervisor: Chuchu Fan

Title: Assitant Professor of Aeronautics and Astronautics

Acknowledgments

I am profoundly grateful to my advisor, Prof. Chuchu Fan, for her immense patience and invaluable guidance during my master's studies. I had a hard time in my first year, struggling to find a community that acknowledged and appreciated my work. Her consistent encouragement, effective advice, and willingness to explore solutions with me were instrumental in helping me through that difficult period.

I would like to extend my thanks to Yue Meng, a friend and lab mate, whose mentorship in technical knowledge and idea brainstorming has been immensely beneficial. I am also thankful to Dr. Kunal Garg for his expert guidance in crafting a technical paper, sentence by sentence. I appreciate my lab mates—Mingxin, Oswin, Songyuan, Charles, Cheng, Yilun, Yongchao, Anjali—and my collaborator, as well as the incoming lab mate, Eric, for their companionship throughout this academic journey.

I am also grateful to my friends—Haike, Yang, Yuxin, Yun, and Mingxin—for enriching my life with their friendship. Leaving China for the first time to live in a new country with a different language and culture was a daunting experience. However, thanks to these friends, I never felt alone.

Lastly, I owe a heartfelt thank you to my family, especially my parents, for their unconditional spiritual support over the past two years. Their presence was a constant reminder of home and strength, which was especially comforting during times of challenge.

Contents

Title page 1

Abstract 3

Acknowledgments 5

List of Figures 11

List of Tables 15

1 Introduction 17

1.1 Practical Constraints . . . 18

1.1.1 Energy Limit . . . 18

1.1.2 Cooperation with Aerial Robots . . . 19

1.2 Contribution . . . 20

1.3 Structure . . . 22

2 Related Work 23

2.1 Methods for TSP . . . 23

2.1.1 Exact methods . . . 23

2.1.2 Approximation and heuristic algorithms . . . 23

2.1.3 End-to-end learning-based methods . . . 24

2.2 Multiple TSP . . . 24

2.3 Energy Constraints . . . 25
2.4 Collaborative Constraints . . . 25
2.4.1 Multi-Agent Path Finding . . . 25
2.4.2 Trucks and drones delivery problem . . . 26

3 Multi-Agent Energy-Constrained Traveling Salesman Problem 27
3.1 Problem Formulation . . . 27
3.2 Methodology . . . 28
3.2.1 Customer assignment . . . 30
3.2.2 Congestion control . . . 33
3.3 Experiments . . . 35
3.3.1 Comparison with Baselines . . . 37
3.3.2 Compare with Exact Algorithm . . . 38
3.3.3 Studies of Components . . . 41
3.3.4 Effectiveness of Resource Distribution . . . 42
3.3.5 Effect of replenishment time . . . 43

4 Multi-Agent Flying Sidekick Traveling Salesman Problem 45
4.1 Problem Formulation . . . 45
4.2 Methodology . . . 48
4.2.1 Phase 1: Set nearest-neighbor method . . . 50
4.2.2 Phase 2: Set traveling salesman problem heuristic . . . 51
4.2.3 Phase 3: Decode solution from heuristic . . . 54
4.3 Experiments . . . 57
4.3.1 Comparison against baselines . . . 57
4.3.2 Validation for set-based methods . . . 60
4.3.3 Factors influencing the algorithm . . . 63

5 Conclusion and Future Work 67

A Mixed-Integer Linear Programming Forumlations **69**

B Theorems and Proofs **73**

B.1 Explaination for Function f and g 73

B.2 Proof for Theorem 1 74

B.3 Proof for Theorem 3 76

References **77**

List of Figures

3.1 A solution of MA-ECTSP in Manhattan: salesmen start from depots to collaboratively visit all customers and always keep their energy above zero. . . . 29

3.2 Illustration of partitions. The orange nodes represent customers, and the blue nodes represent depots. Solid lines indicate the edges remaining after partitioning, and dashed lines represent the edges removed after the partition. In Fig.(a), we disconnect one depot from the root node to form two subtrees with one depot in each. In Fig.(b), we disconnect both depots from the root to form 3 subtrees, each with a depot except the subtree containing the root. 31

3.3 Four Datasets for experiments. (a) The driving road map of Manhattan, New York. (b) The driving road map of Cambridge, Massachusetts (c) Hospitals and Tesla's Supercharger Stations in Massachusetts. (d) The existing benchmark for MDVRP [47] 35

3.4 Results on comparison experiments. (a) The distribution of optimality gaps with a mean value of 12.48%. (b) The change in the percentage of running time for each part in the framework. (c) Comparison on Scalability. All Y-axes are time in seconds. 39

3.5 Results of studies on components. (a) Keep the TSP solver to be LKH, our partition algorithm outperforms all baselines; (b) keep the partition algorithm MST-based, and the LKH solver produces the best solution quality and competitive computation time. 41

3.6 Effect of the dispersion of stations. There are no feasible solutions for cases with 1 or 3 station(s). 42

4.1 An example of the truck and drone routes on a directed graph with node 1∼6. The solid arrows (in all colors) represent the road that the truck must follow. The truck route (A, F, D, E, A) is colored red and the drone route (A, F, C, D, E, B, A), represented as $\langle$(A, F, D, E, A), $\{(2, \mathrm{C}, 3), (5, \mathrm{B}, 6)\}\rangle$, where 2, 3, 4, 5 are indices of nodes F, D, E, A in the truck route, is colored green. The solid green arrows mean the drone is carried by a truck, and the dashed green arrows mean the drone is flying freely. 46

4.2 An Example of the output of each phase in the algorithm. In phase 1, we assign customers, marked by hollow circles, to their closest depots, marked by solid circles. The assignment is shown in Figure (a). In phase 2, we collect nodes within half of the distance limit for drones as the circles shown in Figure (b) and then apply the Set TSP method to find the shortest tour to visit all sets for each truck group starting from its depot as the colored lines shown in Figure (b). In phase 3, based on the visiting order on routes found in phase 2, we optimize the route for truck and drone together. The final results are shown in Figure (c), where the red and blue lines are routes for trucks of different groups, and the green lines are routes for drones when they are separated from the truck to visit customers. 49

4.3 An example of Set TSP versus TSP. Each set represents a set of nodes in the graph, which can be arbitrarily connected inside. Sets are fully connected via nodes at the boundary. The red arrows represent the optimal TSP tour to visit the center nodes of sets in the graph, and the green arrows represent the optimal Set TSP tour to visit every set. Since Set TSP does not require visiting any exact node, it can pass through the set via **shortcuts** represented in dashed green. As a result, the visiting order of sets on the TSP tour and Set TSP tour can be different. 51

4.4 Ablation for parameters θ_{nn} and θ_{tsp} with congestion level δ. Results are averaged over 100 instances of $|\mathcal{C}| = 50$ and $|\mathcal{P}| = 5$ from Manhattan. To emphasize the relative changes, we normalize the results of each congestion level by the minimal value of all θ and report the gap, i.e. $\textsc{Reported}(\theta, \delta) \leftarrow \textsc{Cost}(\theta, \delta)/\min_\theta \textsc{Cost}(\theta, \delta) - 1$. The values are reported in percentages. . . 61

4.5 Results on scalability. Figure (a) fixes the ratio of expected customers per truck group to visit $|\mathcal{C}|/|\mathcal{P}| = 5$, and figure (b) fixes the number of depots $|\mathcal{P}| = 20$. We set the time limit for each instance to 300 seconds. 62

4.6 The influence of relative speed and distance limit. Time in Fig. (b) is plotted on a logarithm scale. 63

4.7 The distribution of the maximal road-geometry ratio of nodes in the Manhattan and Boston graphs. The maximal road-geometry ratio for a node v is defined as $\max_{u \in S_{r/2}} d^{\text{tr}}(u, v)/d(u, v)$. 64

4.8 Ablation of congestion level and congestion area. We see the congestion area to all nodes within the distance $\textsc{Radius}$ to customers. The $\textsc{Radius}$ is reported in the unit of distance limit for drones r. 64

List of Tables

3.1 Results on 1000 Small Instances of 5 Depots, 30 customers, 20 Stations . . . 36

3.2 Results on 1000 Small Instances of 5 Depots, 30 customers, 20 Stations . . . 36

3.3 Results on 100 Large Instances of 10 Depots, 100 customers, 20 Stations . . 37

3.4 Results on 100 Large Instances of 10 Depots, 100 customers, 20 Stations . . 37

3.5 Ablation study on objective functions when $R_c = 0.25$ 43

4.1 Comparison with Baselines. All results are averaged over 100 instances sampled uniformly from all nodes in the graph. 57

4.2 Ablation of the set-based methods. Results are averaged over 100 instances sampled uniformly from Manhattan. 60

4.3 Ablation of the set-based methods. Use the same datasets as Table 2 but increase the congestion level to 5 for roads within a distance of $0.3r$ to customers. 60

4.4 Cost for different numbers of drones per truck on datasets sampling nodes from uniform distribution and Clusters. 63

Chapter 1

Introduction

To optimize robots handling multiple tasks, a crucial step is to plan the mission with the optimal order of tasks. The Traveling Salesman Problem (TSP), well-studied in a wide range of fields including computer science, operation research, and optimization theory, seeks the shortest route for a salesman to visit a set of *customers* exactly once and return to the starting point. In the multi-agent robot system, the Multiple Depot TSP (MDTSP) [1] is a corresponding variant of the classic TSP that adds multiple *depots*, where multiple salesmen start, to visit a set of customers jointly. Such a problem arises in various real-world robotics applications such as logistics scheduling, warehouse robots, healthcare routing for metropolitan customers, and unmanned aerial vehicles (UAVs) [2]–[4].

However, a gap exists between the theoretical model of MDTSP and its practical applications. For instance, robots or electric vehicles may suffer from limited batteries when conducting tasks; ground vehicles may incorporate aerial robots into the systems to enhance their ability. Such scenarios necessitate the introduction of further constraints into the MDTSP to more accurately model real-world problems, which is the focus of the thesis. In this chapter, we elucidate the specific challenges addressed in this thesis, highlight our contributions, and present an outline of the thesis structure. This groundwork lays the foundation for delving deeper into the optimization strategies that can bridge the gap between

theoretical models and practical applications in multi-agent systems.

1.1 Practical Constraints

In the thesis, we focus on two types of restrictions: the energy limit for the moving agents and cooperation with constrained aerial robots.

1.1.1 Energy Limit

In multi-agent systems, particularly those involving autonomous robots or electric vehicles, energy constraints are a critical factor that significantly influences operational efficiency and effectiveness. The energy limit constraint, which refers to the maximum energy capacity available to an agent for completing its route without replenishment, becomes especially pertinent in scenarios involving extensive travel or multiple task assignments.

The optimal routes given by solving an *unconstrained* MDTSP might not be realizable in practice due to each robot's energy consumption and limited energy capability. For instance, when assigning drones or electric vehicles to deliver items from different warehouses, visiting charging stations must be considered. The charging stations also have limited capabilities regarding the number of agents they can host and the total energy resources they can provide. To better model such real-world requirements, we define a new class of MDTSP called Multi-Agent Energy-Constrained TSP (MA-ECTSP) by introducing the energy and resource constraints into MDTSP. In MA-ECTSP, each salesman starts with a finite energy level and consumes energy proportional to the traveled distance. In addition, we introduce *stations* as new nodes where each salesman can replenish their energy levels. The MA-ECTSP seeks the shortest set of routes for m salesmen that start from different depots, jointly visit a set of customers, and return to the depots where they start. Furthermore, each station has a limited energy supply. Hence, there is an additional constraint on the number of salesmen each station can cater to. It is worth noting that MA-ECTSP is also (NP-)hard as any TSP

can be reduced to an MA-ECTSP with $m = 1$ and zero energy consumption rate.

1.1.2 Cooperation with Aerial Robots

In the rapidly evolving landscape of aerial robotics, specifically drones, the reduction they can contribute to road traffic through aerial package delivery has drawn much attention. However, due to their limited battery capacity and carrying ability, drones are usually considered to accomplish tasks with the help of a ground vehicle. A common scenario is a delivery system where trucks carry drones to send packages to customers cooperatively. The Flying Sidekick Traveling Salesman Problem (FSTSP) [5], which is a variant of the famous TSP problem, models the problem of a drone working in tandem with a delivery truck to visit every customer. The problem captures the difficulty of synchronization between the truck and the drone but oversimplifies the path-finding problem between customers for the truck by an edge in the graph.

To better capture the truck-drone delivery problem in the real world, we consider a multi-agent version of FSTSP over a common road network, termed Multi-Agent FSTSP (MA-FSTSP). The FSTSP considers the TSP problem of one truck loaded with one drone on a graph, consisting of only one node to depart and return called *depot* and a set of nodes to visit called *customers*. The **two differences** in MA-FSTSP setting from FSTSP are: (1) we consider multiple trucks carrying multiple drones instead of one truck with a single drone; (2) we consider the problem on real-world road maps which contain (many) nodes other than depots and customers. Those nodes can be considered as any location or address on the map. In MA-FSTSP, multiple truck groups start from different depots to visit a given set of customers. Each truck group consists of one truck and a fleet of drones, in which the drones are loaded on the truck. Similar to TSP, in MA-FSTSP, each customer must be visited exactly once by one member (a truck or a drone) in one of the truck groups. In addition, the trucks and drones are required to synchronize at the same node on the graph for the takeoff and land actions. While the trucks are constrained to move on the roads

in the road network, the drones can fly freely between any pair of nodes. However, due to the limited battery capacity and carrying ability in practice, each drone is restricted to the maximum distance it can travel before landing on the truck. Additionally, a drone is required to visit its truck after serving each customer. We only allow drones to land on the truck from which it initially took off. Given this setup, the MA-FSTSP asks for the route that minimizes the total time for all truck-drone groups to finish the delivery job (i.e., all customers are visited.) It is worth noting that MA-FSTSP is also NP-hard as any TSP can be reduced to an MA-FSTSP by setting the number of truck groups to 1 and the number of drones in each group to 0.

1.2 Contribution

This thesis presents a three-phase hierarchical framework for handling the two variants, balancing solution quality and computational complexity. In the first phase, we break the multi-agent problem into its single-agent version by assigning customers to each of the salesmen. Then, we solve the sub-problems without considering the constraints to determine the customers' visiting order. Finally, based on the given order, we form the optimal feasible solution under the constraints.

For the MA-ECTSP, we allocate customers to salesmen via a heuristic method involving the Minimum Spanning Tree (MST) of a graph consisting of the customers and the depots as the first phase. Then, in the second phase, we use the Lin-Kernighan heuristic (LKH) [6] TSP solver to determine each salesman's visit order in the assigned customers. Finally, in the third phase, each salesman proposes multiple potentially feasible routes by adding charging stations to their customers' routes, and the best feasible solution is selected using a MILP-based congestion control formulation.

Compared with the exact MILP formulation for MA-ECTSP, our framework uses a much smaller MILP, whose number of both integer and real variables grows linearly to the number

of customers instead of quadratic and fourth order. In experiments, our method outperforms selected baselines in both solution quality and scalability on datasets built from Manhattan, Cambridge, and Massachusetts road maps, as well as existing benchmarks. We observe a 5.22%∼14.84% tour length reduction, a more than 79.8x speedup against the best baseline, and a 12.48% mean optimality gap compared with the exact method. Our framework is capable of solving large-scale instances with up to 1100 customers where the exact method times out on 30 customers with the same time limit.

For MA-FSTSP, we assigned the customers to the salesman starting from their closest depots under a set-based distance metric in the first phase. In the second phase, we extend a TSP to a Set TSP by grouping the road nodes within a given distance to the customers to form a set. The Set TSP asks for a route that visits each set exactly once instead of the requirement of visiting each customer exactly once, which is designed to approximate the usage of drones. The Set TSP is solved via Mixed-Integer Linear Programming (MILP). Finally, we plan for trucks and drones simultaneously to find the optimal routes that visit the customers in the same order as the Set TSP routes found in phase 2.

We compare our method with a column generation-based method [7] and a variable neighborhood search method on a dataset of 1024 nodes, 30 customers, 5 depots, and 5 truck groups with 3 drones in each per instance generated from the Manhattan road network, and a dataset of 11000 nodes, 100 customers, 10 depots, and 10 truck groups with 4 drones in each per instance generated from the Boston road network. Our method produces a 11.67% cost reduction on the dataset from Manhattan with a 46.7 times speedup, and a 14.99% cost reduction on the dataset from Boston with a 24.5 times speedup compared to the baseline methods. We also validate the solution quality with an upper bound approximation and a lower bound approximation for MA-FSTSP and get a less than 2.3 empirical optimality gap. Our method scales up to instances with 600 customers on maps of 10000+ nodes within, at most, 5 minutes of computation time. This is as expected since both the first and third phases use algorithms of polynomial time complexity, and the number of binary variables

in the MILP in the second phase grows quadratic to the number of customers and nodes in sets. We validate the set-based methods by comparing the performance with their node version (i.e., standard nearest-neighbor and TSP) on Manhattan and observe cost savings of up to 4.98% in the common case and cost savings of up to 13.77% when the traffic is bad.

1.3 Structure

The rest of the thesis is organized as follows:

1. Chapter 2 provides a literature review for papers on related topics
2. Chapter 3 introduces the method we model energy constraints
3. Chapter 4 introduces the method we model extra aerial robots in the systems
4. Chapter 5 concludes by summarizing the main points developed in this thesis and pointing to several interesting directions for future work.

Chapter 2

Related Work

This chapter discusses the relevant literature on TSP in multi-agent systems. We begin by discussing methods for solving TSP. Then, we discuss how people deal with multi-agent settings, energy constraints, and truck-drone collaboration in related problems.

2.1 Methods for TSP

2.1.1 Exact methods

Exact methods for TSP and its variants, beyond brute force enumeration, include Dynamic Programming [8] and MILP [9]. Existing tools such as Gurobi [10] and Concorde [11] optimize MILP through the Branch and Bound method and Cutting Plane Method for rapid computation. While exact methods ensure optimality, they are computationally intensive, leading to significant scaling challenges.

2.1.2 Approximation and heuristic algorithms

Approximation and heuristic algorithms are significantly more computationally efficient than exact methods but provide only sub-optimal solutions. Among algorithms with worst-case guarantee, the Christofides Algorithm [12] was the state-of-the-art, offering an approximation

ratio of $\frac{3}{2}$ (defined as the ratio of the algorithms' optimal cost to the *theoretical* optimal cost). The Lin-Kernighan heuristic (LKH) algorithm [6] stands out as the best heuristic algorithm for TSP. It begins with a TSP tour and iteratively removes several edges (with 2 or 3 favored in practice) from the tour, then reconnects the remaining sub-tours to find a tour with a lower cost. Recently, a neural version of LKH, dubbed NeuralLKH [13], has been developed, showing superior performance. Metaheuristic algorithms like Simulated Annealing (SA) are also applicable to solving TSP and are more flexible in adapting to its variants.

2.1.3 End-to-end learning-based methods

have recently attracted attention from researchers due to their good performance. The pioneering neural-based approach to solve TSP utilized the Hopfield network [14], which has recently been improved [15]. Another variant of RNN employed for TSP is the Pointer Network [16]. Recently, Graph Neural Network (GNN) has emerged as an efficient method for addressing TSP, as it learns the combinatorial structure of the graph problem better by capturing the node properties against its graph neighbors [17], [18].

2.2 Multiple TSP

Multiple TSP (MTSP) is the basic problem modeling the multi-agent issue, which asks multiple salesmen starting from the same depot to collaboratively visit a set of cities exactly once and come back to the depot. MDTSP is an extension of it by allowing salesmen to start from different depots. Exact algorithms model the problem into MILP [19] or constraint programming [20], and metaheuristic algorithms include Genetic Algorithm (GA) [21], Ant Colony Optimization (ACO) [22], and Artificial Bee Colony algorithm (ABC) [23].

2.3 Energy Constraints

Electric TSP (ETSP) introduces energy constraints and charging stations into standard TSP. The problem is first formally stated by [24] together with the exact MILP formulation. Previous research on energy constraints came together with a time window, known as the Electric TSP with Time Windows (ETSPTW) [25], which is claimed to be easier [24]. Our work can be viewed as a multi-agent variant of the ETSP problem, where we provide an exact MILP formulation and a scalable hierarchical framework.

2.4 Collaborative Constraints

2.4.1 Multi-Agent Path Finding

The Multi-Agent Path Finding (MAPF) [26], [27] problem is a complex challenge in the field of robotics and artificial intelligence, which asks to find paths for multiple agents like robots or vehicles so that they can move from their starting points to designated goals while satisfying constraints. The problem is NP-hard to solve optimally [28] but is handled well by many effective search algorithms [29]–[32] in practice. While most works seek feasible solutions to avoid collisions between agents, there are some works for cooperation constraints between agents. One kind of work models cooperative tasks by forcing agents to satisfy collaborative constraints on feasible solutions. For example, an extension of MAPF to a pairwise collaboration version [33] asks two collaborative agents to meet at some place simultaneously. Another kind of work puts benefits on cooperation and treats it as an optimization problem. [34] allows drones to ride public transits to save energy and optimizes the route when transits' behaviors are given. MA-FSTSP adapts a similar setting that drones can be carried by truck and asks to optimize truck-drone routes simultaneously [35], which incorporates the task allocation problem and TSP constraints in addition to the path-finding problem.

2.4.2 Trucks and drones delivery problem

In practice, especially in drone-assisted delivery tasks, trucks usually start from and end at the same depot, which makes the problem a Traveling Salesman Problem (TSP) or a Vehicle Routing Problem (VRP). The Flying Sidekick TSP (FSTSP) [5] studies the TSP variant with drones. It asks a truck to carry a drone to visit all customers exactly once, as TSP requires, and the drone to synchronize with the truck when taking off and landing. Based on the problem, more variants are studied for practical applications. One of the directions is generalizing FSTSP to a multi-agent version. The problem is first extended to one truck carrying multiple drones [36]–[38], or multiple trucks with one drone on each [39], [40], and then to multiple trucks with multiple drones [19], [41], [42]. All of them are considering a fully connected graph consisting of customers and depots only or with extra nodes as stations, which does not model the complexity of the environment, i.e., road networks. There are several different ways to enhance the flexibility of the drones, including allowing the drone to synchronize with the truck on arcs [43], relaxing the graph to a 2D plane [44], or to a road network [7], [45] where drones can take off and land at arbitrary nodes. Compared with the above works considering the road networks, our framework is far more scalable while keeping the state-of-the-art solution quality.

Chapter 3

Multi-Agent Energy-Constrained Traveling Salesman Problem

In this chapter, we introduce the implementation of our framework on MA-ECTSP. First, we formally state the problem. Then, we show in detail how each phase is designed. Finally, we demonstrate effectiveness and computational efficiency via experiments.

3.1 Problem Formulation

The MA-ECTSP is a variant of the TSP with multiple levels of constraints. The problem asks to find the shortest tours for a group of salesmen to visit a set of customers so each customer is visited exactly once while satisfying the following constraints: 1) the salesman consumes energy proportional to the distance they travel and can replenish their energy at specific locations called *stations*; 2) The salesmen cannot run out of energy; and 3) Each station can only serve a limited number of salesmen due to the limited resources. To clarify, we use the term *customer* aligning with the term in the FSTSP, which may refer to arbitrary targets or destinations in robotic tasks.

Formally, the problem is defined on a complete, undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}) \coloneqq \mathcal{G}(\mathcal{V})$, where $\mathcal{V}$ represents the set of nodes. The vertex set $\mathcal{V}$ is partitioned into the union of three

sets $\mathcal{D}$, $\mathcal{C}$, and $\mathcal{S}$, where $\mathcal{D} = \{d_1, d_2, \cdots, d_m\}$ denotes the set of m depots (i.e., starting and ending locations of the salesmen's tours), $\mathcal{C} = \{c_1, c_2, \cdots, c_n\}$ is the set of n customers, and $\mathcal{S} = \{s_1, s_2, \cdots, s_l\}$ constitutes the set of l stations. Each edge $(i, j) \in E$ is associated with a weight $c(i, j) \geq 0$, representing the cost of traveling from vertex i to vertex j. The energy and resource constraints are encoded as an energy capacity e_i and an energy consumption k_i per unit distance for each salesman i, along with a resource upper bound r_s for each station $s \in S$. We consider all salesmen homogeneous for simplicity, i.e., $k_i = k$ and $e_i = e$ for all $i = 1, 2, \cdots, m$. The cost of a tour is typically defined as the sum of the costs of the edges in the tour, and the total cost is defined as the sum of the costs of all tours. This cost may be interpreted as distance, time, or other pertinent metric. Furthermore, each salesman's energy level is presumed to be fully replenished upon visiting any station.

Problem 1. ***(MA-ECTSP)*** *Given a complete graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} = \mathcal{D} \cup \mathcal{C} \cup \mathcal{S}$ and m salesmen starting from different depots in $\mathcal{D}$, find a set of m tours $\{t_i\}_{i=1}^{|\mathcal{D}|}$, one per salesman, such that: (1) each tour begins and ends at the same depot; (2) each customer in $\mathcal{C}$ is visited exactly once; (3) each salesman maintains a nonnegative energy level throughout the tour; (4) each station s_i is visited at most r_{s_i} times in total for $i = 1, 2, \cdots, l$; and (5) the total cost is minimized.*

3.2 Methodology

Like the original TSP, MA-ECTSP can also be formulated as a MILP (see Appendix A). However, the complexity of such a MILP scales exponentially with the number of customers to the fourth power, the number of depots to the third power, and the number of stations to the second power. Thus, this approach does not scale well for problems involving a large number of depots, customers, or stations.

The intuition of our framework is straightforward: by decomposing the MA-ECTSP into smaller subproblems, we aim to reduce the size of the MILP, the primary bottleneck for

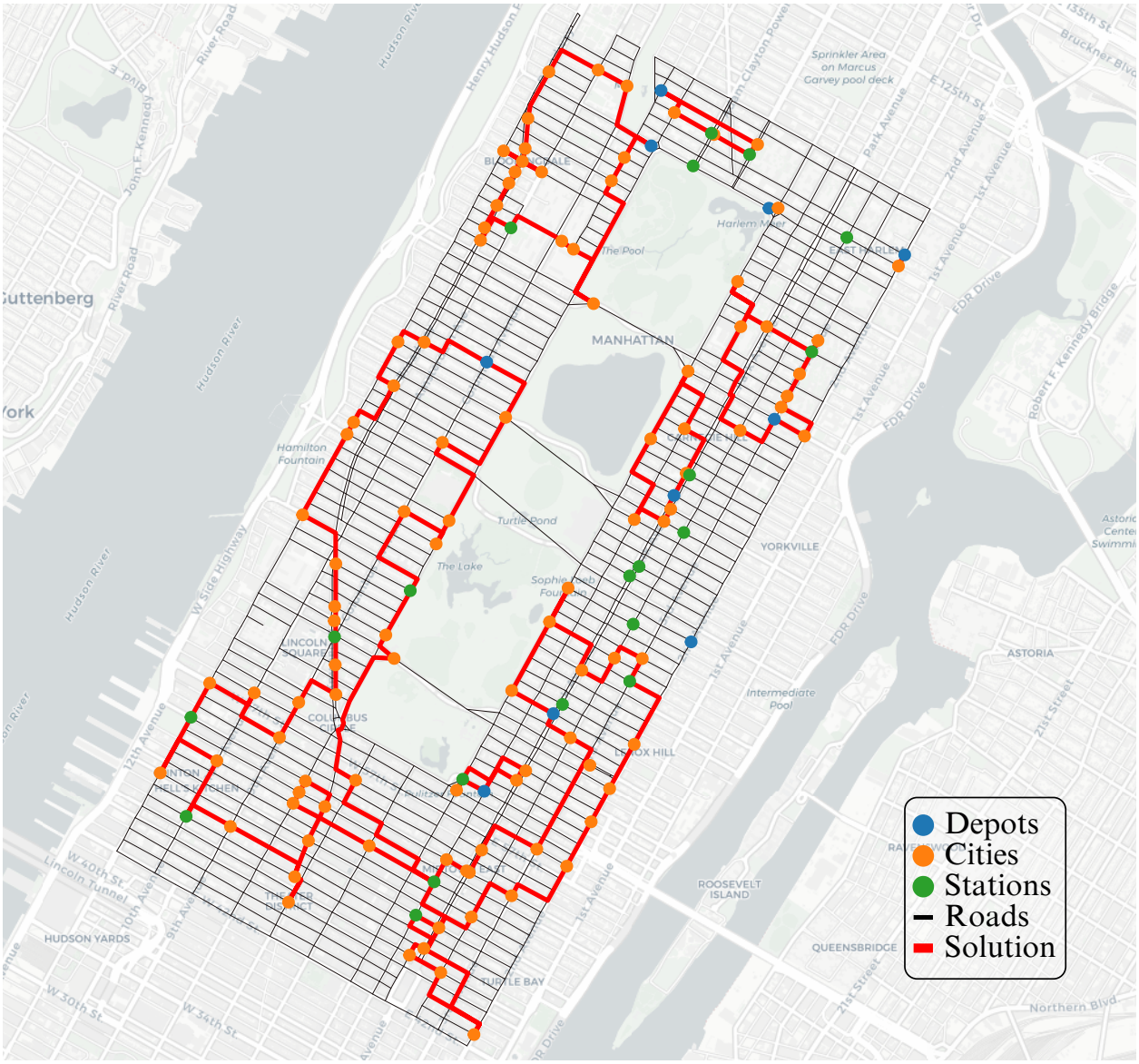


Figure 3.1: A solution of MA-ECTSP in Manhattan: salesmen start from depots to collaboratively visit all customers and always keep their energy above zero.

scaling. To this end, we propose a novel hierarchical framework that utilizes a heuristic and smaller MILP, shown in Alg. 1. Initially, customers are allocated to salesmen (line 3) to form standard TSPs for each (line 5). Then, each TSP is solved without the energy constraints to return a potential tour t_i for salesman i (line 6). For each pair of consecutive customers (u, v) in t_i, we suggest the top-k shortest paths to v from u by passing through a sequence of stations to maintain a positive energy level (lines 7-10). Finally, we collect all paths proposed

Algorithm 1 Framework for solving MA-ECTSP

1: $\mathcal{T}$, $\mathcal{P}$, SOLN $= \emptyset$
2: $l = 0$
3: $\{\mathcal{C}_i\}_{i=1}^{|\mathcal{D}|} = \text{Partition}(\mathcal{C}; \mathcal{D})$ ▷ Assign customers to salemen
4: **for** $i \in \{1, 2, \cdots, m\}$ **do**
5: $\quad \mathcal{G}_i = \mathcal{G}(\mathcal{C}_i \cup \{d_i\})$ ▷ Form a complete graph of $\mathcal{C}_i \cup \{d_i\}$
6: $\quad t_i = \text{TSP}(\mathcal{G}_i)$ ▷ Find the TSP solution of graph $\mathcal{G}_i$
7: $\quad$ **for** $(u, v) \in t_i$ **do**
8: $\quad\quad \mathcal{G}_i' = \mathcal{G}(\{u, v\} \cup \mathcal{S})$ ▷ Form a complete graph of $\{u, v\} \cup \mathcal{S}$
9: $\quad\quad \mathcal{P}_i(u, v) = \text{k-shortest-path}(u, v; \mathcal{G}_i'; k)$ ▷ Find the shortest k paths from u to v
10: $\quad$ **end for**
11: **end for**
12: $\mathcal{P} = \cup_{i=1}^{|D|} \mathcal{P}_i$
13: SOLN $= \text{CongestionControl}(P)$ ▷ Form a solution satisfying all constraints from P
14: **return** SOLN

by all salesmen (line 12) to find a feasible tour for each salesman so that all the constraints, including positive energy level and limited station resources, are satisfied (line 13). Fig. 3.1 shows an illustrative example of route planning in Manhattan.

3.2.1 Customer assignment

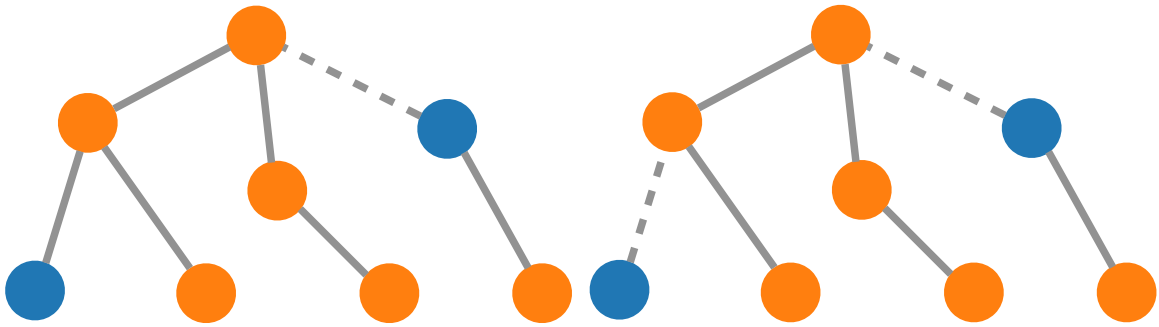
Algorithm 2 Assign Customers to Salesmen

Input: customer set $\mathcal{C}$, depot set $\mathcal{D}$

1: $\mathcal{G} = \mathcal{G}(\mathcal{C} \cup \mathcal{D})$ ▷ Form a complete graph of $\mathcal{C} \cup \mathcal{D}$
2: $\mathcal{T} = \text{MST}(\mathcal{G})$ ▷ Compute a minimum spanning tree
3: Compute minimum weight functions f, g ▷ Using Eq. (3.1), (3.2)
4: $\cup_{i=1}^{|\mathcal{D}|} \mathcal{T}_i = \text{Part}(\mathcal{T}, f, g)$ ▷ Partition tree $\mathcal{T}$ based on f, g
5: **return** $\{\mathcal{C} \cap \mathcal{T}_i\}_{i=1}^{|\mathcal{D}|}$

We first introduce Alg. 2, the customer assignment component in the framework.

To begin with, we construct a complete graph $\mathcal{G} = \mathcal{G}(\mathcal{V})$ with $\mathcal{V} = \mathcal{D} \cup \mathcal{C}$ of all depots $\mathcal{D}$ and customers $\mathcal{C}$ and find its minimum spanning tree $\mathcal{T}(\texttt{rt})$ rooted at some node $\texttt{rt} \in \mathcal{D} \cup \mathcal{C}$. Then, we split $\mathcal{T}(\texttt{rt})$ into m components $\mathcal{T}_i$ by deleting $m - 1$ edges to separate every pair of depots and minimizing the remaining edges' weights using dynamic programming, as explained next. Note that each resulting partition $\mathcal{T}_i$ contains exactly one depot d_i.



(a) Example of partition $\tilde{T}$ (b) Example of partition $\hat{T}$

Figure 3.2: Illustration of partitions. The orange nodes represent customers, and the blue nodes represent depots. Solid lines indicate the edges remaining after partitioning, and dashed lines represent the edges removed after the partition. In Fig.(a), we disconnect one depot from the root node to form two subtrees with one depot in each. In Fig.(b), we disconnect both depots from the root to form 3 subtrees, each with a depot except the subtree containing the root.

Given a rooted tree $\mathcal{T}(u)$ with root node u and m_u depots inside, we define two types of partitions $\tilde{\mathcal{T}}(u)$ and $\hat{\mathcal{T}}(u)$. Partition $\tilde{\mathcal{T}}(u)$ divides the tree $\mathcal{T}(u)$ into m_u subtree(s) such that each subtree contains exactly one depot. If $m_u = 0$, i.e. $\mathcal{T}(u)$ contains no depot, then partition $\tilde{\mathcal{T}}(u)$ does not exist. Partition $\hat{\mathcal{T}}(u)$ divides tree $\mathcal{T}(u)$ into $m_u + 1$ subtrees, where the subtree containing the root node u has no depots inside, and the rest m_u subtrees contain exactly one depot each. If root node u is a depot, partition $\hat{\mathcal{T}}(u)$ does not exist. Fig. 3.2 shows an illustration of two partitions.

Next, define $f : \mathcal{V} \to \mathbb{R}$ such that $f(u)$ represents the minimum total edge weights of $\tilde{\mathcal{T}}(u)$ and $g : \mathcal{V} \to \mathbb{R}$ such that $g(u)$ represents the minimum total edge weights of $\hat{\mathcal{T}}(u)$. If a partition does not exist, we set the corresponding value of f or g to be $+\infty$, i.e., $f(u) = +\infty$ for $T(u)$ contains no depot and $g(u) = +\infty$ if u is a depot. Let $\mathcal{H}(u)$ be the set of children

of node u. Functions f and g can be computed as:

$$f(u) = \begin{cases} \min\limits_{v \in \mathcal{H}(u)} \Big\{ f(v) + c(u,v) + \sum\limits_{v' \in \mathcal{H}(u) \setminus \{v\}} \min\{f(v'), g(v') + c(u,v')\} \Big\}, & u \in \mathcal{C}, \\ \sum\limits_{v \in \mathcal{H}(u)} \min\{f(v), g(v) + c(u,v)\}, & u \in \mathcal{D}, \end{cases} \tag{3.1}$$

$$g(u) = \begin{cases} \sum\limits_{v \in \mathcal{H}(u)} \min\{f(v), g(v) + c(u,v)\}, & u \in \mathcal{C}, \\ +\infty, & u \in \mathcal{D}. \end{cases} \tag{3.2}$$

We offer brief insights here. For a partition $\tilde{\mathcal{T}}(u)$ or $\hat{\mathcal{T}}(u)$, all partitions on its subtrees are also of type $\tilde{\mathcal{T}}$ or $\hat{\mathcal{T}}$. Computing weights for such a partition involves computing the optimal connection from u to its children and weights for corresponding partitions on those subtrees. Using this, we can obtain (3.1)-(3.2).

We can construct the optimal partition $\tilde{\mathcal{T}}(\texttt{rt})$ based on the functions f and g recursively from root to leaves by the following rules. Suppose the partition type of $\mathcal{T}(u)$ is known. For subtree $\mathcal{T}(v)$, $v \in \mathcal{H}(u)$: (1) If $\mathcal{T}(u)$ is partitioned to $\tilde{\mathcal{T}}(u)$, $u \in \mathcal{C}$, and v is the minimizer in Eq. 3.1. $\mathcal{T}(v)$ is partitioned into $\tilde{\mathcal{T}}(v)$ and u, v is connected. (2) Else, $\mathcal{T}(v)$ is partitioned into $\tilde{\mathcal{T}}(v)$ and disconnect to u if $f(v) < g(v) + c(u,v)$, otherwise $\mathcal{T}(v)$ is partitioned into $\hat{\mathcal{T}}(v)$ and (u,v) is connected. We show the correctness and optimality of Alg. 2's output.

Theorem 1. *Given a graph $\mathcal{G} = \mathcal{G}(\mathcal{V})$ with $\mathcal{V} = \mathcal{D} \cup \mathcal{C}$ and an edge-weight function c, the partition $\tilde{\mathcal{T}}(\texttt{rt})$ recovered from value function assignment in (3.1)-(3.2) partitions the minimum spanning tree $\mathcal{T}$ of $\mathcal{G}$ into m subtrees $\{\mathcal{T}_i\}_{i=1}^m$ such that $d_i \in \mathcal{T}_i$, and minimizes the total edge weights in the connected components, i.e., $\sum_{i=1}^{|\mathcal{D}|} \sum_{(u,v) \in \mathcal{T}_i} c(u,v)$.*

Theorem 2. *Alg. 2 for MDTSP has an approximation ratio of 2 for nodes on a 2D plane.*

The proofs of Thm. 1 and Thm. 2 are in Appendix B.

After assigning customers to salesmen, we form graphs for each salesman with corresponding depot and customers and solve TSPs to determine the order of visits for each

salesman. Since we are solving a standard TSP at this stage, any off-the-shelf TSP solver can be plugged in here. Because the solver is called repeatedly, once for each salesman, and the routes have no restriction on the size, it is desirable to use a TSP solver that is both fast and scalable. In this framework, we use the state-of-the-art heuristic solver LKH-3 [46], which efficiently produces solutions with a very small optimality gap and has good scalability.

Next, given a tour of customers, each salesman proposes k paths between each edge along its tour by applying the shortest-path algorithm on the graph consisting of the edge and the stations. That is, each salesman i with a tour t_i proposes $k|t_i|$ paths in total, where $|t_i|$ is the length of the tour t_i. The total number of paths is $\sum_{i=1}^{|\mathcal{D}|} k|t_i| = k(|\mathcal{C}| + |\mathcal{D}|)$, so there are $\prod_{i=1}^{|\mathcal{D}|} \prod_{j=1}^{|t_i|} k = k^{|\mathcal{C}|+|\mathcal{D}|}$ potential solutions. The next step is to solve the *congestion control* problem to find a solution for each salesman that satisfies all the energy constraints.

3.2.2 Congestion control

We say congestion happens at station s when more than r_s salesmen want to visit the station. The congestion control problem aims to plan salesmen tours to avoid congestion at any station. To this end, a salesman i selects a path from the k proposed paths for each edge (u, v) in each tour t_i to satisfy energy requirements for all salesmen and station resource limits. We set this as an optimization problem to find the path assignment that minimizes the total edge weights while satisfying the constraints.

Let $\beta_{i,j,h} \in \{0, 1\}$ be a binary variable indicating whether $p_{i,j,h}$, i.e. h-th path proposed by salesman i for its j-th edge, is chosen and $c_{i,j,h}$ be the corresponding cost. If there is at least one station on the path, we denote the minimum energy needed to arrive at the first station from the j-th node as $q_{1,i,j,h}$ and the maximum energy left after finishing the path (i.e., arriving at the $(j + 1)$-th node) as $q_{2,i,j,h}$. Let $\gamma_{i,j} \in \mathbb{R}_+ \cup \{0\}$ be the energy level of salesman i at the j-th node in tour t_i. Thus, $\beta_{i,j,h} = 1$ is feasible only if corresponding energy constraints are satisfied, i.e., $\gamma_{i,j} \geq q_{1,i,j,h}$ and $\gamma_{i,j+1} \leq q_{2,i,j,h}$. In the case when a salesman visits a station between the j-th and $(j + 1)$-th node, the energy level $\gamma_{i,j+1}$ is independent

of $\gamma_{i,j}$ because the station charges the salesman's energy to its full capacity. On the other hand, if there is no station on the path, we denote the minimum energy needed to travel the path $p_{i,j,h}$ by $q_{3,i,j,h}$. In this case, $\beta_{i,j,h} = 1$ is feasible only if $\gamma_{i,j} - \gamma_{i,j+1} \geq q_{3,i,j,h}$, which means that the energy level $\gamma_{i,j+1}$ depends on the previous energy level $\gamma_{i,j}$. Additionally, we define $q_{3,(\cdot)} = -\infty$ for paths with stations and $q_{1,(\cdot)} = -\infty$, $q_{2,(\cdot)} = +\infty$ for paths without stations, so that the tuple $q_{(\cdot)} = (q_{1,(\cdot)}, q_{2,(\cdot)}, q_{3,(\cdot)})$ is well-defined for each edge. Based on these definitions, the congestion-free tour assignment problem can be posed as:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{|t_i|} \sum_{h=1}^{k} c_{i,j,h} \cdot \beta_{i,j,h}, \tag{3.3a}$$

$$\text{s.t. } \beta_{i,j,h} \in \{0,1\}, i \in [m], j \in [|t_i|], h \in [k], \tag{3.3b}$$

$$\sum_{h=1}^{k} \beta_{i,j,h} = 1, i \in [m], j \in [|t_i|], \tag{3.3c}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{|t_i|} \sum_{h=1}^{k} \beta_{i,j,h} \cdot p_{i,j,h}[s] \leq r_s, s \in [l], \tag{3.3d}$$

$$0 \leq \gamma_{i,j}, i \in [m], j \in [|t_i| + 1], \tag{3.3e}$$

$$\sum_{h=1}^{k} \beta_{i,j,h} \cdot q_{1,i,j,h} \leq \gamma_{i,j}, i \in [m], j \in [|t_i|], \tag{3.3f}$$

$$\gamma_{i,j} \leq \sum_{h=1}^{k} \beta_{i,j,h} \cdot q_{2,i,j-1,h}, i \in [m], j \in [2, |t_i| + 1], \tag{3.3g}$$

$$\sum_{h=1}^{k} \beta_{i,j,h} \cdot q_{3,i,j,h} \leq \gamma_{i,j} - \gamma_{i,j+1}, i \in [m], j \in [|t_i|], \tag{3.3h}$$

where $|t_i|$ refers to the length of salesman i's tour t_i. The objective (3.3a) is the overall cost to minimize. Constraints (3.3b) and (3.3c) ask to choose exactly one path out of k proposed paths for each edge. Constraint (3.3d) is the station's resource limit. Constraints (3.3e)-(3.3h) are the energy constraints for salesmen when passing through the chosen paths.

Now, we analyze the complexity of our proposed MA-ECTSP solver. Given a graph $\mathcal{G} = \mathcal{G}(\mathcal{V})$ with $|\mathcal{V}| = |\mathcal{D} \cup \mathcal{C} \cup \mathcal{S}| = m + n + l$ and parameter k denoting the number

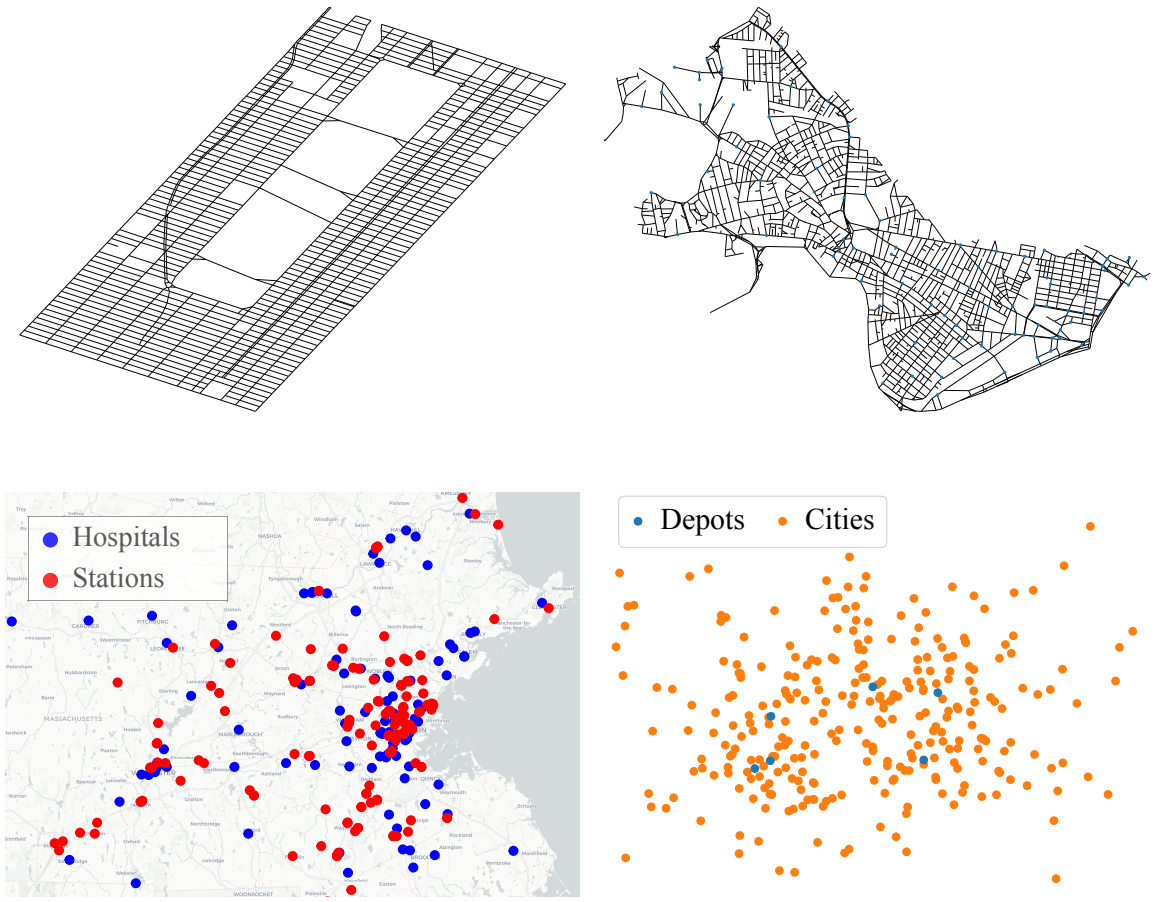


Figure 3.3: Four Datasets for experiments. (a) The driving road map of Manhattan, New York. (b) The driving road map of Cambridge, Massachusetts (c) Hospitals and Tesla's Supercharger Stations in Massachusetts. (d) The existing benchmark for MDVRP [47]

of paths proposed for each edge, the MILP (3.3) has nk real variables, $mk + nk$ integer variables and $5m^2 + 5mn + l$ constraints. Compared with the naive MILP formulation, which has $mn + mn^2 + mn^2(m + n + l)^2$ real variables, $mn^2 + n$ integer variables, and $m(m + n)^2(l^2 + 6m + 5n + 7l + 3)$ constraints, our algorithm has better scalability, which is also validated via experiments presented in the next section.

3.3 Experiments

We empirically validate our method in this section. In section 3.3.1, we compare our method with three baselines: Ant Colony Optimization (ACO) [48], Hybrid Evolutionary Algorithm

Table 3.1: Results on 1000 Small Instances of 5 Depots, 30 customers, 20 Stations

	Manhattan			Cambridge		
Method	Length	Feasible Rate	Time(s)	Length	Feasible Rate	Time(s)
ACO	32.93	0.95	**1.02**	395.75	0.92	**1.04**
HEA	31.65	0.62	21.57	380.12	0.63	21.62
HVNS	28.26	**1.00**	12.95	338.86	**1.00**	14.80
Ours	**24.96**	**1.00**	1.90	**313.45**	**1.00**	1.91

Table 3.2: Results on 1000 Small Instances of 5 Depots, 30 customers, 20 Stations

	Massachusetts			MDVRP Benchmark		
Method	Length	Feasible Rate	Time(s)	Length	Feasible Rate	Time(s)
ACO	830.13	**0.99**	**1.05**	907.83	0.84	**1.26**
HEA	848.34	0.97	10.68	909.78	0.43	14.45
HVNS	748.50	**0.99**	9.71	843.66	**1.00**	15.25
Ours	**703.82**	**0.99**	1.89	**783.06**	0.91	2.04

(HEA) [49], and Hybrid Variable Neighborhood Search (HVNS) [50], on real-world maps and existing benchmarks. In section 3.3.2, we compare our method with a naïve MILP formulation for the scalability and solution quality. In section 3.3.3, we study our framework's partition and TSP solver components to justify our choices. In section 3.3.4, we explore the effectiveness of resource distribution and provide a sufficient condition for the existence of feasible solutions based on the experiments. Finally, in section 3.3.5, we test the influence of the replenishment time.

All experiments were run on a server with 1 AMD Ryzen Threadripper 3990X 64-Core Processor and 4 Nvidia RTX A4000 GPUs. Gurobi 10.0.0 [10] served as the MILP solver.

We set the iterations of LKH to 10 and the number of paths proposed per edge by each salesman $k = 5$. The selection of k is empirical; beyond 5, increasing k extends running time without enhancing performance. For baselines, we adapt the parameters from their papers [48]–[50].

Table 3.3: Results on 100 Large Instances of 10 Depots, 100 customers, 20 Stations

	Manhattan			Cambridge		
Method	Length	Feasible Rate	Time(s)	Length	Feasible Rate	Time(s)
ACO	65.50	0.34	3.63	881.50	0.14	4.45
HEA	50.61	0.01	420.62	824.00	0.01	473.21
HVNS	47.70	0.94	270.19	653.54	0.48	300.75
Ours	**41.74**	**0.98**	**2.54**	**619.43**	**0.56**	**3.11**

Table 3.4: Results on 100 Large Instances of 10 Depots, 100 customers, 20 Stations

	Massachusetts			MDVRP Benchmark		
Method	Length	Feasible Rate	Time(s)	Length	Feasible Rate	Time(s)
ACO	1685.88	0.99	4.16	2156.11	0.41	4.03
HEA	1619.08	0.95	310.27	1931.04	0.09	329.39
HVNS	1373.35	0.99	212.72	1642.98	0.67	287.45
Ours	**1228.60**	**0.99**	**2.78**	**1511.19**	**0.70**	**2.32**

3.3.1 Comparison with Baselines

Datasets: We present experiments on four datasets from real-world maps and existing benchmarks. We use the driving road map of Manhattan [51], Cambridge, and Massachusetts [52]. For the Manhattan map, instances are generated by uniformly sampling depots, customers, and stations from the map. For the Cambridge map, stations are uniformly sampled from Bluebikes stations in 2023 [53], and depots and customers are uniformly sampled from the rest of the map. For the Massachusetts map, stations are uniformly sampled from Tesla's Supercharger stations, and depots and customers are uniformly sampled from hospitals. We adapt the existing MDVRP benchmarks [47] by randomly turning a fraction of customers into stations and uniformly sampling a fixed amount of depots, customers, and stations. Salesmen are restricted to traveling along roads in instances from maps and freely in the 2D space in instances from benchmarks. For each data source, we generate 1000 small instances consisting of depots $|\mathcal{D}| = 5$, customers $|\mathcal{C}| = 30$, and stations $|\mathcal{S}| = 20$, and 100 large instances consisting of $|\mathcal{D}| = 10$, $|\mathcal{C}| = 100$, $|\mathcal{S}| = 20$. We set the resource limit for

each station to be $r = 2$. Due to the different map sizes, the energy capacity of the salesman is set differently, i.e., 4 in Manhattan, 40 in Cambridge, 400 in Massachusetts, and 4 in data from the benchmark.

Metrics: We assess solvers based on total tour length, feasibility rate, and running time. The tour length is defined as the sum of path lengths salesmen traveled in a solution. The feasibility rate is defined as the ratio of feasible solutions found for instances in a dataset. The running time is the duration from formatted data being fed into the solver to the solver outputting the best solution. We report the average tour length and average running time.

Baselines: We use ACO, HEA, and HVNS as baselines. The ACO-based algorithm uses the nearest neighbor partition to split the multi-depot problem into single-depot sub-problems, with ACO employed to find local optima. HEA starts with the nearest neighbor population, iteratively creates offspring by adding minimal incremental density routes from parents, and enhances the solution with variable neighborhood search.HEA was originally designed for MDVRP, so we added the station insertion procedure from the ACO baseline to adapt it to MA-ECTSP. HVNS initializes the solution using a variable neighborhood search in each iteration and searches for the best solution by tabu search. The three baselines represent the main approaches for related problems and are state-of-the-art methods in their categories.

Result: The comparison results are shown in Table 3.1-3.4. Our method effectively produces the best solution quality in all test cases. When the problem size is small, our method's feasibility rate and running time are very close to the best baseline. On test cases with large problem sizes, our method outperforms all other baselines in all evaluation metrics, which also shows good scalability.

3.3.2 Compare with Exact Algorithm

Our framework's solution quality and running time trade-offs are assessed by comparison with an exact algorithm.

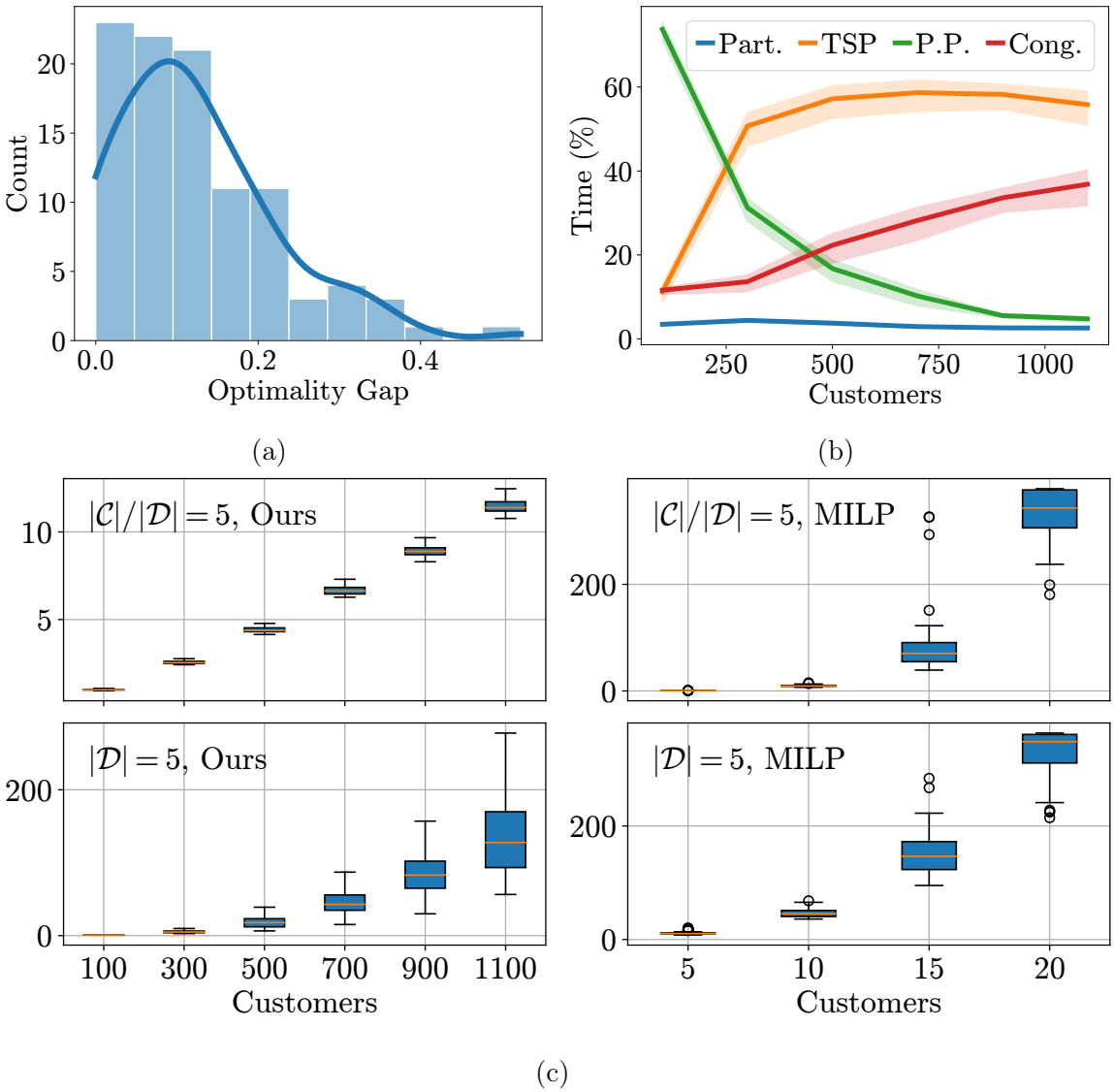


Figure 3.4: Results on comparison experiments. (a) The distribution of optimality gaps with a mean value of 12.48%. (b) The change in the percentage of running time for each part in the framework. (c) Comparison on Scalability. All Y-axes are time in seconds.

Datasets: To measure the solution quality, we randomly generate 100 instances of $|\mathcal{C}| = 15$, $|\mathcal{D}| = 3$, $|\mathcal{S}| = 20$, and $r = 2$ in the unit square $[0, 1]^2$. To measure the scalability of our framework, we vary the size of instances to 100 instances per size. The sizes of problems increase in two ways: (1) fix the number of stations $|\mathcal{S}| = 20$ and the number of salesmen $|\mathcal{D}| = 5$, the number of customers $|\mathcal{C}|$ varies from 100 to 1100 for our framework and from 5 to 20 for the MILP solver; (2) fix the number of stations $|\mathcal{S}| = 20$ and the average customers visited by each salesman, i.e., $|\mathcal{C}|/|\mathcal{D}| = 5$, the number of salesmen $|\mathcal{D}|$ varies from 20 to 220 for our framework and $|\mathcal{D}|$ from 1 to 5 for the MILP solver. To ensure the feasibility rate, we empirically set $r = 0.4 \cdot |\mathcal{C}|/|\mathcal{S}|$.

Baseline: The baseline we use is a naïve MILP.

Metrics: The solution quality is measured by the gap between our tour length and the optimal one. The running time is the same metric as in Sec. 3.3.1.

Results: Our framework achieves a mean optimal gap of 12.48% and worst gap of 52.34% on the dataset with an average 41.7 times speedup, where the distribution is shown in Figure 3.4a. We believe this is a good performance as the worst-case guarantee for standard TSP is around 50%. Figure 3.4c demonstrates the trends of the increment in running time, where our framework shows much better scalability than the MILP solver. Even with more than 1000 customers, the running time of our framework is still acceptably low. We can conclude that our framework achieves good suboptimality and tremendously reduces the running time to scale up to a large problem size.

Comparing the first column in Figure 3.4c, the running time is shorter when there are more salesmen given the same number of customers. We empirically evaluate the running time of different components in our framework. Figure 3.4b shows the changes in relative time consumption for subroutines in the entire algorithm as the number of customers grows from 100 to 1100, while the number of depots is fixed at 5. The TSP solver and the MILP in congestion control take up most of the computation time, which explains the negative correlation between running time and the number of salesmen. Given more salesmen, the

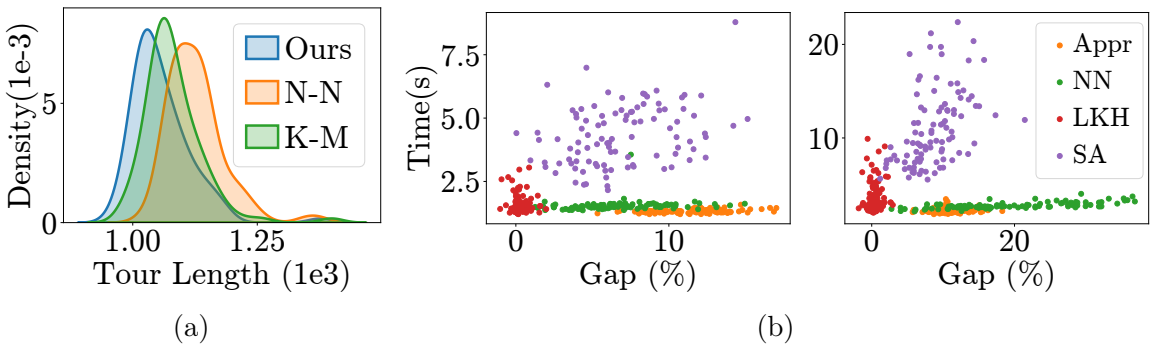


Figure 3.5: Results of studies on components. (a) Keep the TSP solver to be LKH, our partition algorithm outperforms all baselines; (b) keep the partition algorithm MST-based, and the LKH solver produces the best solution quality and competitive computation time.

running time of the partition increases slightly, but the average size of TSP for salesmen decreases, dramatically reducing the overall running time.

3.3.3 Studies of Components

In this section, we validate the effectiveness of our partition algorithm and TSP solver by comparing them with several other potential plugin algorithms.

Datasets: We conduct experiments on 100 randomly generated instances of $|\mathcal{D}| = 5$, $|\mathcal{C}| = 150$, $|\mathcal{S}| = 20$, and 100 randomly generated instances of $|\mathcal{D}| = 5$, $|\mathcal{C}| = 250$, $|\mathcal{S}| = 20$ in the unit square. We set $r = 3$ in both cases.

Metrics: For partition algorithms, we evaluate the algorithms by tour length only since all of them take only polynomial time. We evaluate TSP solvers by tour length, measured by the gap between using a MILP TSP solver and themselves, and the running time.

Baselines: For the partition algorithm, we choose the Nearest Neighbor (N-N) algorithm, which assigns each customer to its closest salesman, and the K-Means clustering algorithm, which first clusters the customers into several groups and then assigns each group to the salesman closest to its cluster center, as baselines. For TSP solvers, we choose the representatives of main approaches, including neural-based solver [17], approximation solver (Christofides algorithm), heuristic solver (LKH), and metaheuristic solver (SA).

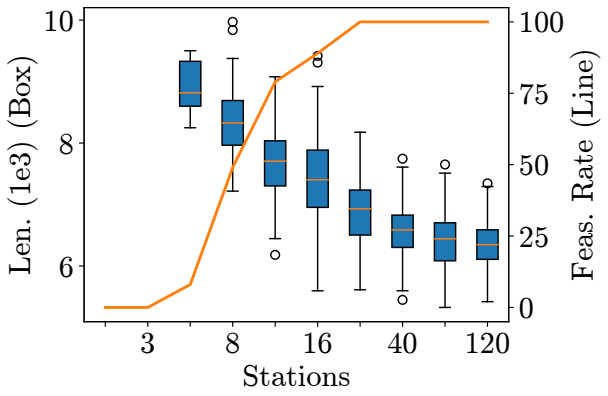


Figure 3.6: Effect of the dispersion of stations. There are no feasible solutions for cases with 1 or 3 station(s).

Results: Results in Figure 3.5a show that our partition method beats both NN and KMeans. The result in the left figure of Figure 3.5b shows that LKH gives much smaller gaps than others within acceptable running time. The right one of Figure 3.5b shows that the LKH solver still produces the best solution with a small running time on large cases, while the neural-based solver has an enormous drop in its solution quality due to the out-of-distribution issue.

Again, we want to emphasize that our framework is adaptable to arbitrary partition algorithms and TSP solvers, allowing for further performance improvements as these components evolve.

3.3.4 Effectiveness of Resource Distribution

In this subsection, we investigate the effectiveness of the resource distribution given the total resources and problem size to guide the station setup.

Datasets: We randomly generate 100 instances in the unit square of $|\mathcal{D}| = 30$, $|\mathcal{C}| = 1200$. Then, we vary $|\mathcal{S}|$ from $1 \sim 120$ and distribute them uniformly in the square, keeping the total amount of resources $r \cdot |\mathcal{S}| = 240$.

Result: As shown in Figure 3.6, adding more stations increases feasibility and shortens tour length, indicating that spreading resources more widely eases their use. Based on the

Table 3.5: Ablation study on objective functions when $R_c = 0.25$

	Optimize Length		Optimize Time		
	Length	Time	Length	Time	Similarity
Manhattan	16.56	19.70	16.64	19.43	0.75
Cambridge	197.73	237.16	199.22	232.38	0.70
Massachusetts	474.70	493.375	476.92	483.11	0.74

observation, we present a sufficient condition for resource density to ensure feasible solutions.

Theorem 3. *Gridding the map with length* $\frac{e}{(1+\sqrt{5})k}$ *into even squares. Suppose in grid* i*, the number of stations is* $N_s[i]$ *and the number of customers is* $N_c[i]$*, then there exists a feasible solution if* $N_s[i] \geq 1$ *and* $N_c[i]/N_s[i] \leq r$ *for all* i*.*

The proof is in Appendix B.3. In practice, the cost of setting up stations needs to be balanced against their benefits for optimal resource allocation.

3.3.5 Effect of replenishment time

In this section, we show that the replenishment time in the objective function has minimal impact on the final solution.

Method: We introduce the parameter $R_c = \frac{w}{k}$, i.e., the rate between consuming energy and replenishing energy, into the naïve MILP to consider the replenishment time. We generate solutions under $R_c = +\infty$, i.e., optimizing tour length without accounting for replenishment time, and $R_c = 0.25$ based on the speed of 180kW charging stations for each instance.

Datasets: We build datasets from those in Sec. 3.3.1 by randomly sampling 2 depots out of all depots and 12 customers out of all customers from each instance to avoid the explosion of solving time. The sizes of the dataset are the same.

Results: The results are shown in 3.5. The length in the table represents the total distance salesmen need to travel, and the time in the table represents the total time salesmen need to spend given $R_c = 0.25$. Optimizing the length, equivalent to not considering the

replenishment time as our previous setting, can be done by **cheating the MILP of** $R_c = +\infty$. The similarity metric, representing the fraction of cases with identical solutions under varying objectives, shows over 70% consistency in solutions whether or not the replenishment time is considered, and very small gaps in the remaining cases.

Chapter 4

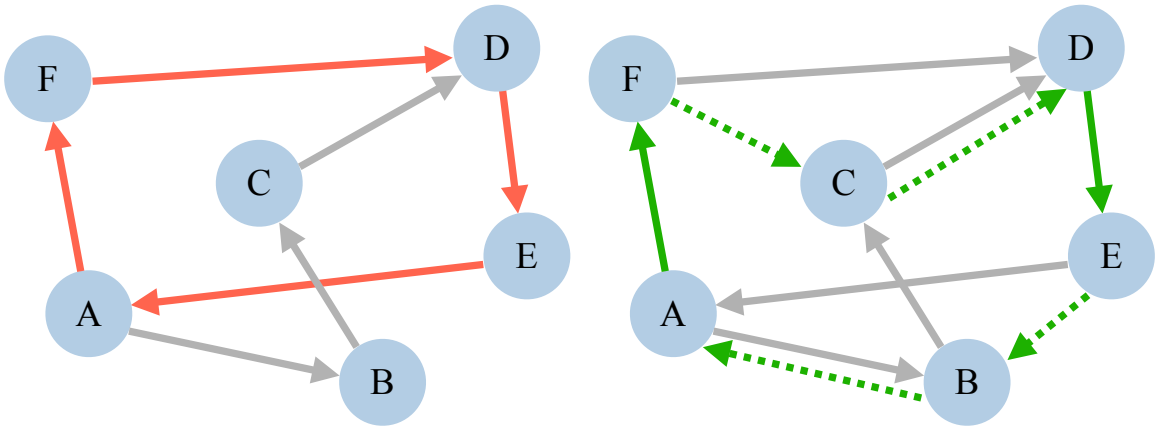
Multi-Agent Flying Sidekick Traveling Salesman Problem

In this chapter, we introduce the implementation of our framework on MA-FSTSP. Following the same structure, we formally state the problem and then show how each phase is designed. Finally, we demonstrate effectiveness and computational efficiency via experiments.

4.1 Problem Formulation

The MA-FSTSP is a multi-agent generalization of FSTSP in both trucks and drones. The problem asks us to find the fastest tours for each truck group to visit a set of customers so that each customer is visited exactly once while satisfying the following constraints: (1) trucks can only move along roads, while drones can fly directly between any locations after leaving the truck; (2) drones can only fly a limited distance before landing back to its truck due to limited battery capacity; (3) drones can be dispatched from the truck and collected by the same truck in a different location; (4) takeoff and land actions only happen at the ends of the road for simplicity; (5) drones can only visit one customer before back to the truck due to the limited payload capacity.

We plan for a fleet of agents comprising m truck groups, denoted as $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_m\}$.



(a) Truck Route (red) (b) Drone Route (green)

Figure 4.1: An example of the truck and drone routes on a directed graph with node 1∼6. The solid arrows (in all colors) represent the road that the truck must follow. The truck route (A, F, D, E, A) is colored red and the drone route (A, F, C, D, E, B, A), represented as $\langle(\text{A, F, D, E, A}), \{(2, \text{C}, 3), (5, \text{B}, 6)\}\rangle$, where 2, 3, 4, 5 are indices of nodes F, D, E, A in the truck route, is colored green. The solid green arrows mean the drone is carried by a truck, and the dashed green arrows mean the drone is flying freely.

Each truck group $\mathcal{T}_i$ consists of a truck t_i carrying a group of k_i drones $\mathcal{D}_i = \{d_1^{(i)}, d_2^{(i)}, \cdots, d_{k_i}^{(i)}\}$. For simplicity, we consider $k_i = k \geq 1$ for all truck groups in this work. The agents operate on a shared road network, represented by a strongly connected directed graph [1] $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of nodes (including depots nodes and customers nodes defined later) and $\mathcal{E}$ is the set of edges.

Each pair of nodes $u, v \in \mathcal{V}$ is associated with a distance $d(u, v) \geq 0$, which may represent the Euclidean distance on a planar map or geometric distance on Earth. Trucks are moving at a constant speed $s_{\text{tr}} > 0$ discounted by the congestion level $\delta(e)$ on each edge $e \in \mathcal{E}$. The congestion level accounts for practical traffic conditions such as traffic jams or varying speed limits, with the travel time across edge e being calculated as $d(e)/(s_{\text{tr}} \cdot \delta(e))$. Drones can be carried by trucks or deployed independently to execute delivery tasks. Drones operate at a constant speed $s_{\text{dr}} > 0$ and are capable of flying directly between any pair of nodes, independent of the road network. Drones can take off from and land on their accompanying

[1] A directed graph is called strongly connected if there is a path in each direction between each pair of graph nodes.

truck at any $v \in \mathcal{G}$. In each delivery task, their operational range is limited to a maximum distance $r > 0$ due to battery constraints, and they are limited to serving one customer due to their loading capacity.

Let $\mathcal{P} = \{p_1, p_2, \cdots, p_m\} \subseteq \mathcal{V}$ be the set of m depots for each truck group to start and end their tours, and $\mathcal{C} = \{c_1, c_2, \cdots, c_n\} \subseteq \mathcal{V}$ be the set of n customers to visit. A customer $c \in \mathcal{C}$ can be visited by any agent starting from any depot $d \in \mathcal{P}$. Furthermore, we assume $\mathcal{C} \cap \mathcal{P} = \varnothing$. Next, we formally define the *truck route* and the *drone route* that constitute a solution to the MA-FSTSP.

Definition 1 (**Truck Route**). *We call a sequence of nodes* $(v_0, v_1, \cdots, v_l)$, $v_j \in \mathcal{V}, \forall j \in \{0, 1, \cdots, l\}$, *a truck route if*

$$v_0 = v_l = p \in \mathcal{P}, \tag{4.1a}$$

$$(v_j, v_{j+1}) \in \mathcal{E}, \quad \forall j \in \{0, 1, \cdots, l-1\}. \tag{4.1b}$$

Intuitively, a truck route forms a cycle in the graph, starting and ending at a specific depot, as illustrated in Fig. 4.1a. Before discussing drone routes, we first introduce the concept of a drone delivery tuple to represent these routes effectively.

Definition 2 (**Drone Delivery Tuple**). *Given a truck route* $(v_0, v_1, \cdots, v_l)$, *a drone delivery tuple* (x, c, y) *describes a delivery task that a drone carried by the truck takes off from* v_x, *visits customer* c, *and lands on the truck at* v_y.

Definition 3 (**Drone Route**). *A drone route is represented by a tuple* $\langle (v_0, v_1, \cdots, v_l), \{(x_1, c_1, y_1), (x_2, c_2, y$
where $(v_0, v_1, \cdots, v_l)$ *is the truck route that the drone accompanies, and* $\{(x_1, c_1, y_1), (x_2, c_2, y_2), \cdots\}$ *comprises drone delivery tuples of the delivery tasks executed by drones.*

An example of the drone route is shown in Fig. 4.1b.

We define the route for a truck group based on the truck route and the drone route definitions.

Definition 4 (**Truck Group Route**). *Given truck group $\mathcal{T}_i$ starting from depot p_i, the route is represented as a $(k+1)$-tuple π_i, where $\pi_i[0] = (v_0, v_1, \cdots, v_l)$ is a truck route that $v_0 = v_l = p_i$, $\pi_i[j] = \{(x_1^{(j)}, c_1^{(j)}, y_1^{(j)}), (x_2^{(j)}, c_2^{(j)}, y_2^{(j)}), \cdots\}$, $j = 1, 2, \cdots k$, are k sets of drone delivery tuples such that $\langle \pi_i[0], \pi_i[j] \rangle$, $j = 1, 2, \cdots k$, are all drone routes.*

The total cost of a truck group route π_i, denoted as $\text{cost}(\pi_i)$, is defined as the minimum time needed for the truck group $\mathcal{T}_i$ to address all drone delivery tasks and return to the depot p_i. Now, we can formally state the MA-FSTSP problem.

Problem 2 (**MA-FSTSP**). *Given a strongly connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and m truck groups starting from different depots in set $\mathcal{D} \subseteq \mathcal{V}$ to visit a set of n customers $\mathcal{C} \subseteq \mathcal{V}$, find a set of m truck group routes $\{\pi_i\}_{i=1}^m$, one for each truck group, to visit every customer $c \in \mathcal{C}$ exactly once that minimizes the total cost $\sum_{i=1}^m cost(\pi_i)$.*

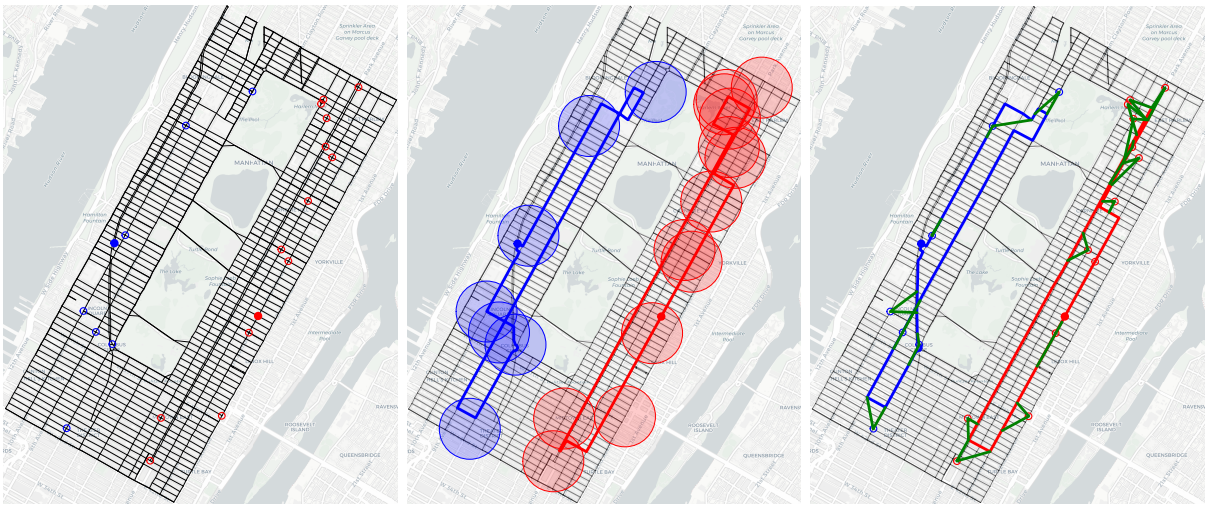
4.2 Methodology

Algorithm 3 Pipeline to Solve MA-FSTSP

Input: Road network $\mathcal{G}$, customer set $\mathcal{C}$, depot $\mathcal{P}$, truck speed s_{tr}, drone speed s_{dr}, drone endurance r, pairwise distance d

```
Initialize solution SOLN as empty set
for (u, v) ∈ C × C do
    d^tr(u, v) ← AdjustedShortestPathLength(u, v; G; δ)
end for
Assign customers {C_i}_{i=1}^{|P|} ← SetNN(C, P; d^tr, d)
for i ∈ {1, 2, ··· , m} do
    S ← {{n ∈ V(G) : w(n, c) < r/2} : c ∈ C_i} ∪ {{p_i}}
                                  ▷ For each customer c collect nearby nodes
    ORDER ← SetTSP(S; d^tr, d; s_tr, s_dr)
    SOLN[i] ← GetRoute(C_i, p_i; ORDER; d^tr, d; s_tr, s_dr)
end for
return SOLN
```

We apply our three-phase framework to solve MA-FSTSP by carefully designing each component based on the problem structure. In phase 1, the problem is broken down to m subproblems (recall that m is the number of depots) of a single truck carrying multiple



(a) Phase 1: assign customers to truck groups (b) Phase 2: solve the set TSP for visiting orders (c) Phase 3: optimize routes for trucks and drones

Figure 4.2: An Example of the output of each phase in the algorithm. In phase 1, we assign customers, marked by hollow circles, to their closest depots, marked by solid circles. The assignment is shown in Figure (a). In phase 2, we collect nodes within half of the distance limit for drones as the circles shown in Figure (b) and then apply the Set TSP method to find the shortest tour to visit all sets for each truck group starting from its depot as the colored lines shown in Figure (b). In phase 3, based on the visiting order on routes found in phase 2, we optimize the route for truck and drone together. The final results are shown in Figure (c), where the red and blue lines are routes for trucks of different groups, and the green lines are routes for drones when they are separated from the truck to visit customers.

drones to visit customers on road networks by allocating customers to truck groups via the nearest-neighbor method on sets. Next, in phase 2, a graph set TSP heuristic is applied to determine the visiting order of customers for each truck group. Finally, in phase 3, a truck-drone route is computed given the restriction of customers' visiting orders. The pseudocode for the proposed algorithm is shown in Alg. 3. Line 1∼4 is the preparation for data. The adjusted shortest path length is computed by incorporating the congestion level into the edge weights, i.e.,

$$d^{\text{tr}}(u,v) = \min_{p \in \textsc{Path}(u,v)} \sum_{e \in p} d(e) \cdot \delta(e), \tag{4.2}$$

where $\textsc{Path}(u,v)$ is the set of all paths from u to v in graph $\mathcal{G}$. Line 5 corresponds to phase 1. Line 7 computes the set for each customer, and lines 8∼9 correspond to phases 2 and 3, respectively. Methods of phase 1∼3 are introduced in Sec. 4.2.1, Sec. 4.2.2, and Sec. 4.2.3,

respectively. An example is shown in Fig. 4.2.

4.2.1 Phase 1: Set nearest-neighbor method

Algorithm 4 Pseudocode for Set Nearest-Neighbor

Input: Road network $\mathcal{G}$, customer set $\mathcal{C}$, depot $\mathcal{P}$, truck speed s_{tr}, drone speed s_{dr}, radius parameter θ, pairwise distance d

```
Initialize assignment {C_p}_{p∈P} as empty sets
for c ∈ C do
    S_c(θ) ← {v ∈ V(G) : d(v, c) < θ}
    for p ∈ P do
        Compute d^set(p, c) by Eq. 4.4
    end for
    p* ← arg min_{p∈P} d^set(p, c)
    C_{p*} ← C_{p*} ∪ {c}
end for
return assignment {C_p}_{p∈P}
```

To assign customers to truck groups, a common strategy used in multiagent systems is the nearest-neighbor algorithm [54]–[56]. However, two geographically close nodes might be distantly connected in road networks, commonly seen in practical scenarios such as places around the highway, one-way roads, and traffic bottlenecks. In such cases, neither geometric nor road map distance gives the most accurate estimation of the time required for the truck group to traverse the path. To this end, we extend the nearest-neighbor method to a set version called the *set nearest-neighbor* to consider both trucks and drones.

Given parameter θ, we collect all nodes within distance θ to a customer $c \in \mathcal{C}$ as:

$$\mathcal{S}_c(\theta) \coloneqq \{n \in \mathcal{V}(\mathcal{G}) : d(n, c) \leq \theta\}. \tag{4.3}$$

If $\theta \leq r$, each node in $\mathcal{S}_c$ is a valid takeoff or land node for a drone to visit customer c. Using this set, we define the distance between a depot p and customer c as

$$d^{\text{set}}(p, c) \coloneqq \min_{v \in \mathcal{S}_c(\theta)} \left\{ d(c, v) \cdot \frac{s_{\text{tr}}}{s_{\text{dr}}} + d^{\text{tr}}(p, v) \right\}, \tag{4.4}$$

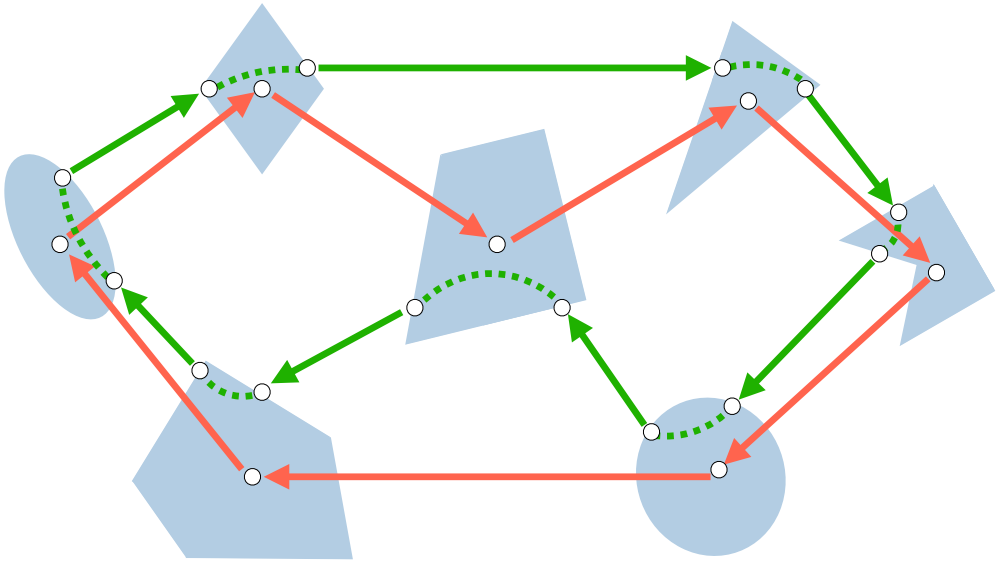


Figure 4.3: An example of Set TSP versus TSP. Each set represents a set of nodes in the graph, which can be arbitrarily connected inside. Sets are fully connected via nodes at the boundary. The red arrows represent the optimal TSP tour to visit the center nodes of sets in the graph, and the green arrows represent the optimal Set TSP tour to visit every set. Since Set TSP does not require visiting any exact node, it can pass through the set via **shortcuts** represented in dashed green. As a result, the visiting order of sets on the TSP tour and Set TSP tour can be different.

where d^{tr} is the adjusted shortest path for the truck defined in Eq. 4.2. The distance d^{set} can be viewed as a hybrid of truck distance on road networks and drone distance on geometry concerning the distance limit of drones. It is reduced to the distance for the truck when $\theta \to 0$. Based on d^{set}, we assign customers to the truck group with the closest starting depot. The pseudocode for this assignment is given in Alg. 4. Next, each group finds their own shortest tours in phase 2-3.

4.2.2 Phase 2: Set traveling salesman problem heuristic

Following the same insight in Sec. 4.2.1, we extend the TSP heuristic to a set version as well to take the usage of drones into account, which is formally stated as:

Problem 3 (Set TSP). *Given n location sets $V_1, V_2, \cdots, V_n$ along with the traveling costs between each pair of locations, find a possible route with the lowest total cost that visits each*

location set exactly once and returns to the original starting location set, where a consecutive visit to one or more location(s) in a location set is called a visit to the location set.

For the truck group $\mathcal{T}_i$, approximating the task of designing a tour that visits each customer in $\mathcal{C}_i$, starting and ending at depot p_i, can effectively be modeled by a set TSP as follows. (1) The location sets are $\{S_c(\theta) : c \in \mathcal{C}_i\} \cup \{\{p_i\}\}$, where $S_c(\theta)$ is defined in Eq. 4.3. We choose $\theta = r/2$ so all pairs of nodes in $S_c(\theta)$ can be a drone delivery task's start and end nodes. (2) The traveling cost from $u \in S_c(\theta)$, $c \in C_i$ to v is defined as:

$$w(u,v) = \begin{cases} \max\{\frac{d(u,c)+d(c,v)}{s_{\text{dr}}}, \frac{d(u,v)}{s_{\text{dr}}}\}, & v \in S_c(\theta); \\ \frac{d(u,v)}{s_{\text{tr}}}, & v \notin S_c(\theta). \end{cases} \tag{4.5}$$

The cost is the time needed to move from node u to node v and visit customer c by the truck group if $u, v \in S_c(\theta)$. The visiting order of customers in the optimal route found for the set TSP is used as the heuristic in Phase 3.

The Set TSP is an NP-hard problem since the TSP is a special case when each set only contains one node, and the total traveling cost can be computed in polynomial time. To solve it, we extend the Gavish-Graves (GG) formulation [57] of Mixed-Integer Linear Programming (MILP) for TSP.

For simplicity, we define $\bar{\mathcal{C}}_i \coloneqq \mathcal{C}_i \cup \{p_i\}$. Let $\beta_{\bar{c},\bar{c}'} \in \{0,1\}$ be a binary variable indicating whether the next customer or depot is $\bar{c}'$ after visiting customer or depot $\bar{c}$, and $y_{\bar{c},\bar{c}'} \in \mathbb{R}_+$ be the corresponding network flow. The TSP property of visiting each set exactly once can be expressed as

$$\beta_{\bar{c},\bar{c}} = y_{\bar{c},\bar{c}} = 0, \forall \bar{c} \in \bar{\mathcal{C}}_i, \tag{4.6a}$$

$$\sum_{\bar{c}' \in \bar{\mathcal{C}}_i} \beta_{\bar{c},\bar{c}'} = \sum_{\bar{c}' \in \bar{\mathcal{C}}_i} \beta_{\bar{c}',\bar{c}} = 1, \forall \bar{c} \in \bar{\mathcal{C}}_i, \tag{4.6b}$$

$$y_{\bar{c},\bar{c}'} \leq |\mathcal{C}_i| \cdot \beta_{\bar{c},\bar{c}'}, \forall \bar{c}, \bar{c}' \in \bar{\mathcal{C}}_i, \tag{4.6c}$$

$$\sum_{\bar{c} \in \bar{\mathcal{C}}_i} y_{p_i,\bar{c}} = |\mathcal{C}_i|, \tag{4.6d}$$

$$\sum_{\bar{c} \in \bar{\mathcal{C}}_i} y_{\bar{c},p_i} = 0, \tag{4.6e}$$

$$\sum_{\bar{c}' \in \bar{\mathcal{C}}_i} y_{\bar{c}',\bar{c}} - \sum_{\bar{c}' \in \bar{\mathcal{C}}_i} y_{\bar{c},\bar{c}'} = 1, \forall \bar{c} \in \bar{\mathcal{C}}_i. \tag{4.6f}$$

The constraint 4.6a forbids self-loops, 4.6b forces the solution to be a collection of cycles, and 4.6c∼4.6f ask the flow to reduce 1 unit each step along the cycle from $|\mathcal{C}_i|$ to 0, which only allows the existence of one cycle.

Let $\gamma_{u,v}$ be a binary variable indicating whether a drone tour starts from u to v for $u, v \in \mathcal{S}_c$ and $\delta_{v,u}$ be a binary variable indicating whether a truck picks up a drone at v and carry it to u to dispatch it. The constraints are:

$$\sum_{u,v \in \mathcal{S}_c} \gamma_{u,v} = 1, \forall c \in \mathcal{C}_i, \tag{4.7a}$$

$$\sum_{v \in \mathcal{S}_{\bar{c}}} \sum_{u \in \mathcal{S}_{\bar{c}'}} \delta_{v,u} = \beta_{\bar{c},\bar{c}'}, \forall \bar{c}, \bar{c}' \in \bar{\mathcal{C}}_i, \tag{4.7b}$$

$$\sum_{u \in \mathcal{S}_{\bar{c}}} \gamma_{u,v} = \sum_{\bar{c}' \in \bar{\mathcal{C}}_i} \sum_{w \in \mathcal{S}_{\bar{c}'}} \delta_{v,w}, \forall \bar{c} \in \bar{\mathcal{C}}_i, v \in \mathcal{S}_{\bar{c}} \tag{4.7c}$$

$$\sum_{v \in \mathcal{S}_{\bar{c}}} \gamma_{u,v} = \sum_{\bar{c}' \in \bar{\mathcal{C}}_i} \sum_{w \in \mathcal{S}_{\bar{c}'}} \delta_{w,u}, \forall \bar{c} \in \bar{\mathcal{C}}_i, u \in \mathcal{S}_{\bar{c}}. \tag{4.7d}$$

Constraint 4.7a encodes the TSP constraint, which asks for exactly one drone visit starting from a $u \in S_c$ and ending at a $v \in S_c$ to each customer $c \in \mathcal{C}_i$. Constraint 4.7b aligns the node-level route with the set-level route. If $\bar{c}$ is visited followed by the visit of $\bar{c}'$, i.e.

$\beta_{\bar{c},\bar{c}'} = 1$, then the truck route should pass from a $v \in S_{\bar{c}}$, where it picks up the drone visiting $\bar{c}$, to a $u \in S_{\bar{c}'}$, where it dispatches the drone to visit $\bar{c}'$. Otherwise, such a sub-route does not exist in the truck route. Constraint 4.7c means that if the drone lands at $v \in S_{\bar{c}}$ after visiting $\bar{c}$, it is picked up at v by the truck. It synchronizes the truck and the drone at the dispatching node. Constraint 4.7d, similarly, synchronizes the truck and the drone at the landing node.

The objective function minimizes the sum of costs of the edges (defined in Eq. 4.5) traversed by the route

$$\sum_{c \in \bar{\mathcal{C}}_i} \sum_{u \in \mathcal{S}_c} \left(\sum_{v \in \mathcal{S}_c} w(u,v) \cdot \gamma_{u,v} + \sum_{c' \in \bar{\mathcal{C}}_i} \sum_{v \in \mathcal{S}_{c'}} w(u,v) \cdot \delta_{u,v} \right). \tag{4.8}$$

γ and δ together form the adjacent matrix of the truck group tour on the fully connected graph $\mathcal{G}(\{p_i\} \cup_{c \in \mathcal{C}_i} S_c(\theta))$, from which we can extract the order of customers to visit.

4.2.3 Phase 3: Decode solution from heuristic

Even when the customers' visiting order is given, determining the optimal route for the truck group is an NP-hard problem. To approximate the optimal solution in polynomial time, we only consider the action of dispatching all drones simultaneously. Intuitively, when using multiple drones outperforms using one drone, the customers are close to each other within $O(r)$ distance, which means that the assumption does not sacrifice the performance a lot if r is small.

Based on the assumption above, dynamic programming can finish the decoding procedure within a polynomial time. Given visiting order $\mathcal{O}_i$ for truck group $\mathcal{T}_i$, we first define $\textsc{Time}(u, v; s, t)$ to be the minimum time needed for the truck group to visit customers $\{\mathcal{O}_i[s], \mathcal{O}_i[s+1], \cdots, \mathcal{O}_i[s+t-1]\}$ with t drones. The drones are dispatched together from u, and the last one is collected at node v. The initial state $\textsc{Time}(u, v; s, 1)$ is the time for

the truck with one drone to visit the customer $\mathcal{O}_i[s]$ starting from u to v, which is

$$\textsc{Time}(u, v; s, 1) = \begin{cases} \frac{d^{\text{tr}}(u,\mathcal{O}_i[s])+d^{\text{tr}}(\mathcal{O}_i[s],v)}{s_{\text{tr}}}, & d(u, \mathcal{O}_i[s]) + d(\mathcal{O}_i[s], v) > r; \\ \max\{\frac{d(u,\mathcal{O}_i[s])+d(\mathcal{O}_i[s],v)}{s_{\text{dr}}}, \frac{d^{\text{tr}}(u,v)}{s_{\text{tr}}}\}, & \text{otherwise.} \end{cases} \tag{4.9}$$

In the first case, the drone cannot finish the delivery task due to the energy limit given the position of u, v, and $O_i[s]$. We set $\textsc{Time}(u, v; s, 1)$ as the time needed for the truck to finish the delivery task. In the second case, where the drone can finish the delivery task, we set $\textsc{Time}(u, v; s, 1)$ as the minimum time in which the truck can move from u to v along the road network and the drone can fly from u to $O_i[s]$ and then to v. For $t \geq 2$, if condition $d(u, \mathcal{O}_i[s + t - 1]) + d(\mathcal{O}_i[s + t - 1], v) \leq r$ is satisfied, i.e., the t-th drone can finish the delivery task, $\textsc{Time}(u, v; s, t)$ has the transition function

$$\begin{aligned} \textsc{Time}(u, v; s, t) = \min_{w \in S_{\mathcal{O}_i[s+t-2]}} \{\max\{&\textsc{Time}(u, w; s, t-1) + \frac{d^{\text{tr}}(w, v)}{s_{\text{tr}}}, \\ &\frac{d(u, \mathcal{O}_i[s + t - 1]) + d(\mathcal{O}_i[s + t - 1], v)}{s_{\text{dr}}}\}\}. \end{aligned} \tag{4.10}$$

It computes the minimum time needed to finish the t delivery tasks by identifying the optimal node w to collect the $(t - 1)$-th drone, which minimizes the overall time the truck takes to travel from u to v while ensuring that the first $t - 1$ drones are collected. If the condition is not satisfied, i.e., the drone cannot finish the delivery task, we set $\textsc{Time}(u, v; s, t) = +\infty$.

Then, we define $\textsc{Value}(s, u)$ as the optimal cost starting from the depot p_i to the current node u after serving the first s customers. The initial state

$$\textsc{Value}(0, v) = d^{\text{tr}}(p_i, v), \forall v \in \mathcal{V}(\mathcal{G}), \tag{4.11}$$

which is the adjusted shortest path length from depot p_i to end node u. Given the number

of drones k, the transition function takes the k preceding states into account,

$$\textsc{Value}(s, v) = \max_{\substack{0 \le t \le k \\ u, w \in \mathcal{V}(\mathcal{G})}} \textsc{Value}(s - t, u) + \textsc{Time}(u, w; s - t, t) + d^{\text{tr}}(w, v). \tag{4.12}$$

In this way, $\textsc{Value}(|\mathcal{O}_I|, p_i)$ is the optimal cost for truck group $\mathcal{T}_i$, and the corresponding route can be reconstructed by tracing backward.

Here, we analyze the complexity of our 3-phase method. Suppose there is a lower bound l for the distance between nodes on the graph; then each node has at most 6 neighbors within distance l. So the number of nodes we collect to form sets $|\mathcal{S}_\theta| \le 7\frac{\pi\theta^2}{\pi l^2} = O(\theta^2)$ grows in quadratic to the radius θ. The time complexity for the first phase is $O(|\mathcal{C}||\mathcal{P}||\mathcal{S}_\theta|) = O(mn\theta^2)$ and for the third phase is $O(k|\mathcal{C}||\mathcal{S}_\theta| + k|\mathcal{C}||\mathcal{S}_\theta|^3) = O(kn\theta^6)$, where k is the number of drones per truck. Since phase 3 is executed for each truck group, the overall complexity for the polynomial part is $O(mn\theta^2 + kmn\theta^6) = O(kmn\theta^6)$. The second phase is a MILP with $(|\mathcal{C}| + 1)^2 = O(n^2)$ binary variables β, $(|\mathcal{C}| + 1)|\mathcal{S}_\theta|^2 = O(n\theta^4)$ binary variables γ, and $(|\mathcal{C}| + 1)^2|\mathcal{S}_\theta|^2 = O(n^2\theta^4)$ binary variables δ, which are $O(n^2\theta^4)$ binary variables in total, and $O(n^2)$ continuous variables y.

To address the NP-hard problem scalably, certain trade-offs in solution optimality are made. The first phase estimates the cost post-allocation based on set distances between nodes. This approach often deviates from the optimal solution as it does not account for the spatial distribution of other customers. The second phase approximates the FSTSP with multiple drones with standard TSP, which is effective when the customers are widely spaced. However, it is not a good heuristic for areas with high customer density since it does not consider the number of drones. The last phase only considers a limited subset of possible drone routes, constrained by predetermined truck routes. Intuitively, the most effective route in the set is a near-optimal solution.

Table 4.1: Comparison with Baselines. All results are averaged over 100 instances sampled uniformly from all nodes in the graph.

	Manhattan(part)		Manhattan (full)		Boston	
Method	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)
CG [7]	1.41	175.34	-	-	-	-
HC-VNS	1.21	4.17	26.91	191.64	146.72	2507.9
Truck-Only	1.75	**0.15**	25.55	**0.93**	137.70	34.2
Ours	**0.89**	0.23	**23.85**	3.71	**115.94**	**17.67**
LB	0.60	-	10.85	-	61.34	-

4.3 Experiments

In this section, we report experiments to validate our method. In section 3.3.1, we compare our method with four baselines: a column generation-based method [7], a hill-climbing algorithm, an upper bound, and a lower bound implemented by us. In section 4.3.2, we validate our set extension of nearest-neighbor and TSP heuristic. In section 4.3.3, we do the sensitivity analysis to explore the influence of congestion level, truck/drone speed, distance limit for drones, and number of trucks, drones, depots, and customers.

All experiments are performed on a server running on Ubuntu 20.04.6 LTS with an AMD Ryzen Threadripper 3990X 64-Core Processor. The MILP was solved by Gurobi 10.0.0 [10].

Unless mentioned otherwise, we use the drone traveling distance limit $r = 0.5$, speeds $s_{\text{tr}} = s_{\text{dr}} = 1$, congestion level $\delta(e) = 1$, $\forall e \in \mathcal{E}$, set radius $\theta_{\text{nn}} = \theta_{\text{tsp}} = r/2$, and the number of iterations for hill-climbing algorithm ROUNDS $= 5000$.

4.3.1 Comparison against baselines

Baselines. To the best of our knowledge, the only method for multiple trucks and drones on the road networks is the column generation-based heuristic [7]. It sets the partition problem as the master problem and the routing problem for each truck group as the pricing problem, approximated via Variable Neighborhood Search (VNS).

We also implement a VNS-based method incorporated with the widely used meta-heuristic method, the Hill-Climbing algorithm [58], for the problem. It assigns customers by nearest-neighbor method, initiates the route for each truck group from truck-only routes, and iteratively replaces the current route with the best route in its neighborhood, i.e., the best route among three classes of routes: (1) changes a drone's takeoff/land location from current route, (2) changes the agent to visit a customer from the truck to a drone from current route, (3) switch the visiting order of two consecutive customers from current route.

Besides, we report an upper bound and a lower bound for the optimal cost as two baselines. The upper bound is the optimal truck-only solution cost. The lower bound for the Manhattan datasets, including both the partial map and the full map, is the optimal solution cost given $k = +\infty$ and $s_{\text{dr}} = +\infty$, which is computed by solving an equivalent Set MATSP:

Problem 4 (**Set-MATSP-v1**). *Given a strongly connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and m trucks starting from different depots in set $\mathcal{D} \subseteq \mathcal{V}$ to visit a set of n customers $\mathcal{C} \subseteq \mathcal{V}$, where a visit to customer $c \in \mathcal{C}$ is defined as reaching a node $v \in S_c(\frac{r}{2})$, find a set of m truck routes, one for each truck, to visit every customer $c \in \mathcal{C}$ at least once that minimizes the total cost. The cost of a truck route is the total weight of the edges it passes.*

However, computing the optimal solution for Set-MATSP-v1 is also very hard. We approach the optimal cost by the lower bound from solving its MILP formulation via Gurobi for 600 seconds. The Boston map is too large for Gurobi to get a reasonable lower bound in 3600 seconds, so we further relaxed the Set-MATSP not to consider the traveling cost within $S_c(\frac{r}{2})$ for all $c \in \mathcal{C}$, i.e., only considering the traveling cost between $S_c(\frac{r}{2})$. Formally, we state the problem in an equivalent form:

Problem 5 (**Set-MATSP-v2**). *Given a strongly connected directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a fully connected graph $\mathcal{G}'(\mathcal{D} \cup \mathcal{C})$, where $\mathcal{D} \subseteq V$ is the depots set and $\mathcal{C} \subseteq V$ is the customers*

set and the edge weight of $(u, v) \in \mathcal{G}'$ *is defined as*

$$w_{\mathcal{G}'}(u,v) = \begin{cases} \min_{s \in S_u(\frac{r}{2}), t \in S_v(\frac{r}{2})} d^{tr}(s,t), u, v \in \mathcal{C}, \\ \min_{s \in S_u(\frac{r}{2})} d^{tr}(s,v), u \in \mathcal{C}, v \in \mathcal{D}, \\ \min_{t \in S_v(\frac{r}{2})} d^{tr}(u,t), u \in \mathcal{D}, v \in \mathcal{C}, \\ d^{tr}(u,v), u, v \in \mathcal{D}, \end{cases}$$

, find a set of m *cycles on* $\mathcal{G}'$*, one starts from each* $d \in \mathcal{D}$ *(can be self cycle), to visit each* $c \in \mathcal{C}$ *exactly once that minimize the total weights of edges passed.*

We report the optimal cost of Set-MATSP-v2 for the Boston dataset from solving its MILP formulation.

Datasets. The methods are evaluated on datasets created from three real-world road networks. We use the Manhattan road map [51] and the Cambridge road map from OpenStreetMap [52] and extract part of the Manhattan map to form a partial Manhattan map to evaluate methods that cannot handle the full map.

We sample $|\mathcal{C}| = 6$ customers and $|\mathcal{P}| = 2$ depots each instance on tiny map of $|\mathcal{V}(G)| = 20$ nodes, $|\mathcal{C}| = 30$ customers and $|\mathcal{P}| = 5$ depots on Manhattan of $|\mathcal{V}(\mathcal{G})| = 1024$ nodes, and $|\mathcal{C}| = 100$ customers and $\mathcal{P} = 10$ depots on Boston of $|\mathcal{V}(\mathcal{G})| = 11000$ nodes. We set the number of drones on each truck to be 2, 3, and 4 on the tiny map, Manhattan and Boston.

Metrics. We evaluate the methods by the tour costs for all truck groups and the running time for each method. The tour cost is the sum of the minimal time needed for each truck group to visit assigned customers. The running time is recorded since the input of formatted datasets.

Results. The results are shown in table 4.1. Baseline CG suffers from the scalability problem and cannot produce solutions within the time limit of 5 minutes. Our method outperforms baselines in both solution quality and computational efficiency. On the dataset from the full Manhattan graph, our method gives an 11.67% reduction in cost and a 46.74

Table 4.2: Ablation of the set-based methods. Results are averaged over 100 instances sampled uniformly from Manhattan.

		$\|\mathcal{C}\| = 30$		$\|\mathcal{C}\| = 40$		$\|\mathcal{C}\| = 50$	
θ_{nn}	θ_{tsp}	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)
0	0	26.78	1.54	28.76	1.86	31.33	2.60
0	$r/2$	26.42	6.56	28.31	11.68	30.72	19.54
$r/2$	0	26.53	1.55	28.09	1.90	30.49	2.54
$r/2$	$r/2$	26.19	6.58	27.57	12.31	29.77	19.59

Table 4.3: Ablation of the set-based methods. Use the same datasets as Table 2 but increase the congestion level to 5 for roads within a distance of $0.3r$ to customers.

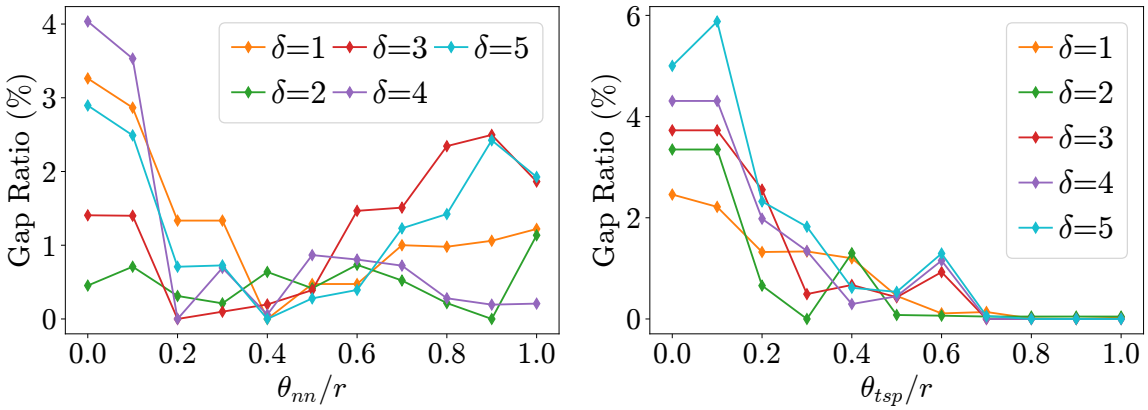
		$\|\mathcal{C}\| = 30$		$\|\mathcal{C}\| = 40$		$\|\mathcal{C}\| = 50$	
θ_{nn}	θ_{tsp}	Cost	Time(s)	Cost	Time(s)	Cost	Time(s)
0	0	33.18	1.32	38.03	1.91	42.18	2.42
0	$r/2$	32.77	11.42	36.94	20.12	39.47	33.47
$r/2$	0	31.58	1.35	34.86	1.90	39.02	2.40
$r/2$	$r/2$	31.44	11.77	33.98	17.99	36.37	29.91

times speed up. On the dataset from the Boston graph, it reduces 14.99% cost with a 23.77 times speed up.

4.3.2 Validation for set-based methods

The value of θ trades off between the computational efficiency and the performance of the heuristic. When we set $\theta = 0$, it is reduced to the standard TSP heuristic for a truck group. In this section, we validate the Set NN and Set TSP and explore the influence of the difference values of θ.

Effectiveness. To show that both set-based methods can contribute to the improvement in results, we compare four pairs of $\theta_{\text{nn}}, \theta_{\text{tsp}} \in \{(0,0), (0, r/2), (r/2, 0), (r/2, r/2)\}$, corresponding to NN + TSP, NN + Set TSP, Set NN + TSP, and Set NN + Set TSP, on Manhattan map. We vary the number of customers $|\mathcal{C}| \in \{30, 40, 50\}$ with fixed number of depots $|\mathcal{P}| = 5$. We first do the test with $\delta = 1$. In practice, customers are usually at places



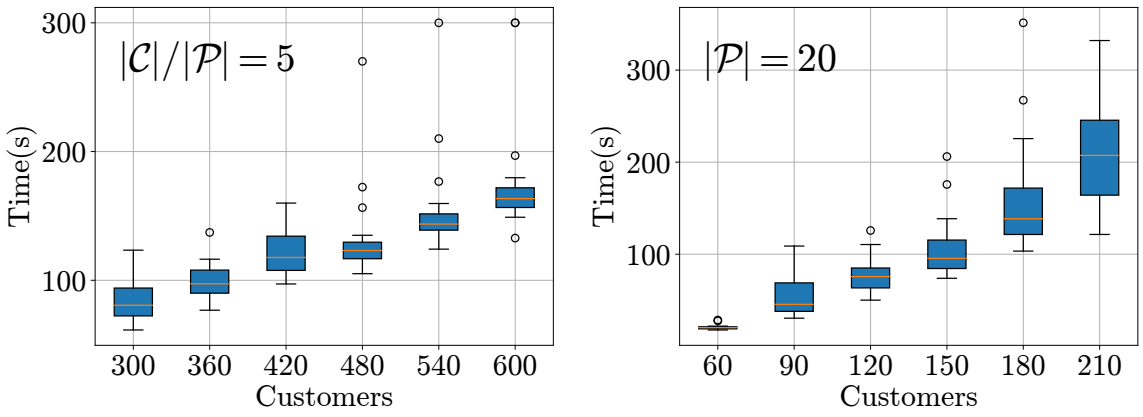
(a) Set NN parameter θ_{nn}

(b) Set TSP parameter θ_{tsp}

Figure 4.4: Ablation for parameters θ_{nn} and θ_{tsp} with congestion level δ. Results are averaged over 100 instances of $|\mathcal{C}| = 50$ and $|\mathcal{P}| = 5$ from Manhattan. To emphasize the relative changes, we normalize the results of each congestion level by the minimal value of all θ and report the gap, i.e. $\textsc{Reported}(\theta, \delta) \leftarrow \textsc{Cost}(\theta, \delta) / \min_\theta \textsc{Cost}(\theta, \delta) - 1$. The values are reported in percentages.

like office buildings, schools, and resident areas, where traffic is quite heavy. To model it, we test the four pairs of parameters with the level of congestion $\delta(e) = 5$ for roads e within geometry distance $0.3r$ to customers. The results are shown in Tab. 4.2 and Tab. 4.3. Extension to the set-based methods trades off between the computation cost and the solution performance. Both Set NN and Set TSP show their contribution to the reduction of cost, and the significance increases along with the increment of the density of customers as well as the congestion level. While the Set NN does not take much extra computation time, the Set TSP takes 20 times longer than the TSP.

Influence of θ and δ. We further explore the influence of θ for set NN and set TSP under different congestion levels with fixed congestion area radius $0.3r$. For the significance of the difference, we use the dataset used in the effectiveness experiments with $|\mathcal{C}| = 50$. To avoid considering invalid nodes for drones to take off or land, θ_{nn} varies from 0 to r when θ_{tsp} is fixed at 0, and θ_{tsp} varies from 0 to r when θ_{nn} is fixed at 0. The congestion level δ varies from 1 to 5. The results are shown in Fig. 4.4a and Fig. 4.4b. The results are generally good for $\theta_{\text{nn}} \in [0.3, 0.6]$ depending on the congestion level. The reason is that since

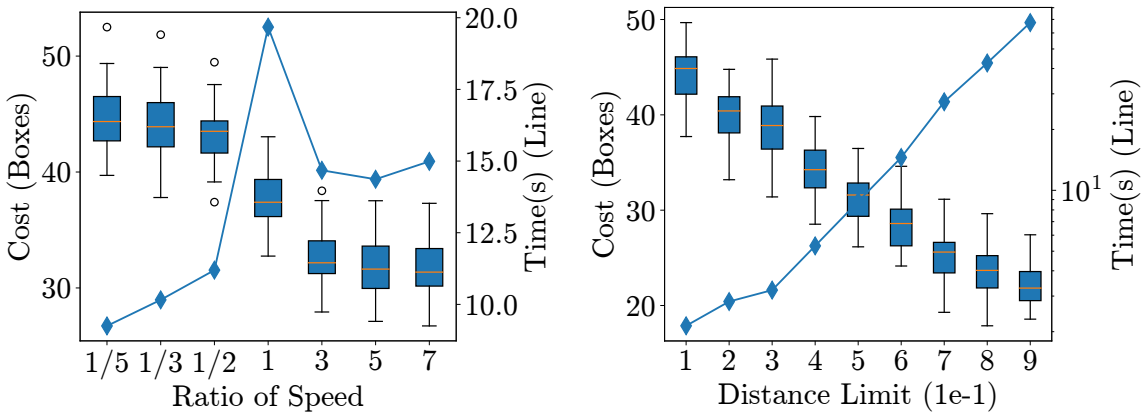


(a) Depots

(b) Customers

Figure 4.5: Results on scalability. Figure (a) fixes the ratio of expected customers per truck group to visit $|\mathcal{C}|/|\mathcal{P}| = 5$, and figure (b) fixes the number of depots $|\mathcal{P}| = 20$. We set the time limit for each instance to 300 seconds.

the traveling distance limit for the drone is r, a take-off or land action should happen within $0.5r$. Because the drone only takes off and lands at the ends of edges, the actual take-off action or land action happens strictly closer than $0.5r$ to the customer. Hence, when taking θ_{nn} around 0.4, the set distance metric is a better heuristic for the cost after partition than other θ_{nn} values. Compared with the cost of $\theta_{\text{nn}} = 0$ and $\theta_{\text{nn}} = 1$, the Set NN is effective in improving the solution cost by producing better assignments when the number of nodes considered is within the proper range. From curves for θ_{tsp}, increasing the value of θ_{tsp} generally decreases the solution cost. The local maximum happens at $\theta_{\text{tsp}} \in \{0.4, 0.6\}$ due to the mismatch between the nodes considered by the Set TSP and the nodes considered by the third phase. The relative changes also become more significant as the congestion level δ around customers increases. This means that the set-based methods are more effective for scenarios with heavy traffic. In Fig. 4.4b, the impact of congestion level after $\theta_{\text{tsp}} \geq 0.5$ quickly reduce to 0. Since we choose the congestion area radius to be $0.3r$ and $s_{\text{tr}} = s_{\text{dr}}$, it is preferred to dispatch the drone outside the congestion area when $\theta_{\text{tsp}} \geq 0.4r$. In this case, the truck route has very little or even no overlap with the congestion area; hence, it is not influenced by the congestion level. We can conclude that the set-based methods successfully



(a) Relative speed $s_{\text{dr}}/s_{\text{tr}}$

(b) Distance limit r

Figure 4.6: The influence of relative speed and distance limit. Time in Fig. (b) is plotted on a logarithm scale.

Table 4.4: Cost for different numbers of drones per truck on datasets sampling nodes from uniform distribution and Clusters.

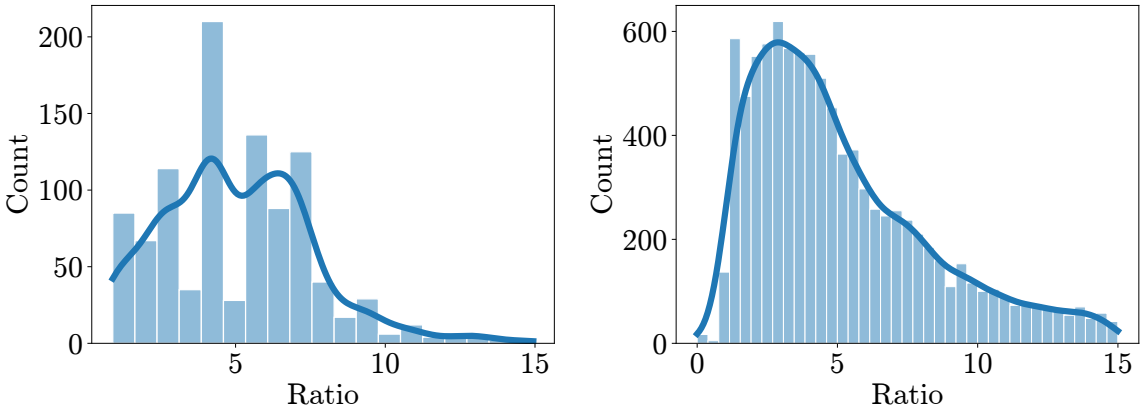
	Cost				
Distribution	$k=1$	$k=2$	$k=3$	$k=4$	$k=5$
Uniform	30.16	30.16	29.98	29.96	29.95
Clusters	29.76	29.75	29.52	29.50	29.49

considered the usage of drones and the advantages of not being influenced by road traffic over ground vehicles.

4.3.3 Factors influencing the algorithm

In this section, we study the influence of various factors and parameters in our algorithm, which includes the number of customers $|\mathcal{C}|$, the number of depots $|\mathcal{P}|$, the level of congestion δ and the area where congestion happens, the number of drones per truck k, the relative speed $s_{\text{dr}}/s_{\text{tr}}$ between trucks and drones, and the distance limit r for drones.

Scalability. To measure the scalability of our method, we vary the number of depots $|\mathcal{P}|$ and the number of customers $|\mathcal{C}|$ in two ways: (1) fix the ratio of $|\mathcal{C}| : |\mathcal{P}| = 5$, vary the $|\mathcal{P}|$ from 10 to 100; (2) fix the number of depots $|\mathcal{P}| = 20$, vary the number of customers



(a) Manhattan (b) Boston

Figure 4.7: The distribution of the maximal road-geometry ratio of nodes in the Manhattan and Boston graphs. The maximal road-geometry ratio for a node v is defined as $\max_{u \in S_{r/2}} d^{\text{tr}}(u, v)/d(u, v)$.

$|\mathcal{C}|$ from 60 to 210. All datasets are sampled randomly from the Boston road network. The results are shown in Fig. 4.5a and Fig. 4.5b. When fixing the ratio $|\mathcal{C}|/|\mathcal{P}|$, which is also the expected size for each Set TSP, the running time grows linearly as the number of depots $|\mathcal{P}|$ grows, which is also the growth of the number of Set TSP to solve. When fixing the number of depots $|\mathcal{P}|$, the running time grows exponentially with the number of cities $|\mathcal{C}|$. So, the bottleneck for our method to scale up is the Set TSP phase.

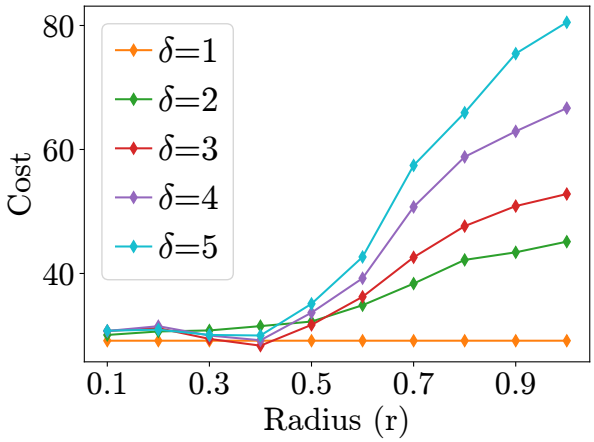


Figure 4.8: Ablation of congestion level and congestion area. We see the congestion area to all nodes within the distance RADIUS to customers. The RADIUS is reported in the unit of distance limit for drones r.

Congestion Level. We vary the congestion level from 1 to 5 and the radius RAD around the customers from 0 to r. Experiments are conducted on datasets from the Manhattan with $|\mathcal{C}| = 50$ and $|\mathcal{P}| = 5$, with parameters $\theta_{\text{nn}} = \theta_{\text{tsp}} = r/2$. The results are shown in Fig. 4.8. When RADIUS $\leq 0.4r$, the impact of increasing RADIUS and δ is not significant, which means that the optimal locations for drones to take off and land are out of the area with radius $0.4r$ around customers. When RADIUS ≥ 0.5, both factors influence the cost in a sublinear manner, which means that the increase of RADIUS and δ also result in different routes for truck groups.

Relative Speed. We vary the relative speed $s_{\text{dr}}/s_{\text{tr}}$ from 0.2 to 7 on instances of $|\mathcal{P}| = 15$, $|\mathcal{C}| = 75$ from the Manhattan road network when there is no congestion. The result is shown in Fig. 4.6a. The cost decreases as the relative speed increases, and the biggest improvement happens between ranges 0.5 and 3. The marginal utility diminishes quickly when $s_{\text{dr}}/s_{\text{tr}} > 3$ since the time consumption for trucks is much higher than for drones. The increment also happens when $0.2 \leq s_{\text{dr}}/s_{\text{tr}} \leq 0.5$, which means that even if the drones are quite slow, the truck group still gets benefits from using drones to visit nodes instead of trucks. Such nodes are near their neighbors in geometry but far reach through road network. From the statistics in Fig. 4.7a and Fig. 4.7b, nodes with large geometry-road distance ratio to some of their neighbors are quite common in the map. The maximum running time is reached at $s_{\text{dr}}/s_{\text{tr}} = 1$ when there are more routes to choose from since neither drones nor trucks have a dominant advantage.

Distance Limit for Drones. The limit r varies from 0.1 to 0.9 in the experiment, which uses the same dataset as experiments for the relative speed. The result is shown in Fig. 4.6b. The cost decreases linearly as the r increases while the running time increases exponentially.

Number of Drones. We test different number of drones $k \in \{1, 2, 3, 4, 5\}$ on 2 datasets on Manhattan road networks with $|\mathcal{P}| = 15$ and $|\mathcal{C}| = 75$. One dataset is generated by randomly sampling from all nodes on the graph, and the other one is generated by first

randomly sampling several nodes as the center of clusters and then sampling other nodes within the distance $3r$ to them. The results are shown in Tab. 4.4.

Chapter 5

Conclusion and Future Work

This thesis introduces a three-phase hierarchical framework designed to address practical constraints in Traveling Salesman Problems (TSP) within multi-agent systems. We focus on two prevalent constraints: energy limitations and the cooperation of aerial robots. We developed two novel problem formulations to encapsulate these constraints: the Multi-Agent Energy-Constrained TSP (MA-ECTSP) and the Multi-Agent Flying Sidekick TSP (MA-FSTSP). In Chapter 3, we introduce algorithms for MA-ECTSP that deconstruct the problem using a minimum spanning tree (MST) and tackle the energy constraints by selecting feasible path combinations through Mixed Integer Linear Programming (MILP). Chapter 4 extends the partitioning method and traditional TSP to a set version to accurately model drone behaviors in MA-FSTSP. Our proposed methodology generates solutions of superior quality compared to existing baselines and demonstrates the capability to tackle significantly larger problem sizes than those manageable by the Mixed Integer Linear Programming (MILP) approach. These advancements are substantiated by rigorous experimental validation on real-world datasets.

Currently, the three phases of our framework depend significantly on the specific structure or inherent properties of the problem at hand. Given that many variants of the TSP in multi-agent systems can be formulated as MILP, we are motivated to develop a more general

heuristic for the initial phase, i.e., the assignment of customers to salesmen, to accommodate a wider range of problems. Although our method can manage scenarios involving up to thousands of customers, it still struggles to handle some large-scale practical applications. To bridge this gap, we aim to enhance the scalability of our framework while maintaining high solution quality by integrating learning-based methods into our approach.

Appendix A

Mixed-Integer Linear Programming Forumlations

We include the MILP formulations for MA-ECTSP in this chapter.

Let $\beta_{d,u,v,i,j} \in \{0,1\}$ be a binary variable indicating whether a salesman who starts from depot d passes the edge (i,j) when traveling from depot or customer u to depot or customer v. Let $c_{i,j}$ be the weights of edge (i,j). The objective function is to minimize the total weights of selected edges:

$$\min \sum_{d\in\mathcal{D}} \sum_{u,v\in\mathcal{C}\cup\mathcal{D}} \sum_{(i,j)\in\mathcal{E}} c_{i,j} \cdot \beta_{d,u,v,i,j}. \tag{A.1}$$

Let $\gamma_{d,u} \in \{0,1\}$ be the binary variable indicating whether depot or customer u is visited by the salesman who starts from depot a. Then, each customer should be visited exactly once, i.e.,

$$\sum_{d\in\mathcal{D}} \gamma_{d,u} = 1, \qquad u \in \mathcal{C}\cup\mathcal{D} \tag{A.2a}$$

$$\gamma_{d,d} = 1, \qquad a \in \mathcal{D}. \tag{A.2b}$$

To guarantee the TSP properties of salesmen, we adapt the GG formulation [57] with binary $\phi_{d,u,v} \in \{0,1\}$, which indicates whether the tour starts from depot d will consecutively visit depots or customers u and v, and real variable $f_{d,u,v} \in \mathbb{R}_+$ w.r.t. $\phi_{d,u,v}$.

$$\phi_{d,u,u} = 0, d \in \mathcal{D}, u \in \mathcal{C} \cup \mathcal{D}, \tag{A.3a}$$

$$\phi_{d,u,v} = 0, d \in \mathcal{D}, u \in \mathcal{C} \cup \mathcal{D}, v \in \mathcal{D}, \tag{A.3b}$$

$$\sum_{u \in \mathcal{C} \cup \mathcal{D}} \phi_{d,u,v} = \sum_{u \in \mathcal{C} \cup \mathcal{D}} \phi_{d,v,u} = \gamma_{d,v}, d \in \mathcal{D}, v \in \mathcal{C} \cup \mathcal{D} \setminus \{d\}, \tag{A.3c}$$

$$\gamma_{d,v} \leq \sum_{u \in \mathcal{C} \cup \mathcal{D}} \phi_{d,u,d} = \sum_{u \in \mathcal{C} \cup \mathcal{D}} \phi_{d,d,u} \leq 1, d \in \mathcal{D}, v \in \mathcal{C} \cup \mathcal{D}, \tag{A.3d}$$

$$\sum_{u \in \mathcal{C} \cup \mathcal{D}} \gamma_{d,u} - \phi_{d,u,d} \geq 1, d \in \mathcal{D}, \tag{A.3e}$$

$$f_{d,u,v} \leq (|\mathcal{C}| + |\mathcal{D}|) \cdot \phi_{d,u,v}, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}, \tag{A.3f}$$

$$\sum_{v \in \mathcal{C} \cup \mathcal{D}} \gamma_{d,v} - f_{d,d,v} = 1, d \in \mathcal{D}, \tag{A.3g}$$

$$\sum_{u \in \mathcal{C} \cup \mathcal{D}} f_{d,u,v} - f_{d,v,u} = \gamma_{a,v}, d \in \mathcal{D}, v \in \mathcal{C}. \tag{A.3h}$$

Then, we add the constrained selection of edges from the whole graph. We connect the variables $\beta_{d,u,v,i,j}$ and variables $\phi_{d,u,v}$ to address the TSP property and multi-agent partition restriction as

$$\sum_{j \in \mathcal{V}} \beta_{d,u,v,u,j} = \phi_{d,u,v}, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}, \tag{A.4a}$$

$$\sum_{i \in \mathcal{V}} \beta_{d,u,v,i,v} = \phi_{d,u,v}, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}. \tag{A.4b}$$

To ensure the selected edges form paths, we ask for the in-degree equal to the out-degree at

every node except for the start and end nodes, i.e.,

$$\sum_{j\in\mathcal{V}} \beta_{d,u,v,i,j} = \sum_{j\in\mathcal{V}} \beta_{d,u,v,j,i} \leq 1, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}, i \in \mathcal{V} \setminus \{u, v\}, \tag{A.5a}$$

$$\sum_{i\in\mathcal{V}} \beta_{d,u,v,i,u} = \sum_{j\in\mathcal{V}} \beta_{d,u,v,v,j} = 0, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}. \tag{A.5b}$$

Also, we need to get rid of self-loop

$$\beta_{d,u,v,i,i} = 0, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}, i \in \mathcal{V}. \tag{A.6}$$

Finally, we do not want to visit any other customers or depots along the path from u to v, i.e.

$$\sum_{j\in\mathcal{V}} \beta_{d,u,v,i,j} = 0, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}, i \in \mathcal{V} \setminus \{u, v\}. \tag{A.7}$$

Next, we start to address the energy constraints. The first energy constraint we need to meet is the resource limitation for each station, which is

$$\sum_{d\in\mathcal{D}} \sum_{u,v\in\mathcal{C}\cup\mathcal{D}} \sum_{j\in\mathcal{V}} \beta_{d,u,v,s,j} \leq r_s, s \in \mathcal{S}. \tag{A.8}$$

To express the energy constraints for each salesman, we define non-negative real variables e_u, representing the energy level of a salesman when visiting the depot or customer u. Initially, the energy for all salesman is E,

$$e_d = E, d \in \mathcal{D}. \tag{A.9}$$

Every salesman should not run out of energy, i.e.,

$$e_u \geq 0, u \in \mathcal{C} \cup \mathcal{D} \tag{A.10}$$

In case of the edge (i, j) selected, if both i and j are stations, then the distance should not

exceed the maximum distance a full energy salesman can travel, i.e.,

$$c_{i,j} \cdot \beta_{d,u,v,i,j} \leq E, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}, i, j \in \mathcal{S}. \tag{A.11}$$

If the start node i is not a station but a customer or depot u, then the distance between u and j should not take more than the energy remaining at u, i.e.,

$$c_{u,j} \cdot \beta_{d,u,v,u,j} \leq e_u, d \in \mathcal{D}, u, v \in \mathcal{C} \cup \mathcal{D}, j \in \mathcal{V}. \tag{A.12}$$

And if the arriving node j is not a station but a customer v, then the energy remains is bounded by full minus the consumption, which is

$$c_{i,v} \cdot \beta_{d,u,v,i,v} \leq E - e_v, d \in \mathcal{D}, u \in \mathcal{C} \cup \mathcal{D}, v \in \mathcal{C} \cup \mathcal{D} \setminus \{d\}, i \in \mathcal{S}. \tag{A.13}$$

If both i and j are not stations, i.e. $i = u$ and $j = v$, then

$$(c_{u,v} + E) \cdot \beta_{d,u,v,u,v} \leq E + e_u - e_v, d \in \mathcal{D}, u \in \mathcal{C} \cup \mathcal{D}, v \in \mathcal{C} \cup \mathcal{D} \setminus \{d\}. \tag{A.14}$$

For the path returning back to the depot, we can get two restrictions conditioned on the departing node following the discussion above,

$$(c_{u,d} + E) \cdot \beta_{d,u,d,u,d} \leq E + e_u, d \in \mathcal{D}, u \in \mathcal{C} \cup \mathcal{D}, \tag{A.15a}$$

$$c_{i,d} \cdot \beta_{d,u,d,i,d} \leq E, d \in \mathcal{D}, u \in \mathcal{C} \cup \mathcal{D}, i \in \mathcal{S}. \tag{A.15b}$$

Finally, combine objects (A.1) and constraints (A.2)-(A.15), we get the MILP formulation for MA-ECTSP.

Appendix B

Theorems and Proofs

B.1 Explaination for Function f and g

By the optimality, for tree $\mathcal{T}(u)$, its optimal partitions $\tilde{\mathcal{T}}(u)$ and $\hat{\mathcal{T}}(u)$ should have all subtrees of its child partitioned optimally, i.e. optimal $\tilde{\mathcal{T}}(v)$ or optimal $\hat{\mathcal{T}}(v)$ for all $v \in H(u)$, where $\mathcal{H}(u)$ represents the set of children of node u in $\mathcal{T}(\texttt{rt})$. Based on this observation, we briefly discuss the kind of partitions children nodes can have based on whether the root u is a customer or a depot in the tree $\mathcal{T}(u)$.

1. If the root u is a customer and has type $\tilde{\mathcal{T}}$, i.e., there is a depot connected to u after the partition. Then the depot is contained in a subtree $\mathcal{T}(v)$ for a child $v \in \mathcal{H}(u)$. Thus, u and v are connected and have the same partition type $\tilde{\mathcal{T}}$, which means node v contributes $f(v) + c(u, v)$ to value function $f(u)$. Other children cannot connect to both u and any depot in their subtrees simultaneously to avoid the scenario that u connects to two depots. So, each child either connects to u with partition type $\hat{\mathcal{T}}$ or disconnects to u with partition type $\tilde{\mathcal{T}}$, which contributes $\min\{f(v'), g(v') + c(u, v')\}$ to the value of $f(u)$.

2. If the root u is either a depot partitioned with type $\tilde{\mathcal{T}}$ or a customer partitioned with type $\hat{\mathcal{T}}$, every child $v \in \mathcal{H}(u)$ cannot connect to both u and any depot in their subtrees

at the same time, which contributes $\min\{f(v), g(v) + c(u, v)\}$ to the value of $f(u)$ or $g(u)$ as argued in 1.

3. $g(u) = +\infty$ for every depot u as it must contain a depot (itself).

B.2 Proof for Theorem 1

Lemma 1. *Given the input as in Theorem 1, the assignment of partition type for each node based on the output of Algorithm 2 is consistent with the resulting partition.*

Proof. We prove the lemma by contradiction. Assume u is the node with an inconsistent partition type and the minimum subtree.

If u is a leaf node in $T(\texttt{rt})$, then $f(u) = +\infty$ for $u \in C$ and $g(u) = +\infty$ for $u \in D$. The former results in type $\hat{T}$, and the latter results in type $\tilde{T}$, matching the resulting partition in both cases. So u cannot be a leaf node, and by minimum assumption, all its children should have consistent types.

If u is assigned type $\tilde{T}$ but does not connect to any depot in the subtree, then $u \in C$. By Eq. (3.1), there exists a child v of u connected to u and assigned type $\tilde{T}$, which means v also has an inconsistent type, contradicting the minimum assumption.

If u is assigned type $\tilde{T}$ but connects to more than one depot in the subtree, then u can be either a customer or a depot. If u is a customer, then by Eq. (3.1), the reconstruction only assigns one such v to connect to u and has type $\tilde{T}$. So the other child connects to u, and a depot in its subtree has the wrong type $\hat{T}$, contradicting the minimum assumption. If u is a depot, then all children cannot both connect to u and be assigned $\tilde{T}$, which means the child connects to it and a depot in the subtree has the wrong type $\hat{T}$, contradicting the minimum assumption.

If u is assigned type $\hat{T}$, then $u \in C$ by the assignment rule. If u connects to any depot in the subtree, then by Eq. (3.2), every child either connects to u or is assigned type $\tilde{T}$. The

child connecting to v and a depot in its subtree has the wrong type $\hat{T}$, contradicting the minimum assumption. □

Lemma 2. *Given the input as in Theorem 1, the reconstruction based on the value function assignment in Eq. (3.1)-(3.2) produce the minimum* $\tilde{T}(\texttt{rt})$ *and* $\hat{T}(\texttt{rt})$.

Proof. We prove the lemma by induction. Let $N = m + n$, where m is the number of depots, and n is the number of customers. When $N = 1$, there is either no customer to visit or no depot for a salesman to start from, so the optimality holds trivially. Suppose the algorithm gives an optimal solution for all $N \in \{1, 2, \cdots, m + n - 1\}$. Now, for the case when $N = m + n$, assume a partition with smaller total weights exists. We prove that this assumption leads to a contradiction; hence, the statement holds for the case when $N = m+n$. Let v be the root node of the smallest subtree that v has the same partition type in both the optimal partition and our partition, but the optimal partition gives a smaller total weight of the remaining edges. Such node v exists because the root is given the same partition type and, by assumption, does not minimize the weights. Let W_e denote the total weights of remaining edges in the subtree rooted at v under our partition and W_o of that under the optimal partition. By assumption $W_o < W_e$. Now, we construct a partition for the subtree rooted at v with total weights no larger than W_o and no smaller than W_e to get the contradiction. First, assign all child nodes of v the same partition type as the optimal partition and apply our reconstruction rule to get partitions for their subtrees. Then, connect v and its children in the same way as the optimal partition. The new partition is correct because both the connections between v and its children and its children's partition types are the same as the optimal partition. Let W denote the total weights of the new partition. By the induction hypothesis, all subtrees rooted at nodes in $C(v)$ are optimal, which means the total weight of remaining edges in the subtree rooted at v is no larger than W_o, i.e., $W \leq W_o$. On the other hand, by Eq. (3.1)-(3.2), our partition selects the optimal way of connecting v and its children, which means $W_e \leq W$. So, we have $W_e \leq W \leq W_o < W_e$, a contradiction. Hence, the statement holds for $N = m + n$, and by induction, it holds for all

N. □

Proof for Thm. 1. By lemma 1, all nodes have consistent partition types. If the partition is incorrect, a connected component exists with either more than one depot or zero depots. In both cases, the component's root has an inconsistent partition type, contradicting lemma 1. So, the partition is correct. By lemma 2, $\tilde{T}(\texttt{rt})$ is the optimal partition. □

B.3 Proof for Theorem 3

A sufficient condition for existence is the density of stations to be large enough, which is

Theorem 3. *Gridding the map with length* $\frac{e}{(1+\sqrt{5})k}$ *into even squares. Suppose in grid* i*, the number of stations is* $N_s[i]$ *and the number of customers is* $N_c[i]$*, then there exists a solution if* $N_s[i] \geq 1$ *and* $N_c[i]/N_s[i] \leq r$ *for all* i*.*

Proof. Since the number of squares is even, there is a tour to visit each grid exactly once. Within grid i, there is a valid way to visit all customers since $N_c[i]/N_s[i] \leq r$ and the longest distance in a grid is $\frac{\sqrt{2}e}{(1+\sqrt{5})k} < \frac{e}{2k}$. For two grids to visit consecutively, the longest two-segment poly-line is $\frac{e}{k}$, which means it is valid to visit any customer between the visit of two stations in two grids. □

References

[1] E. Benavent and A. Martínez, "Multi-depot multiple TSP: a polyhedral study and computational results," *Annals of Operations Research*, vol. 207, pp. 7–25, 2013.

[2] S. Yadlapalli, W. A. Malik, S. Darbha, and M. Pachter, "A Lagrangian-based algorithm for a multiple depot, multiple traveling salesmen problem," *Nonlinear Analysis: Real World Applications*, vol. 10, no. 4, pp. 1990–1999, 2009.

[3] P. Oberlin, S. Rathinam, and S. Darbha, "A transformation for a heterogeneous, multiple depot, multiple traveling salesman problem," in *2009 American control conference*, IEEE, 2009, pp. 1292–1297.

[4] K. Sundar and S. Rathinam, "An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2015, pp. 366–371.

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

[6] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations research*, vol. 21, no. 2, pp. 498–516, 1973.

[7] J. Gao, L. Zhen, G. Laporte, and X. He, "Scheduling trucks and drones for cooperative deliveries," *Transportation Research Part E: Logistics and Transportation Review*, vol. 178, p. 103 267, 2023.

[8] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied mathematics*, vol. 10, no. 1, pp. 196–210, 1962.

[9] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *Journal of the ACM (JACM)*, vol. 7, no. 4, pp. 326–329, 1960.

[10] Gurobi Optimization, LLC, *Gurobi Optimizer Reference Manual*, 2023. URL: https://www.gurobi.com.

[11] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. USA: Princeton University Press, 2007, ISBN: 0691129932.

[12] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," in *Operations Research Forum*, Springer, vol. 3, 2022, p. 20.

[13] L. Xin, W. Song, Z. Cao, and J. Zhang, "NeuroLKH: Combining deep learning model with Lin-Kernighan-Helsgaun heuristic for solving the traveling salesman problem," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7472–7483, 2021.

[14] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities.," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[15] Y. Luo, "Design and Improvement of Hopfield network for TSP," in *Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science*, 2019, pp. 79–83.

[16] J. Perera, S.-H. Liu, M. Mernik, M. Črepinšek, and M. Ravber, "A Graph Pointer Network-Based Multi-Objective Deep Reinforcement Learning Algorithm for Solving the Traveling Salesman Problem," *Mathematics*, vol. 11, no. 2, p. 437, 2023.

[17] W. Kool, H. van Hoof, and M. Welling, "Attention, Learn to Solve Routing Problems!" In *International Conference on Learning Representations*, 2018.

[18] X. Bresson and T. Laurent, "The Transformer Network for the Traveling Salesman Problem," *CoRR*, vol. abs/2103.03012, 2021.

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

[20] M. Vali and K. Salimifard, "A constraint programming approach for solving multiple traveling salesman problem," in *The Sixteenth International Workshop on Constraint Modelling and Reformulation*, 2017, pp. 1–17.

[21] Z. Wang, X. Fang, H. Li, and H. Jin, "An improved partheno-genetic algorithm with reproduction mechanism for the multiple traveling salesperson problem," *IEEE Access*, vol. 8, pp. 102 607–102 615, 2020.

[22] M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "Modification of the ant colony optimization for solving the multiple traveling salesman problem," *Romanian Journal of Information Science and Technology*, vol. 16, no. 1, pp. 65–80, 2013.

[23] V. Pandiri and A. Singh, "A hyper-heuristic based artificial bee colony algorithm for k-interconnected multi-depot multi-traveling salesman problem," *Information Sciences*, vol. 463, pp. 261–281, 2018.

[24] A. Ceselli and G. Righini, "The Electric Traveling Salesman Problem: properties and models," Tech. Rep., Nov. 2020. DOI: 10.13140/RG.2.2.17712.99848.

[25] R. Roberti and M. Wen, "The electric traveling salesman problem with time windows," *Transportation Research Part E: Logistics and Transportation Review*, vol. 89, pp. 32–52, 2016.

[26] R. Stern, N. Sturtevant, A. Felner, S. Koenig, H. Ma, T. Walker, J. Li, D. Atzmon, L. Cohen, T. Kumar, *et al.*, "Multi-agent pathfinding: Definitions, variants, and benchmarks," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 10, 2019, pp. 151–158.

[27] O. Salzman and R. Stern, "Research challenges and opportunities in multi-agent path finding and multi-agent pickup and delivery problems," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1711–1715.

[28] J. Yu and S. LaValle, "Structure and intractability of optimal multi-robot path planning on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 27, 2013, pp. 1443–1449.

[29] M. Čáp, P. Novák, A. Kleiner, and M. Seleckỳ, "Prioritized planning algorithms for trajectory coordination of multiple mobile robots," *IEEE transactions on automation science and engineering*, vol. 12, no. 3, pp. 835–849, 2015.

[30] M. Barer, G. Sharon, R. Stern, and A. Felner, "Suboptimal variants of the conflict-based search algorithm for the multi-agent pathfinding problem," in *Proceedings of the International Symposium on Combinatorial Search*, vol. 5, 2014, pp. 19–27.

[31] E. Lam, P. Le Bodic, D. Harabor, and P. J. Stuckey, "Branch-and-cut-and-price for multi-agent path finding," *Computers & Operations Research*, vol. 144, p. 105 809, 2022.

[32] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial Intelligence*, vol. 219, pp. 40–66, 2015.

[33] N. Greshler, O. Gordon, O. Salzman, and N. Shimkin, "Cooperative multi-agent path finding: Beyond path planning and collision avoidance," in *2021 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, IEEE, 2021, pp. 20–28.

[34] S. Choudhury, K. Solovey, M. J. Kochenderfer, and M. Pavone, "Efficient large-scale multi-drone delivery using transit networks," *Journal of Artificial Intelligence Research*, vol. 70, pp. 757–788, 2021.

[35] S. Choudhury, K. Solovey, M. Kochenderfer, and M. Pavone, "Coordinated multi-agent pathfinding for drones and trucks over road networks," *arXiv preprint arXiv:2110.08802*, 2021.

[36] C. C. Murray and R. Raj, "The multiple flying sidekicks traveling salesman problem: Parcel delivery with multiple drones," *Transportation Research Part C: Emerging Technologies*, vol. 110, pp. 368–398, 2020.

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

[38] R. G. Mbiadou Saleu, L. Deroussi, D. Feillet, N. Grangeon, and A. Quilliot, "An iterative two-step heuristic for the parallel drone scheduling traveling salesman problem," *Networks*, vol. 72, no. 4, pp. 459–474, 2018.

[39] D. Sacramento, D. Pisinger, and S. Ropke, "An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones," *Transportation Research Part C: Emerging Technologies*, vol. 102, pp. 289–315, 2019.

[40] W.-C. Chiang, Y. Li, J. Shang, and T. L. Urban, "Impact of drone delivery on sustainability and cost: Realizing the UAV potential through vehicle routing optimization," *Applied energy*, vol. 242, pp. 1164–1175, 2019.

[41] F. Tamke and U. Buscher, "A branch-and-cut algorithm for the vehicle routing problem with drones," *Transportation Research Part B: Methodological*, vol. 144, pp. 174–203, 2021.

[42] C. Chen, E. Demir, and Y. Huang, "An adaptive large neighborhood search heuristic for the vehicle routing problem with time windows and delivery robots," *European journal of operational research*, vol. 294, no. 3, pp. 1164–1180, 2021.

[43] H. Li, J. Chen, F. Wang, and Y. Zhao, "Truck and drone routing problem with synchronization on arcs," *Naval Research Logistics (NRL)*, vol. 69, no. 6, pp. 884–901, 2022.

[44] J. G. Carlsson and S. Song, "Coordinated logistics with a truck and a drone," *Management Science*, vol. 64, no. 9, pp. 4052–4069, 2018.

[45] M. Lin, Y. Chen, R. Han, Y. Chen, *et al.*, "Discrete optimization on truck-drone collaborative transportation system for delivering medical resources," *Discrete Dynamics in Nature and Society*, vol. 2022, 2022.

[46] K. Helsgaun, *An Extension of the Lin-Kernighan-Helsgaun TSP Solver for Constrained Traveling Salesman and Vehicle Routing Problems: Technical report*, English. Roskilde Universitet, Dec. 2017, p. 60.

[47] C. R. C. in Distribution Management, *Mdvrp*, http://neumann.hec.ca/chairedistributique/data/mdvrp/, 2023.

[48] S. Zhang, W. Zhang, Y. Gajpal, and S. Appadoo, "Ant colony algorithm for routing alternate fuel vehicles in multi-depot vehicle routing problem," *Decision Science in Action: Theory and Applications of Modern Decision Analytic Optimisation*, pp. 251–260, 2019.

[49] B. Peng, L. Wu, Y. Yi, and X. Chen, "Solving the multi-depot green vehicle routing problem by a hybrid evolutionary algorithm," *Sustainability*, vol. 12, no. 5, p. 2127, 2020.

[50] M. E. H. Sadati and B. Çatay, "A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem," *Transportation Research Part E: Logistics and Transportation Review*, vol. 149, p. 102 293, 2021.

[51] F. Blahoudek, T. Brázdil, P. Novotnỳ, M. Ornik, P. Thangeda, and U. Topcu, "Qualitative controller synthesis for consumption Markov decision processes," in *International Conference on Computer Aided Verification*, Springer, 2020, pp. 421–447.

[52] OpenStreetMap contributors, *Planet dump retrieved from https://planet.osm.org*, https://www.openstreetmap.org, 2017.

[53] Bluebikes, *System data*, https://s3.amazonaws.com/hubway-data/current_bluebikes_stations.csv, 2023.

[54] W. Ho, G. T. Ho, P. Ji, and H. C. Lau, "A hybrid genetic algorithm for the multi-depot vehicle routing problem," *Engineering applications of artificial intelligence*, vol. 21, no. 4, pp. 548–557, 2008.

[55] S. Salhi, A. Imran, and N. A. Wassan, "The multi-depot vehicle routing problem with heterogeneous vehicle fleet: Formulation and a variable neighborhood search implementation," *Computers & Operations Research*, vol. 52, pp. 315–325, 2014.

[56] S. Geetha, P. Vanathi, and G. Poonthalir, "Metaheuristic approach for the multi-depot vehicle routing problem," *Applied Artificial Intelligence*, vol. 26, no. 9, pp. 878–901, 2012.

[57] B. Gavish and S. C. Graves, "The travelling salesman problem and related problems," 1978.

[58] S. Chinnasamy, M. Ramachandran, M. Amudha, and K. Ramu, "A review on hill climbing optimization methodology," *Recent Trends in Management and Commerce*, vol. 3, no. 1, 2022.