A Hierarchical Framework for Solving the Constrained Multiple Depot Traveling Salesman Problem

Ruixiao Yang and Chuchu Fan, *Member, IEEE*

***Abstract*—The Multiple Depot Traveling Salesman Problem (MDTSP) is a variant of the NP-hard Traveling Salesman Problem (TSP) with more than one salesman to jointly visit all destinations, commonly found in task planning in multi-agent robotic systems. Traditional MDTSP overlooks practical constraints like limited battery level and inter-agent conflicts, often leading to infeasible or unsafe solutions in reality. In this work, we incorporate energy and resource consumption constraints to form the Constrained MDTSP (CMDTSP). We design a novel hierarchical framework to obtain high-quality solutions with low computational complexity. The framework decomposes a given CMDTSP instance into manageable sub-problems, each handled individually via a TSP solver and heuristic search to generate tours. The tours are then aggregated and processed through a Mixed-Integer Linear Program (MILP), which contains significantly fewer variables and constraints than the MILP for the exact CMDTSP, to form a feasible solution efficiently. We demonstrate the performance of our framework on both real-world and synthetic datasets. It reaches a mean 12.48% optimality gap and 41.7x speedup over the exact method on common instances and a 5.22%∼14.84% solution quality increase with more than 79.8x speedup over the best baseline on large instances where the exact method times out.**

***Index Terms*—Combinatorial optimization, multi-robot systems, path planning for multiple mobile robots or agents, planning, scheduling, and coordination, task planning.**

I. 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 *cities* (or destinations) 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 cities jointly. Such a problem arises in various real-world robotics applications such as logistics scheduling, warehouse robots, healthcare routing for metropolitan cities, and unmanned aerial vehicles (UAVs) [2], [3], [4].

The optimal routes given by the solution of an *unconstrained* MDTSP might not be realizable in practice due to the energy consumption and limited energy capability of each robot (also called salesmen henceforth). For example, due to the limited battery capacity, 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 in terms of 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 Constrained MDTSP (CMDTSP) by introducing the energy and resource constraints into MDTSP. In CMDTSP, 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 CMDTSP seeks the shortest set of routes for m salesmen that start from different depots, jointly visit a set of cities, 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 CMDTSP is also (NP-)hard as any TSP can be reduced to a CMDTSP with $m = 1$ and zero energy consumption rate.

Literature on MDTSP is very scarce, and existing works on related problems either use metaheuristic algorithms such as Ant-Colony Optimization-based methods [5], [6], [7], Simulated Annealing-based methods [8], [9], [10], Evolutionary Algorithm [11], and neighborhood search [12] or directly solve a Mixed-Integer Linear Programming (MILP) [4], [13], [14]. While the former methods have large optimality gaps, the latter methods do not scale to large problems due to the high computational complexity.

In this letter, we propose a novel hierarchical framework for solving CMDTSP, balancing solution quality and computational complexity. We first allocate cities to salesmen via a heuristic method involving the Minimum Spanning Tree (MST) of a graph consisting of the cities and the depots. Then, we use a TSP solver to determine each salesman's visit order in the assigned cities. Each salesman then proposes multiple potential feasible routes by adding charging stations to their routes of cities. Finally, the proposed solutions are collected, and the best feasible solution is selected using a MILP-based congestion control formulation. The numbers of both integer and real variables grow linearly to the number of cities in our MILP instead of quadratic and fourth order in the exact method. In experiments, our framework outperforms selected baselines in both solution quality and

Manuscript received 10 December 2023; accepted 10 April 2024. Date of publication 16 April 2024; date of current version 6 May 2024. This letter was recommended for publication by Associate Editor E. Pastore and Editor C.-B. Yan upon evaluation of the reviewers' comments. This work was supported by National Science Foundation CAREER under Grant CF-2238030. *(Corresponding author: Ruixiao Yang.)*

The authors are with the Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: ruixiao@mit.edu; chuchu@mit.edu).

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2024.3389817, provided by the authors.

Digital Object Identifier 10.1109/LRA.2024.3389817

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, and 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 cities where the exact method times out on 30 cities with the same time limit. The code, proof for theorems, and MILP formulation for CMDTSP are publicly available in the full version of the letter uploaded on the project website.

The main **contributions** of the letter are as follows: (1) We formulate CMDTSP as a new variant of MDTSP to model energy consumption and replenishment of salesmen in the real world; (2) We propose a novel hierarchical pipeline for solving CMDTSP and show that the overall computational complexity of the proposed algorithm is much lower than a pure MILP formulation; (3) We illustrate through various numerical experiments against multiple baselines on hybrid and synthetic datasets that the proposed method produces high-quality solutions while maintaining scalability.

II. Related Work

A. Methods for TSP and Its Variants

Exact methods for TSP and its variants, beyond brute force enumeration, include Dynamic Programming [15] and MILP [16]. Existing tools such as Gurobi [17] and Concorde [18] 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.

Approximation and heuristic algorithms are significantly more computationally efficient than exact methods, but they provide only sub-optimal solutions. Among algorithms with worst-case guarantee, the Christofides Algorithm [19] 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 [20] stands out as the best heuristic algorithm for TSP. It begins with a TSP tour and iteratively removes several edges (with 2 or 3 being 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 [21], 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.

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 [22], which has recently been improved [23]. Another variant of RNN employed for TSP is the Pointer Network [24]. 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 [25], [26].

None of the existing approaches can be directly applied to guarantee the correct results for CMDTSP.

B. CMDTSP-Related TSP Variants

Multiple TSP (MTSP) is the basic problem modeling the multiagent 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 [27] or constraint programming [28], and metaheuristic algorithms include Genetic Algorithm (GA) [29], Ant Colony Optimization (ACO) [30], and Artificial Bee Colony algorithm (ABC) [31].

Electric TSP (ETSP) introduces energy constraints and charging stations into standard TSP. The problem is first formally stated by [32] 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) [33], which is claimed to be easier [32]. Our work can be viewed as a multiagent variant of the ETSP problem, where we provide an exact MILP formulation and a scalable hierarchical framework.

III. Problem Formulation

The CMDTSP 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 cities (destinations) so each city 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 *city* aligning with the term in the TSP, which may refer to arbitrary targets or destinations in robotic tasks.

Formally, the problem is defined on a complete, undirected graph $\mathcal{G} = (V, E) := \mathcal{G}(V)$, where V represents the set of vertices. The vertex set V is partitioned into the union of three sets D, C, and S, where $D = \{d_1, d_2, \ldots, d_m\}$ denotes the set of m depots (i.e., starting and ending locations of the salesmen's tours), $C = \{c_1, c_2, \ldots, c_n\}$ is the set of n cities, and $S = \{s_1, s_2, \ldots, s_l\}$ constitutes the set of l stations. Each edge $(i, j) \in E$ is associated with a weight $c(i, j) \geq 0$, which represents 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 $s \in S$. In this letter, we consider all salesmen homogeneous, i.e., $k_i = k$ and $e_i = e$ for all $i = 1, 2, \ldots, m$. 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 any other pertinent metric. Furthermore, each salesman's energy level is presumed to be fully replenished upon visiting any station.

Problem 1 (CMDTSP): Given a complete graph $\mathcal{G} = (V, E)$ where $V = D \cup C \cup S$ and m salesmen starting from different depots in D, find a set of m tours $\{t_i\}_{i=1}^{|D|}$, one per salesman, such that: (1) each tour begins and ends at the same depot; (2) each city in 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 $i = 1, 2, \ldots, l$; and (5) the total cost is minimized.

It is also a general version of a more commonly studied subproblem, Multiple Depots TSP (MDTSP) [2], [34].

IV. Methodology

Similar to the original TSP, CMDTSP can also be formulated as a MILP (see project website). However, the complexity of

Algorithm 1: Framework for Solving CMDTSP.

1: T, P, Solution $= \emptyset$
2: $l = 0$
3: $\{C_i\}_{i=1}^{|D|} =$ Partition$(C; D)$ ▷ Assign cities to salemen
4: **for** $i \in \{1, 2, \ldots, m\}$ **do**
5: $\mathscr{G}_i = \mathscr{G}(C_i \cup \{d_i\})$
▷ Form a complete graph of $C_i \cup \{d_i\}$
6: $t_i =$ TSP$(\mathscr{G}_i)$ ▷ Find the TSP solution of graph $\mathscr{G}_i$
7: **for** $(u, v) \in t_i$ **do**
8: $\mathscr{G}_i = \mathscr{G}(\{u, v\} \cup S)$
▷ Form a complete graph of $\{u, v\} \cup S$
9: $P_i(u, v) =$ k-shortest-path$(u, v; \mathscr{G}_i; k)$
▷ Find the top-k shortest paths from u to v
10: **end for**
11: **end for**
12: $P = \cup_{i=1}^{|D|} P_i$
13: Solution $=$ CongestionControl(P)
▷ Form a solution satisfying all constraints from P
14: **return** Solution

such a MILP scales exponentially with the number of cities 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, cities, or stations.

The intuition of our framework is straightforward: by decomposing the CMDTSP into smaller subproblems, we aim to reduce the size of the MILP, the primary bottleneck for scaling. To this end, we propose a novel hierarchical framework that utilizes a heuristic and smaller MILP, shown in Algorithm 1. Initially, cities 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 cities (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 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. 1 shows an illustrative example of route planning in Manhattan.

A. City Assignment

We first introduce Algorithm 2, the city assignment component in the framework.

To begin with, we construct a complete graph $\mathscr{G} = \mathscr{G}(V)$ with $V = D \cup C$ of all depots D and cities C and find its minimum spanning tree $T(\texttt{rt})$ rooted at some node $\texttt{rt} \in D \cup C$. Then, we split $T(\texttt{rt})$ into m components 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 T_i contains exactly one depot d_i.

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

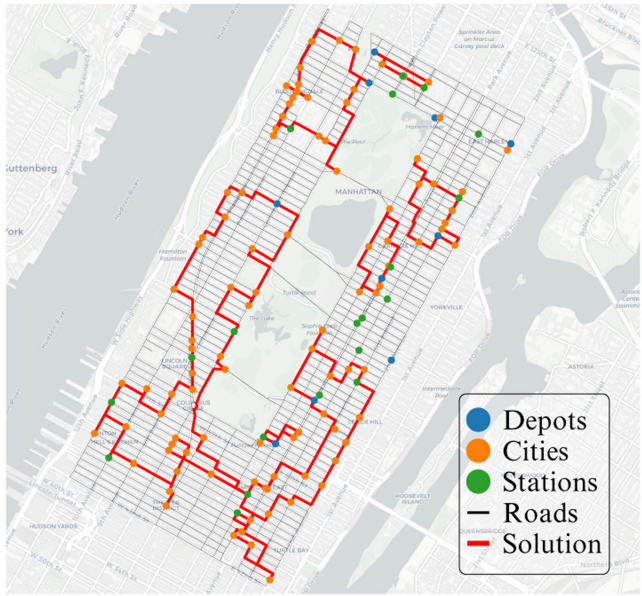

Fig. 1. Solution of CMDTSP in Manhattan: salesmen start from depots to collaboratively visit all cities and always keep their energy above zero.

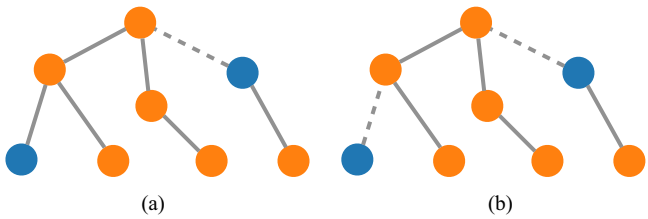

Fig. 2. Illustration of partitions. The orange nodes represent cities, and the blue nodes represent depots. Solid lines indicate the edges remaining after partitioning, and dashed lines represent the edges removed after the partition. In Fig. (a), we disconnect one depot from the root node to form two 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.

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 $\hat{T}(u)$ does not exist. Fig. 2 shows an illustration of two partitions.

Next, define $f : V \to \mathbb{R}$ such that $f(u)$ represents the minimum total edge weights of $\tilde{T}(u)$ and $g : V \to \mathbb{R}$ such that $g(u)$ represents the minimum total edge weights of $\hat{T}(u)$. 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 $H(u)$ be the set of children of node u. Functions f and g can be computed as:

$$f(u) = \begin{cases} \min\limits_{v \in H(u)} \{f(v) + c(u, v) \\ \quad + \sum\limits_{v' \in H(u) \setminus \{v\}} \min\{f(v'), g(v') + c(u, v')\} \Big\}, & u \in C, \\ \sum\limits_{v \in H(u)} \min\{f(v), g(v) + c(u, v)\}, & u \in D, \end{cases} \tag{1}$$

Algorithm 2: Assign Cities to Salesmen.

Input: City set C, depot set D

1: $\mathscr{G} = \mathscr{G}(C \cup D)$ ▷ Form a complete graph of $C \cup D$

2: $T = \text{MST}(\mathscr{G})$ ▷ Compute a minimum spanning tree

3: Compute minimum weight functions f, g ▷ Using Equation (1), (2)

4 $\cup_{i=1}^{|D|} T_i = \text{Part}(T, f, g)$ ▷ Partition tree T based on f, g

5: **return** $\{C \cap T_i\}_{i=1}^{|D|}$

$$g(u) = \begin{cases} \sum\limits_{v \in H(u)} \min\{f(v), g(v) + c(u,v)\}, & u \in C, \\ +\infty, & u \in D. \end{cases} \tag{2}$$

We offer brief insights here and provide the derivation details in the full version of the letter uploaded on the project website. For a partition $\tilde{T}(u)$ or $\hat{T}(u)$, all partitions on its subtrees are also of type $\tilde{T}$ or $\hat{T}$. 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 (1)-(2).

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

Theorem 1: Given a graph $\mathscr{G} = \mathscr{G}(V)$ with $V = D \cup C$ and an edge-weight function c, the partition $\tilde{T}(\texttt{rt})$ recovered from value function assignment in (1)-(2) partitions the minimum spanning tree T of $\mathscr{G}$ into m subtrees $\{T_i\}_{i=1}^m$ such that $d_i \in T_i$, and minimizes the total edge weights in the connected components, i.e., $\sum_{i=1}^{|D|} \sum_{(u,v) \in T_i} c(u,v)$.

Theorem 2: Algorithm 2 for MDTSP has an approximation ratio of 2 for nodes on 2D-plane.

The proof of Theorem 1 and Theorem 2 is provided in the full version of the letter uploaded on the project website.

After assigning cities to salesmen, we form graphs for each salesman with corresponding depot and cities and solve TSPs to determine the order of visits for each salesman. This is a standard TSP; any off-the-shelf TSP solver can be plugged in here. Since 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 work, we use the state-of-the-art heuristic solver LKH-3 [35], which efficiently produces solutions with a very small optimality gap and has good scalability.

Next, given a tour of cities, 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 $\sum_{i=1}^{|D|} k|t_i| = k(|C| + |D|)$, so there are $\prod_{i=1}^{|D|} \prod_{j=1}^{|t_i|} k = k^{|C|+|D|}$ 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.

B. 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 resource limits at all stations. We set up 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:

$$\min \sum_{i=1}^{m} \sum_{j=1}^{|t_i|} \sum_{h=1}^{k} c_{i,j,h} \cdot \beta_{i,j,h}, \tag{3a}$$

$$\text{s.t. } \beta_{i,j,h} \in \{0,1\}, i \in [m], j \in [|t_i|], h \in [k], \tag{3b}$$

$$\sum_{h=1}^{k} \beta_{i,j,h} = 1, i \in [m], j \in [|t_i|], \tag{3c}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{|t_i|} \sum_{h=1}^{k} \beta_{i,j,h} \cdot p_{i,j,h}[s] \leq r_s, s \in [l], \tag{3d}$$

$$0 \leq \gamma_{i,j}, i \in [m], j \in [|t_i| + 1], \tag{3e}$$

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

$$\gamma_{i,j} \leq \sum_{h=1}^{k} \beta_{i,j,h} \cdot q_{2,i,j-1,h}, i \in [m], j \in [2, |t_i| + 1], \tag{3g}$$

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

where $|t_i|$ refers to the length of salesman i's tour t_i. The objective (3a) is the overall cost to minimize. Constraints (3b) and (3c) ask to choose exactly one path out of k proposed paths for each edge. Constraint (3d) is the station's resource limit.

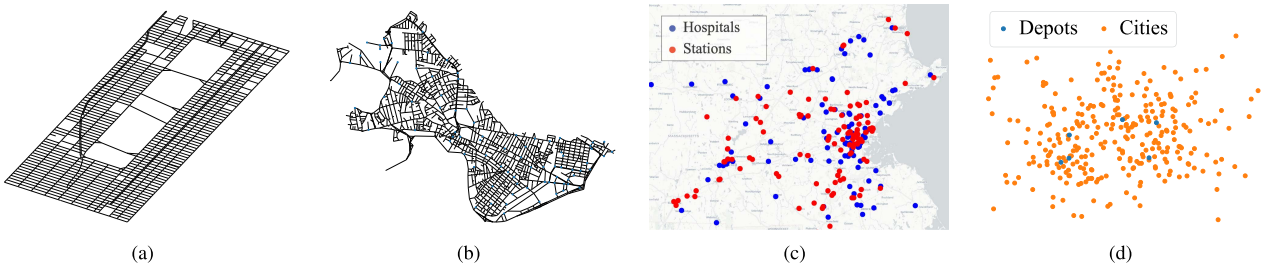

Fig. 3. Four Datasets for experiments. (a) The driving road map of Manhattan, New York. (b) The driving road map of Cambridge, Massachusetts. (c) Hospitals and Tesla's Supercharger Stations in Massachusetts. (d) The existing benchmark for MDVRP [36].

Constraints (3e)–(3h) are the energy constraints for salesmen when passing through the chosen paths.

Now, we analyze the complexity of our proposed CMDTSP solver. Given a graph $\mathscr{G} = \mathscr{G}(V)$ with $|V| = |D \cup C \cup S| = m + n + l$ and parameter k denoting the number of paths proposed for each edge, the MILP (3) has nk real variables, $mk + nk$ integer variables and $5\tilde{}m^2 + 5mn + l$ 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.

V. Experiments

We empirically validate our method in this section. In section V-A, we compare our method with three baselines, Ant Colony Optimization (ACO) [7], Hybrid Evolutionary Algorithm (HEA) [11], and Hybrid Variable Neighborhood Search (HVNS) [12], on real-world maps and existing benchmarks. In section V-B, we compare our method with a naïve MILP formulation for the scalability and solution quality. In section V-C, we study our framework's partition and TSP solver components to justify our choices. In section V-D, 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 V-E, 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 [17] 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 letters [7], [11], [12].

A. 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 [37], Cambridge, and Massachusetts [38]. For the Manhattan map, instances are generated by uniformly sampling depots, cities, and stations from the map. For the Cambridge map, stations are uniformly sampled from Bluebikes stations in 2023 [39], and depots and cities 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 cities are uniformly sampled from hospitals. We adapt the existing MDVRP benchmarks [36] by randomly turning a fraction of cities into stations and uniformly sampling a fixed amount of depots, cities, 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 $|D| = 5$, cities $|C| = 30$, and stations $|S| = 20$, and 100 large instances consisting of $|D| = 10$, $|C| = 100$, $|S| = 20$. 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. An example from each data source is provided in Fig. 3.

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 CMDTSP. HVNS initializes the solution by 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 Tables I and II. 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.

B. Compare With Exact Algorithm

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

Datasets: To measure the solution quality, we randomly generate 100 instances of $|C| = 15$, $|D| = 3$, $|S| = 20$, 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 $|S| = 20$ and the number of salesmen

TABLE I
RESULTS ON 1000 SMALL INSTANCES OF 5 DEPOTS, 30 CITIES, 20 STATIONS

	Manhattan			Cambridge			Massachusetts			MDVRP Benchmark		
Method	Length	Feas.	Time(s)	Length	Feas.	Time(s)	Length	Feas.	Time(s)	Length	Feas.	Time(s)
ACO	32.93	0.95	**1.02**	395.75	0.92	**1.04**	830.13	**0.99**	**1.05**	907.83	0.84	**1.26**
HEA	31.65	0.62	21.57	380.12	0.63	21.62	848.34	0.97	10.68	909.78	0.43	14.45
HVNS	28.26	**1.00**	12.95	338.86	**1.00**	14.80	748.50	**0.99**	9.71	843.66	**1.00**	15.25
Ours	**24.96**	**1.00**	1.90	**313.45**	**1.00**	1.91	**703.82**	**0.99**	1.89	**783.06**	0.91	2.04

The best results are bolded.

TABLE II
RESULTS ON 100 LARGE INSTANCES OF 10 DEPOTS, 100 CITIES, 20 STATIONS

	Manhattan			Cambridge			Massachusetts			MDVRP Benchmark		
Method	Length	Feas.	Time(s)	Length	Feas.	Time(s)	Length	Feas.	Time(s)	Length	Feas.	Time(s)
ACO	65.50	0.34	3.63	881.50	0.14	4.45	1685.88	0.99	4.16	2156.11	0.41	4.03
HEA	50.61	0.01	420.62	824.00	0.01	473.21	1619.08	0.95	310.27	1931.04	0.09	329.39
HVNS	47.70	0.94	270.19	653.54	0.48	300.75	1373.35	0.99	212.72	1642.98	0.67	287.45
Ours	**41.74**	**0.98**	**2.54**	**619.43**	**0.56**	**3.11**	**1228.60**	**0.99**	**2.78**	**1511.19**	**0.70**	**2.32**

The best results are bolded.

$|D| = 5$, the number of cities $|C|$ varies from 100 to 1100 for our framework and from 5 to 20 for the MILP solver; (2) fix the number of stations $|S| = 20$ and the average cities visited by each salesman, i.e., $|C|/|D| = 5$, the number of salesmen $|D|$ varies from 20 to 220 for our framework and $|D|$ from 1 to 5 for the MILP solver. To ensure the feasibility rate, we empirically set $r = 0.4 \cdot |C|/|S|$.

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 Section V-A.

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 Fig. 4(a). We believe this is a good performance as the worst-case guarantee for standard TSP is around 50%. Fig. 4(c) 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 cities, 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 be able to scale up to a large problem size.

Comparing the first column in Fig. 4(c), the running time is shorter when there are more salesmen given the same number of cities. We empirically evaluate the running time of different components in our framework. Fig. 4(b) shows the changes in relative time consumption for subroutines in the entire algorithm as the number of cities 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 running time of the partition increases slightly, but the average size of TSP for salesmen decreases, dramatically reducing the overall running time.

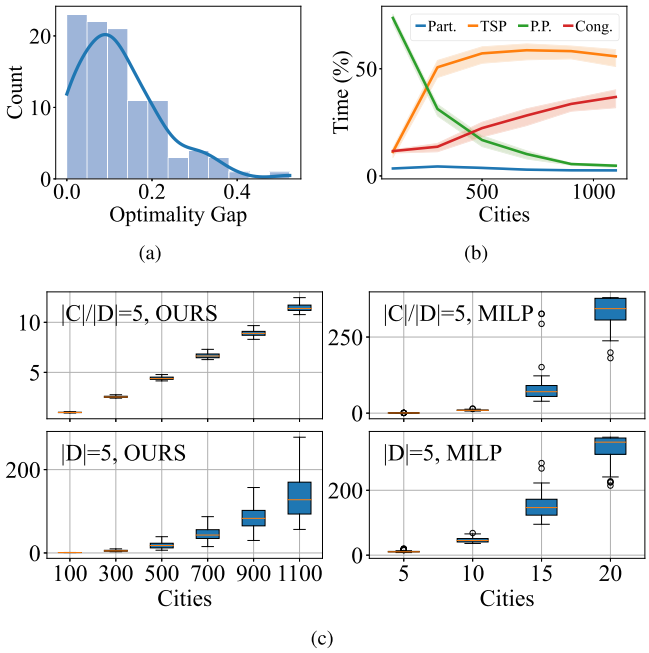

Fig. 4. Results on comparison experiments. (a) The distribution of optimality gaps with a mean value of 12.48%. (b) The change in the percentage of running time for each part in the framework. (c) Comparison on Scalability. All Y-axes are time in seconds.

C. 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 $|D| = 5$, $|C| = 150$, $|S| = 20$, and 100 randomly generated instances of $|D| = 5$, $|C| = 250$, $|S| = 20$ 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 city to its closest salesman, and the K-Means clustering algorithm, which first

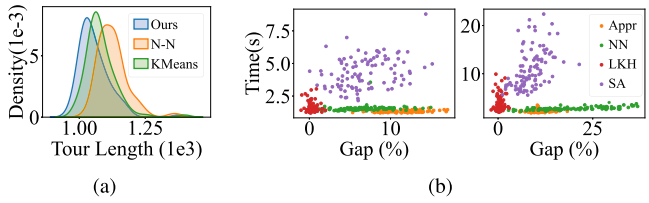

Fig. 5. Results of studies on components. (a) Keep the TSP solver to be LKH, our partition algorithm outperforms all baselines; (b) keep the Partition algorithm to be MST based, LKH solver produces the best solution quality and competitive computation time.

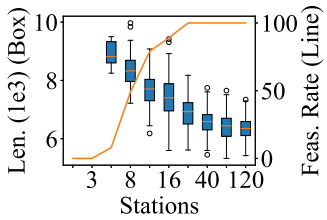

Fig. 6. Effect of the dispersion of stations. There are no feasible solutions for cases with 1 or 3 station(s).

clusters the cities 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 [25], approximation solver (Christofides algorithm), heuristic solver (LKH), and metaheuristic solver (SA).

Results: Results in Fig. 5(a) show that our partition method beats both NN and KMeans. The result in the left figure of Fig. 5(b) shows that LKH gives much smaller gaps than others within acceptable running time. The right one of Fig. 5(b) 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.

D. 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 $|D| = 30$, $|C| = 1200$. Then, we vary $|S|$ from $1 \sim 120$ and distribute them uniformly in the square, keeping the total amount of resources $r \cdot |S| = 240$.

Result: As shown in Fig. 6, adding more stations increases feasibility and shortens tour length, indicating that spreading resources more widely eases their use. Based on the 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 cities 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.

TABLE III
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

The proof can be found on the website. In practice, the cost of setting up stations needs to be balanced against their benefits for optimal resource allocation.

E. 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 180 kW charging stations for each instance.

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

Results: The results are shown in Table III. 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.

VI. CONCLUSION

We define a new variant of MDTSP called CMDTSP to capture the energy constraints in real-world applications and propose a novel framework to solve them. Our method efficiently produces high-quality solutions compared to baselines and solves much larger problems than the MILP formulation, verified by the experiments. For future work, we would like to enhance scalability using a neural network for congestion control and improve the partition part with a better heuristic.

REFERENCES

[1] E. Benavent and A. Martínez, "Multi-depot multiple TSP: A polyhedral study and computational results," *Ann. Operations Res.*, 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 Anal., Real World Appl.*, 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 *Proc. IEEE Amer. Control Conf.*, 2009, pp. 1292–1297.
[4] K. Sundar and S. Rathinam, "An exact algorithm for a heterogeneous, multiple depot, multiple traveling salesman problem," in *Proc. IEEE Int. Conf. Unmanned Aircr. Syst.*, 2015, pp. 366–371.

[5] T. Ramadhani, G. F. Hertono, and B. D. Handari, "An ant colony optimization algorithm for solving the fixed destination multi-depot multiple traveling salesman problem with non-random parameters," in *Proc. AIP Conf.*, 2017, vol. 1862, Art. no. 030123.
[6] S. Ghafurian and N. Javadian, "An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems," *Appl. Soft Comput.*, vol. 11, no. 1, pp. 1256–1262, 2011.
[7] S. Zhang, W. Zhang, Y. Gajpal, and S. Appadoo, "Ant colony algorithm for routing alternate fuel vehicles in multi-depot vehicle routing problem," in *Decision Science in Action: Theory and Applications of Modern Decision Analytic Optimisation*. Berlin, Germany: Springer, 2019, pp. 251–260.
[8] T. S. Rao, "A simulated annealing approach to solve a multi traveling salesman problem in a FMCG company," *Mater. Today, Proc.*, vol. 46, pp. 4971–4974, 2021.
[9] Y. Zhang, X. Han, Y. Dong, J. Xie, G. Xie, and X. Xu, "A novel state transition simulated annealing algorithm for the multiple traveling salesmen problem," *J. Supercomput.*, vol. 77, pp. 11827–11852, 2021.
[10] Y. Zhou, W. Xu, Z.-H. Fu, and M. Zhou, "Multi-neighborhood simulated annealing-based iterated local search for colored traveling salesman problems," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 16072–16082, Sep. 2022.
[11] 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, 2020, Art. no. 2127.
[12] M. E. H. Sadati and B. Çatay, "A hybrid variable neighborhood search approach for the multi-depot green vehicle routing problem," *Transp. Res. Part E, Logistics Transp. Rev.*, vol. 149, 2021, Art. no. 102293.
[13] M. Diaby, "Linear programming formulation of the multi-depot multiple traveling salesman problem with differentiated travel costs," in *Traveling Salesman Problem, Theory and Applications*. New York, NY, USA: InTech, 2010, pp. 257–282.
[14] D. Scott, S. G. Manyam, D. W. Casbeer, and M. Kumar, "Market approach to length constrained min-max multiple depot multiple traveling salesman problem," in *Proc. IEEE Amer. Control Conf.*, 2020, pp. 138–143.
[15] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *J. Soc. Ind. Appl. Math.*, vol. 10, no. 1, pp. 196–210, 1962.
[16] C. E. Miller, A. W. Tucker, and R. A. Zemlin, "Integer programming formulation of traveling salesman problems," *J. ACM*, vol. 7, no. 4, pp. 326–329, 1960.
[17] Gurobi Optimization LLC, "Gurobi optimizer reference manual," 2023. [Online]. Available: https://www.gurobi.com
[18] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "The traveling salesman problem," in *The Traveling Salesman Problem*. Princeton, NJ, USA: Princeton Univ. Press, 2011.
[19] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," *Oper. Res. Forum*, vol. 3, no. 1, p. 20, 2022.
[20] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Res.*, vol. 21, no. 2, pp. 498–516, 1973.
[21] 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," in *Proc. Int. Conf. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 7472–7483.
[22] J. J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proc. Nat. Acad. Sci.*, vol. 79, no. 8, pp. 2554–2558, 1982.
[23] Y. Luo, "Design and improvement of hopfield network for TSP," in *Proc. Int. Conf. Artif. Intell. Comput. Sci.*, 2019, pp. 79–83.
[24] 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, 2023, Art. no. 437.
[25] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!," in *Proc. Int. Conf. Learn. Representations*, 2018.
[26] X. Bresson and T. Laurent, "The transformer network for the traveling salesman problem," *CoRR*, vol. abs/2103.03012, 2021.
[27] 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," *Comput. Ind. Eng.*, vol. 129, pp. 14–30, 2019.
[28] M. Vali and K. Salimifard, "A constraint programming approach for solving multiple traveling salesman problem," in *Proc. 16th Int. Workshop Constraint Modelling Reformulation*, 2017, pp. 1–17.
[29] Z. Wang, X. Fang, H. Li, and H. Jin, "An improved parthenogenetic algorithm with reproduction mechanism for the multiple traveling salesperson problem," *IEEE Access*, vol. 8, pp. 102607–102615, 2020.
[30] M. Yousefikhoshbakht, F. Didehvar, and F. Rahmati, "Modification of the ant colony optimization for solving the multiple traveling salesman problem," *Romanian J. Inf. Sci. Technol.*, vol. 16, no. 1, pp. 65–80, 2013.
[31] V. Pandiri and A. Singh, "A hyper-heuristic based artificial bee colony algorithm for k-interconnected multi-depot multi-traveling salesman problem," *Inf. Sci.*, vol. 463, pp. 261–281, 2018.
[32] A. Ceselli and G. Righini, "The electric traveling salesman problem: Properties and models," University of Milan, Milan, Italy, Tech. Rep. 2434/789142, 2020, doi: 10.13140/RG.2.2.17712.99848.
[33] R. Roberti and M. Wen, "The electric traveling salesman problem with time windows," *Transp. Res. Part E, Logistics Transp. Rev.*, vol. 89, pp. 32–52, 2016.
[34] K. Sundar and S. Rathinam, "Generalized multiple depot traveling salesmen problem–polyhedral study and exact algorithm," *Comput. Operations Res.*, vol. 70, pp. 39–55, 2016.
[35] K. Helsgaun, "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems," Roskilde Universitet, Tech. Rep., Dec. 2017.
[36] C. R. C. in Distribution Management, "MDVRP," 2023. [Online]. Available: http://neumann.hec.ca/chairedistributique/data/mdvrp/
[37] F. Blahoudek, T. Brázdil, P. Novotny, M. Ornik, P. Thangeda, and U. Topcu, "Qualitative controller synthesis for consumption Markov decision processes," in *Proc. Int. Conf. Comput. Aided Verification*, 2020, pp. 421–447.
[38] OpenStreetMap contributors, "Planet dump," 2017. [Online]. Available: https://planet.osm.org, https://www.openstreetmap.org
[39] Bluebikes, "System data," 2023. [Online]. Available: https://s3.amazonaws.com/hubway-data/current_bluebikes_stations.csv