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

Ruixiao Yang and Chuchu Fan

***Abstract*— The Multiple Depot Traveling salesman Problem (MDTSP) is a variant of the well known NP-hard Traveling Salesman Problem (TSP) with more than one salesmen to collaboratively visit all destinations, which widely encounters in task or mission planning in multi-agent robotic systems. Traditional MDTSP does not consider constraints such as limited battery level or inter-agent conflicts that are widely seen in practical problem, leading to the high risk of generating infeasible or unsafe solution in practice. In this work, we incorporate realistic constraints on energy and resource consumption into MDTSP to form the Constrained MDTSP (CMDTSP). We design a novel hierarchical framework to solve such CMDTSP with provide high-quality solution and low computational complexity, addressing the problem of lacking good heuristics when solving CMDTSP. The framework decomposes a given large CMDTSP problem into manageable sub-problems, each handled by a salesman individually via an existing solver for Traveling Salesman Problem (TSP) and heuristic search to generate tours. The proposed solutions are then aggregated and processed through a relatively small Mixed-Integer Linear Program (MILP) to form a feasible solution that meets the constraints. Compared with exact method, We reduce the number of real variables in MILP from forth order of cities to linear and the number of integer variables from quadratic to linear. We demonstrate the advantages of our framework on solution quality and running time over existing methods by experiments on both real road maps as well as synthetic datasets. Our framework gives a mean optimality gap 12.48% on small dataset, and a 5.22%∼14.84% solution quality increase with more than 79.8x speedup over the best baseline on large dataset where exact method times out. The code, proof, and MILP formulation are publicly available at https://mit-realm.github.io/CMDTSP/**

I. Introduction

To optimize robots in handling multiple tasks, a crucial step is to plan the mission with the optimal order of tasks. The Traveling Salesman Problem (TSP), a well-studied problem in a wide range of fields including computer science, operation research, and optimization theory, asks to find the shortest route for a salesman to visit a set of *cities* (or destinations) exactly once and return to the start 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 collaboratively visit a set of cities. Such a problem 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 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).

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) in mission planning tasks. For example, in the task of assigning drones or electric vehicles to deliver items from different warehouses, visiting charging stations must be taken into account due to the limited battery capacity. The charging stations also have limited capabilities in terms of the number of agents they can host and the total energy resource 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 a new set of nodes, called *stations*, that each salesman can visit to replenish their energy level. The CMDTSP asks for 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 trivially 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] and Simulated Annealing-based methods [7], [8], [9] or directly solve a Mixed-Integer Linear Programming (MILP) [4], [10], [11]. 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 paper, we propose a novel hierarchical framework for solving CMDTSP, providing a balance between 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 the visit order of assigned cities for each salesman. Each salesman then proposes multiple potential feasible routes by adding charging stations to their routes of cities. Finally, the proposed solutions are collected and an optimal solution for each salesman is computed using a MILP-based congestion control formulation. The number of both integer and real variables grows linearly to the number of cities in our MILP instead of quadratic and forth order in the exact method. In experiments, our framework outperforms selected baselines

in both solution quality and scalability on hybrid datasets built from Manhattan and Cambridge road maps as well as synthetic datasets. We observe a 5.22%∼14.84% tour length reduction and 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 main **contributions** of the paper 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 besides brute force enumeration includes Dynamic Programming [12] and Mixed-Integer Linear Programming (MILP) [13], [14]. Existing tools like Gurobi [15] and Concorde [16] optimize MILP through Branch and Bound (B&B) method and Cutting Plane Method (CMP) for fast computation. Exact methods are guaranteed to find the optimal solution but are computationally expensive which leads to severe scaling problems.

Approximation and heuristic algorithms are much more computationally efficient than exact methods but only provide sub-optimal solutions. Among algorithms with worst-case guarantee, the Christofides Algorithm [17] was the state-of-the-art algorithm with approximation ratio $\frac{3}{2}$ (defined as the ratio of the algorithms' optimal cost and *theoretical* optimal cost), which is recently improved to $\frac{3}{2}-10^{-36}$ [18]. The best heuristic algorithm for TSP is the Lin-Kernighan heuristic (LKH) algorithm [19]. Starting from a TSP tour, it iteratively removes several edges (2 or 3 are favored in practice) from the tour and reconnects the remaining sub-tours to find a tour with a lower cost. Recently, the neural version of LKH called NeuralLKH is developed and shows better performance [20]. Metaheuristic algorithms like Simulated Annealing (SA) are also applicable to solve TSP and are more flexible to be adapted to its variants.

End-to-End Learning-based methods have recently attracted attention from researchers due to their good performance. The first neural-based approach to solve TSP applies Hopfield network [21], and is recently improved on TSP [22]. Another variant of RNN used for TSP is Pointer Network [23], [24]. Recently, Graph Neural Network (GNN) has provided an efficient method for TSP since it learns the combinatorial structure of the graph problem better by capturing the node properties against its graph neighbors [25], [26], [27].

None of the existing approaches can be directly applied to give guaranteed 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. Exact algorithms models the problem into MILP [28], [29] or constraint programming [30], and metaheuristic algorithms includes Genetic Algorithm (GA) [31], Ant Colony Optimization (ACO) [32], [33], and Artificial Bee Colony algorithm (ABC) [34], [35].

Electric TSP (ETSP) introduces energy constraints and charging stations into standard TSP. The problem is first formally stated by [36], which also proposed the exact MILP formulation. Previous research on energy constraints came together with a time window, known as Electric Traveling Salesman Problem with Time Windows (ETSPTW) [37] which is claimed to be easier by [36]. Our work is a multiagent variant of the ETSP problem, where we provide an exact MILP formulation and a scalable hierarchical solution.

III. Problem Formulation

The CMDTSP is a variant of the TSP problem 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 raise 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 expression in the TSP, which can refer to arbitrary targets or destinations in robotic tasks.

Formally, the problem is defined on a complete undirected graph $\mathscr{G}=(V,E):=\mathscr{G}(V)$, where V is the set of vertices. The vertex set V is partitioned into the union of three sets D, C, and S, where $D=\{d_1,d_2,\cdots,d_m\}$ is the set of m depots (i.e., starting and ending locations of the salesmen's tours), $C=\{c_1,c_2,\cdots,c_n\}$ is the set of n cities, and $S=\{s_1,s_2,\cdots,s_l\}$ is 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, and a resource upper bound r_s for each station $s\in S$. 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 can represent distance, time, or any other measure that is relevant to the problem. We also assume that each salesman's energy level is fully replenished upon visiting any station.

Problem 1: (**CMDTSP**) Given a complete undirected graph $\mathscr{G}=(V,E)$ 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 for each salesman, such that: (1) each tour starts from and ends to the same depot; (2) each city in C is visited exactly once; (3) every salesman has a non-negative energy

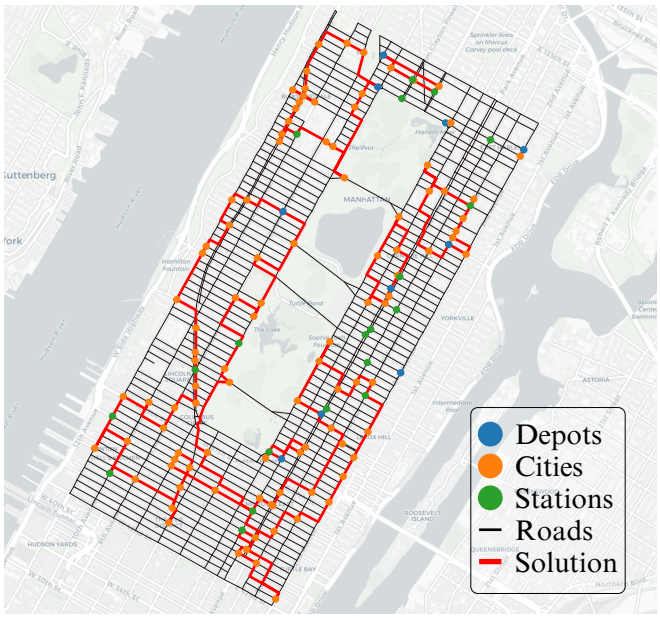

Fig. 1. A solution of CMDTSP in Manhattan: salesmen start from depots to collaboratively visit all cities and always keep their energy above zero.

Algorithm 1 Framework for solving CMDTSP

T, P, Solution $= \emptyset$
$l = 0$
$\{C_i\}_{i=1}^{|D|} =$ Partition(C; D)
▷ Assign the cities to each salesman
for $i \in \{1, 2, \cdots, m\}$ **do**
  $\mathscr{G}_i = \mathscr{G}(C_i \cup \{d_i\})$
  ▷ Form a complete graph of $C_i \cup \{d_i\}$
  $t_i =$ TSP($\mathscr{G}_i$) ▷ Find the TSP solution of graph $\mathscr{G}_i$
  for $(u, v) \in t_i$ **do**
    $\mathscr{G}_i' = \mathscr{G}(\{u, v\} \cup S)$
    ▷ Form a complete graph of $\{u, v\} \cup S$
    $P_i(u, v) =$ k-shortest-path(u, v; $\mathscr{G}_i'$; k)
    ▷ Find the top-k shortest paths from u to v
  end for
end for
$P = \cup_{i=1}^{|D|} P_i$
Solution $=$ CongestionControl(P)
▷ Form a solution satisfying all constraints from P
return Solution

level during the tour; (4) 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.

It is also a general version of a more commonly studied sub-problem, Multiple Depots TSP (MDTSP) [2], [38].

IV. Methodology

Similar to classical TSP problems, CMDTSP can be formulated as a mixed-integer linear program (MILP) (see website). However, the complexity of such MILP scales exponentially to the fourth order of cities, third order of depots, and second order of stations and therefore, does not scale well for the problems with a large number of cities.

Algorithm 2 Assign Cities to Salesmen

Input: City set C, depot set D

$\mathscr{G} = \mathscr{G}(C \cup D)$ ▷ Form a complete graph of $C \cup D$
$T =$ MST($\mathscr{G}$) ▷ Compute a minimum spanning tree
Compute minimum weight functions f, g
▷ Using Eq. (1), (2)
$\cup_{i=1}^{|D|} T_i =$ Part(T, f, g) ▷ Partition tree T based on f, g
return $\{C \cap T_i\}_{i=1}^{|D|}$

The intuition of our framework is straightforward: break down a CMDTSP into smaller subproblems to reduce the size of MILP, which is the bottleneck for scaling up. We propose a novel hierarchical framework that utilizes a heuristic and smaller MILP. The cities are first assigned to each salesman to form standard TSPs. Then, each TSP is solved for each salesman i without the energy constraints to return a potential tour t_i. For each pair of consecutive cities (u, v) in t_i, we suggest the top-k shortest paths that get to v from u by traveling through a sequence of stations to maintain a positive energy level. Finally, we collect all paths proposed by all salesmen to find a feasible tour for each salesman so that all the constraints including positive energy level and limited station resources are satisfied. The framework is shown in Algorithm 1 and an illustrative example of route planning in Manhattan is shown in Figure 1.

A. City assignment

We first introduce Algorithm 2 to assign the cities to every salesman. First, 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 minimize the total weights of remaining edges using dynamic programming as explained next. Note that each resulting partition T_i contains exactly one depot d_i. Given a rooted tree $T(u)$ rooted at u of any (sub)tree, we define two types of partitions $\tilde{T}(u)$ and $\hat{T}(u)$. The first type of partition $\tilde{T}(u) = \cup_{i=1}^{m} T_i(u)$ is a partition of $T(u)$ such that each connected component contains exactly one depot. If $T(u)$ contains no depot, then such a partition does not exist. The second type of partition $\hat{T}(u) = \cup_i T_i'(u)$ is a partition of $T(u)$ such that all depots and the root u are separated, i.e., each connected component either contains exactly one depot or the root. For a depot u as a root node, $\hat{T}(u)$ does not exist. By definition, $\tilde{T}(u) = \hat{T}(u)$ when $u \in D$.

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. Thus, functions f and g are computed by the following rules:

$f(u)$

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

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

We offer brief insights here and provide the details of the derivation on website. For a partition of type $\tilde{T}$ or $\hat{T}$, every partition on a subtree is also of these types. Computing weights for the partition of the tree $T(u)$ involves computing weights of partition for all u's children, plus the optimal way of connection to form a valid partition. Using this, we can obtain (1)-(2).

After computing $f(\texttt{rt})$, we can construct the partition $\tilde{T}(\texttt{rt})$ using the functions f and g by the following rules. If $u\in C$ with the partition of type $\tilde{T}$, then we can find a $v\in H(u)$ such that u choose to update $f(u)$ in Eq. (1), which means v is partitioned by $\tilde{T}$ and connects to u. Each of the rest of the children $v'\neq v$ is partitioned by $\tilde{T}$ and does not connect to u if $f(v')<g(v')+c(u,v')$. If $u\in C$ with partition type $\hat{T}$ or $u\in D$ with partition type $\tilde{T}$ ($u\in D$ only have partition type $\tilde{T}$ as $f(u)<g(u)=+\infty$), each $v\in H(u)$ is partitioned by $\tilde{T}$ and does not connect to u if $f(v)<g(v)+c(u,v)$. This construction is carried from root $\texttt{rt}$ of the tree $T(\texttt{rt})$ to the leaves, leading to the partition $\tilde{T}(\texttt{rt})$. Now we show the correctness and optimality of Alg. 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 connected components $\{T_i\}_{i=1}^m$ such that $d_i\in T_i$, and minimizes the total edge weights in the connected components, i.e., $\sum_{i=1}^{|D|}\sum_{(u,v)\in T_i} c(u,v)$.

Algorithm 2 runs in $O(|C\cup D|^2)$ and has an approximation ratio of 2 for MDTSP.

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

Proof: Let G be a graph, $M(G)$ be the corresponding MST, and $C(G)$ be the corresponding TSP tour. Let $CM(G)$ be the optimal TSP tour recover from $M(G)$, and $()^*$ denote the optimal solution. Suppose algorithm 2 gives partition $\mathscr{P}(G)=\cup G_i$ while the optimal partition is $\mathscr{P}^*(G)=\cup G_i^*$. Suppose E is the set of edges that are removed in algorithm 2 in $M(G)$. For simplicity, we also use the set to represent the total edge weights in the set. The algorithm cost is

$$\sum_i C^*(G_i) \le \sum_i CM(G_i) \le \sum_i 2M(G_i) \le 2(M(G)-E).$$

Since for the optimal partition $\mathscr{P}^*$, there exists a set of edge $E'\subseteq M(G)$ to reconnect the subgraphs. By algorithm 2, $E\ge E'$. Using this, we obtain

$$\sum_i C^*(G_i^*) \ge \sum_i M(G_i^*) \ge M(G)-E' \ge M(G)-E.$$

Hence, we can conclude $\sum_i C^*(G_i) \le 2\sum_i C^*(G_i^*)$. ■

After assigning cities to salesmen, we form graphs for each depot with corresponding cities and solve TSPs on the graph to determine the order of visits for each salesman. This is a standard TSP and 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 [39] 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 finding algorithm on the graph consisting of the edge and the stations. That is, each salesman i with a tour t_i proposes $k\times|t_i|$ paths in total, where $|t_i|$ is the length of the tour t_i. Hence, the total number of paths is $\sum_{i=1}^{|D|} k|t_i| = k(|C|+|D|)$ paths so that 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 that congestion happens at station s when more than r_s salesmen want to visit the station. The congestion control problem asks to find tours for salesman such that there is no congestion at any station. To this end, a salesman i chooses a path out of the k proposed paths for each edge $(u(v),v)$ in each tour t_i to meet the energy constraints for all the salesmen and the resource constraints for all the 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 $\gamma_{i,j}\in\mathbb{R}_+$ be the energy level of salesman i at the $j-$th node in tour t_i and $b_{i,j,h}$ be a binary vector indicating the stations included in the h-th path $p_{i,j,h}$ proposed by salesman i for its j-th edge along its tour, i.e. $b_{i,j,h}[s]=1$ if the path passes through station s. Let $\beta_{i,j,h}\in\{0,1\}$ be a binary variable indicating whether $p_{i,j,h}$ 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 $(j+1)$-the node) as $q_{2,i,j,h}$. Thus, for *path* to be feasible, the corresponding constraints $\gamma_{i,j}\ge q_{1,i,j,h}$ and $\gamma_{i,j+1}\le q_{2,i,j,h}$ should be satisfied. 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 $q_{3,i,j,h}$. Thus, for a direct path from j-th to $(j+1)$-th node to be feasible, it is required that $\gamma_{i,j}-\gamma_{i,j+1}\ge q_{3,i,j,h}$. In this case, 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 function (3a) represents the overall cost of all the tours that should be minimized. The constraints (3b) and (3c) encode choosing exactly one path out of k proposed paths for each edge. Constraint (3d) is the resource constraint for stations and constraints (3e)-(3h) are the energy constraints for the salesmen when traveling 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 $5m^2+5mn+l$ 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.

V. Experiments

In this section, we report experiments for validating our method. In the first part, we compare our method with three baselines, namely, Ant Colony Optimization (ACO) based algorithm [41], Hybrid Evolutionary Algorithm (HEA) [42], and Hybrid Variable Neighborhood Search (HVNS) [43], on both hybrid data and synthetic data. In the second part, we compare our method with a naïve MILP formulation for scalability and quality of the solution. In the third part, we study the partition component and TSP solver component in our framework to justify our choices.

Part one experiments were run on a server with one AMD Ryzen Threadripper 3990X 64-Core Processor and the rest experiments were run on a desktop with an Intel Core CPU I5-13600KF and an Nvidia RTX 3080 GPU with 12GB of RAM. Gurobi 10.0.0 [15] was used as the MILP solver. The neural-based solver was trained on TSP 50 provided by [26].

As to the hyperparameter setting, we set the iterations of LKH to 10 and the number of paths proposed per salesman $k = 5$. The selection of k is empirical as we notice that in the experiments, increasing k beyond 5 increases the running time but does not improve the performance. For baselines, we adapt the parameters from their papers [41], [42], [43].

A. Comparison with Baselines

Datasets: We present experiments on four datasets consisting of hybrid data and synthetic data. We use the real-world Manhattan road map from [44] and the Cambridge road map extracted from planet OSM [45]. For the Manhattan map, hybrid 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 [46] and depots and cities are uniformly sampled from the rest of the map. A synthetic data is adapted from existing MDVRP benchmarks [40] by randomly turning a fraction of cities into stations and uniformly sampling a fixed amount of depots, cities, and stations. The final synthetic dataset is generated by sampling depots, cities, and stations uniformly in a unit square. Salesmen are restricted to traveling along roads in hybrid instances, while can travel freely in the 2D space in synthetic instances. We generate datasets of two sizes, with 1000 instances of small size and 100 instances of large size. Small instances consist of depots $|D| = 5$, cities $|C| = 30$, and stations $|S| = 20$, while large instances consist of $|D| = 10$, $|C| = 100$, $|S| = 20$. In all cases, a station is allowed to be visited no more than $r = 2$ times. Due to the different map size, the energy capacity of salesman is set differently, i.e. 4 in Manhattan, 40 in Cambridge, 4 in synthetic data from benchmark, 1 in synthetic data.

Metrics: We use tour length, feasibility rates, and running time as metrics to evaluate a solver for CMDTSP. 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 found for the instance. We report the average tour length and average running time.

Baselines: We use ACO, HEA, HVNS as baselines. ACO based algorithm consists of a nearest neighbor partition parts to divide the multi-depot to single-depot sub-problems and an ant-colony optimization to seek local optimum. HEA initializes population by nearest neighbor and generate the offspring by adding routes with minimum incremental density from parents iteratively. The solution is enhanced by applying variable neighborhood search. HEA is originally designed for MDVRP, so we add the station insertion procedure from ACO baseline to adapt it to CMDTSP. HVNS initialize the solution by variable neighborhood search in each iteration and search for the best solution by tabu search. The three baselines represent the main approaches for related problems and are state of the arts methods in their categories.

Result: The comparison results are shown in Table I and Table II. Our method is sufficiently effective to produce the best solution quality in all test cases. When the problem size is small, our method performs the second best feasibility rate

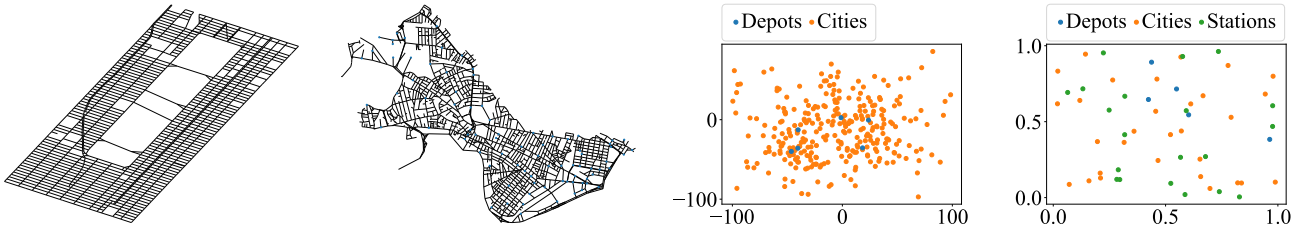

Fig. 2. Four Datasets for experiments. (a) The driving road map of Manhattan, New York. (b) The driving road map of Cambridge, Massachusetts (c) The existing benchmark for MDVRP [40] (d) An example of randomly generated data in the unit square.

TABLE I

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

	Manhattan			Cambridge			MDVRP Benchmark			Uniform		
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**	907.83	0.84	**1.26**	7.98	**1.00**	**1.13**
HEA	31.65	0.62	21.57	380.12	0.63	21.62	909.78	0.43	14.45	6.93	0.59	17.69
HVNS	28.26	**1.00**	12.95	338.86	**1.00**	14.80	843.66	**1.00**	15.25	6.67	**1.00**	16.80
Ours	**24.96**	**1.00**	1.90	**313.45**	**1.00**	1.91	**783.06**	0.91	2.04	**5.68**	**1.00**	1.74

TABLE II

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

	Manhattan			Cambridge			MDVRP Benchmark			Uniform		
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	2156.11	0.41	4.03	14.27	0.93	4.88
HEA	50.61	0.01	420.62	824.00	0.01	473.21	1931.04	0.09	329.39	13.73	0.17	393.03
HVNS	47.70	0.94	270.19	653.54	0.48	300.75	1642.98	0.67	287.45	9.95	0.99	239.44
Ours	**41.74**	**0.98**	**2.54**	**619.43**	**0.56**	**3.11**	**1511.19**	**0.70**	**2.32**	**8.90**	**1.00**	**3.00**

and running time, and is very close to the best baseline. On test cases with large problem size, our method out perform all other baselines in all evaluation metrics, which also shows the good scalability.

B. Compare with Exact Algorithm

We explore the trade-off made by our approximate framework by comparing with an exact algorithm in solution quality and running time.

Datasets: To measure the solution quality, we use the randomly generated 100 instances of $|C| = 15$, $|D| = 3$, $|S| = 20$, and $r = 2$. To measure the scalability of our framework, we vary the size of instances with 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 $|D| = 5$, 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 tour length. The

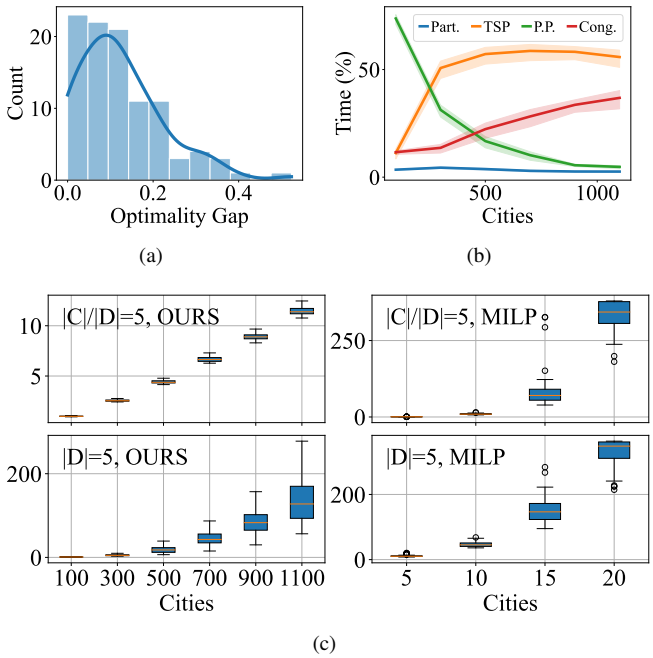

Fig. 3. Results on comparison experiments. (a) The distribution of optimality gaps with mean value 12.48%. (b) The change of percentage of running time for each part in the framework. (c) Comparison on Scalability. All Y-axes are time in seconds.

running time is the duration from formatted data being fed into the solver to the solver outputting the best solution found for the instance.

Results: Our framework achieves a mean optimal gap of 12.48% and worst gap of 52.34% on the dataset, where the distribution is shown in Figure 3(a). We believe that this is a good performance as the worst-case guarantee for standard TSP is around 50%. Figure 3(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 a good suboptimality and tremendously reduce the running time to be able to scale up to large problem size.

Comparing the first column in Figure 3(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. Figure 3(b) shows the changes of 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.

C. Studies of Components

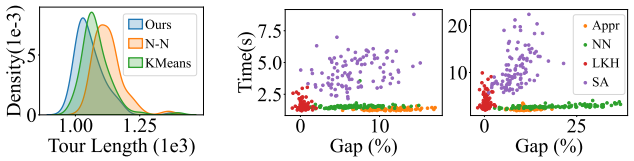

Fig. 4. Results of studies on components. (a) Keep TSP solver to be LKH, our partition algorithm outperforms all baselines; (b) keep Partition algorithm to be MST based, LKH solver produce best solution quality and competitive computation time.

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

Datasets: We conduct experiments on 100 randomly generated instances of $|D| = 5$, $|C| = 150$, $|S| = 20$, and $r = 3$. To test the scalability of different TSP solvers, we further randomly generate 100 instances of $|D| = 5$, $|C| = 250$, $|S| = 20$, and $r = 3$.

Metrics: For partition algorithms, we evaluate the algorithms by tour length only since all of them takes only polynomial time. For TSP solvers, we evaluate them by tour length, which is measured by the gap between using a MILP TSP solver and themselves, as well as 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 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 [26], approximation solver (Christofides algorithm), heuristic solver (LKH), and metaheuristic solver (SA).

Results: Results in Figure 4(a) show that our partition method beats both NN and KMeans. The result in the left figure of Figure 4(b) shows that LKH gives much smaller gaps than others within acceptable running time. And the right one of Figure 4(b) shows that LKH solver still produces the best solution with a small running time on large cases while the neural-based solver has a enormous drop in its solution quality due to the out-of-distribution issue.

We want to emphasize here again that our framework is flexible with arbitrary partition algorithms and TSP solvers, which means that it can performs better as the development of both components.

D. Effectiveness of Resource Distribution

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

Datasets: We use the random dataset with 1200 instances of fixed $|D| = 30$, $|C| = 1200$. We randomly and equally divide the instances into 12 parts, with the number of stations $|S|$ in each part varies from $1 \sim 120$, where the total amount of resources $r \cdot |S| = 240$.

Metrics: We record the tour length and the running time for each part of the dataset.

Result: As shown in Figure 5, the increase in stations leads to the increase of feasibility rate and the decrease of the tour length, which means that a more dispersed distribution of resources facilitates easier utilization. It suggests that the resource should be distributed as dense as possible. In practice, there is a cost to set up a station so the overall optimal resource allocation needs to balance the gain and cost of setting up stations.

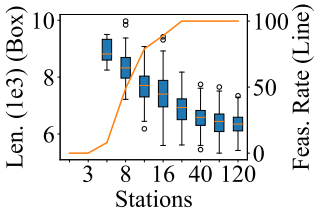

Fig. 5. Effect of the dispersion of stations. There are no feasible solutions for cases with 1 or 3 station(s).

VI. Conclusion

We define a new variant of MDTSP to capture the energy constraints in real-world applications and propose a novel framework to solve them. Our method is able to efficiently produce high-quality solutions compared against baselines and solve much larger problems than MILP formulation, which is verified by the experiments. For future work, we would like to improve the congestion control part with a neural network to get better scalability 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," *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] 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 *AIP Conference Proceedings*, vol. 1862, no. 1. AIP Publishing LLC, 2017, p. 030123.

[6] S. Ghafurian and N. Javadian, "An ant colony algorithm for solving fixed destination multi-depot multiple traveling salesmen problems," *Applied Soft Computing*, vol. 11, no. 1, pp. 1256–1262, 2011.

[7] T. S. Rao, "A simulated annealing approach to solve a multi traveling salesman problem in a fmcg company," *Materials Today: Proceedings*, vol. 46, pp. 4971–4974, 2021.

[8] 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," *The Journal of Supercomputing*, vol. 77, pp. 11 827–11 852, 2021.

[9] Y. Zhou, W. Xu, Z.-H. Fu, and M. Zhou, "Multi-neighborhood simulated annealing-based iterated local search for colored traveling salesman problems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 16 072–16 082, 2022.

[10] M. Diaby, "Linear programming formulation of the multi-depot multiple traveling salesman problem with differentiated travel costs," *Traveling Salesman Problem, Theory and Applications. InTech, New York, NY*, pp. 257–282, 2010.

[11] 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 *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 138–143.

[12] 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.

[13] 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.

[14] G. Dantzig, *Linear programming and extensions*. Princeton university press, 1963.

[15] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023. [Online]. Available: https://www.gurobi.com

[16] D. L. Applegate, R. E. Bixby, V. Chvátal, and W. J. Cook, "The traveling salesman problem," in *The Traveling Salesman Problem*. Princeton university press, 2011.

[17] N. Christofides, "Worst-case analysis of a new heuristic for the travelling salesman problem," Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, Tech. Rep., 1976.

[18] A. R. Karlin, N. Klein, and S. O. Gharan, "A (slightly) improved approximation algorithm for metric tsp," in *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, 2021, pp. 32–45.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," *Advances in neural information processing systems*, vol. 28, 2015.

[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, p. 437, 2023.

[25] E. Khalil, H. Dai, Y. Zhang, B. Dilkina, and L. Song, "Learning combinatorial optimization algorithms over graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[26] W. Kool, H. Van Hoof, and M. Welling, "Attention, learn to solve routing problems!" *arXiv preprint arXiv:1803.08475*, 2018.

[27] X. Bresson and T. Laurent, "The transformer network for the traveling salesman problem," *arXiv preprint arXiv:2103.03012*, 2021.

[28] K. Sundar and S. Rathinam, "Algorithms for heterogeneous, multiple depot, multiple unmanned vehicle path planning problems," *Journal of Intelligent & Robotic Systems*, vol. 88, pp. 513–526, 2017.

[29] 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.

[30] 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.

[31] 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.

[32] W. Liu, S. Li, F. Zhao, and A. Zheng, "An ant colony optimization algorithm for the multiple traveling salesmen problem," in *2009 4th IEEE conference on industrial electronics and applications*. IEEE, 2009, pp. 1533–1537.

[33] 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.

[34] P. Venkatesh and A. Singh, "Two metaheuristic approaches for the multiple traveling salesperson problem," *Applied Soft Computing*, vol. 26, pp. 74–89, 2015.

[35] 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.

[36] A. Ceselli and G. Righini, "The electric traveling salesman problem: properties and models," Technical Report 2434/789142-University of Milan https://doi. org/10.13140 ..., Tech. Rep., 2020.

[37] 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.

[38] K. Sundar and S. Rathinam, "Generalized multiple depot traveling salesmen problem—polyhedral study and exact algorithm," *Computers & Operations Research*, vol. 70, pp. 39–55, 2016.

[39] K. Helsgaun, "An extension of the lin-kernighan-helsgaun tsp solver for constrained traveling salesman and vehicle routing problems," *Roskilde: Roskilde University*, vol. 12, 2017.

[40] C. R. C. in Distribution Management, "mdvrp," http://neumann.hec.ca/chairedistributique/data/mdvrp/, 2023.

[41] 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.

[42] 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.

[43] 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. 102293, 2021.

[44] 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.

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

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