使用定制成本矩阵的分层旅行商问题进行亚马逊最后一英里配送轨迹预测

郭晓彤、莫百川、王清怡 [1]*Department of Civil and Environmental Engineering, Massachusetts Institute of Technology77 Massachusetts Ave, Cambridge, MA, USA*

arXiv:2302.02102v1 [math.OC] 4 Feb 2023

摘要

为应对亚马逊最后一英里路径规划挑战赛，Permission Denied 团队提出了一种采用定制成本矩阵的分层旅行商问题（TSP）优化方案。高层 TSP 用于求解区域序列，低层 TSP 则用于求解区域内停靠点序列。该方案对成本矩阵进行了调整，以考虑最短行驶时间之外的路径规划模式。最后，通过后处理调整序列，使其符合常见的路径规划模式；例如，当旅行时间相近时，驾驶员通常会优先前往包裹较多的停靠点。模型在从训练集中随机选取的 1223 条路线中进行了测试，得分为 0.0381。在给定的 13 条模型应用集路线中，得分为 0.0375。

1. 引言

本报告介绍了 Permission Denied 团队参加亚马逊最后一英里路径规划研究挑战赛时的思考过程、所选方法论以及预期结果。总的来说，团队在提交最终方案前经历了四个阶段。

描述性分析：收到挑战后，我们进行了全面的描述性分析。首个重要发现是，在大多数情况下，驾驶员会在前往另一个区域的停靠点之前，完成当前区域内的所有配送任务。只有在存在回溯时，这一规律才会被打破。进一步查看得分后，这一判断得到证实：假设区域序列和区域内停靠点序列正确，仅因某些区域被重复访问而造成的得分损失只有 0.009。如果区域序列正确，但各区域内的停靠点顺序被打乱，平均得分约为 0.02。因此，准确预测区域序列最为重要，团队决定采用分层方法：先求解区域序列，再求解区域内停靠点序列。这样大幅缩小了问题规模，因为大多数路线约有 150 个停靠点（最多可达 250 个），而区域数量介于 6 到 47 个之间。其次，我们研究了区域转移概率。由于大多数区域在训练集中仅出现一次，频数制表的尝试并不成功。另一方面，74% 的区域转移会选择旅行时间最短的区域，这使得逐步预测算法有望奏效。接下来，我们研究了包裹尺寸、包裹数量、配送时间窗与序列顺序之间的相关性，但未发现明显的关系。

[1]按姓氏字母顺序

也就是旅行时间最短的区域，这使得逐步预测算法有望奏效。接下来，我们研究了包裹尺寸、包裹数量、配送时间窗与序列顺序之间的相关性，但未发现明显的关系。

基准： 我们建立了一个基准模型，用于了解解决方案的质量和预期性能。由于大多数驾驶员都会遵循亚马逊提供的导航，因此最短距离巡回路线自然成为一种基准。团队分别求解基于巡回路线的旅行商问题（起点和终点车站均为 INIT），以生成区域序列；以及基于路径的旅行商问题（不计从最后一个区域到 INIT 的距离），以生成区域内停靠点序列，作为基准。在每个区域内，团队求解一个基于路径的旅行商问题，路线从距离上一个区域最近的停靠点出发，前往距离下一个区域最近的停靠点。

模型尝试： 两种朴素旅行商问题解的得分都还算合理（约为 0.06）。为提升性能，团队尝试了机器学习模型。首先，团队发现，正确预测首个区域可以显著提升旅行商问题的性能，因此构建了一个神经网络，根据旅行时间、距离、包裹数量和尺寸等信息预测首个区域。其次，团队探索了用于生成序列的纯机器学习模型，包括根据先前预测的停靠点预测下一个元素的短视方法，以及对整个序列进行编码和解码的序列到序列（seq2seq）方法。第三，团队考虑了不同的训练方法，包括传统的交叉熵损失、自定义加权损失，以及使用策略梯度的强化学习。最后，团队通过对非连续区域编号添加惩罚成本，改进了一些基准旅行商问题模型。由于样本量较小（6k），机器学习技术无法胜过基准模型。在尝试了各种建模技术后，团队决定将旅行商问题解作为最终提交结果。

超参数搜索与后处理：定制成本矩阵涉及一些超参数，团队在给定的训练集上对这些参数进行了搜索。最后，我们还确定了一些后处理模式，以进一步提升解决方案的质量。

最终提交的模型有以下亮点：

- 分层建模——为缩小每个优化问题的规模，我们将问题拆解为区域路径规划和区域内停靠点路径规划。

- 定制旅行商问题成本矩阵——为考虑最短距离以外的因素，我们对成本矩阵进行了调整，使旅行商问题的性能提升了近 0.01。

- 通过后处理匹配行为模式——我们会反转某些旅行商问题的访问序列，以适应配送模式；例如，在其他条件相同的情况下，包裹较多的停靠点会优先于包裹较少的停靠点访问。

- 稳定的超参数——成本超参数具有良好的泛化能力，无需重新训练。

技术报告的其余部分将回顾相关文献及其与研究问题的契合度；详细介绍所选模型，并讨论预期结果。

2. 文献综述

这个问题本质上是一个车辆路径问题，但传统车辆路径问题的目标是使行驶距离最短，而这里关注的问题则是寻找与观测序列最相似的序列。机器学习和运筹学这两个研究领域对车辆路径问题进行了广泛研究。本文将回顾这两个领域的相关文献，并讨论各类算法在解决当前问题时的优缺点。

2.1. 运筹学

给定一组需要访问的位置，可以通过求解旅行商问题（TSP）来找到成本或距离最小的路线。Applegate 等人（2011）介绍了旅行商问题的概述及其历史。尽管旅行商问题是组合优化中一个众所周知的 NP-hard 问题，但现成的整数优化求解器（如 Gurobi 和 GLPK）能够高效地求解实际问题实例。我们在求解旅行商问题时采用的一种关键方法是割平面法（Marchand 等人，2002），该方法最初由 Dantzig 等人（1954）应用于旅行商问题。

2.2. 机器学习

可使用两种架构对输入序列重新排序：逐步预测或序列到序列（seq2seq）。逐步预测是指根据先前停靠点的信息以及候选停靠点，逐个预测停靠点。由于候选停靠点的信息至关重要，前馈神经网络并不适合，因为它无法为候选项分配特征。因此，我们采用了带有备选项特定效用的前馈神经网络（Wang 等人，2020）。该架构将离散选择模型与神经网络联系起来，并使用神经网络为每个候选项生成效用，然后选择“效用”最高的候选项。随后，将选中的停靠点反复输入算法，以预测下一个停靠点，直至序列结束，从而形成一个序列。该算法的优势在于其处理粒度是停靠点，而不是序列。因此，对于机器学习算法的成功至关重要的样本量，显著大于 seq2seq 模型。该算法的缺点是具有短视性，在做出选择时只能看到下一步的候选项。

近年来，人们开发了许多 seq2seq 预测算法，主要用于自然语言处理（NLP）任务。与逐步预测相比，seq2seq 模型由一个编码器和一个解码器组成。序列中的所有元素都会在解码开始前完成编码，因此模型能够获得全局视图。编码器和解码器的架构通常采用循环神经网络的变体（例如长短期记忆网络）（Sutskever 等人，2014），或注意力机制（Vaswani 等人，2017）。大多数 seq2seq 问题关注的是将一个序列映射到另一个序列，而本文关注的问题则是对输入序列重新排序。指针网络旨在解决这类问题：解码器利用自注意力指向输入元素之一（Vinyals 等人，2015）。作者使用了一个

指针网络来解决旅行商问题，并取得了与旅行商问题求解器相近的性能。原始指针网络的一个缺点是对输入顺序较为敏感。因此，作者增加了另一个编码模块，以消除这种影响（Vinyals 等人，2016）。然而，在我们的实验中，可以通过将输入集合按有意义的顺序排列来利用这种依赖关系，从而提升性能。例如，按照旅行商问题的序列对输入停靠点排序，可以加快模型收敛并提高得分。不过，在上述论文中，网络使用了 100 万个训练样本。鉴于训练集仅包含 6000 条路线，旅行商问题求解结果的得分未能得到提升。

原始指针网络使用交叉熵损失（监督学习）。在这个问题中，由于评分的计算方式，交叉熵损失的效率很低：损失只考虑正确位置的概率，而对所有其他位置预测错误所产生的损失都相同。但评分函数除了正确性之外，还会考虑相似度。评分函数不可微，因此无法直接用作损失函数并进行梯度下降。另一种训练方法是基于策略梯度的强化学习（Ma 等人，2019；Bello 等人，2019）。使用广为人知的 REINFORCE 算法，我们可以直接优化不可微的评分函数。研究人员发现，与监督学习相比，这种方法在旅行商问题上具有相同的样本效率和更强的泛化能力（Joshi 等人，2019）。然而，针对这个特定问题，在给定样本量和信息的情况下，使用强化学习进行训练也无法优于旅行商问题的解法。

2.3. 所提方法

我们提出的方法基于传统旅行商问题，并采用定制的距离矩阵，该矩阵隐含地包含了亚马逊最后一公里配送中司机的行驶行为。与现有的旅行商问题框架不同，后者旨在最小化车辆行驶总距离；我们对距离矩阵进行了调整，并生成了使调整后的总行驶距离最小化的最优路线。

3. 方法论

3.1. 数据

我们观察到，大多数驾驶员倾向于在前往下一个区域之前，先访问某一区域内的所有停靠点。因此，我们将问题分为两个子问题：第一个是确定区域序列，第二个是识别区域内停靠点序列。

实际区域序列根据各区域首次出现的顺序生成。图 1 展示了一个示例。对于没有区域 ID 的停靠点（由于数据缺失），我们将其区域 ID 填为距离（基于旅行时间）最近的停靠点的区域 ID。

观察区域序列时，可以发现三个重要特性：

- 司机最有可能先完成一个“主要区域”，然后再前往下一个“主要区域”。主要区域定义为点号前的区域 ID。例如，“A-2.2A”的主要区域是“A-2”。例如，在图 1 中，司机先完成主要区域“A-2”，然后是“A-1”，最后是“P-13”。

- 在特定的主要区域内，两个相邻的“内部区域”ID 最有可能“相差 1”。“内部区域”定义为点号后的区域 ID。例如，“A-2.2A”的内部区域是“2A”。“相差 1”的定义如下。给定两个内部区域 ID“XY”和“AB”，其中 X 和 A 是数字，Y 和 B 是字符，则有

$$|X - A| + |\texttt{ord}(Y) - \texttt{ord}(B)| = 1 \tag{1}$$

其中 ord(·) 函数返回一个表示 Unicode 字符的整数。例如，“1A”和“1B”相差一，“1A”和“2A”也相差一。但“1A”和“2B”相差二。

- 当司机完成一个“主要区域”并前往另一个区域时，相邻的两个主要区域 ID 最有可能“相差一”。例如，在图 1 中，司机先完成主要区域“A-2”，然后前往“A-1”。这两个主要区域 ID 相差一。

Zone sequence	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Zone id	INIT	A-2.2A	A-2.1A	A-1.1A	A-1.2A	A-1.3A	A-1.3B	A-1.1B	A-1.1C	A-1.2C	A-1.3D	A-1.3C	A-1.2D	A-1.1D	P-13.2A	P-13.1C	P-13.2C	P-13.3C	P-13.3D

图 1：区域序列示例。“INIT”表示配送站

为验证这三项性质，我们计算了这些规则在数据集中的成立频率。在所有相邻的区域 ID 对中，87.67 % 的区域 ID 对具有相同的主要区域 ID（性质 1）。在特定主要区域内的所有相邻区域 ID 对中，82.49% 相差一（性质 2）。在所有主要区域 ID 发生变化的相邻区域 ID 对中，96.17% 的变化会导致两个主要区域 ID 相差一（性质 3 ）。这些统计数据支持这三项性质，这意味着**区域 ID 包含大量有助于序列估计的信息**。

我们使用的另一项信息是计划服务时间和包裹体积。关于如何使用这些信息的详情，请参见第 3.3 节。

我们还从 OpenStreetMap 收集了外部数据源。具体而言，我们提取了每个停靠点周围的交通信号灯和高速公路匝道数量。遗憾的是，这并未帮助改进模型，因此最终提交中不再使用这些数据。

为了验证模型，我们将 6,112 条路线随机划分为训练数据集（4,889 条路线）和测试数据集（1,223 条路线），尽管我们提出的解决方案不需要训练过程。

3.2. 旅行商问题的形式化

观察到驾驶员会先访问同一区域内的所有停靠点，然后再前往下一个区域，我们通过求解一个采用修改后行驶时间矩阵的标准旅行商问题，先生成区域序列，再求解多个路径旅行商问题，以确定区域内的停靠点序列。

首先，我们给出用于生成区域序列的标准旅行商问题的数学表述。对于包含 n 个区域的路线实例，区域集合的索引为 $[n] = \{1, ..., n\}$，初始车站位置以索引 0 表示。令 V 表示所有需要访问的位置的集合，其中包括初始

车站，即 $V = \{0, 1, ..., n\}$。 t_{ij}表示任意两个位置之间的旅行时间，即 $\forall i \neq j \in V$。任意两个区域之间的旅行时间，按这两个区域之间所有可能停靠点对的平均旅行时间计算。该问题的决策变量为 $x_{ij} \in \{0, 1\}$，$\forall i,\ j \in V$。 $x_{ij} = 1$表示司机在访问位置 i后将访问位置 j。因此，旅行商问题可表述为：

$$\min \quad \sum_{i=0}^{n} \sum_{j=0}^{n} t_{ij} x_{ij} \tag{2a}$$

$$\text{s.t.} \quad \sum_{i=0}^{n} x_{ij} = 1 \quad \forall j \in V \tag{2b}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \quad \forall i \in V \tag{2c}$$

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1 \quad \forall S \subset V, S \neq \emptyset, V \tag{2d}$$

$$\sum_{i \notin S} \sum_{j \in S} x_{ij} \geq 1 \quad \forall S \subset V, S \neq \emptyset, V \tag{2e}$$

$$x_{ii} = 0 \quad \forall i \in V \tag{2f}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V \tag{2g}$$

目标函数（2a）旨在最小化巡回路线的总旅行时间。约束（2b）和（2c）确保最优巡回路线中的每个访问地点恰好有一个前驱和一个后继。约束（2d）和（2e）用于消除最优巡回路线中的子回路。约束（2f）避免出现自环，约束（2g）则保证决策变量为二进制变量。

问题（2）是一个整数线性规划（ILP），由于约束（2d）和（2e），其约束数量呈指数级增长。为高效求解该问题，我们将约束（2d）和（2e）均实现为惰性约束，即只有在当前最优解中识别出子回路时，才将这些约束添加到问题中。

为考虑区域序列中的观测结果（第 3.1 节），我们提出了三种启发式方法来修改行驶时间矩阵，该矩阵是生成最优区域序列的输入。

1. 对于从初始站到区域 i的旅行时间，如果该区域既不在按照旅行时间计算的距离初始站最近的 h 个区域之内，也不在按照欧几里得距离计算的距离初始站最近的 h 个区域之内，则将旅行时间修改为 $t_{0i} * \alpha$，其中 α 和 h 均为第一种启发式方法的参数。2. 对于任意两个区域 i 和 j之间的旅行时间，如果区域 i 和区域 j 不属于同一个“主要区域”，则将旅行时间修改为 $t_{ij} * \beta$，其中 β 是第二种启发式方法的参数。3. 对于任意两个区域 i 和 j之间的旅行时间，如果它们属于同一个“主要区域”，且其区域 ID 中点号后的部分之差不等于 1，则将旅行时间修改为 $t_{ij} * \gamma$，

其中 γ 是第三种启发式方法的参数。

在最终提交的算法中，我们采用网格搜索方法来确定全部四个启发式参数的取值： $h = 9$, $\alpha = 1.04$, $\beta = 3.8$, $\gamma = 2.5$。

使用修改后的行驶时间矩阵求解问题 (2)，可得到最优区域序列[2] S^* =(0， s_1，.....， s_n)，其中 s_i 表示离开初始站后，按最优序列访问的第 i 个区域。然后，我们使用基于路径的旅行商问题求解区域内停靠点序列。给定一组需要访问的位置 V，以及起始位置 v_o 和结束位置 v_d，可将路径旅行商问题表述如下：

$$\min \quad \sum_{i=0}^{n}\sum_{j=0}^{n} t_{ij}x_{ij} \tag{3a}$$

$$\text{s.t.} \quad \sum_{i=0}^{n} x_{ij} = 1 \quad \forall j \in V \setminus \{v_o, v_d\} \tag{3b}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \quad \forall i \in V \setminus \{v_o, v_d\} \tag{3c}$$

$$\sum_{j\in V} x_{v_o j} = \sum_{i\in V} x_{i v_d} = 1 \tag{3d}$$

$$\sum_{j\in V} x_{v_d j} = \sum_{i\in V} x_{i v_o} = 0 \tag{3e}$$

$$\sum_{i\in S}\sum_{j\notin S} x_{ij} \geq 1 \quad \forall S \subset V, S \neq \emptyset, V \tag{3f}$$

$$\sum_{i\notin S}\sum_{j\in S} x_{ij} \geq 1 \quad \forall S \subset V, S \neq \emptyset, V \tag{3g}$$

$$x_{ii} = 0 \quad \forall i \in V \tag{3h}$$

$$x_{ij} \in \{0, 1\} \quad \forall i, j \in V \tag{3i}$$

路径旅行商问题 (3) 与标准旅行商问题 (2) 类似，但起始位置 v_o 没有前驱，结束位置 v_d没有后继，这由约束条件 (3d) 和 (3e) 表示。完整序列根据生成的区域序列，按照算法1生成；最终实现中使用了一个启发式参数 $k = 3$ 。

值得一提的是，所有旅行商问题实例均使用开源整数线性规划求解器 GLPK 求解，该求解器以 Julia 编程语言（Bezanson et al., 2017）和优化软件包 JuMP（Dunning et al., 2017）实现。在生成完整的停靠点序列 $S^*_{complete}$ 后，我们进入后处理阶段，进一步提升序列性能。

[2]不失一般性，我们可以假设序列从编号为 0.

算法1 根据生成的区域序列生成完整序列。

输入：最优区域序列 $S^* = (0, s_1, ..., s_n)$，启发式参数 k。

函数 CompletePathGeneration(S^*)

$S^*_{complete} \leftarrow \{0\}$ ▷ 使用初始车站初始化完整序列

对于 $s_i = s_1, ..., s_n$，**执行**

查找前一个访问区域 s_{i-1} 和下一个访问区域 s_{i+1}

计算区域 s_{i-1} 和区域 s_{i+1} 中任意停靠点 $v \in s_i$ 到所有停靠点的平均出行时间

将区域 s_i 中相对于区域 s_{i-1} 最近的 k 个停靠点作为集合 M

将区域 s_i 中相对于区域 s_{i+1} 最近的 k 个停靠点作为集合 N

在 $M \times N$ 中任意一对停靠点之间求解 k^2 路径旅行商问题 (3)。

将路径 S^*_i 作为区域 i 中旅行时间最短的最优序列

将序列 S^*_i 追加到完整序列 $S^*_{complete}$

返回 $S^*_{complete}$

3.3. 后处理

通过旅行商问题求解停靠点序列后，我们发现，大多数得分较高（即性能较差）的路线都是由于序列部分或完全反向所致（例如，序列 A-B-C-D 被错误地估计为 D-C-B-A）。因此，我们提出一种后处理方法，以纠正因反向而导致的错误估计。

我们从数据集中观察到两个特性：

- 大多数驾驶员倾向于先服务商业区。一个可能的原因是，在商业建筑中配送包裹也需要更长时间。先服务这些地点，可以让行程结束时的总服务时间更易于控制。因此，我们预计前几个停靠点的计划服务时间会长于最后几个停靠点。

- 大多数驾驶员倾向于先配送大件包裹。这可能是因为在车辆中装载大件包裹并不燃油高效。

基于这些特性，对于旅行商问题生成的每个停靠点序列，我们都会检查是否需要将其反向。给定一条生成路线 i，令 p_i^+ （分别为 p_i^- ）表示路线 i中前（分别为后）p% 停靠点的平均计划服务时间。如果 i ，我们将反向路线

$$\frac{p_i^-}{p_i^+} \geq \theta, \tag{4}$$

其中， p 和 θ 是表示停靠点比例和阈值的超参数。我们基于测试集上的交叉验证设置 $p = 15$ 和$\theta = 1.22$ 。公式 5表示，在生成的序列中，如果最后几个停靠点的计划服务时间过长，则可能出现反转错误，需要将整个序列反向以进行纠正。

通过公式 5处理后，我们修正了所有已经反向的序列。对于其余序列，我们进一步根据包裹体积检查是否需要反向。具体而言，给定一条生成路线 i，令 v_i^+ （分别为 v_i^- ）表示路线 i中前（分别为后）15%停靠点的包裹列总量（深度××宽度××高度）。如果

$$\frac{v_i^-}{v_i^+} \geq \eta, \tag{5}$$

其中使用了 $\eta = 3$ 。

后处理后，会进行序列有效性检查。具体而言，我们检查估计序列中的第一个停靠点是否为配送站，以及估计序列中的停靠点 ID 是否与实际序列相同。如果这两项标准中任意一项不满足，我们就按区域 ID 对停靠点排序，以此生成序列，从而确保区域 ID 相同的停靠点彼此相邻。

4. 结果与结论

4.1. 性能

尽管提交的方案不需要训练模型，我们仍将给定的训练集拆分为训练集（4889）和测试集（1223），以便对机器学习模型进行自评估。因此，所有自评估均在测试集上完成。为缩短评估时间，我们使用 Cython 实现了评分函数。与挑战主办团队提供的 Python 评估代码相比，我们的实现评估同一组路线所需的计算时间仅为其三分之一。

图 2 展示了最终算法生成的得分分布。路线性能得分服从指数分布，大多数路线的得分低于 0.1。这 1223 条测试路线的平均路线得分为 0.0381。在给定的模型应用集中，13 条路线的得分为 0.0375。

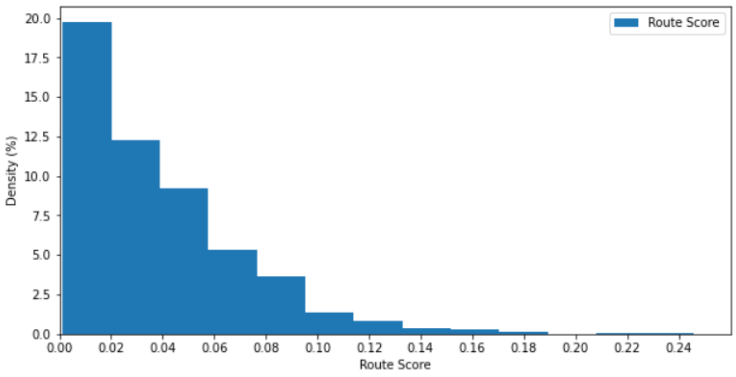


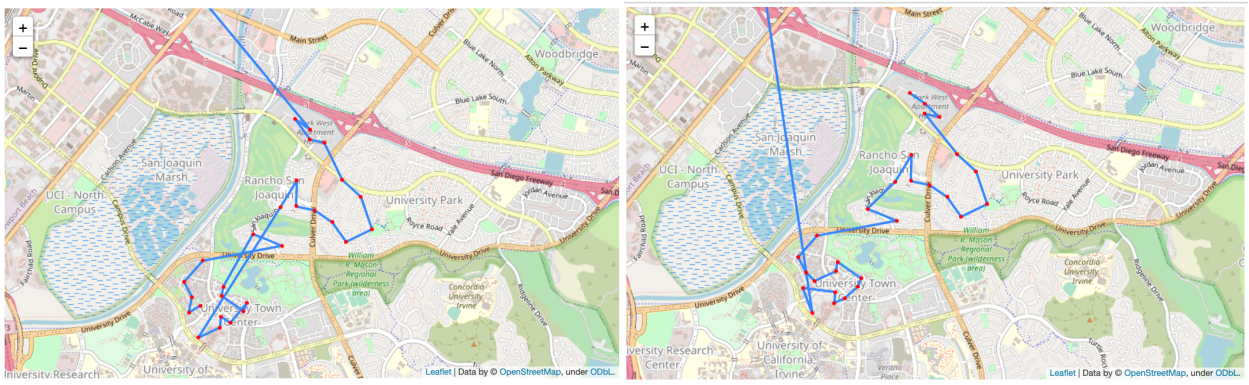
图 2：路线评分性能。

4.2. 讨论

区域序列对得分影响最大。我们发现，如果区域序列预测完全准确，即使区域内的停靠点 ID 顺序被打乱，平均路线得分仍可达到 0.0206。因此，我们的大部分工作都集中在预测区域序列，而不是停靠点序列。

区域 ID 的三项特性（见第 3.1 节）可能意味着驾驶员很可能会遵循规划路线，很少偏离。由于使用区域 ID 是为了“帮助简化路线规划流程”（引自 Slack 问答），我们认为亚马逊规划路线的方式会使区域 ID 呈现出明显的规律。因此，这个问题的主要挑战在于还原亚马逊的路线规划方式。这也解释了为什么在当前给定的信息和样本量下，旅行商问题比机器学习方法效果更好。

逆转问题仍然存在。图 3 展示了一个反向预测示例。由于我们无法将首个区域预测准确率提高到 35% 以上，后处理后，反向问题仍然存在。后处理使我们在测试集上的得分从 0.391 降至 0.381。然而，如果我们能对逆转问题实现 100% 的纠正率（即始终选择得分较低的那个），得分可以降至 0.280，这表明还需要采用一些进一步的纠正方法。请注意，我们曾尝试将周边高速公路匝道的数量用作新指标，也尝试使用机器学习预测首个区域，但这些方法并未提升模型性能。



(a) 实际路线　　(b) TSP 预测路线

图 3：反向预测示例

参考文献

˜Applegate, D. L., Bixby, R. E., ChvA¡tal, V., & Cook, W. J. (2011). 旅行商问题。普林斯顿大学出版社。URL：https://doi.org/10.1515/9781400841103。doi:doi:10.1515/ 9781400841103。

Bello, I.、Pham, H.、Le, Q. V.、Norouzi, M. 和 Bengio, S.（2019年）。利用强化学习进行神经组合优化。第五届学习表征国际会议，*ICLR 2017* - 研讨会论文集，（第 1–15 页）。arXiv:1611.09940。

Bezanson, J.、Edelman, A.、Karpinski, S. 和 Shah, V. B.（2017年）。Julia：一种数值计算的新方法。*SIAM Review*，*59*，65–98。网址：http://www.siam.org/journals/sirev/59-1/100067.html。doi:10.1137/141000671。arXiv:1411.1607。

Dantzig, G.、Fulkerson, R. 和 Johnson, S.（1954年）。大规模旅行商问题的求解。美国运筹学会期刊，*2*，393–410。网址：http://www.jstor.org/stable/ 166695。

Dunning, I., Huchette, J., & Lubin, M. (2017年). JuMP：一种用于数学优化的建模语言。*SIAM Review*, *59*, 295–320. doi:10.1137/15M1020575.

Joshi, C. K., Laurent, T., & Bresson, X. (2019年). 旅行商问题的学习范式, . (pp. 1–9). URL: http://arxiv.org/abs/1910.07210. arXiv:1910.07210.

Ma, Q., Ge, S., He, D., Thaker, D., & Drori, I. (2019年). 基于图指针网络和层次强化学习的组合优化, . URL: http://arxiv.org/abs/1911.04936. arXiv:1911.04936.

Marchand, H., Martin, A., Weismantel, R., & Wolsey, L. (2002). 整数与混合整数规划中的割平面。离散应用数学, *123*, 397–446. doi:10.1016/S0166-218X(01)00348-1.

Sutskever, I., Vinyals, O., & Le, Q. V.（2014）。使用神经网络进行序列到序列学习。载于第二十七届神经信息处理系统国际会议论文集——第二卷NIPS'14（第 3104–3112 页）。美国马萨诸塞州剑桥：MIT出版社。

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I.（2017年）。Attention Is All You Need。神经信息处理系统进展，*2017*年*12*月，5999–6009。网址：http://arxiv.org/abs/1706.03762。arXiv:1706.03762。

Vinyals, O., Bengio, S., & Kudlur, M. (2016). 顺序很重要：集合的序列到序列。第 *4* 届国际学习表征会议，*ICLR 2016* - 会议分会场论文集，（第 1–11 页）。arXiv:1511.06391。

Vinyals, O., Meire, F., & Navdeep, J. (2015). PointerNetworks。神经信息处理系统进展，（第 1–9 页）。

Wang, S., Mo, B., & Zhao, J. (2020年). 用于选择分析的深度神经网络：采用备选项特定效用函数的架构设计。《交通研究*C*辑：新兴技术》，*112*，234– 251。网址：https://www.sciencedirect.com/science/article/pii/S0968090X19310381。doi:https: //doi.org/10.1016/j.trc.2020.01.012。