Two-stage route planning algorithm for last mile delivery

Dang Bao Le

A Thesis

in

The Department

of

Mechanical, Industrial and Aerospace Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Applied Science (Industrial Engineering) at

Concordia University

Montréal, Québec, Canada

August 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Dang Bao Le**

Entitled: **Two-stage route planning algorithm for last mile delivery**

and submitted in partial fulfillment of the requirements for the degree of

Master of Applied Science (Industrial Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Claudio Contardo

_____ Examiner
Dr. Brigitte Jaumard

_____ Supervisor
Dr. Ivan Contreras

_____ Co-supervisor
Dr. Andrew Delong

Approved by _____
Dr. Martin D. Pugh, Chair
Department of Mechanical, Industrial and Aerospace Engineering

_____ 2022

_____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Two-stage route planning algorithm for last mile delivery

Dang Bao Le

This thesis reports an application of using two stage optimization framework to solve the clustered asymmetric traveling salesman problem to compete in the 2021 Amazon Last Mile Routing Research Challenge. This approach implicitly leverages tacit knowledge encoded in delivery data to learn a data-driven routing algorithm capable of predicting high quality routes. Given a set of features and a set of decision targets corresponding to routes taken by experienced drivers, the model seeks to learn routing models that provide near-optimal decisions for a set of high quality observed routes. The thesis presents and computationally compares the algorithmic approaches capable of learning feature-dependent cost functions for the base routing model. The results of extensive computational experiments based on real data provided by Amazon involving 6,112 historical routes show that this approach is comparable in score with the prize winners methods.

Keywords: *Last mile delivery; Traveling salesman; Inverse optimization; Structured prediction*

Acknowledgments

I would like to give a special thank to my supervisors, Dr. Ivan Contreras and Dr. Andrew Delong, as well as Dr. Daria Terekhov for their guidance and support throughout my program. It had been an amazing journey with many ups and downs, but at the end of the day I have gained so much more than I ever wished for before entering the program.

Thank you Dr. Contreras for your support and advice throughout my undergraduate and graduate studying career. It has been 7 years since I know you and I truly admire your knowledge and passion in your field of research. You are truly my role model for the past many years and the reason why I always try to better myself and become passionate about operation research, programming and academic topics in general.

I also want to thank Dr. Delong for giving me valuable advice for as long as I know you. You have been an amazing mentor and I truly appreciate you checking up on my mental health throughout the program. I will always be grateful for that.

I wish you all the best in your next adventure and it would be amazing if our paths cross again.

I would also like to thank my parents and my sister for always be by my side even though I am not near you. I love you guys always. Without you I would not have been the person I am today.

To my amazing girlfriend Thu, you have always been my cheerleader and best companion. I look forward to have a family with you.

would also like to thank Concordia University and my supervisors for their financial support. Finally, to everyone whom I did not mention by name but that contributed toward this work, thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Global retail sales growth has been growing and taken up more retail market share. According to eMarketer, online retail sales will reach $6.17 trillion by 2023, with ecommerce websites taking up to 22.3% of total retail sales. Although retail had a tough year in 2020, ecommerce sales still increased growth globally, with Latin America saw 25% growth to $85 billion dollar, Russia, the UK and Philippines saw more than 20% ecommerce sales growth in 2021, according to Keenan (2022). One of the most crucial parts in the ecommerce supply chain is last mile delivery, where products get delivered from the last distribution center to final customer. Precedence (2022) highlighted in their report that last mile logistical efficacy represents one of the main challenges for key players engaging in freight delivery and distribution, where last mile can represent up to 40% of total parcel delivery cost. The global last mile delivery transportation market size is expected to increase 3 fold by 2030 to a value of $424 billion industry, as shown in Figure 1.1.

In the quest for achieving technology advantage over competitors, Amazon has invested in alternative technologies for last mile delivery, e.g., drone delivery. In a December 1, 2013 interview on American television program 60 Minutes, Amazon CEO Jeff Bezos announced that Amazon would change ecommerce shopping by allowing users to receive items within 30 minutes of ordering. According to Mr. Bezos, this service would be powered by unmanned autonomous drones and could be offered as soon as 2015. Unfortunately, there has been minimal progress with Amazon struggling to make drone delivery a reality. According to Bloomberg in 2022, there were five crashes over the course of a four-month period at the company testing site in Oregon.

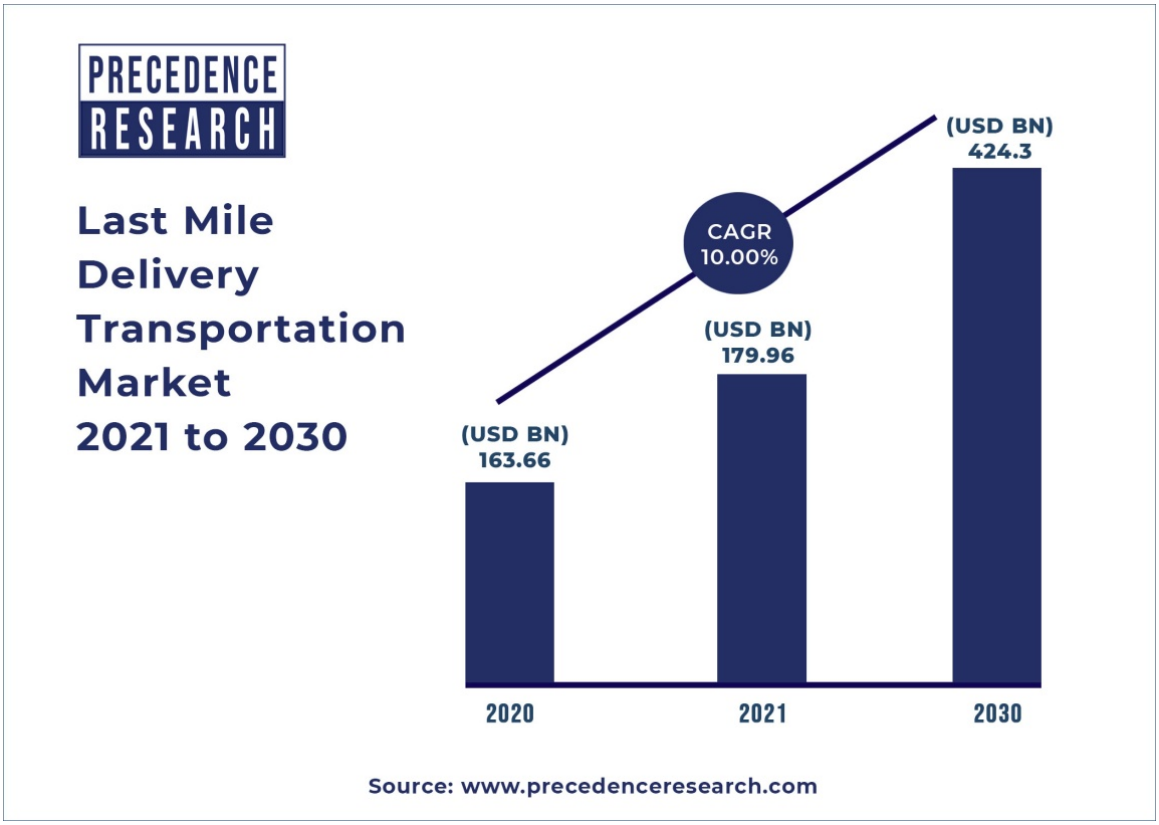


Figure 1.1: Last mile delivery transportation market 2021 to 2030

Due to the technical challenges of emerging technologies, Amazon has returned its focus to the use of automotive vehicles and existing road infrastructure with its introduction of Amazon Delivery Service Partner Program in 2018. According to Soper (2020), Amazon delivered more than 1.8 billion packages via more than 40,000 Prime-branded vehicles from 2018 to 2020. According to Amazon official website, Amazon (2022), it costs as little as CDN$ 15,000 to start the delivery partnership, without requiring any experience, and a potential annual revenue of 1 to 4 million Canadian Dollars.

With the every growing demand of quick and efficient services, Amazon has focused greatly on expanding their supply chain network, however there are still challenges in their last mile delivery system that differ from the traditional logistical network. Last mile delivery rely on many nuance factors that can change quickly, such as unpredictability in transit, availability of customer, route fuel efficiency, and environmental regulations.

1.1 Motivation

The operations research literature has been covering the Traveling Salesman Problem (TSP) and its many variants extensively for decades, which has led to important advancement in both solution quality and computational cost. However, there remains an important gap between theoretical route planning and real-life route execution that most optimization-based approaches are unable to bridge. This gap relates to the fact that in real-life operations, the quality of a route is not exclusively defined by its theoretical length, duration or cost, but also the multitudes of factors that affect the driver ability to carry out the delivery route safely, securely and conveniently under real-life conditions.

Furthermore, experienced drivers usually have tacit knowledge about the geography, infrastructure and consumer they deliver to. They know which roads are hard to navigate, when traffic is bad, when and there they can easily park, which stops can be conveniently served together, and many other things that are hard to formalize in an optimization model. Most traditional route planning tools used in the industry therefore do not utilize this knowledge, which cause drivers to frequently deviate from planned route sequences for their own convenience.

1.2 The Last Mile Delivery Challenge

To address this mismatch between theory and practice, in March 2021 Amazon announced a competition called the Last Mile Routing Research Challenge (further referred to as the Amazon Challenge) with the goal of developing innovative approaches to derive solutions to the route sequencing problem faced when using their private fleet of vehicles. The key reason for the challenge was the gap between the theory and practice of route planning, particularly the observation that experienced drivers deviate from routes recommended to them by classical optimization models/algorithms.

Given a benchmark set of instances with real data of delivery requests and their characteristics (e.g., service time, package dimensions, and time windows), the goal was to generate new high quality route sequences by learning from historical route sequences that are operated and ranked by actual experienced drivers. Competition participants were encouraged to “develop innovative approaches leveraging artificial intelligence, machine learning (ML), deep learning, and

other non-conventional methods to produce solutions to the route sequencing problem which outperform traditional, optimization-driven operations research methods in terms of solution quality and computational cost" (Amazon, 2021).

1.3 Contribution of the Research

In this thesis, we study the TSP in last mile delivery context. We aim to find optimal route planning decisions using traditional combinatory optimization methods as well as machine learning topics to leverage historical routing data and driver tacit knowledge. In this problem, we assume a deterministic network with known traveling time and number of nodes. The route needs to start at a common depot and then visit all planned stops before returning to the depot to be considered a valid route.

Furthermore, we develop a new two level heuristic that improves the solver efficiency while maintaining the same solution quality as its one level solver counterpart. The motivation behind this strategy is to encode the zone data structure provided by Amazon within a decision tree model and learn historical driver patterns.

We believed that our heuristic would have won the second place in the Amazon Challenged that we participated in, however there was a bug that result in a crash and therefore we never received the final ranking. The computational results reported in later section were done after the competition had completed.

1.4 Thesis Structure

This dissertation studies a new methodology to learn characteristic of historical good quality route data and extract hidden feature to recommend new routes that retain equivalent attributes.

The remainder of the thesis is structured as follows. Chapter 2 provides the background and literature review for TSPs and extensions with an analysis of the Technical Proceedings of the 2021 Last Mile Routing Research Challenge. Chapter 3 describes the proposed routing model and solution framework used during the competition. Chapter 4 presents the results of computational

experiments obtained with a 5-fold cross validation and with the Amazon competition official evaluation dataset. We also analyze the model's performance when compared to with the prize winners and the proposed model strength and weakness. Finally, we summarize our research and directions for future research in Chapter 5.

Chapter 2

Literature Review

In this chapter, we provide a preliminary description of the Amazon Routing Challenge, following with an introduction to the TSP and its variations and finally the methods used by prize winners of the Amazon Challenge.

2.1 The Amazon Challenge

The Amazon Challenge (AC) challenges participants to build a model that uses known delivery data and historically observed high, medium and low-quality route sequences to propose high-quality route sequence for new deliveries for which no observed route sequence is known yet. Figure 2.1 depicts the competition objective provided by its organizer.

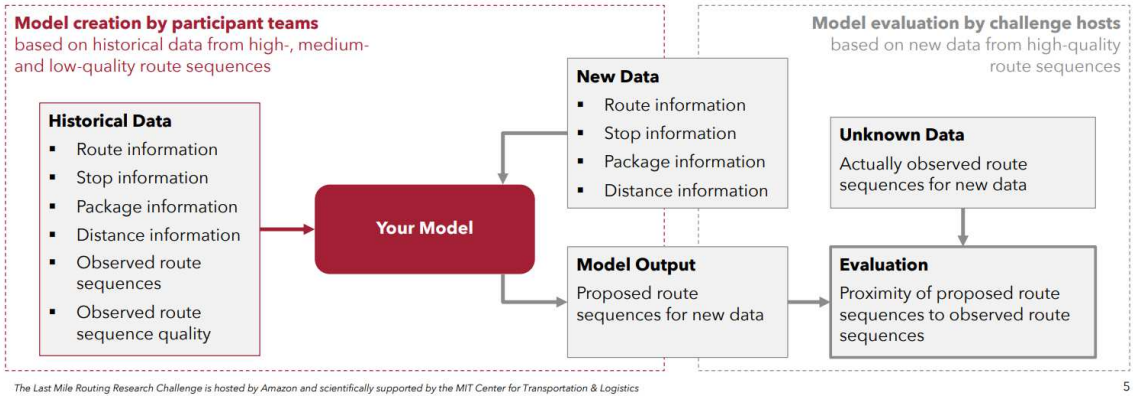


Figure 2.1: The Amazon Routing Challenge Model flow

The historical data given to participants comprises of approximately 6,100 routes. Each route comprises of multiple stops and each stops comprises of multiple packages. For each provided route, participants are given route information, stop information and package information. The route information has the following properties:

- **RouteID**: Unique identifier for group of stops that needs to be served on a joint route
- **StationCode**: Unique identifier of the delivery station that a route starts from
- **Date**: Date of route performance
- **Departure Time**: Time of the day at which route departs
- **Executor Capacity**: Volumetric capacity of delivery vehicle
- **Stops**: List of stops to be served on the route
- **Observed sequence**: Historical route that the stops were served
- **Route Type**: Categorical value denoting quality of the route sequence (High, Medium, Low).

Similarly, the organizer provide each stop with the following information:

- **StopID**: Unique identifier of each stop on a route (unique within each route).
- **Latiture/Longtitude**: Geo-coordinate of the stop.
- **Type**: Categorical variable denoting type of stop, Either station or delivery.
- **ZoneId**: Unique identifier of the geographical planning area that the stop falls into.
- **Packages**: Served at each stop.
- **Distance**: To any other stop on the same route.

Finally, for each package the following information are provided:

- **PackageID**: Unique identifier for each package. Unique over entire data.
- **Status**: Categorical variable depending on the delivery status of the package.

- **Time window**: Start and end time of the delivery time window constraint on some package.
- **Planned service time**: Time that serving each package is expected to take.
- **Dimensions**: Maximum length, width and height of the package.

2.1.1 Route characteristic

The dataset contains 6,112 routes from five major cities and its surrounding geographical areas: Los Angeles, Seattle, Boston, Chicago and Austin. Approximately half of the routes are from Los Angeles. The routes are executed from 17 unique depots, with some cities having multiple depots. The historical routes spanned from July 19, 2018 to August 26, 2018 with start time range from 10:00 to 18:00. There are three types of vehicles used, with capacity of 4.2 m^3, 3.3 m^3 and 3.1 m^3. The vehicle average load is 74.2% of their capacity. Each route contains a quality score of either High, Medium or Low with unknown scoring criteria. Table 2.1 remarks the route quality distribution. The route duration ranges from 6 to 10 hours and includes between 33 to 238 stops in the route, with an average of 145 stops per routes. From plotting the given GPS location of each stop, we can approximate that the stops included both residential, such as houses and apartment buildings, to commercial locations, such as mall and office complexes. Each stop can contains one to multiple packages, with the largest stop contains 25 packages.

Route Quality	Number of routes
High	2718
Medium	3292
Low	102

Table 2.1: Route quality distribution

2.1.2 Evaluation of submission

All submissions are evaluated based on a combination of solution accuracy (i.e., the proximity of predicted route sequence to actual high-quality route sequence), total route time (as a measure of route efficiency), and total computational time. Furthermore, all participants have to make their code available publicly under Apache 2.0 or MIT open-source license, thus eliminating the possible

usage of commercial software such as CPLEX or Gurobi.

During the evaluation phase, the organizer compares the proposed sequences generated by the participants' model for a set of previously unknown evaluation routes with known high-quality route sequences for the same set of evaluation routes. In order to fairly judge each team submission and compare how close do the participant model-proposed sequences match with known high quality routes , the competition implemented a combined scoring metric that use *Sequence Deviation (SD)* and *Edit Distant with Real Penalty (ERP)*. The SD score measure how different the proposed sequence from benchmark sequence, ranging from 0 to 1, with the score of 0 indicate that proposed and historical route are identical. This score only captures the differences in ordering of stops, regardless of the physical distance between stops. The (ERP) measures the variable of Levenshetin distance, the number of single element operations (insertion, deletions and substitution) required to transform the proposed sequence into benchmark sequence. The operations are weighted by the physical distance between stops involved, thus the further the stop from its optimal location, the worse ERP score it gets. Figure 2.2 shows the combination of the two metrics

$$score = \frac{SD(A,B) \cdot ERP_{norm}(A,B)}{ERP_e(A,B)}$$

Figure 2.2: The Amazon Routing Challenge score function

2.2 Traveling salesman problem

Generally speaking, given a delivery vehicle and a set of stops that need to be served, the problem consists of finding a Hamiltonian circuit visiting each stop exactly once. When the quality of the route is defined only by its cost (or length) and such cost is assumed to be symmetric, the problem corresponds to the well-known *symmetric traveling salesman problem* (TSP).

Biggs et al. (1986) noted in their book that the TSP was mathematically formulated in the 19th century by the Irish Mathematician William Rowan Hamilton and by the British mathematician Thomas Kirkman . Flood (1956) introduced the traveling salesman problem, or TSP, as finding a permutation P of integers from 1 to n that minimize the sum of interaction costs between every consecutive pair . Since there are $(n-1)!$ possibilities to consider, the problem is to find an efficient

method to minimize permutation cost.

Applegate et al. (1998) stated the TSP as: given a finite number of "cities" along with the cost of traveling between each pair of them, find the cheapest way of visiting all of the cities and returning to your starting point. This definition of TSP gives a great connecting example to current application for TSP nowadays. Though we are not all traveling salesman, this problem interest those who want to optimize their route, either by considering cost, distance or time. If one has multiple tourist attraction that they wish to visit in a day, then one automatically tries to think about the shortest distance possible to save travel time.

2.2.1 Variations of the Traveling Salesman Problem

When the cost of travel between stops depends on the time at which vehicles arrives at each stop (or their position in the sequence), the problem generalizes to the *time-dependent TSP* (Vander Wiel and Sahinidis, 1995). If stops have to be visited within a specified time window, the problem correspond to the *TSP with time windows* (Baldacci et al., 2012) and to the even more general *time-dependent TSP with time windows* (Vu et al., 2020). However, none of these problems are capable of capturing the underlying structure of delivery routes given in the historical data. Moreover, taking into account Amazon's real-life operations described in this competition, the quality of a route should not only be defined by its cost but also consider experience-based expectations of dynamic traffic conditions, availability of safe and convenient parking, and customer availability. Therefore, more complex functions are needed to better approximate observed routes.

A close look at the sequencing structure of historical routes revealed that the vast majority follow the following pattern. If a vehicle enters to a given zone, it tends to visit all stops of that zone consecutively before moving to the next zone. This sequencing structure is captured by a TSP variant known as the *clustered TSP* (CTSP) (Chisman, 1975; Laporte and Palekar, 2002), in which delivery stops are assumed to be partitioned into a set of mutually exclusive clusters. The CTSP consists of determining a minimum cost Hamiltonian cycle (or circuit) in which the stops of any cluster are contiguous. The CTSP can be transformed into a TSP by adding an arbitrarily large constant M to inter-cluster costs, or by subtracting M from intra-cluster costs. Lu et al. (2020) provide a computational study to compare several state-of-the-art TSP solvers with adhoc heuristic

algorithms specifically designed for the CTSP. The results on a set of benchmark instances show that the genetic algorithm of Nagata and Kobayashi (2013) and the LK-based heuristic of Helsgaun (2000) are capable of producing better solutions with shorter computational times as compared to specialized CTSP heuristics. Our proposed solution algorithm does not use any of these algorithms given that they are not released under the licensing agreements considered in this competition. Instead, we describe in the next section our own constructive and local search heuristics for an extension of the CTSP.

In the CTSP, it is assumed that there are no predefined sequencing constraints among different clusters. However, a deep analysis of the zone sequencing structure in observed routes revealed that the majority of routes followed specific zone sequencing patterns that could not be modeled with the CTSP. As a consequence, in this paper we study an extension of the CTSP in which precedence constraints are incorporated into the CTSP. We refer to this problem as the *clustered ATSP with precedence constraints* (CTSPP). To the best of our knowledge, this specific TSP extension has not been studied in the literature.

2.2.2 Algorithm to solve large a scale TSP

Korte and Vygen (2012) study the cutting plane approach combined with a branch-and-bound scheme that can solve TSP instances with several thousand cities optimally. Branch-and-bound is a technique for simulation complete enumeration of all possible solutions without having to consider them one by one. However, branch-and-bound is no better than complete enumeration of all possible solution in worst case, thus it is often combined with a cutting plane method that results in a branch-and-cut method.The idea of the cutting plane algorithm is to add a set of constraints to the assignment problem that prevent the formation of a subtour. Dantzig et al. (1954) demonstrated the power of their cutting-plane method by solving a 49-city instance of the TSP, which was an impressive size in 1954.

Helsgaun (2000) and Taillard and Helsgaun (2019) describe some of the most sophisticated implementations and extensions of Lin-Kernighan (LKH) based heuristics for finding good-quality solutions for very large-scale instances, whereas Nagata and Kobayashi (2013) present a powerful genetic algorithm that does not rely on a LKH based local search. If the costs are asymmetric, as it is

the case for the Amazon instances, the problem correspond to the so-called *asymmetric TSP* (ATSP). Kanellakis and Papadimitriou (1980) present an effective adaptation of the LKH TSP heuristic for the ATSP. Roberti and Toth (2012) surveys the most effective models and exact algorithms for solving the ATSP.

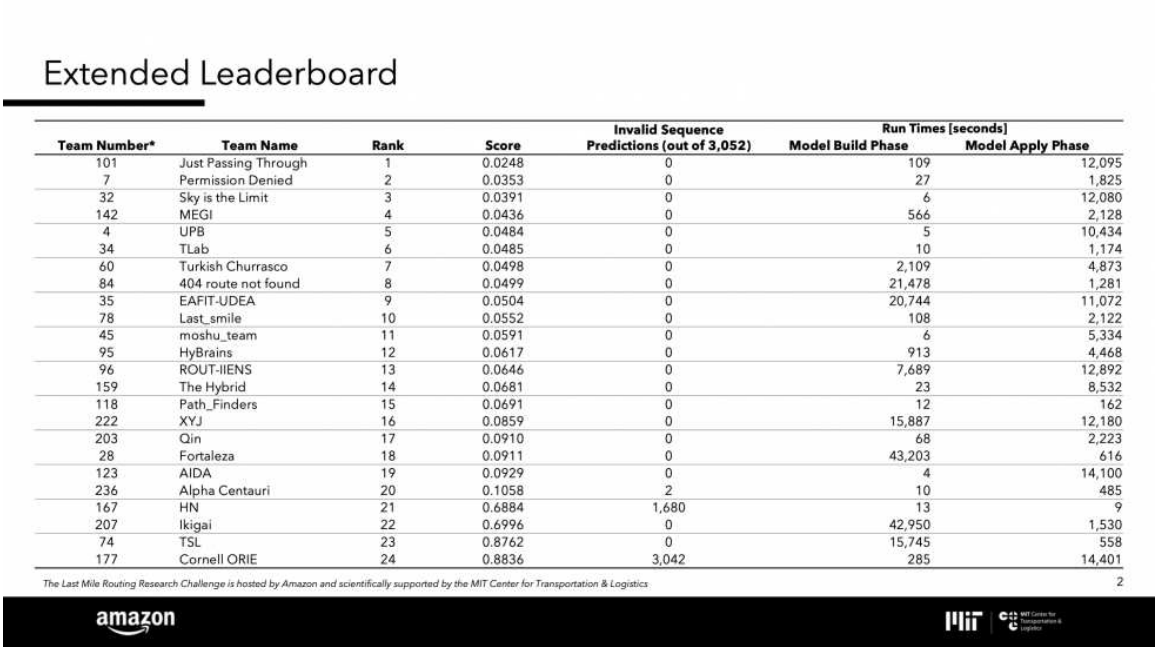
With the recent increase of interest and advancement in machine learning and deep learning architecture, novel methods that utilize Graph Neural Networks have had recent success in solving combinatorial optimization problems. Bengio et al. (2018) provide a through survey on the recent attempts, both from machine learning and operation research communities, at leveraging machine learning to solve combinatorial optimization problems. They argue that even on state of the art TSP algorithms, many decisions are made in a heuristic ways, such as cutting plane selection, thus leaving room for machine learning to assist in making these decisions. Gasse et al. (2019) propose a graph convolutional neural network model for learning branch-and-bound variable selection policies. They train the proposed model on a series of hard problems and claim that the new approach produces policies that improve over expert-design branching rules implemented in state-of-the-art solver (SCIP Bestuzheva et al. (2021)) on large problems. The proposed algorithm is called Learn2Branch.

2.3 Amazon Routing Challenge Technical Proceeding

In this section, we present the methodology and observations that prize winners of the Amazon Challenge noted in their proceedings paper. Figure 2.3 shows the competition final leader board. We believe that this section is important to the reader because it justifies our approach that will be presented in Chapter 3. We want to emphasize the fact that these observations presented below from winning teams and our observation in chapter 3 were made independently during the duration of the competition and were not shared among teams, even though they are similar in concept.

2.3.1 Third place winner

The third-place prize of $25,000 was awarded to Okan Arslan and Rasit Abay of team Sky is the limit with the article ”Data-Driven Vehicle Routing in Last Mile Delivery”. Their method involves



Extended Leaderboard

Team Number*	Team Name	Rank	Score	Invalid Sequence Predictions (out of 3,052)	Run Times [seconds] Model Build Phase	Model Apply Phase
101	Just Passing Through	1	0.0248	0	109	12,095
7	Permission Denied	2	0.0353	0	27	1,825
32	Sky is the Limit	3	0.0391	0	6	12,080
142	MEGI	4	0.0436	0	566	2,128
4	UPB	5	0.0484	0	5	10,434
34	TLab	6	0.0485	0	10	1,174
60	Turkish Churrasco	7	0.0498	0	2,109	4,873
84	404 route not found	8	0.0499	0	21,478	1,281
35	EAFIT-UDEA	9	0.0504	0	20,744	11,072
78	Last_smile	10	0.0552	0	108	2,122
45	moshu_team	11	0.0591	0	6	5,334
95	HyBrains	12	0.0617	0	913	4,468
96	ROUT-IIENS	13	0.0646	0	7,689	12,892
159	The Hybrid	14	0.0681	0	23	8,532
118	Path_Finders	15	0.0691	0	12	162
222	XYJ	16	0.0859	0	15,887	12,180
203	Qin	17	0.0910	0	68	2,223
28	Fortaleza	18	0.0911	0	43,203	616
123	AIDA	19	0.0929	0	4	14,100
236	Alpha Centauri	20	0.1058	2	10	485
167	HN	21	0.6884	1,680	13	9
207	Ikigai	22	0.6996	0	42,950	1,530
74	TSL	23	0.8762	0	15,745	558
177	Cornell ORIE	24	0.8836	3,042	285	14,401

Figure 2.3: Final leader board Amazon (2021)

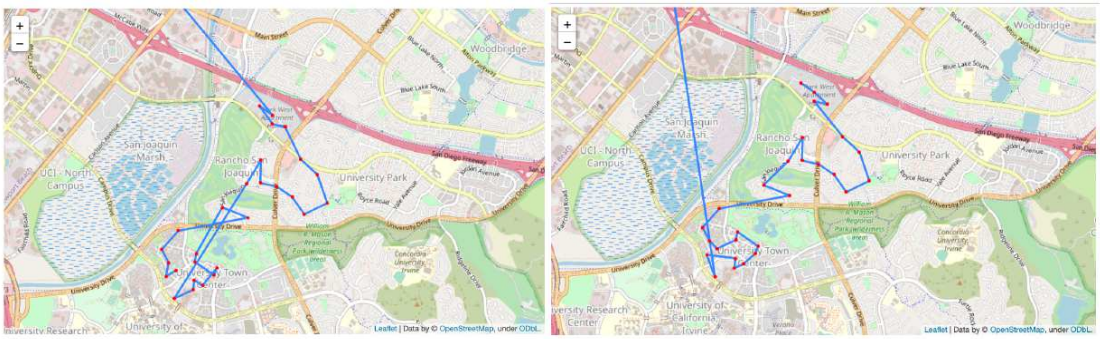
solving the traveling salesman problem on a transformed graph, which they use the `zone_ids` for modifying travel times between node pairs. With the observation that drivers generally visits the customers in the same `zone_ids` before going to another zone, they define η_{ij} to be the token of difference of nodes i and j, which measure the dissimilarity between the `zone_ids` of the two given nodes. Having $\eta_{ij} = 0$ implies that nodes i and j share the same `zone_ids`. Their network transformation is based on modifying travel times between stops to discourage an arc from appearing in the solution by multiplying constant values to arc cost based on their `zone_ids`. Furthermore, they introduces 3 'discouragement multipliers' a_1, a_2, a_3 in their model, with $a_1 < a_2 < a_3$. Parameter a_1 discourage for stops in the same zone, a_2 is the multipliers between two stops i and j whose `zone_ids` share three tokens and a_3 discourage between all other stop pairs they neither two stops is the depot. With this approach, Arslan and Abay decided to skip the model build phase and encoded all penalty terms as constants for final submission, which is shown in their GitHub Code https://github.com/rasitabay/lastmile_challenge.

2.3.2 Second place winner

The second prize team is from Massachusetts Institute of Technology, MIT, comprised of three PhD students, Xiaotong Guo, Baichuan Mo and Qingyi Wang of team Permission Denied with the article ”Last-Mile delivery trajectory prediction using hierarchical TSP with customized cost matrix”. Their team proposes a hierarchical Traveling Salesman Problem (TSP) optimization framework with customized cost matrix. The upper level TSP model solves for zone sequence, while the lower level TSP solves the intra-zonal stop sequence. Similar to team Sky is the limit, Guo et al observe that in most circumstances, the driver finished all deliveries in one zone before moving on the stops in another zone, which they further emphasizes that getting zone sequence correct is the most important factor. By introducing the hierarchical approach, they greatly reduces the scale of the problem since majority of routes have up to 250 stops, while number of zones is only between 6 and 47. Guo et al also realize the importance of predicting first zone correctly and construct a neural network to predict first zone based on travel time, distance, package count and size. However their attempt at using machine learning is not successful because the training set only contains 6,000 routes, not enough to train a meaningful machine learning model. Their method starts with a sequence of heuristic that modifies travel time matrix as the input for generating optimal zone sequence. The travel time between pair of zones is calculated by the average travel time between all possible pairs of stops between two zones. Then they solve the intra-zonal stop sequence using path-based TSP. Finally, in the conclusion section, Guo et al noted that because they are not able to increase the fist-zone prediction accuracy, they still faces with the reversal problem presented in Figure 2.4, which the actual route is the reversal of the predicted route.

2.3.3 First place winner

Finally, the top prize of USD 100,000 was awarded to Professor William Cook, Professor Stephan Helf and Professor Emeritus Keld Helsgaun for team Just Passing Through, with their approach published in the article ”Local Search with Learned Constraints for Last Mile Routing” Matthias et al. (2021). Their optimization method utilizes a simple penalty-based local-search algorithm in combination with routing constraints that are learned from historical data. Cooks et al



(a) Actual route (b) Predicted route by TSP

Figure 2.4: Example of the route with wrong first stop that leads to the reversal problem. Matthias et al. (2021)

started by obtaining the initial score of 0.07030, for 1,107 High and Delivered training instances using pure ATSP solver, with 0 being the lowest, and observed that driver tours are on average 12.9% longer than routes that minimize travel time. Similar to the teams presented above, Cooks et al also point out the possible clustering of route's stops in a tour using `zone_ids` and modify the code to force the stops in each zones to appear consecutively in any optimal tour. They again run the train instances with Concorde algorithm and obtain the mean score of 0.04866. Furthermore, their research achieve a better match by adding new restrictions at the inter-zone level. The first constraint is *precedence constraint* that require zone a to be visited before zone b. Then a *path constraint* is a stronger restriction that zone a immediately precedes zone b in a tour, such that $visit(a) = visit(b) - 1$. Finally, the *neighbor constraints* requires that zone a is either immediately before or immediately after zone b in a tour. Furthermore, they permit logical combinations of these constraints, such that for two specified constraints (A, B), a solution tour must satisfy either A or B. Together with these constraints, they also form super clusters, super-super clusters to further guiding the algorithm towards sequences followed by actual driver.

2.3.4 Machine learning perspective

We observed that there were many teams reported the use of machine learning and reinforcement learning in their technical proceeding paper and we want to report some of the findings here.

Anselmo R. Pitombeira-Neto Matthias et al. (2021) approached the problem from perspective

of inverse reinforcement learning where they tried to learning a parameterized cost function that drivers were trying to minimize. They reported the mean score of 0.0958 in a holdout sample with 815 routes in the conclusion section, with the main limitation was that the loss function is computationally expensive and non differentiable, so it was hard to use more powerful approximation technique.

Ashutosh Nayak and Ahmad Hemmati in their paper named ”Delivery Location Dependent Traveling Salesman Problem Using Multi-head Attention based Actor-Critic Method” attempted to use deep learning based reinforcement learning model to solve TSP with time window constraints by building an attention based Actor-Critic Model. Unfortunately they could not run the model and did not report any computational result in their proceeding.

Finally, Andres Regal tried using graph neural network (GNNs) in his article ”Using graph neural networks to solve route sequencing problems” Matthias et al. (2021) by preprocessing the original networks into a directed bipartite graph to represent the sequential nature of designing a route. They reported an average score of 0.88 when testing using high quality routes only.

Chapter 3

Methodology

In this section we present the methodology used in solving the cluster asymmetric traveling salesman problem (ATSP) during the Amazon Routing Challenge Competition. Similar to other ML frameworks, our approach requires two main phases: model build and model apply. During the model build phase, our algorithm aims to learn zone precedence constraints, some of them involving disjunctive conditions. During the model apply phase, we then incorporate the learned precedence constraints to the CTSPP to generate sequencing decisions that mimic observed patterns of historical routes. To approximately solve the CTSPP, we develop a fast two-stage heuristic algorithm.

This section is structured as follows. We first present a summary of our analysis of the zone sequencing structure of observed routes. We then provide the details of the model build phase, including a data cleaning step and the considered decisions tress for learning cluster precedence constraints. Finally, we provide the details of the model apply phase, including a formal definition and a mathematical programming formulation for the CTSPP and the proposed constructive and local search heuristics.

3.1 An Analysis of the Sequencing Structure of Observed Routes

There are a total of 6,112 historical routes given in the competition. These can be classified into high, medium and low quality routes. Taking into account their coordinates and zoneID's, we identified superclusters of stops associated with five different major urban agglomerations in

the contiguous United States. Table 3.1 provides descriptive information on the historical delivery routes per urban agglomeration, each of them identified with its main city from now on.

	No. of historical routes			No. stops	No. zones	No. depots
	High	Medium	Low			
Los Angeles	1,393	1,438	57	414,440	12,410	6
Chicago	381	609	12	162,410	6,939	4
Seattle	542	521	16	155,781	4,677	3
Boston	306	610	13	140,622	8,361	3
Austin	96	114	4	31,274	3,198	1
Total	2,718	3,292	102			17

Table 3.1: Descriptive information on historical delivery routes per city

After plotting several observed routes, we realized Zone IDs encode an underlying hierarchical structure of the geographical planning areas considered by Amazon in each city. We conjecture that cities are partitioned into several *Regions*, each of them further partitioned into several *Areas*. Each area within each region contains exactly three *Zones*. Figure 3.1 describes how regions, areas and zones unique identifiers are encoded in Zone IDs. For example, given zone ID H-18.1D, the first three alphanumerical characters (i.e., H-18) provide the region ID, the last alphabetical character (i.e., D) provides the area ID, and the second to last numerical character (i.e., 1) provides the actual zone ID. Moreover, we note that zones of the same area are contiguous as well as area pairs of the same region. Delivery stops always have a zone ID associated with them and each zone ID can be served by one or more depots from the same city.

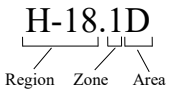


Figure 3.1: Identification of Regions, Areas, and Zones in Amazon zoning.

The following observations summarize our findings on sequencing patterns at different levels. They were obtained after performing an in-depth analysis of the sequencing structure of observed routes using data visualization tools available in open-source Python packages and Google maps.

(1) In the majority of routes, when a vehicle enters a zone it tends to visit all stops of that zone consecutively before serving stops of different zones. There are clearly cases in which a

vehicle enters and leaves multiple times the same zone. However, the proportion of zones which are served with almost perfect zone clusters is sufficiently large to justify the use of this hypothesis in our model.

(2) When a delivery vehicle enters a given area, it tends to visit all three zones of such area before moving to another area. Moreover, for a significant number of routes the zone sequencing decisions within an area tend to follow an increasing (or decreasing) numerical order. For example, if a vehicle enters area B, it will likely visit the zones in either the order 1B $\rightarrow$ 2B $\rightarrow$ 3B, or 3B $\rightarrow$ 2B $\rightarrow$ 1B.

(3) When a vehicle enters a given region, it tends to visit all areas of such region before moving to another region (if any). Moreover, for a significant number of routes the area sequencing decisions inside a region tend to follow a increasing (or decreasing) alphabetical order. For example, if a vehicle enters region A-4 and has to visit stops in areas: B, C, D, E, G, H, it will likely visit the areas in either the order B $\rightarrow$ C $\rightarrow$ D $\rightarrow$ E $\rightarrow$ G $\rightarrow$ H, or H $\rightarrow$ G $\rightarrow$ E $\rightarrow$ D $\rightarrow$ C $\rightarrow$ B.

(4) Moreover, when a vehicle transitions from one area to the next, it tends to reverse the numerical ordering when visiting the zones. For example, if a vehicle transitions from area B to area C, it will likely visit the zones of these two areas in either the order 1B $\rightarrow$ 2B $\rightarrow$ 3B $\rightarrow$ 3C $\rightarrow$ 2C $\rightarrow$ 1C, or 3B $\rightarrow$ 2B $\rightarrow$ 1B $\rightarrow$ 1C $\rightarrow$ 2C $\rightarrow$ 3C.

(5) If a vehicle has to visit two or more regions in the same route, it tends to visit all areas and zones of one region consecutively before serving stops of different regions. Moreover, for a significant number of routes, the region sequencing decisions tend to follow an increasing (or decreasing) alphanumerical order which depends on the associated depot. For example, if a vehicle has to visit A-4 and A-3 regions, it will either likely visit the regions in either the order A-4 $\rightarrow$ A-3, or A-3 $\rightarrow$ A-4.

(6) We did not observe any consistent time dependency features presented in the dataset, even with time window constraints and route start time information provided. The given routes were from 2018, which were 3 years prior to the challenge start time.

Figure 3.2 depicts an example of the observed zone sequencing decisions for route *RouteID_84f72f6e-3b04-4883-8e14-632fba58a7f0* associated with depot DLA5 in Los Angeles, CA. To simplify the illustration of the above sequencing patterns, the number of stops as well as the size, shape and spatial interaction of the regions, areas and zones depicted in this figure do not match the actual ones. There are more visited areas in the actual route, however, this figure only show a subset of areas and zone that the driver visited for this route. We also want to highlight that the driver most of the time only visit subset of areas and zone in any given region and thus this is the behavior we aim for.

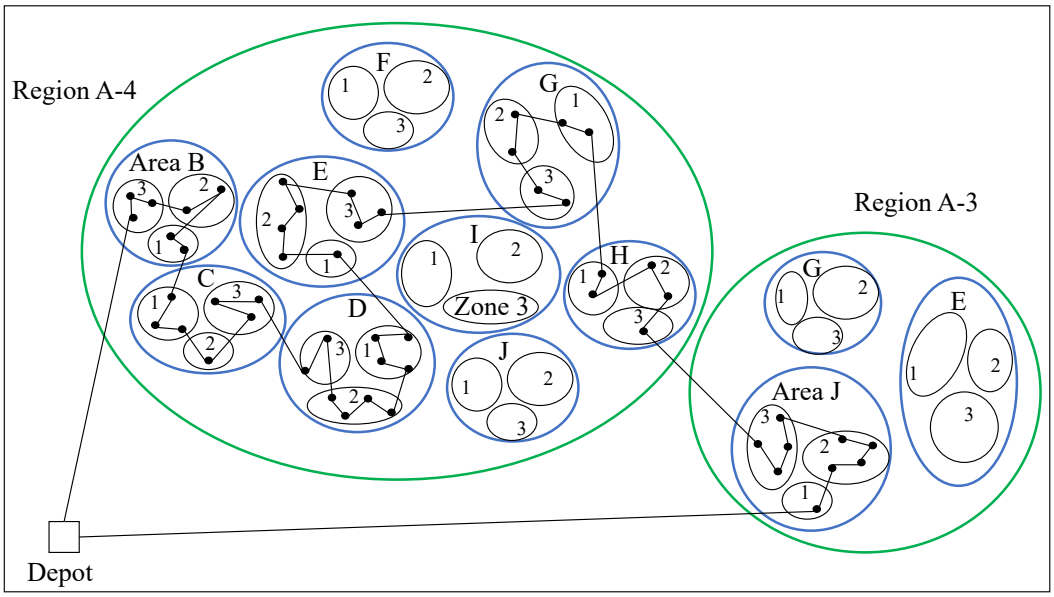


Figure 3.2: Example of zone sequencing structure of observed high quality route.

We stress that the main objective of the above identified patterns is to provide some base guidelines on how zone sequencing decision should be made. There are numerous observed routes in which a subset (or all) of these patterns do not apply. We noted that one common reason on why these patterns would not apply is when there exist a significant number of stops with time windows constraints. Another common reason is when serving densely populated and commercial areas. Finally, we conjecture that all above identified sequencing patterns are likely a direct consequence of the base algorithms used by Amazon to partition stops that need to be served each day to create a set of vehicle routes.

In the next section, we describe how ML techniques can be used during the model build phase to identify and learn these sequencing patterns from observed routes. These learned patterns are used to derive precedence constraints that are then used during the model apply phase.

3.2 Model Build Phase

The goal of our model build phase is to identify a set of region and area precedence constraints for each depot observed in the historical data. Given that our model build and apply phases heavily rely on the zone IDs given in the input files, we need to make sure all stops have zone IDs that conform to the template described in the previous section. We note that a very small portion of stops in the input file have either missing or non-conforming zone IDs. Therefore, we use a k-nearest neighbor model with the stop's longitude and latitude values as features. In particular, when there exist a stop s without a correct zone ID, we find the 5-nearest neighbors with respect to the Euclidean distance metric and assign s to the most dominant zone ID among these neighbors. This step employs the *KNeighborsClassifier* implemented in *scikit-learn* (Pedregosa et al., 2011).

During the model build phase, we build two direction classifiers for each observed station code (i.e., depot). The first classifier stores the region precedence constraints and the second one stores the area precedence constraints. From the observations noted in section 3.1, we want to learn the zone ids sequence direction at both region and area levels and for that we decide that a simple look up table would suffice this objective. We called these look up tables as region direction classifier (RCLF) and area direction classifier (ACLF). For the ease of simplicity during implementation phase, we decided to use scikit-learn implementation of decision tree model to encode our look-up table and hash the categorical variable, region and area ids in this case, using Python built-in hash function to match with the input requirement. Furthermore, all classifier have class weight parameter set to 'balanced' because we found it give the best precision score in preliminary experiment.

A detailed explanation for each classifier is provided below.

3.2.1 Building region direction classifier

Algorithm 1 provides a pseudo-code of the logic behind building the region classifier. For each input route, the model takes the first and last stop of the observed sequence denoted as src and dst, respectively. If src and dst belong to the same region code, denoted by $region(src)$ and $region(dst)$, then the model compares the two string values using a natural order and record the observed direction of either ascending or descending and the corresponding region code. Function *regionCode* return the first string of the zone id, for example *regionCode('R-1.3A')* = 'R'.

Algorithm 1: RCLF

Result: Region Direction Classifier for depot D
Input: D - The station code, L - List of routes starting at depot D;
$X \leftarrow$ empty list, $y \leftarrow$ empty list;
foreach *route r in L* **do**
    $Q^* \leftarrow$ zoneIDs of optimal sequence
    $src \leftarrow Q^*_1$, $dst \leftarrow Q^*_{-1}$
    $Ascending \leftarrow -1$
    $Descending \leftarrow 1$
    if *regionCode(src) == regionCode(dst)* **then**
        X += hash(regionCode(src))
        if *src* $\leq$ *dst* **then**
            y += $Ascending$
        else
            y += $Descending$
        end
    end
end
clf $\leftarrow$ DecisionTreeClassifier(classweight='balanced');
clf.fit(X,y);

For example, if a route starts at stop with zone ID R-5.1G and finishes at zone R-6.2C, then the region classifier would receive one observation of ascending order in region R. This observation can then be used during the model apply phase to predict interactions that we have not seen before during the model train phase. For instance, if a validation route serves regions R-7 and R-8, we can predict that the route should visit R-7 and then R-8.

3.2.2 Building area direction classifier

Algorithm 2 provides a pseudo-code for our area classifier model. Similar to Algorithm 1, this model also uses src as the zone ID of the first stop of the observed sequence. However, the dst variable is the last consecutive stop belong to the same region and area as src. For example, if src zone ID is H-3.3G, then dst zone ID is the last consecutive stop with a zone ID that matches the pattern H-3.*G. Then, the model records the zone ID direction and uses it to construct the classifier. We also utilize scikit-learn implementation of decision tree model classifier to encode the area direction look up table for simplicity. We use function areaCode(src) to get area code of a given zone id.

Algorithm 2: ACLF

Result: Area Direction Classifier for depot D
Input: D - The station code, L - List of routes starting at depot D;
$X \leftarrow$ empty list, $y \leftarrow$ empty list
$Ascending \leftarrow -1$
$Descending \leftarrow 1$
foreach *route r in L* **do**
  $Q^* \leftarrow$ zone IDs of optimal sequence
  $src \leftarrow Q^*_1$
  $idx \leftarrow$ last index of consecutive stops with same region and area
  $dst \leftarrow Q^*_{idx}$;
  if *src* $\leq$ *dst* **then**
    $direction \leftarrow Ascending$
  else
    $direction \leftarrow Descending$
  end
  X += hash(areaCode(src))
  y += direction
end
clf $\leftarrow$ DecisionTreeClassifier(classweight='balanced');
clf.fit(X,y) ;

3.3 Model Apply Phase

In this section, we first formally define the proposed routing model and provide a mathematical programming formulation. We then describe the two-stage heuristic algorithm used to generate feasible routes during model apply phase.

3.3.1 The Clustered Asymmetric Traveling Salesman Problem with Precedence Constraints (CTSPP)

The CTSPP is formally defined as follows. Let $D = (V, A)$ be a complete directed graph where $V = \{0, v_1, \cdots, v_n\}$ is a node set partitioned into m clusters $V_0, V_1, \cdots, V_m$, where V_0 is a singleton corresponding to the depot (node 0), and $A = \{(v_i, v_j) : v_i, v_j \in V, i \neq j\}$ is an arc set. We denote as $C = \{1, \cdots, m\}$ to the cluster set and $T = \{0, \cdots, m\}$ to the set of positions in the cluster sequence, where the 0 position is always reserved for the depot. For each $(i, j) \in A$, we define an (asymmetric) arc cost $c_{ij}(u, w)$, that depends on historical delivery data and on new delivery data encoded via a vector of features u and associated weights w. For each $k \in C$ and $t \in T$, we also define a sequence-dependent cluster cost $f_{kt}(u, w)$. The CTSPP seeks to determine a minimum cost Hamiltonian circuit on D in which the nodes of any cluster V_k are contiguous and cluster precedence constraints are satisfied. The main difficulty of the CTSPP stems from the inherent interrelation between two level of the decision process. The first level considers the cluster sequencing decisions, whereas the second level deals with the node sequencing decisions.

For each $(i, j) \in A$, we define the binary variable x_{ij} equal to one if and only if the vehicle goes directly from node i to node j. For each $k \in C$ and $t \in T$, we define the binary variable y_{kt} equal to one if and only if cluster k is assigned to position t at the cluster sequence level. Let $E(S) = \{(i, j) \in A : i, j \in S\}$ denote the set of arcs incident to node set S. The CTSPP can be stated with the following integer nonlinear program (INLP):

$$(INLP) \quad \text{minimize} \qquad \sum_{(i,j)\in A} c_{ij}(u, w)x_{ij} + \sum_{k\in C}\sum_{t\in T} f_{kt}(u, w)y_{kt}$$

$$\text{subject to} \quad \sum_{(i,j)\in\delta(i)^+} x_{ij} = 1 \qquad i \in V \tag{1}$$

$$\sum_{(i,j)\in\delta(j)^-} x_{ij} = 1 \qquad j \in V \tag{2}$$

$$\sum_{(i,j)\in E(S)} x_{ij} \leq |S| - 1 \qquad \forall S \subset V, 2 \leq |S| \leq n-1 \tag{3}$$

$$\sum_{(i,j)\in E(V_k)} x_{ij} = |V_k| - 1 \qquad \forall k \in C, |V_k| \geq 2 \tag{4}$$

$$\sum_{t\in T} y_{kt} = 1 \qquad k \in C \tag{5}$$

$$\sum_{k\in C} y_{kt} = 1 \qquad t \in T \tag{6}$$

$$\sum_{i\in V_k} \sum_{j\in V_\ell} x_{ij} = \sum_{t\in T} y_{kt-1} y_{\ell t} \qquad \forall k, \ell \in C, k \neq \ell \tag{7}$$

$$\text{Precedence constraints on } y_{kt} \text{ and } y_{k't} \text{ for all cluster pairs } (k, k') \tag{8}$$

$$x_{ij}, y_{kt} \in \{0, 1\} \qquad (i,j) \in A, k \in C, t \in T. \tag{9}$$

The objective minimizes the sum of the arc and cluster costs. Constraints (1)–(3) are usual constraints needed to model the ATSP, whereas constraints (4) ensure that nodes in the same cluster are visited contiguously. Constraints (5)–(6) are assignment constraints at the cluster-level sequence. They ensure each cluster is assigned to exactly one position and that each position contains exactly one cluster. Constraints (7) are (nonlinear) linking constraints to ensure the node sequencing decisions are aligned with the cluster sequencing decisions. Constraints (8) represent the precedence constraints for all cluster pairs obtained from the learned sequencing patterns during the model build phase. Finally, constraints (9) are the usual integrality conditions. During the first submission, we used a relaxation of the CTSPP in which constraints (5)–(8) and y_{kt} variables were not taken into account. Moreover, constraints (4) were indirectly considered with a intra-cluster penalization feature. Other features involving time windows constraints and other info given in historical data were used to define several linear families of arc-cost functions $c_{ij}(u, w)$. However, experiments performed after the first submission revealed that direct incorporation of constraints (4), in combination with cluster precedence constraints (8), significantly outperformed the model used during the first submission.

3.3.2 A Two-stage Heuristic for the CTSPP

We next describe our two-stage heuristic algorithm to solve the CTSPP. It uses a sequential approach to construct feasible delivery routes by decomposing the cluster and node sequencing decisions. The first stage uses the learned region and area precedence constraints from the model build phase to determine cluster sequencing decisions. Given a fixed cluster sequence, the second stage uses a greedy deterministic constructive heuristic and a local search to optimize node sequencing decisions.

Algorithm 3 provides a pseudo-code for the first stage. The input consists of the region ordering given by the classifier and initial area and zone orderings for the first region that needs to be visited. The algorithm starts by clustering the stops of the input route with the same zone ID (excluding the depot). For each region, the algorithm then takes all area IDs present in the given route and orders them according to the current area order AO. Finally, the ordering of zones within each area is done according to the current zone order ZO. Following the patterns described in Section 3.2, the area and zone orders are reversed from ascending to descending (and vise-versa) when moving between different regions and areas, respectively.

Algorithm 3: GetSequence

```
Result: Cluster sequence
Input: s - input route, RO - region order, AO - area order, ZO - zone order
S ← s.stops, zoneIDs ← s.zoneID, RegionCodes ← s.RegionCode,
  zoneIDList ← empty list;
Sort RegionCodes according to order RO;
foreach region r in RegionCodes do
    Q ← all areas in region r;
    Sort Q acroding to order AO;
    foreach area a in Q do
        Z ← all zoneIDs in area a;
        Sort Z acroding to order ZO;
        zoneIDList.append(Z);
        Reverse zone order ZO;
    end
    Reverse area order AO;
end
```

The input of the second stage consists of the cluster sequencing decisions obtained from the first

stage. During the constructive step of the second stage, we obtain an initial feasible node sequence by starting the route from the depot and visiting all nodes from the first cluster before moving to the next one. We use a closest insertion mechanism in which at every iteration, we identify the closest unvisited node within the current cluster with respect to the current node and visit it. Once we finish visiting all nodes of the current cluster, we move to the next cluster by identifying again the closest node that belongs to the next cluster. Once we finish visiting all nodes of all clusters, we finish the route by going back to the depot. This provides an initial solution which is then improved with a local search. We consider 3-opt and 4-opt neighborhoods and explore them sequentially. At every iteration of the local search, 3-opt and 4-opt neighborhoods consider only feasible routes (with respect to clustering constraints) which differ with respect to the current one by adding/subtracting exactly three and four arcs, respectively. We use a best improvement strategy and we only explore 4-opt moves when the current solution cannot be improved with a 3-opt move.

The overall two-stage heuristic is depicted in Algorithm 4. Given an input route r, along with the region and area classifiers from the model build phase, the algorithm starts by determining the region ordering (RO) and area ordering (AO) of the first region that needs to be visited. Preliminary computational experiments showed our algorithm works best with the region ordering applied as a hard constraint, i.e., the region ordering should never change with respect to what is predicted by the region classifier. However, area ordering should be applied as a soft constraint. That is, although the area classifier identifies the order in which a set of areas of the same region should be served, we still check if it would be better to follow the opposite ordering. In order to override the classifier's choice, we add a condition stating the improvement on the routing cost should be at least 2% as compared to the cost of the classifier's ordering.

Algorithm 4: Model Apply

Result: Proposed stop sequence
Input: r - input route, RCLF- region classifier, ACLF - area classifier
$RO \leftarrow$ RCLF.predict(r), $AO \leftarrow$ ACLF.predict(r);
$bestDist \leftarrow \infty$, $bestSequence \leftarrow$ empty list;
foreach *initial area ordering AO in [acending,decending]* **do**
  foreach *initial zone ordering ZO in [acending,decending]* **do**
    $ClusterSequence \leftarrow$ GetSequence(r, RO, ao, zo);
    $currentDist, StopSequence \leftarrow$ SolveFixedCTSPP($r, ClusterSequence$);
    if *ao = AO* **then**
      $currentDist \leftarrow currentDist * 0.98$;
    end
    if *currentDist* $<$ *bestDist* **then**
      $bestDist \leftarrow currentDist$;
      $bestStopSequence \leftarrow StopSequence$;
    end
  end
end
Return $bestStopSequence$

Chapter 4

Computational Experiments

In this section we present the score distribution for the 6-fold cross-validation experiment we conducted during the competition to validate the heuristic performance. Next we compare our heuristic with the Amazon competition prize winners using the 5-fold cross validation method on the training dataset. Finally, we show the result comparing out method with the competition prize winner using the actual validation set from Amazon Routing Challenge, which was released one year after the competition in May 2022.

4.1 Competition submission result using 5-fold cross validation

Our data-driven algorithm was coded in a combination of *Python* and *C* and run on an Intel Xeon E5-2687W v3 processor at 3.10 GHz with a limit of 64 GB of RAM under Linux environment. To speed up the performance of our algorithm, we used the multi-core programming tools of *Python Dask* package. We evaluated the performance of the algorithm using a k-fold cross validation scheme with $k = 6$. During our computational experiments, we used all 6,010 high and medium quality routes provided in the competition. We observed there is no significant conflict in terms of precedence constraints in these two sets of routes. Each fold took about five minutes to run, including model build and model apply phases, which is well within the time limit of the Amazon Competition. The competition limits runtime of 12 hours for model build phase and 5 hours for model build phase.

Table 4.1 gives the results of our computational experiments using the six-fold cross validation, where the model is trained using about 5,000 routes and is then tested with about 1,000 routes. The minimum (best), average, and maximum (worst) obtained average route scores are 0.0278, 0.0293, and 0.0302, respectively. Figure 4.1 shows the obtained route score histogram for the first-fold. From this figure, we note our algorithm performs very well on most of the test routes, with around half of the test routes having scores smaller than 0.018. We noted that in most of the routes were our algorithm performed very well, they conform the patterns described in Section 3.2. In the worst-case scenario, the score is as high as 0.32, which clearly indicates a rather poor performance on some of the more challenging routes. We noted that in most routes were our algorithm performed poorly, the observed routes either do not conform the predicted cluster patterns or there were several time windows constraints violations.

Given that our main contribution is a two-stage approach, it is important that we verify that a low score is still achievable, in principle, by the second stage, so long as the first stage succeeds; in other words, we want to show that good overall scores can be achieved even if these two ”levels” of the problem are solved in separate stages. In fact, we found that the first stage (zone- or cluster-level stage) is most important, and that if it is predicted correctly the second stage (stop-level stage) can achieve surprisingly low scores. Specifically, we verified this by running an experiment where we trained our algorithm on all medium and high quality routes and tested it in all high quality routes. The results are given in Figure 4.2 and show that even if all high quality routes are used for both training and testing, the results are still very similar to the six-fold cross validation, with an average route score of 0.0278.

k	**Score**	**Test routes**	**Train routes**	**Total**
1	0.0287	1,002	5,008	6,010
2	0.0278	1,002	5,008	6,010
3	0.0302	1,002	5,008	6,010
4	0.0297	1,002	5,008	6,010
5	0.0300	1,001	5,009	6,010
6	0.0297	1,001	5,009	6,010

Table 4.1: Computational result using a six-fold cross validation

In order to assess the potential of our proposed model (i.e., the CTSPP), we run an additional set

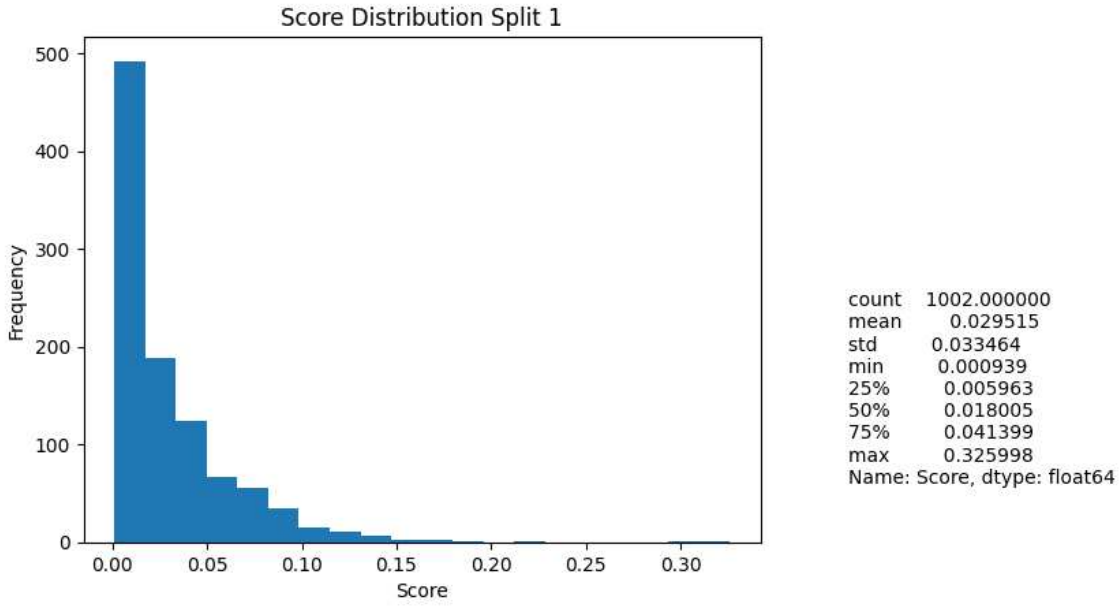


Figure 4.1: One-fold route score histogram

of experiments in which we assume we know the optimal clustering decisions. For each high quality route, we analyzed and recorded the observed cluster sequences. Whenever a cluster is visited more than once in the cluster sequence, we only keep the position in the cluster sequence for the largest *block* of stops of such cluster. We use this observed cluster sequence in the CTSPP to fix the associated y_{kt} variables to one (or zero) and solve the remaining subproblem (on the x_{ij} variables). We developed an exact branch-and-cut algorithm and implemented it using the *Callable Library* of CPLEX 20.1. In our opinion, the obtained results are remarkable. The average route score for all high quality routes is only 0.0065 when using the branch-and-cut algorithm to optimize the node sequencing decisions, and 0.0074 when using Algorithm 4 instead. This means that if we find an effective way to improve the cluster sequencing decisions, specially for routes in which our algorithm performs poorly, there is a potential to significantly improve our current best obtained results. On this line, we developed (and experimented) with a third neighborhood in our local search to modify the cluster sequencing decisions. However, a composite function that uses the total route cost and a penalization term for time windows constraints violations is not enough to effectively guide the search to consistently improve route scores. We conjecture that more sophisticated composite functions are needed to significantly improve our results.

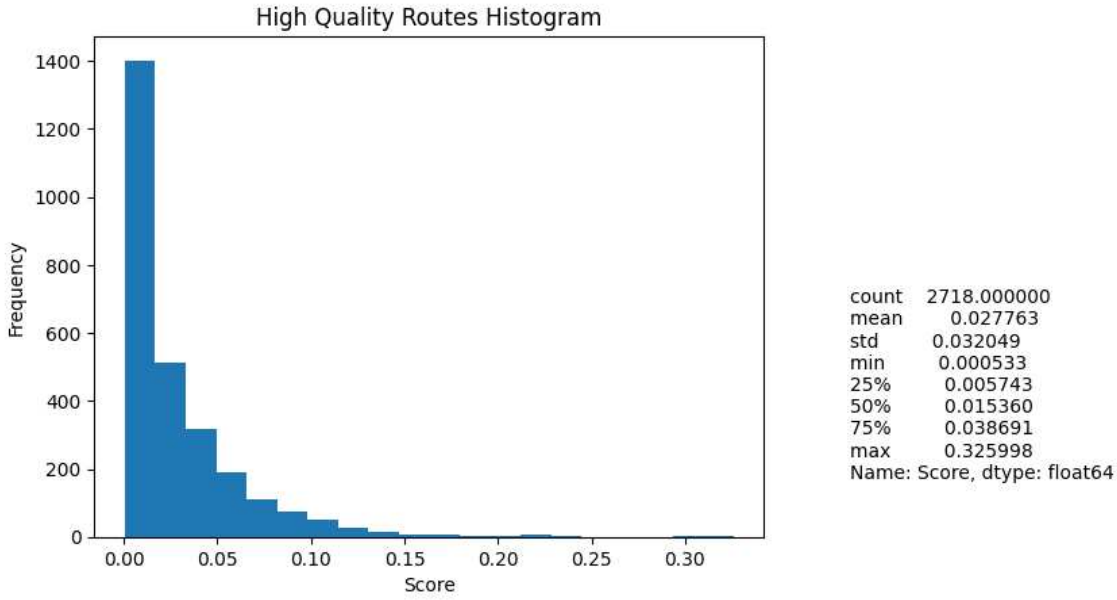


Figure 4.2: High quality route score histogram

A potential line of research we are currently investigating is to identify a family of features to define arc-cost functions $c_{ij}(u, w)$ and sequence-dependent costs functions $f_{kt}(u, w)$ that can provide a better approximation for the observed high quality routes. Having access to expert-knowledge of drivers, would be highly beneficial for this endeavor. Another line we are interested in is to study how our approach can be extended to train using instances on a set of depots and test it on unseen depots. Our current learning mechanism relies on the assumption that the testing set will contain routes associated with regions and areas seen previously during training.

4.2 Comparison with prize winners using 5 fold cross validation

In this section, we present the computational result for our method when compared to the prize winners from Amazon Routing Challenge. Even though our methodology showed competitive result, we overlooked the possibility of having test depots that were not present in train set. Our code failed to complete evaluation and was disqualified from the competition. However because the competition require participants to publish their code for open source, we are able to run the prize winners code and obtain a comparison score.

We decided to run the code from top 3 teams in Figure 2.3 to make a fair comparison, however

the second team code cannot be compiled and therefore removed from the result. We used a 5-fold cross validation procedure on the historical dataset of 6,001 routes to run the experiment. The test result score is presented in Table 4.2.

Baselines:	Score	Std. Dev
1st place team (Cook et al.)	0.0237	±0.0014
Our final submission	0.0280	±0.0015
2nd place team (Permission denied)	-	-
3rd place team (Sky is the limit)	0.0428	±0.0022
Pure travel time	0.0524	±0.0019

Table 4.2: Five-fold test score and standard deviation comparison

4.3 Comparison with prize winners using actual competition validation dataset

In this section, we present the score comparison using the actual validation set from Amazon with our proposed method and the prize winner solutions, which is given in Table 4.3. This experiment shows that Cook et al. team solution performs well in the validation set and their method score similarly with the 5-fold cross validation score. On the other hand, our final submission score deteriorate when comparing with the 5-fold cross validation, which indicate that there are potential over fitting issues or there are routes in the validation set that does not conform with our assumption listed above.

By comparing this result with Amazon published final leader board in Figure 2.3, we can confidently state that our proposed method could have been among the top three teams. Note that the scores presented below differ slightly than Figure 2.3 and we expect that it's due to numerical round up.

Baselines:	Validation Score
1st place team (Cook et al.)	0.0249
Our final submission	0.0389
2nd place team (Permission denied)	-
3rd place team (Sky is the limit)	0.0395

Table 4.3: Amazon held-out test set test score comparison

Chapter 5

Conclusion and Future Research

In this thesis, we propose a novel method that combines the clustered traveling salesman problem and a two stage heuristic to learn precedence constraints from historical data and leverage this tacit information for route planning. It furthermore has speed advantages by not having weight-sensitive features that require extensive training. It was unfortunate that our code did not consider an edge case where a test depot doesn't appear in training set and lead to an exception error. We believed that our approach would have produced a strong final score and be awarded top 3 place.

In the future, we want to continue our work with this topic by extending the model to incorporate time window constraints, which is one feature that we did not include yet. By doing this, we hope to reduce the "long tail" phenomenal presented in our histograms.

Also, our model cannot scale from one depot station to another, given the fact that precedent constraints are learned per depot. We believe that we can incorporate a clustering mechanism that can relate unseen depots to an existing one and apply the rules accordingly.

References

Amazon (2021). Amazon last mile challenge. https://routingchallenge.mit.edu/about-the-challenge/. Accessed: 2022-01-07.

Amazon (2022). Amazon delivery service partner program. https://logistics.amazon.ca/marketing/opportunity. Accessed: 2022-01-07.

Applegate, D., Bixby, R., Chvatal, V., and Cook, W. (1998). On the solution of traveling salesman problems. *Documenta Mathematica Journal der Deutschen Mathematiker-Vereinigung*, page 645–656.

Baldacci, R., Mingozzi, A., and Roberti, R. (2012). New state-space relaxations for solving the traveling salesman problem with time windows. *INFORMS Journal on Computing*, 24(3):356–371.

Bengio, Y., Lodi, A., and Prouvost, A. (2018). Machine learning for combinatorial optimization: a methodological tour d'horizon. *CoRR*, abs/1811.06128.

Bestuzheva, K., Besançon, M., Chen, W.-K., Chmiela, A., Donkiewicz, T., van Doornmalen, J., Eifler, L., Gaul, O., Gamrath, G., Gleixner, A., Gottwald, L., Graczyk, C., Halbig, K., Hoen, A., Hojny, C., van der Hulst, R., Koch, T., Lübbecke, M., Maher, S. J., Matter, F., Mühmer, E., Müller, B., Pfetsch, M. E., Rehfeldt, D., Schlein, S., Schlösser, F., Serrano, F., Shinano, Y., Sofranac, B., Turner, M., Vigerske, S., Wegscheider, F., Wellner, P., Weninger, D., and Witzig, J. (2021). The SCIP Optimization Suite 8.0. Technical report, Optimization Online.

Biggs, N., Lloyd, E. K., and Wilson, R. J. (1986). *Graph theory 1736 - 1936*. Clarendon Press.

Chisman, J. A. (1975). The clustered traveling salesman problem. *Computers and Operations Research*, 2(2):115–119.

Dantzig, G., Fulkerson, R., and Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the Operations Research Society of America*, 2(4):393–410.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, 4(1):61–75.

Gasse, M., Chételat, D., Ferroni, N., Charlin, L., and Lodi, A. (2019). Exact combinatorial optimization with graph convolutional neural networks. *CoRR*, abs/1906.01629.

Helsgaun, K. (2000). An effective implementation of the lin–kernighan traveling salesman heuristic. *European Journal of Operational Research*, 126(1):106–130.

Kanellakis, P.-C. and Papadimitriou, C. H. (1980). Local search for the asymmetric traveling salesman problem. *Operations Research*, 28(5):1086–1099.

Keenan, M. (2022). Global ecommerce: Stats and trends to watch to succeed internationally (2022).

Korte, B. and Vygen, J. (2012). *Combinatorial Optimization Theory and algorithms*. Springer Berlin Heidelberg.

Laporte, G. and Palekar, U. (2002). Some applications of the clustered travelling salesman problem. *Journal of the Operational Research Society*, 53(9):972–976.

Lu, Y., Hao, J.-K., and Wu, Q. (2020). Solving the clustered traveling salesman problem via tsp methods.

Matthias, W., Steven, P., and Joseph, N. (2021). Technical proceedings of the amazon last mile routing research challenge. *MIT Megacity Logistics Lab*.

Nagata, Y. and Kobayashi, S. (2013). A powerful genetic algorithm using edge assembly crossover for the traveling salesman problem. *INFORMS Journal on Computing*, 25(2):346–363.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Precedence, R. (2022). Last mile delivery transportation market size to hit us$ 424.3 bn by 2030. https://www.globenewswire.com/news-release/2022/03/24/2409950/0/en/Last-Mile-Delivery-Transportation-Market-Size-to-Hit-US-424-3-Bn-by-2030.html. Accessed: 2022-01-07.

Roberti, R. and Toth, P. (2012). Models and algorithms for the asymmetric traveling salesman

problem: an experimental comparison. *EURO Journal of Transportation Logistics*, 1:113–133.

Soper, T. (2020). Amazon delivery service partner program creates nearly 85k jobs, 1,300 small businesses in 2 years. https://www.geekwire.com/2020/amazon-delivery-service-partner-program-creates-nearly-85k-jobs-1300-small-businesses-2-years/. Accessed: 2022-01-07.

Taillard, E. D. and Helsgaun, K. (2019). Popmusic for the travelling salesman problem. *European Journal of Operational Research*, 272(2):420–429.

Vander Wiel, R. J. and Sahinidis, N. V. (1995). Heuristic bounds and test problem generation for the time-dependent traveling salesman problem. *Transportation Science*, 29(2):167–183.

Vu, D. M., Hewitt, M., Boland, N., and Savelsbergh, M. (2020). Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. *Transportation Science*, 54(3):703–720.