arXiv:2305.02169v1 [cs.DS] 3 May 2023

学习增强型环、树、花及（几乎）其他所有场景下的在线旅行商问题

Evripidis Bampis ✉ ID索邦大学，法国国家科学研究中心，LIP6，法国巴黎 F-75005

Bruno Escoffier ✉ ID索邦大学，法国国家科学研究中心，LIP6，法国巴黎 F-75005 法国大学研究院，法国巴黎

Themis Gouleakis ✉ ID新加坡国立大学，新加坡，新加坡

Niklas Hahn ✉ ID索邦大学，法国国家科学研究中心，LIP6，法国巴黎 F-75005

Kostas Lakis ✉ ID苏黎世联邦理工学院，瑞士苏黎世

Golnoosh Shahkarami ！ ID马克斯·普朗克计算机科学研究所，萨尔大学，萨尔布吕肯，德国。

Michalis Xefteris ！ ID索邦大学、法国国家科学研究中心、LIP6，F-75005 法国巴黎。

摘要

我们研究带预测的在线旅行商问题（OLTSP）。在 OLTSP 中，一系列起初未知的请求会随时间到达度量空间中的各个点（位置）。目标是从度量空间中的某个特定点（原点）出发，在尽可能缩短总耗时的同时服务所有请求。服务器以单位速度移动，或在某个位置“等待”（速度为零）。我们考虑两种变体：在开放变体中，服务完最后一个请求即达成目标；在闭合变体中，服务器还必须返回原点。我们采用一种预测模型，该模型最初是为直线上的 OLTSP 提出的，其中预测对应于请求的位置 [23], ，并将其扩展到更一般的度量空间。

我们首先提出一种受先前工作 [13] 启发的基于预言机的算法框架。该框架使我们能够为一般度量空间设计在线算法，并为其提供竞争比保证：在完美预测下，竞争比优于经典情形中可能达到的最佳保证（一致性）。此外，随着误差增大，算法性能会平滑退化（平滑性），但始终保持在经典情形中已知最佳竞争比的常数倍以内（鲁棒性）。

将问题归约为设计合适且高效的预言机后，我们介绍如何针对一般度量空间以及特定度量空间（环、树和花）实现这一目标；对于后者，由此得到的算法具有可处理性。我们的算法在几乎所有情况下都具有紧确的一致性保证，其平滑性保证受到的误差线性依赖影响，而我们证明这种依赖是必要的。最后，我们给出优于先前结果的鲁棒性保证。

2012年ACM主题分类 计算理论 → 算法设计与分析

关键词与短语 旅行商问题、在线算法、预测、竞争分析

资助本研究部分由法国国家研究署（ANR）资助，资助项目编号为 ANR-19-CE48-0016。

1 引言

在经典旅行商问题（TSP）中，给定一组位置及其两两之间的距离，目标是找出一条访问所有位置的最短巡回路线。TSP 是计算机科学中最基础且研究最深入的问题之一 [30]。我们关注该问题在度量空间中的在线版本，即在线旅行商问题（OLTSP），该问题由 Ausiello 等人在一篇开创性论文中提出 [10]。在 OLTSP 中，输入会随着时间推移而陆续到达，也就是说，旅客（或服务器）在行程中会收到需要前往访问的新请求（位置）。旅客获知请求的时间称为该请求的释放时间（或释放日期）。目标是最小化总行程时间，并假设旅客在任何时刻要么以单位速度移动，要么在某个位置“等待”（速度为零）[1]。我们研究两种变体：闭合变体要求服务器在服务完所有请求后返回原点；开放 变体则不要求服务器在服务完所有请求后返回原点。一系列论文研究了 OLTSP 在不同度量空间中的多种变体，包括一般度量空间 [10, 13]、直线 [10, 17, 23]、半直线 [9, 13, 18]、环 [13, 27] 和星形空间 [13]。

研究 OLTSP 及其变体的动机，源自许多不同领域的应用，例如物流和机器人技术 [8, 37]。在竞争分析框架下，在线算法的性能通常通过竞争比来评估。竞争比定义为：对于所有输入实例，在线算法的成本与最优离线算法成本之比的最大值。根据定义，最优离线算法预先知道整个输入。然而，人们承认，竞争分析方法可能过于悲观，因为它是基于最坏情况实例计算的，这赋予了对手很大的能力。因此，许多论文试图限制对手的能力 [18]，或者为在线算法提供额外知识，从而赋予其更大的能力 [1, 13, 26]。

近年来，随着机器学习方法和人工智能在基于数据预测和学习未知信息（即在线算法中的未来输入）方面取得显著成功，学习增强（*LA*）算法框架应运而生 [32]。这项研究旨在利用未来输入的预测（这些预测可能由学习算法获得），从而在预测足够准确时获得可证明的更优竞争比，同时即使预测误差任意大，也能维持最坏情况保证。具体而言，算法会获得一个（可能有误的）输入预测，研究目标是设计具有良好性能保证的算法：预测准确时表现良好（一致性），预测错误时性能不至于太差且有界（鲁棒性），并且竞争比会随预测误差逐步恶化（平滑性）。我们将在第 2 节给出更精确的定义。

近期的研究提出了多种方法来解决带预测的 OLTSP（LA-OLTSP）。在 [23], 中，Gouleakis 等人研究了直线上的 OLTSP 学习增强框架。他们引入了一种预测模型，其中预测对应于请求的位置。他们针对闭合情形和未完结情形都提出了 LA 算法。

[1] 需要注意的是，这种可能速度的选择是无损一般性的，因为任何最大速度有界的设置都可以约化为这种设置。具体而言，如果最大速度为某个 $S > 0$，我们可以将最大速度归一化为 1，并将所有距离乘以 S。此外，我们的设置可以模拟任何其他算法可以自由选择单位速度或其他某个速度 $0 < s < 1$ 的设置。以 s 的速度移动，等价于将时间划分为长度为 $dt \to 0$ 的区间，并且在每个区间内仅以单位速度移动时间 $s \cdot dt$。

开放变体，且具备一致性、平滑性和鲁棒性。Bampis 等人 [13], 考虑了位置预测完美的情况，并研究了不同的度量空间（一般度量空间、半直线、环和星形），提出了具有竞争性的在线算法和下界。在 [25], 中，Hu 等人为 OLTSP 提出了三种不同的预测模型。在其中两种模型中，每个请求都关联着对其释放时间和位置的预测，而在第三种模型中，预测仅为最后一个请求的释放时间。在 [16], 中，Bernardini 等人研究了对每个请求的释放时间和位置均有预测的 OLTSP。他们引入了一种新的误差度量——覆盖误差，并且还研究了其他在线图问题。最近，Chawla 和 Christou [19]研究了具有请求释放时间和位置预测的在线时间窗旅行商问题。

在本文中，我们采用 [23]的预测模型。我们提出了一个通用的、基于预言机的框架，使我们能够针对不同度量空间中的问题闭合和开放变体，设计具有一致性、平滑性和鲁棒性的学习增强型算法。我们还给出了一些有趣的下界。

1.1 我们的贡献与技术

本文提出了一种新方法，利用有关一般度量空间中请求位置的预测来改善 OLTSP 的竞争比。我们的算法在大多数情形下都能给出紧确的竞争比保证。此外，我们还展示了如何在特定度量空间（即环、树和花）中获得多项式时间算法或固定参数可处理算法。

Bampis 等人 [13] 针对一般度量给出了一种算法，在预测完美（位置已知）的情况下，其竞争比为 $3/2$，且该界是紧确的。首先，在第 3 节中，我们修改了这一算法（仍假设预测完美），并提出了基于预言机的主要框架，该框架的竞争比为 3/2，我们称之为 *Strategically Wait And Go*（SWAG，伪代码见算法 1）。其主要思想是考虑请求排列的一个适当子集，称为支配排列，该子集由所谓的支配预言机 给出，而非考虑所有排列。由于瓶颈在于所考虑排列集合的基数，这样便能缩短运行时间。对排列集合的这一限制保留了 $3/2$ 的一致性。

接下来，我们介绍主要算法 *Learning-Augmented Strategically Wait AndGo*（LA-SWAG，伪代码见算法 2）。该算法不假设预测完美。LA-SWAG 具有一致性、平滑性和鲁棒性。更正式地说，我们将在第 4 节证明以下结论。

▶**定理 1 (**一致性与平滑性)。 *LA-SWAG* 对于闭合和开放 *LA-OLTSP* 的竞争比均至多为 $3/2 + 5\eta$。

这里， η 是预测误差（稍后将给出其形式化定义），用于刻画请求的预测位置与实际位置之间的距离归一化和。注意，对于 $\eta = 0$，我们得到一致性为 $3/2$；除树上的开放变体外，该结果在所有情形下都是紧的。

此外，我们证明了开放变体在$\eta \in [0, 1/3]$ 时的平滑性下界为 $3/2 + \eta/2$ （命题 14），这意味着 η 的线性依赖是必要的。

关于鲁棒性， [23] 中的算法在线上可达到 3-鲁棒性。 LA-SWAG改进了这一界限，在特定度量下进一步提升。

▶**定理 2 (**闭式鲁棒性)。 *LA-SWAG* 在一般度量空间中的闭合 *LA-OLTSP* 上具有2.75-鲁棒性，在欧几里得空间和树中则具有2.5-鲁棒性。

▶ **定理 3 (**开放式鲁棒性)。 *LA-SWAG* 在一般度量空间中的开放 *LA-OLTSP* 上具有$(3 - 1/6)$-鲁棒性，在树中具有$(3 - 1/3)$-鲁棒性。

我们关于 2.5-鲁棒性和（$3-1/3$）-鲁棒性的分析即使在线上也是紧的（注 18 和 21）。此外，我们还针对开放变体中任意算法的一致性/鲁棒性权衡给出了一个负面结果（引理 22）。

我们工作的主要技术贡献见第 5 节和第 6 节，在于实现了我们此前提到的支配预言机。从较高层面来看，如果满足以下条件，我们称排列 π_{dom} 在时间 t支配另一个排列 π：假设 q 是在时间 t下 π 中的第一个未释放请求，则在 π_{dom}中行进至 q所经过的距离不更长，并且 π 中位于 q 之前的请求的某个超集会在 q之前被访问。此外， π_{dom} 所诱导的路径不长于 π。这两个关键事实使我们能够保持 $3/2$-一致性。

对于一般度量，我们采用一种与经典旅行商问题 [15, 24]的 $O(n^2 2^n)$ 动态规划解的定义中所使用的方法非常相似的思想来实现这种支配。也就是说，对于任意可能在 q之前已由 π 服务的已释放请求子集，我们分别为 q之前和之后的部分构造两条最优路径（不考虑释放时间），然后将它们连接起来，得到一个支配排列。我们将所得的集合称为一般支配集。任何排列都受到与截至 q所服务请求的正确猜测相对应的排列的支配。由此，整体上得到一个单指数时间算法。

▶ **定理 4**（一般度量）。 *LA-SWAG* 借助一个使用一般支配集的预言机 D，可在单指数时间内运行；对于 *LA-OLTSP* 的闭合变体，其竞争比为$\min\{3/2+5\eta, 2.75\}$；对于开放变体，其竞争比为$\min\{3/2+5\eta, 3-1/6\}$。

我们在特定度量下缩短运行时间的核心思路是：实际上，我们不必尝试所有可能的请求子集来确定在 q 之前服务哪些请求。例如，对于树，我们首先证明一个关于最优解（带释放时间）的结构性结果。具体而言，我们证明，可以假设从叶节点到原点（视为根）的路径上各请求的服务顺序非常特殊：沿着该路径从叶节点走向原点时，依次遇到请求的顺序。基于这一事实，我们可以设计一个支配预言机，它大致能为在 q 之前访问的任意叶节点子集提供一个排列。因此，我们得到一种固定参数可处理算法，其参数为树的叶节点数量 l。

▶**定理 5 (树)**。 存在适用于树的 *LA-SWAG* 支配预言机，其时间复杂度为：闭合变体下为 $O(2^l \cdot n^3)$，开放变体下为 $O(2^l \cdot n^4)$，其中l 是输入树的叶节点数。

作为研究更一般图的一步，我们通过考察环来处理环这一概念。虽然我们无法得到关于在线最优解的完全相同的结构性结果，但我们证明了一个非常相似的结论。具体而言，我们证明，最优解至多只能利用环的循环特性一次。经过这样一次循环遍历后，我们可以假设环被分成两半，从而得到一棵我们知道如何处理的树。

▶**定理 6 (环)**。 存在适用于环的 *LA-SWAG* 支配预言机，其时间复杂度为：闭合变体下为 $O(n^3)$，开放变体下为 $O(n^5)$。

最后，我们结合前面两组思路来处理花。花本质上由若干个环（花瓣）和一条半直线（主干）组成，它们都连接到原点。在这种情况下，每个单个环仍然可以假设只被环路遍历一次。结果表明，对于每个花瓣，我们至多需要考虑 6 种选项。

▶**定理 7** (花)。 存在一个适用于花的支配预言机 *LA-SWAG*，其在闭合变体下的时间复杂度为 $O(6^p \cdot n^3)$，在开放变体下的时间复杂度为 $O(6^p \cdot n^5)$，其中 p 是输入花的花瓣数。

我们的结果汇总于表1和表2中。在闭合变体中，即使在线段上，也存在 $3/2$ 的下界 [23]。在开放变体中，我们在附录（命题44）中证明，即使在线段上，也存在 ≈ 1.468 的下界；此外，即使在环上，也存在 $3/2$ 的下界 [13]。

表1 LA-SWAG的一致性、平滑性、鲁棒性和运行时间保证。

	平滑性（$\eta = 0$ 的一致性）	鲁棒性		运行时间	
	闭合/开放	闭合	Open	闭合	Open
Tree	$3/2 + 5\eta$	2.5	$3 - 1/3$	$O\left(2^l \cdot n^3\right)$	$O\left(2^l \cdot n^4\right)$
Ring		2.75	$3 - 1/6$	$O(n^3)$	$O(n^5)$
花		2.75	$3 - 1/6$	$O\left(6^p \cdot n^3\right)$	$O\left(6^p \cdot n^5\right)$
欧几里得		2.5	$3 - 1/6$	$O(n^2 \cdot 2^n)$	$O(n^2 \cdot 2^n)$
通用		2.75	$3 - 1/6$	$O\left(n^2 \cdot 2^n\right)$	$O\left(n^2 \cdot 2^n\right)$

表 2 LA-SWAG 与其他可处理算法的一致性。带 * 的上界由固定参数可处理算法给出，否则由多项式时间算法给出。紧界以粗体表示。

	一致性			
	闭合		Open	
	先前工作	**本文**	先前工作	**本文**
Line	**3/2** [23]	**3/2**	5/3 [23]	3/2
Star	**7/4+ϵ** [13]	**3/2***	2 [13]	3/2*
Tree	2 [10]	**3/2***	2 [13]	3/2*
Ring	5/3 [13]	**3/2**	2 [13]	**3/2**
花	2 [10]	**3/2***	2 [13]	**3/2***

1.2 相关工作

该问题的离线版本中，位置和释放时间均已提前知晓，已有研究在 [17, 37] 中探讨了闭合和开放变体。对于 OLTSP， [10] 在一般度量空间中分别提出了闭合变体的 2-竞争算法和开放变体的 2.5-竞争算法。具体来说，在线上，闭合和开放变体的下界分别为 1.64 [10] 和 2.04 [17] 。

除了在线旅行商问题之外，还有一些研究探索了学习增强设置。带有预测的在线缓存问题由 [32], 进行了研究，初步结果由 [3, 39, 41]得到了改进。采用学习增强方法，研究人员为滑雪租赁问题 [2, 22, 38, 40] 以及调度问题 [4, 11, 34, 36]开发了算法。此外，还有关于经典数据结构 [29]、布隆过滤器 [33], 、路由问题 [14, 21, 28], 、在线选择和匹配问题 [5, 20]以及更一般的在线原始-对偶算法框架 [12]的学习增强算法研究。该领域已有一篇综述 [35]以及一份更新后的论文列表 [31] 。

2 预备知识

在线旅行商问题（OLTSP）。

OLTSP 的输入由一个度量空间 M 组成，该空间包含一个特殊点$\mathcal{O}$ （原点），以及一个由 $Q = \{q_1, ..., q_n\}$ 表示的 n 个请求集合。每个请求 q_i 都是一个有序对（x_i, t_i），其中 x_i 是 M 中的一个点，$t_i \geq 0$ 是一个实数。我们使用 t 来量化时间。数字 t_i 表示请求 q_i 可以被服务的时刻之后的时间点（释放时间）。一台在时间 $t = 0$位于原点、能够以单位速度移动的服务器，必须在所有请求释放后为其提供服务，目标是最小化总完成时间（完工时间）。

我们考虑一大类连续度量空间 M，其对应的距离度量 $d(x, y)$ 定义为从 $x \in M$到 $y \in M$的最短路径，并且关于 M连续，如 [10]中所示。我们称这类空间为一般度量空间（或一般度量）。连续度量空间的释放时间可以是任意非负实数。

在本文余下部分中，我们用 $|\mathrm{ALG}|$表示在线算法 ALG 的总完成时间，用 $|\mathrm{OPT}|$表示最优（离线）解 OPT 的总完成时间。回顾一下，如果对于所有实例都有 $|\mathrm{ALG}| \leq \rho \cdot |\mathrm{OPT}|$，则称算法 ALG 是 ρ-竞争的。

学习增强算法。

为了衡量预测的质量，我们将定义一个预测误差 η。LA 算法有三个主要性质。我们在此采用 [23] 中的形式化定义。我们称一个算法为

- 如果它在 $\eta = 0$时是 α-竞争性 的，则称其为α-一致的；如果无论
- η如何，它始终是 β-竞争性 的，则称其为β-鲁棒的；对于连续函
- 数 $\gamma(\eta)$，如果它是 $\gamma(\eta)$-竞争性的，则称其为γ-平滑的。

一般而言，如果 c 是在没有预测的情况下可达到的最佳竞争比，那么理想情况下应有 $\alpha < c$、$\beta \leq k \cdot c$，其中 k 为某个常数；同时，函数 γ 应随着误差 η 从 α 增加到 β。

我们的预测设定。

请求集合为 $Q = \{q_1, \ldots, q_n\}$。正如我们所提到的，每个请求 q_i都有一个对应的释放时间 t_i和位置 x_i。我们有一组预测 $P = \{p_1, \ldots, p_n\}$，其中 p_i 预测了请求 q_i的位置 x_i。算法将这些预测以及请求数量 n 作为离线输入。x_i 和 t_i 的实际值只有在时间点 t_i才会变得已知。

预测的质量可能有所不同，且算法对此一无所知。我们可以通过定义度量 η 来评估质量。从本质上讲，η 衡量预测位置与实际位置之间的所有距离之和，并将其按服务所有请求的最短路径长度进行归一化。

▶**定义 8**（预测误差）。实例的预测误差定义为 $\eta = \sum_{i=1}^{n} d(x_i, p_i) / F$，其中 F 是服务所有请求的最短路径长度（在闭合情形下还需返回原点）。

注意，预测误差具有尺度不变性（即，如果将所有距离乘以一个常数因子，预测误差也不会改变），而 F 充当归一化因子。

3 基于预言机的框架：SWAG 算法

在本节中，我们为完美预测的情形定义一种基于预言机的算法 SWAG。预言机向算法提供一组请求的排列。我们证明，如果预言机满足某些条件，那么 SWAG 的竞争比为 3/2。

SWAG 实际上是对 [13] 中通用算法的（略微）修改。后一个算法的原理如下：

- 首先，在$\mathcal{O}$ 处等待，直到选定的时间 T。这个时间 T 取决于请求的位置及其释放时间。然后，选择一条服务请求的路线，使某个准则最小化；该准则涉及相应路线的长度，以及在时间 T已释放的路线部分所占的比例。接着沿着这条路线行进，并在尚未释放的请求处等待。

第一步中选定时间 T 的计算方法是：对所有 $n!$种请求排列计算一些数值。我们在此展示的改进是：无需考虑排列的全集，只需考虑满足某种性质的、经过精心挑选（理想情况下规模较小）的子集，就能获得相同的竞争比（即 3/2）。这个排列子集由预言机提供给算法。

因此，基于这一框架，要推导出高效的 3/2-竞争算法，只需构造一个高效的预言机。我们将在第6节举例说明，对于直线或环，可以设计一个多项式时间预言机，构造一个由排列组成的多项式规模子集，从而为这些度量空间得到一个多项式时间的 3/2-竞争算法。

更正式地说，我们考虑 SWAG，并采用以下记号。对于请求的一种给定顺序 σ（其中 $\sigma[i]$ 表示该顺序中的第 i 个请求），我们记：

- 用 ℓ_σ 表示与 σ 对应的路线长度（从 $\mathcal{O}$ 出发），即开放情形下的 $\ell_\sigma = d(\mathcal{O}\ \sigma[1]) + \sum_{j=1}^{n-1} d(\sigma[j], \sigma[j+1])$，闭合情形下的 $\ell_\sigma = d(O, \sigma[1]) + \sum_{j=1}^{n-1} d(\sigma[j], \sigma[j+1]) + d(\sigma[n], \mathcal{O})$；用 $\alpha_\sigma(t)$表示在时间 t 时，与 σ 对应的路线中完全释放前缀的最大长度占 ℓ_σ的比例。
- 更正式地说，如果所有 n 请求都已释放，那么对于所有 σ， $\alpha_\sigma(t) = 1$；否则，如果请求 $\sigma[1], \ldots, \sigma[k-1]$ 在 t时释放，但 $\sigma[k]$ 尚未释放，那么路线截至 $\sigma[k]$均已完全释放，并且

$$\alpha_\sigma(t) = \big(d(\mathcal{O}, \sigma[1]) + \sum_{j=1}^{k-1} d(\sigma[j], \sigma[j+1])\big)/\ell_\sigma \ .$$

请注意，这一定义要求 $\ell_\sigma > 0$。如果 $\ell_\sigma = 0$，即所有请求都位于 $\mathcal{O}$，则设定$\alpha_\sigma = 1$。

[13] 中适用于一般度量的算法会考虑请求的所有可能排列，以求得等待时间 T 以及一个合适的后续排列。借助 SWAG，我们在此基础上进一步构建。关键的是，在任意时间点 t，它只关注排列的一个子集 S，以确定T 以及后续要采用的排列。这些子集背后的核心思想如下。为了达到相同的竞争比，对于任意可能的排列 σ，只需保证集合 S 中包含一个排列 σ'，使其所对应的巡回路线 / 路径不长于 σ 所对应的路线，并且在时间 t 尚未释放的路线部分不长于 σ所对应的部分。我们称σ' 支配 σ，并将在下文正式定义这一概念。此外，由于已发布部分仅在有新请求释放时才会发生变化，因此只需在这种情况下更新该子集。

算法 1 Strategically Wait And Go (SWAG)

输入：离线：请求位置 x_1、……、x_n；在线：释放时间 t_1、……、t_n。参数：预言机 D，它会输出请求的一组排列。1 调用预言机 D，在 $t = 0$处获取初始排列集合 $S(0)$。设置 $S = S(0)$。2 while true do 3 在每个释放时间 t_i，请求一个新的排列集合 $S(t_i)$，并更新$S = S(t_i)$。对于每个 $\sigma \in S$，计算 ℓ_σ 和 $\alpha_\sigma(t_i)$。4 如果存在 $\exists \sigma_0 \in S$，使得 (1) $t \geq \ell_{\sigma_0}/2$ 且 (2) $\alpha_{\sigma_0}(t) \geq 1/2$，则设置 $T = t$ 并退出循环。5 end 6 在时间 T：计算一个顺序 σ_1，使其在所有顺序 $\sigma' \in S$ 中最小化 $(1 - \beta_{\sigma'})\ell_{\sigma'}$，其中 $\beta_{\sigma'} = \min\{\alpha_{\sigma'}(T)\ 1/2\}$。沿着与 σ_1对应的巡回路线/路径前进。按此顺序处理请求；如果某个请求尚未释放，则在该请求位置等待。

▶**定义 9** (支配排列)。 设 σ 为请求 n 的一个排列， t 为给定的时间。我们将 *Dom(σ, t)* 定义为在时间 t支配排列 σ的所有排列构成的集合。当且仅当满足以下条件时，排列 $\sigma' \in Dom(\sigma, t)$ 成立：

$$\ell_{\sigma'} \leq \ell_\sigma \ , \tag{1}$$

and

$$(1 - \alpha_{\sigma'}(t))\ell_{\sigma'} \leq (1 - \alpha_\sigma(t))\ell_\sigma \ . \tag{2}$$

我们还称 σ' 为 σ 在时间 t 的对应支配排列。

为便于阐述，我们假设排列的子集由预言机 $\mathcal{D}$ 提供给算法。我们将在第 5 节和第 6 节中讨论如何针对具体情况实现此类预言机。

我们证明，当 $\mathcal{D}$ 是支配预言机时， SWAG with an oracle $\mathcal{D}$ 是 $3/2$-一致的。形式化地，我们有如下结论。

▶ **定义 10** (支配预言机)。 若预言机 *D* 在时间 t 输出一个排列集合 $S(t)$，则称其为支配预言机，当且仅当
1. $S(t) \subseteq S(t')$ 对于每个 $t \leq t'$，并且2. 对于所有 t，存在一个排列 $\sigma' \in S(t)$ 使得 $\sigma' \in Dom(\sigma_{\mathrm{OPT}}, t)$。这里， σ_{OPT}是与最优解中请求服务顺序相对应的排列。

▶**引理 11**。 *SWAG* 在使用支配预言机时，对于具有完美预测的 *OLTSP* 闭合和开放变体，均为 $3/2$-一致的。

证明。 我们将 $S(t)$ 记为算法在时间 t时所考虑的排列集合（即 S 在 SWAG的描述中），并将 t_q 记为请求 q的释放时间。注意，我们偶尔省略对 t 的依赖关系时，意指 $t = T$。

首先，我们证明该算法总是会终止，因为使用支配预言机时， SWAG 中的 while 环路总会终止。当所有请求都已释放时，对于请求的每个排列 σ，都有 $\alpha_\sigma(t) = 1$。由方程（2）和支配预言机的条件 2 可知，存在 $\sigma' \in S(t)$，使得 $(1 - \alpha_{\sigma'}(t))\ell_{\sigma'} \leq 0$。然后

$\alpha_{\sigma'}(t) = 1$（当 $\ell_{\sigma'} = 0$ 时，我们还有 $\alpha_{\sigma'}(t) = 1$，根据 α 的定义）。因此，存在一个时间 t'，使得 $t' = \ell_{\sigma'}/2$，且环路最迟在 $t = t'$ 终止。

在时间 T，存在一个排列 $\sigma_0 \in S(T)$，满足 $T \geq \ell_{\sigma_0}/2$ 和 $\alpha_{\sigma_0}(T) \geq 1/2$。于是 $\beta_{\sigma_0} = 1/2$，且 $(1-\beta_{\sigma_0})\ell_{\sigma_0} = \ell_{\sigma_0}/2 \leq T$。根据 σ_1 的定义，我们有 $(1-\beta_{\sigma_1})\ell_{\sigma_1} \leq (1-\beta_{\sigma_0})\ell_{\sigma_0}$。因此得到

$$(1-\beta_{\sigma_1})\ell_{\sigma_1} \leq T \ . \tag{3}$$

现在考虑最优解 OPT 及其对应的排列 σ_{OPT}。考虑时间 t 时 OPT 的一个对应支配排列 $\sigma'(t) \in S(t)$（支配预言机的条件 2）。由于 σ' 始终表示一个支配 OPT 的排列，因此我们在记号中省略 OPT。由方程（2）可得

$$|\,\mathrm{OPT}\,| \geq T + (1-\alpha_{\sigma_{\mathrm{OPT}}}(T))\ell_{\sigma_{\mathrm{OPT}}} \geq T + (1-\alpha_{\sigma'(T)}(T))\ell_{\sigma'(T)} \ .$$

我们区分 $\alpha_{\sigma'(T)}(T)$ 的值的两种情况。

如果 $\alpha_{\sigma'(T)}(T) \leq 1/2$，那么 $\beta_{\sigma'(T)}(T) = \alpha_{\sigma'(T)}(T)$，并且

$$|\,\mathrm{OPT}\,| \geq T + (1-\alpha_{\sigma'(T)}(T))\ell_{\sigma'(T)} = T + (1-\beta_{\sigma'(T)}(T))\ell_{\sigma'(T)} \ .$$

由于 σ_1 是 $(1-\beta)\ell$ 在 T 处的极小化元，我们得到

$$|\,\mathrm{OPT}\,| \geq T + (1-\beta_{\sigma_1})\ell_{\sigma_1} \ .$$

否则，如果 $\alpha_{\sigma'(T)}(T) > 1/2$，那么 $\beta_{\sigma'(T)}(T) = 1/2$。我们研究两种子情况。

- **情况 1：** $T \neq t_q$ 对每个请求 q 都成立。由于算法仅在释放一个新请求时才更新集合 $S(t)$，OPT 的对应支配排列 $\sigma'(T)$ 在 $T-\epsilon'$、$\forall \epsilon' \in (0,\ \epsilon_1]$ 对于足够小的 $\epsilon_1 > 0$（我们可以选择 ϵ_1，使得在 $[T-\epsilon', T], \forall \epsilon' \in (0,\ \epsilon_1])$ 内没有新请求被释放）时相同。因此，我们有 $\sigma'(T) = \sigma'(T-\epsilon'),\ \forall \epsilon' \in (0,\ \epsilon_1]$。

  现在我们考察 OPT 在时间 T 时位于其路线上的位置。假设它（严格地）处于这条路线的后半段。那么，在 $T-\epsilon$ 时（对于足够小的 $\epsilon_1 \geq \epsilon > 0$），它已经处于路线的后半段。但此时 $T-\epsilon \geq \ell_{\sigma_{\mathrm{OPT}}}/2$（因为 OPT 已经走完了路线的一半）。因此，由方程（1）可得 $T-\epsilon \geq \ell_{\sigma_{\mathrm{OPT}}}/2 \geq \ell_{\sigma'(T-\epsilon)}/2$。此外，由于在 $[T-\epsilon, T]$ 内没有新请求释放，我们有 $\alpha_{\sigma'(T-\epsilon)}(T-\epsilon) = \alpha_{\sigma'(T-\epsilon)}(T) > 1/2$。那么，$T-\epsilon$ 将满足算法起始时间的两个条件，这与 T 的定义矛盾。因此，在 T 时，OPT 位于其路线的前半段。所以 $|\mathrm{OPT}| \geq T + \ell_{\sigma_{\mathrm{OPT}}}/2 \geq T + \ell_{\sigma'(T)}/2 = T + (1-\beta_{\sigma'(T)}(T))\ell_{\sigma'(T)}$。由于 σ_1 是在 T 时使 $(1-\beta)\ell$ 最小化的极小化元，我们得到 $|\mathrm{OPT}| \geq T + (1-\beta_{\sigma_1})\ell_{\sigma_1}$。

- **情况 2：** $T = t_q$ 对于某个请求 q。现在我们可以找到一个（足够小的）$\epsilon_1 > 0$，使得在 $[T-\epsilon_1, T]$ 中不会有新请求释放。然后，我们可以找到另一个（足够小的）ϵ_2，满足 $0 < \epsilon_2 \leq \epsilon_1$，使得 OPT 在 $[T-\epsilon_2, T)$ 时仅位于其路线的前半段，或在 $[T-\epsilon_2, T)$ 时仅位于其路线的后半段。令 $\epsilon \in (0,\ \epsilon_2]$。现在我们考察 OPT 在时间 $T-\epsilon$ 沿其路线所处的位置。

  - 如果 OPT 在 $T-\epsilon$ 处位于前半段，那么 $|\mathrm{OPT}| \geq T-\epsilon+\ell_{\sigma_{\mathrm{OPT}}}/2 \geq T-\epsilon+\ell_{\sigma'(T)}/2 = T-\epsilon+(1-\beta_{\sigma'(T)}(T))\ell_{\sigma'(T)}$。该不等式对 $\epsilon \in (0, \epsilon_2]$ 的任意取值都成立，因此取极限 $\epsilon \to 0$，得到 $|\mathrm{OPT}| \geq T + (1-\beta_{\sigma'(T)}(T))\ell_{\sigma'(T)}$。由于 σ_1 是在 T 处使 $(1-\beta)\ell$ 取极小值的极小化元，因此得到 $|\mathrm{OPT}| \geq T + (1-\beta_{\sigma_1})\ell_{\sigma_1}$。

- 如果 OPT 在 $T-\epsilon$ 处（严格地）位于其路线的后半段，那么考虑 OPT 在 $T-\epsilon$ 处对应的支配排列，该排列包含在 $S(T-\epsilon)$、$\sigma'(T-\epsilon)$ 中。
  假设 $\alpha_{\sigma'(T-\epsilon)}(T-\epsilon) > 1/2$。那么，由于在 $T-\epsilon$ 时 OPT（严格来说）处于后半段，我们有 $T-\epsilon \geq \ell_{\sigma_{\mathrm{OPT}}}/2$（因为 OPT 已经走过了路线的一半）。因此，由方程（1）可得 $T-\epsilon \geq \ell_{\sigma_{\mathrm{OPT}}}/2 \geq \ell_{\sigma'(T-\epsilon)}/2$。那么，$T-\epsilon$ 将满足算法开始时间的两个条件，这与 T 的定义矛盾。因此，如果在 $T-\epsilon$ 时 OPT（严格来说）处于其路线的后半段，则有 $\alpha_{\sigma'(T-\epsilon)}(T-\epsilon) \leq 1/2$。

  因此，$\alpha_{\sigma'(T-\epsilon)}(T-\epsilon) \leq 1/2$ 且 $\beta_{\sigma'(T-\epsilon)}(T-\epsilon) = \alpha_{\sigma'(T-\epsilon)}(T-\epsilon)$。由方程（1）可得

$$\begin{aligned} |\,\mathrm{OPT}\,| &\geq T-\epsilon+(1-\alpha_{\sigma_{\mathrm{OPT}}}(T-\epsilon))\ell_{\sigma_{\mathrm{OPT}}} \\ &\geq T-\epsilon+(1-\alpha_{\sigma'(T-\epsilon)}(T-\epsilon))\ell_{\sigma'(T-\epsilon)} \\ &= T-\epsilon+(1-\beta_{\sigma'(T-\epsilon)}(T-\epsilon))\ell_{\sigma'(T-\epsilon)}\ . \end{aligned}$$

  注意，在时间 T，集合 $S(T)$ 还包含排列 $\sigma'(T-\epsilon)$，该排列在 $T-\epsilon$ 处支配 σ_{OPT}（支配预言机的条件 1）。因此，对于每个 $\epsilon \in (0,\epsilon_2]$（对于固定排列，$\alpha$ 和 β 是时间的非递减函数），有 $\alpha_{\sigma'(T-\epsilon)}(T-\epsilon) \leq \alpha_{\sigma'(T-\epsilon)}(T)$，且 $\beta_{\sigma'(T-\epsilon)}(T-\epsilon) \leq \beta_{\sigma'(T-\epsilon)}(T)$。由此可得 $|\,\mathrm{OPT}\,| \geq T-\epsilon+(1-\beta_{\sigma'(T-\epsilon)}(T))\ell_{\sigma'(T-\epsilon)}$。这意味着 $|\mathrm{OPT}\,| \geq T-\epsilon+(1-\beta_{\sigma_1})\ell_{\sigma_1}$，因为 σ_1 是 $(1-\beta)\ell$ 在 T 处的极小化元。由于上述不等式对每个 $\epsilon \in (0,\epsilon_2]$ 都成立，因此对于 $\epsilon \to 0$，可得 $|\,\mathrm{OPT}\,| \geq T+(1-\beta_{\sigma_1})\ell_{\sigma_1}$。

因此，在所有情况下，我们都有：

$$|\,\mathrm{OPT}\,| \geq T+(1-\beta_{\sigma_1}(T))\ell_{\sigma_1}\ . \tag{4}$$

现在我们来看算法输出的解 ALG 的值 $|\mathrm{ALG}\,|$。我们区分两种情况：

- 如果 ALG 在 T 之后不等待，则有 $|\,\mathrm{ALG}\,| = T+\ell_{\sigma_1}$。根据定义，$\beta_{\sigma_1}(T) \leq 1/2$，因此 $|\,\mathrm{ALG}\,| \leq T+2(1-\beta_{\sigma_1}(T))\ell_{\sigma_1}$。将方程（3）和（4）分别乘以系数 $1/2$和 $3/2$ 后相加，得到 $T+2(1-\beta_{\sigma_1}(T))\ell_{\sigma_1} \leq 3|\,\mathrm{OPT}\,|/2$。因此，$|\,\mathrm{ALG}\,| \leq 3|\,\mathrm{OPT}\,|/2$。
- 否则，ALG 在 T 之后等待某个请求被释放。设 t^* 为 ALG 最后一次等待的时间。由于 σ_1 的比例 $\alpha_{\sigma_1}(T)$ 在 T 时（即 ALG 开始时）已完全释放，因此 ALG 在 t^*之后最多需要行进距离 $(1-\alpha_{\sigma_1}(T))\ell_{\sigma_1}$。因此

$$|\,\mathrm{ALG}\,| \leq t^*+(1-\alpha_{\sigma_1}(T))\ell_{\sigma_1} \leq t^*+(1-\beta_{\sigma_1}(T))\ell_{\sigma_1}\ , \tag{5}$$

  这里我们用到了这样一个事实：根据定义，$\beta_{\sigma_1}(T) \leq \alpha_{\sigma_1}(T)$。我们有 $t^* \leq |\,\mathrm{OPT}\,|$，因为一个请求在 t^*时释放。将等式 (3) 和 (4) 相加，得到 $2(1-\beta_{\sigma_1}(T))\ell_{\sigma_1} \leq |\,\mathrm{OPT}\,|$。将这两个不等式代入等式 (5)，得到 $|\,\mathrm{ALG}\,| \leq 3|\,\mathrm{OPT}\,|/2$。

◀

现在来考察 SWAG 的运行时间，并证明以下引理。

▶**引理 12.** 如果 N 是预言机 D 在每个时刻 t 输出的排列数的最大值，且 $T_{\mathcal{D}}$ 是（由预言机）计算所有排列所需的总时间，那么 *SWAG* 的运行时间为 $O(\max\{n^2 \cdot N, T_{\mathcal{D}}\})$。

证明。 对于每个释放时间 t_i 以及每个排列 $\sigma \in S(t_i)$，算法必须计算长度 ℓ_σ 以及相应巡回路线中已释放部分所占的比例，α_σ。它可以

通过 $O(n)$ 次操作完成这一任务，并且必须重复此过程 $n+1$ 次，每次有新的请求释放时都要执行一次（并且在 $t=0$时也要执行一次）。如果 N 是预言机输出的排列的最大数量，那么不计入预言机工作量时，算法的总运行时间为 $O(n^2 \cdot N)$。此外，假设对于预言机 $\mathcal{D}$ 的所有调用，$\mathcal{D}$ 计算所有排列所需的总计算时间为 $T_{\mathcal{D}}$，那么 SWAG 的总运行时间为 $O(\max\{n^2 \cdot N, T_{\mathcal{D}}\})$。◀

4 LA-OLTSP 的性能保证：LA-SWAG 算法

本节讨论不完美预测；更具体地说，我们将展示如何调整SWAG，从而也获得平滑性和鲁棒性的上界。我们首先介绍算法，然后进行一致性-平滑性分析，最后进行鲁棒性分析。

算法 2 学习增强型 Strategically Wait And Go（LA-SWAG） 输入：离线：预测的请求位置 p_1、……、p_n 在线：释放时间 t_1、……、t_n，真实请求位置 x_1、……、x_n 参数：一个预言机 D，它输出请求的排列集合 1（中断规则）在任意时间 t：如果所有请求都已释放，则沿着一条服务所有未服务请求的最优路径行进（在闭合变体中返回 O），然后中断。 2 运行 SWAG，直到服务器的起始时间 T，并完成 σ_1 的计算。 3 在时间 T，沿着巡回路线/路径 σ_1 行进，并按以下顺序服务请求： 4 对于 $i=1$、……、n，首先前往$p_{\sigma_1[i]}$；如果$q_{\sigma_1[i]}$ 尚未释放，则在此处等待，直到其释放；然后前往$x_{\sigma_1[i]}$ 并服务该请求。 5 结束 6 在闭合版本中，返回 O。

我们注意到，如果预测完全准确，那么 LA-SWAG 至少与 SWAG 一样好（它的工作方式相同，唯一的区别是，在所有请求都释放后，它会以最优方式服务剩余的未服务请求）。因此，特别是只要预言机满足引理 11 的条件，它也具有 3/2-一致性。

我们还注意到，该算法可以用更巧妙的方式服务请求：算法不必先前往请求的预测位置，再前往其真实位置；如果请求已经释放（或一经释放），算法可以直接前往其真实位置。不过，即使进行这一修改，我们给出的下界仍然成立。

关于 LA-SWAG的运行时间，显然，如果 T_{TSP} 是计算服务请求子集的最优路径所需的时间，那么根据引理 12，我们得到以下结论。

▶**推论13**。 如果 N 是预言机 D 在每个时间 t输出的排列的最大数量， $T_{\mathcal{D}}$ 是（由预言机）计算所有排列所需的总时间，而T_{TSP} 是计算服务请求子集的最优路径所需的时间，那么 *LA-SWAG* 的运行时间为 $O(\text{最大值}\{n^2 \cdot N, T_{\mathcal{D}}, T_{TSP}\})$。

请注意，对于我们考虑的情况，我们为 T_{TSP} 实现了适当的界。我们将在第6节中介绍如何针对每种度量空间实现这一点。

4.1 平滑性

我们现在证明，LA-SWAG 关于预测中误差度量 η 是平滑的。回顾一下，$\eta = \sum_{i=1}^{n} d(x_i, p_i)/F$ 表示服务所有请求的最短旅行商巡回路线的长度（在闭合情形下，还需返回原点），且不考虑释放时间。为方便起见，令 $\Delta = \eta F = \sum_{i=1}^{n} d(x_i, p_i)$。

▶**定理 1** (一致性与平滑性)。 *LA-SWAG 在闭合和开放 LA-OLTSP 中的竞争比至多为* $3/2 + 5\eta$。

证明。设 I 是该问题的一个实例，包含请求 q_1、……、q_n，预测位置 p_1、……、p_n 以及实际位置 x_1、……、x_n。记 I_H 为该问题的一个相应（假设）实例，其中预测位置 p_1、……、p_n 将成为实际位置（因此 I_H 是一个具有完美预测的实例）。记 ALG_{I_H}（分别为 OPT_{I_H}）为 SWAG 在 I_H 上输出的解（分别为 I_H 上的最优解）。根据引理 11，有 $|\mathrm{ALG}_{I_H}|_{I_H} \leq 3/2\,|\mathrm{OPT}_{I_H}|_{I_H}$，因为在 I_H 中预测是完美的。

现在，我们首先断言 $|\mathrm{ALG}|_I \leq |\mathrm{ALG}_{I_H}|_{I_H} + 2\Delta$。如前所述，LA-SWAG 中的中断规则不会增加解的值，因此 $|\mathrm{ALG}|_I \leq |\mathrm{ALG}'|_I$，其中 ALG' 是 I 上的解，它会在 LA-SWAG 的步骤 4 中执行整个 for 循环。注意，现在 ALG_{I_H} 和 ALG' 在同一时间 T 开始，并按相同顺序 σ_1 服务请求。令 t^* 为 ALG' 针对实际实例 I 等待预测 $p_{\sigma_1[s]}$ 的最后时刻；若 ALG' 从未等待预测，则令 $t^* = T$。在第一种情况下，ALG_{I_H} 无法在 t^* 之前服务相应的请求 $\sigma_1[s]$（因为该请求尚未释放）。在第二种情况下，ALG_{I_H} 在 t^* 之前不会离开原点。因此，为方便起见，记 $s = 0$、$\sigma_1[0] = 0$ 和 $p_0 = \mathcal{O}$（若 ALG' 从未等待预测），则对于闭合情形有（开放情形只需去掉项 $d(p_n, \mathcal{O})$）：

$$|\,\mathrm{ALG}_{I_H}\,|_{I_H} \geq t^* + \sum_{i=s}^{n-1} d(p_{\sigma_1[i]}, p_{\sigma_1[i+1]}) + d(p_{\sigma_1[n]}, \mathcal{O}) \ .$$

由于假设的算法 ALG' 在预测和请求之间移动，我们得到关于 $|\mathrm{ALG}'|_I$ 的以下界

$$|\,\mathrm{ALG}'\,|_I \leq t^* + \left(\sum_{i=s}^{n-1} d(p_{\sigma_1[i]}, x_{\sigma_1[i]}) + d(x_{\sigma_1[i]}, p_{\sigma_1[i+1]}) \right) + d(p_{\sigma_1[n]}, x_{\sigma_1[n]}) + d(x_{\sigma_1[n]}, \mathcal{O}) \ .$$

利用三角不等式 $d(x_{\sigma_1[i]}, p_{\sigma_1[i+1]}) \leq d(x_{\sigma_1[i]}, p_{\sigma_1[i]}) + d(p_{\sigma_1[i]}, p_{\sigma_1[i+1]})$，我们得到

$$\begin{aligned} |\,\mathrm{ALG}'\,|_I &\leq t^* + \left(\sum_{i=s}^{n-1} d(p_{\sigma_1[i]}, p_{\sigma_1[i+1]}) + 2d(x_{\sigma_1[i]}, p_{\sigma_1[i]}) \right) + 2d(p_{\sigma_1[n]}, x_{\sigma_1[n]}) + d(p_{\sigma_1[n]}, \mathcal{O}) \\ &\leq |\,\mathrm{ALG}_{I_H}\,|_{I_H} + 2\Delta \ . \end{aligned}$$

现在，我们将注意力转向 $|\mathrm{OPT}_I|_I$ 和 $|\mathrm{OPT}_{I_H}|_{I_H}$。我们的目标是证明 $|\mathrm{OPT}_I|_I \geq |\mathrm{OPT}_{I_H}|_{I_H} - 2\Delta$。由于 $|\mathrm{OPT}_{I_H}|_{I_H} \leq |\mathrm{OPT}_I|_{I_H}$，只需证明 $|\mathrm{OPT}_I|_I \geq |\mathrm{OPT}_I|_{I_H} - 2\Delta$。我们可以假设 OPT 按照某个顺序 σ 积极地前往请求，并在请求释放后立即离开。与之前类似，令 t^* 为 OPT_I 在实例 I_H 中的巡回路线导致等待的最后时间，并令 $q_{\sigma[s]}$ 为该请求（或者对于 $s = 0$，为原点，这也意味着 $t^* = 0$）。由此可知，

$$|\,\mathrm{OPT}_I\,|_{I_H} = t^* + \sum_{i=s}^{n-1} d(p_{\sigma[i]}, p_{\sigma[i+1]}) + d(p_{\sigma[n]}, \mathcal{O}) \ .$$

但是，由于请求 $q_{\sigma[s]}$ 在 t^* 之前尚未释放（情况 $s=0$ 意味着 $t^*=0$，因此结论仍然成立），我们也有

$$\begin{aligned}|\operatorname{OPT}_I|_I &\geq t^* + \sum_{i=s}^{n-1} d(x_{\sigma[i]}, x_{\sigma[i+1]}) + d(x_{\sigma[n]}, \mathcal{O}) \\ &\geq t^* + \left(\sum_{i=s}^{n-1} -d(x_{\sigma[i]}, p_{\sigma[i]}) + d(p_{\sigma[i]}, p_{\sigma[i+1]}) - d(p_{\sigma[i+1]}, x_{\sigma[i+1]}) \right) \\ &\quad - d(x_{\sigma[n]}, p_{\sigma[n]}) + d(p_{\sigma[n]}, \mathcal{O}) \\ &\geq |\operatorname{OPT}_I|_{I_H} - 2\Delta \ ,\end{aligned}$$

其中，第二个不等式用到了三角不等式。由此可得一般算法的平滑性。我们有 $|\operatorname{ALG}|_{I_H} \leq 3/2|\operatorname{OPT}_{I_H}|_{I_H}$，这意味着 $|\operatorname{ALG}|_I - 2\Delta \leq 3/2(|\operatorname{OPT}_I|_I + 2\Delta)$。由此可得 $\frac{|\operatorname{ALG}|_I}{|\operatorname{OPT}_I|_I} \leq 3/2 + 5\eta$. ◀

LA-SWAG 其平滑性与误差呈线性关系（此外，一致性界已达到最优值 3/2）。现在我们证明，对于开放情形，这种线性依赖是必需的。

▶**命题 14** (平滑性下界)。对于预测误差为 $\eta \in [0, 1/3]$ 的实例，任何算法在开放变体上的竞争比都不可能优于 $(3/2 + \eta/2)$。

证明。令 $\eta \in [0, 1/3]$，并定义 ϵ 使得 $\eta = \epsilon/(2-\epsilon)$ (即 $\epsilon = 2\eta/(1+\eta)$)。注意到 $\epsilon \in [0, 1/2]$。考虑图1所示的图。

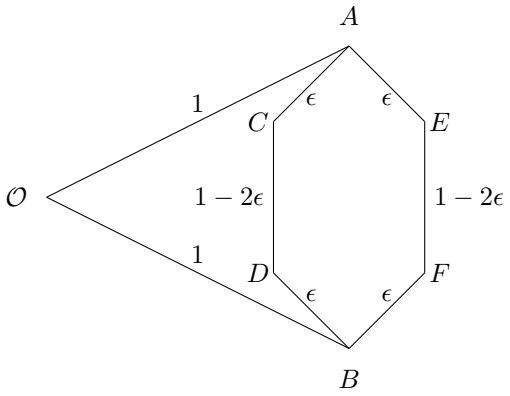


图1 开放变体的一个示例：该度量空间中，竞争比必须随 η 线性增长，直至后者达到 $1/3$。

有两个请求，q_1 预测位于位置 $p_1 = A$，另一个 q_2 预测位于$p_2 = B$。不失一般性，假设在时间 $t=1$，算法位于 $[O, B]$。随后，q_1 于 $t=1$ 在其预测位置（$x_1 = p_1 = A$）释放。注意，在时间 $2-\epsilon$，算法与 A 的距离至少为 ϵ（特别是，它不能位于（C，A]）或（E，A]）上。如果它位于 $[B, E]$，我们就在 D 上释放第二个请求 q_2。否则，我们就在 F 上释放该请求。不失一般性，假设出现第一种情况。那么，算法至少需要一个时间单位才能服务这两个请求（在 $t = 2-\epsilon$，它不在（B，C）上，且与A的距离至少为 ϵ）。因此 $|\mathrm{ALG}| \geq 3 - \epsilon$。

最优解是在 $t=1$ 服务 q_1，并在 $t = 2-\epsilon$. 服务 q_2。注意，$\eta = \frac{\epsilon}{2-\epsilon}$ 确实是预测中的误差。然后

$$|\operatorname{ALG}| \geq \frac{3-\epsilon}{2-\epsilon}|\operatorname{OPT}| = \left(\frac{3}{2} + \frac{\eta}{2}\right)|\operatorname{OPT}| \ .$$

◀

4.2 鲁棒性

现在我们来分析鲁棒性。首先给出 LA-SWAG 鲁棒性的上界，然后通过一些下界来补充这一分析。

▶ 备注 15（3-鲁棒性）。容易看出， LA-SWAG 具有 3-鲁棒性（在闭合和开放变体中均如此）。确实，设 t_f 为最终请求释放的时间。当然，$|\mathrm{OPT}| \geq t_f$。那么，在 t_f时，服务器距离原点至多为 $|\mathrm{OPT}|$。由于有中断规则，算法在$t = t_f$之后会以最优方式服务剩余请求，因此，先返回原点再遵循 OPT 是其成本的一个上界。所以， $|\mathrm{ALG}| \leq t_f + |\mathrm{OPT}| + |\mathrm{OPT}| \leq 3|\mathrm{OPT}|$，带预测的算法的鲁棒性至多为 3。

现在，我们进一步改进这一界限，涵盖闭合和开放情形，以及一般度量和特定度量空间。

我们先处理闭合变体。

闭合变体。

▶**引理 16。** *LA-SWAG 在一般度量空间中的闭合在线旅行商问题中具有2.75-鲁棒性。*

证明。 令 $T \geq 0$ 为 LA-SWAG第 2 步中计算出的时间。它取决于请求的预测位置和（实际）释放时间。令 $c \in (0, 1]$ 为稍后确定的实数常数。我们用 ρ表示比值 $|\mathrm{ALG}|/|\mathrm{OPT}|$。

- 如果 $T > |\mathrm{OPT}|$，那么由于中断规则，ALG 会在时间$t = t_f \leq |\mathrm{OPT}|$从原点出发，沿着一条最优巡回路线行进。因此， $|\mathrm{ALG}| \leq 2|\mathrm{OPT}|$并且

$$\rho \leq 2 \ . \tag{6}$$

- 如果 $0 \leq T \leq c\cdot|\mathrm{OPT}|$，那么存在一种请求顺序 σ_0，使得该通用算法会持续跟踪此顺序，并且 $T \geq \ell'_{\sigma_0}/2$, $\alpha'_{\sigma_0}(T) \geq 1/2$ 以及 $\beta'_{\sigma_0}(T) = 1/2$，其中带撇号的参数 ℓ', α', β'是根据预测位置和实际释放时间计算的。

  令 σ_1 表示我们的算法在 T 时将遵循的巡回路线。因此，我们有：

$$\big(1 - \beta'_{\sigma_1}(T)\big)\ell'_{\sigma_1} \leq \big(1 - \beta'_{\sigma_1}(T)\big)\ell'_{\sigma_1} = \ell'_{\sigma_1}/2 \leq T \ .$$

  由于 $\beta' \leq 1/2$，有 $1 - \beta'_{\sigma_1}(T) \geq 1/2$且 $\ell'_{\sigma_1} \leq 2T$。因此，在假设$T \leq c\cdot|\mathrm{OPT}|$的情况下，我们得到做爱片

$$\ell'_{\sigma_1} \leq 2c\cdot|\mathrm{OPT}| \ . \tag{7}$$

  在 T 之后，算法按照请求 σ_1的顺序执行。它首先前往某个请求的预测位置，并等待该请求释放。然后，它前往该请求的实际位置并为其提供服务。所有请求释放后，算法会计算并沿着一条服务剩余请求并返回原点的最优路径行进。

  预测请求距离原点的最远距离为 $L_P \leq \ell'_{\sigma_1}/2$。根据 (7)，我们得到 $L_P \leq c\cdot|\mathrm{OPT}|$。另一方面，实际请求距离原点的最远距离为 $L_R \leq \ell_{\mathrm{OPT}}/2 \leq |\mathrm{OPT}|/2$，其中 ℓ_{OPT}是最优巡回路线的长度。

  现在让我们考虑 ALG 在时间$t = t_f$距离原点最远能有多远。在巡回路线中，ALG 会在请求的预测位置和实际位置之间移动，直到

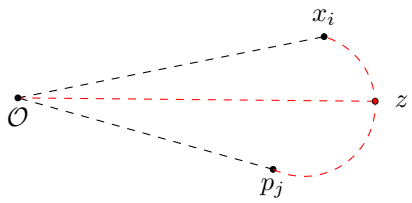


图 2 当服务器从距离最远的预测位置p_j 前往距离最远的（实际）请求 q_i 所在位置 x_i，或反向移动时，一般度量的情况。算法（z）始终位于 x_i 和 p_j之间的最短路径上。

时间 t_f。下文中，我们将 ALG 在 t_f 时的位置记为 z。最坏情况发生在 ALG 从预测请求中最远的请求前往实际请求中最远的请求，或反之（见图 2）时。根据三角不等式，我们有 $d(z, \mathrm{O}) \leq d(\mathrm{O}\ x_i) + d(x_i, z)$，以及 $d(z, \mathrm{O}) \leq d(\mathrm{O}\ p_j) + d(p_j, z)$。因此，我们有 $2d(z, \mathrm{O}) \leq d(\mathrm{O}\ x_i) + d(\mathrm{O}\ p_j) + d(x_i, z) + d(p_j, z)$。由于算法位于 x_i 和 p_j之间的最短路径上，我们得到 $d(x_i, p_j) = d(x_i, z) + d(p_j, z)$。再次利用三角不等式可得$d(x_i, p_j) = d(x_i, z) + d(p_j, z) \leq d(\mathrm{O}\ x_i) + d(\mathrm{O}\ p_j)$。因此，

$$2d(z, \mathcal{O}) \leq d(\mathcal{O}, x_i) + d(\mathcal{O}, p_j) + d(x_i, z) + d(p_j, z) \leq 2(d(\mathcal{O}, x_i) + d(\mathcal{O}, p_j)) \ ,$$

这意味着

$$d(z, \mathcal{O}) \leq d(\mathcal{O}, x_i) + d(\mathcal{O}, p_j) \leq L_R + L_P \leq (1/2 + c) \cdot |\,\mathrm{OPT}\,| \ . \tag{8}$$

在时间 $t = t_f$ 返回原点然后跟随 OPT 的策略显然是 $|\,\mathrm{ALG}\,|$的一个上界，因此由 (8) 以及 $t_f \leq |\mathrm{OPT}\,|$ 这一事实可得，$|\mathrm{ALG}\,| \leq t_f + d(z, \mathcal{O}) + |\,\mathrm{OPT}\,| \leq (2.5 + c) \cdot |\,\mathrm{OPT}\,|$ 且

$$\rho \leq 2.5 + c \ . \tag{9}$$

否则，$c \cdot |\,\mathrm{OPT}\,| < T \leq |\,\mathrm{OPT}\,|$. 那么，在 $t = |\,\mathrm{OPT}\,|$时，ALG 距离原点至多为 $(|\,\mathrm{OPT}\,| - T)$。因此，以下策略显然是 $|\,\mathrm{ALG}\,|$的上界：一种在 $t = |\,\mathrm{OPT}\,| + |\,\mathrm{OPT}\,| - T$ 时位于原点，然后跟随 OPT 服务所有请求的算法。由此，我们得到

$$\rho \leq \frac{|\,\mathrm{OPT}\,| + |\,\mathrm{OPT}\,| - T + |\,\mathrm{OPT}\,|}{|\,\mathrm{OPT}\,|} < \frac{(3 - c)|\,\mathrm{OPT}\,|}{|\,\mathrm{OPT}\,|} = 3 - c \ . \tag{10}$$

可以看出，导致式 (6) 的情况不会影响鲁棒性的确定，因此可以忽略。我们同时优化式 (9) 和式 (10)。显然，当 $c = 0.25$ 时可取到最优值，对应的鲁棒性为 2。75。 ◀

此外，我们注意到，对于欧几里得空间和树这两类特殊度量空间，几乎沿用相同的证明就能得到更好的鲁棒性保证。具体而言，我们得到以下引理。

▶**引理 17**。 *LA-SWAG* 对于欧几里得空间和树中的闭合在线旅行商问题是2.5-鲁棒的。

证明。 证明过程与引理 16 的证明几乎相同。唯一的区别在于，由于成立的是 $d(z, \mathcal{O}) \leq$ 最大值$\{d(\mathcal{O}\ x_i)\ d(\mathcal{O}\ p_j)\} \leq$最大值$\{c \cdot |\,\mathrm{OPT}\,|, \ 1/2 \cdot |\,\mathrm{OPT}\,|\}$ 在这些

度量空间中（因为算法位于 x_i 和 p_j 之间的一条最短路径上），我们实际上可以得到比不等式（9）更好的上界，即

$$\rho \leq \frac{t_f + \max\{d(\mathcal{O}, x_i), d(\mathcal{O}, p_j)\} + |\,\mathrm{OPT}\,|}{|\,\mathrm{OPT}\,|} \leq 2 + \max\{1/2, c\} \ . \tag{11}$$

我们可以再次忽略（6）。结合（10）和（11），我们得到鲁棒性 2.5，其中 $c = 0.5$。 ◀

由引理 16 和 17，我们可以直接得到以下关于LA-SWAG 在闭合变体中的鲁棒性的定理。

▶**定理 2** (闭式鲁棒性)。 *LA-SWAG* 在一般度量空间中对闭合 *LA-OLTSP* 具有2.75-鲁棒性，在欧几里得空间和树中具有2.5-鲁棒性。

▶ 注释 18（闭合变体 2.5-鲁棒性分析的紧致性）。 LA-SWAG 引理 17 中给出的 2.5-鲁棒性界限即使在线上也是紧的，如下例所示。共有两个请求。真实请求位置为 $x_1 = 1$， $x_2 = 0$，分别在时间 1 和 2 释放，预测为 $p_1 = 0$， $p_2 = -1$。注意，在时间 1，我们的算法将开始向预测 p_2移动，因为巡回路线的一半$\mathcal{O} \to p_2 \to p_1 \to \mathcal{O}$ 已“释放”，并且已经过去了足够的时间（其长度的一半）。[2] 它将继续移动，并在时间 2 到达位置 -1 ，因为 p_2对应的请求直到该时间才会释放。然后，在到达这个不利位置后，ALG 只能返回原点并复制 OPT。我们看到， $|\mathrm{ALG}\,| = 2 + |-1| + 2 = 5$ 而 $\mathrm{OPT}\, = 2$，因此该算法的竞争比为 2.5。

开放变体。

事实上，即使对于开放变体（只假设满足三角不等式），我们也可以证明一个优于 3 的鲁棒性界。诀窍是考虑另一种收尾方式：前往 OPT 最后服务的请求，记作 q_ℓ，然后反向复制 OPT。显然，这种复制所需时间也至多为 $|\mathrm{OPT}\,|$ 时间，因为原点处可能没有请求。因此，剩下要证明的是，我们可以相对快速地到达 q_ℓ 或 原点。

▶ **引理 19**。 *LA-SWAG* 对于一般度量空间中的开放式在线旅行商问题，具有 $\left(3 - \frac{1}{6}\right)$-鲁棒性。

证明。注意，在最终释放时间 t_f， LA-SWAG 位于从某个预测到某个请求的位置或反向（或者从原点到某个预测的位置，这种情况显然成立，也可以视为一个请求位置）的某条最短路径上。现在，设 p 为该预测， q 为实际位置为 x 的请求。我们不区分移动是从 p 到 x 还是反向进行的。考虑路径 P_1，它从 z （时间 t_f 时服务器所在的位置）移动到 p ，然后到达 O；以及路径 P_2 ，它从 z 移动到 x ，然后到达 x_ℓ，即 OPT 服务的最后一个请求的位置。现在，令 $d_p = d$（z， p）和 $d_x = d$（z， x）。由于 ALG 位于从 p 到 x 或反向的最短路径上，因此有 $d_p + d_x = d$（p， x）。注意，我们已经有了

$$\begin{aligned} min\{|P_1|, |P_2|\} &\leq \frac{|P_1| + |P_2|}{2} = \frac{d_p + d_x + d(p, \mathcal{O}) + d(x, x_\ell)}{2} \\ &\leq d(p, \mathcal{O}) + \frac{d(x, x_\ell) + d(x, \mathcal{O})}{2} \leq \frac{|\,\mathrm{OPT}\,|}{2} + d(p, \mathcal{O}) \ . \end{aligned}$$

[2] 事实上，这是我们稍后定义的预言机将提供给 LA-SWAG 的唯一排列。为确保这一点，我们可以将 p_1 向原点左侧扰动任意小的 ϵ ，然后令 $\epsilon \to 0$。因此，它肯定会选择遵循这一排列。

这意味着我们最多可以在 $(2.5+\gamma)|\mathrm{OPT}|$ 内完成整体任务，其中 $\gamma = {}^{d_{(p,\mathcal{O})}}|\mathrm{OPT}|$。现在，我们可以应用闭合情形证明中的相同思路。也就是说，令 $T = c\cdot|\mathrm{OPT}|$，其中 T 是算法的开始时间。我们看到，$d(p,\mathcal{O}) \le 2c\cdot|\mathrm{OPT}|$，否则算法启动所需的条件在 T 时将无法满足。此外，性能的一个平凡上界为 $(3-c)|\mathrm{OPT}|$。因此，一方面我们有 $\frac{|\mathrm{ALG}|}{|\mathrm{OPT}|} \le 2.5+2c$，另一方面有 $\frac{|\mathrm{ALG}|}{|\mathrm{OPT}|} \le 3-c$，并且这些界同时成立。因此，通过设定 $c=\frac{1}{6}$，整体上我们有 $\frac{|\mathrm{ALG}|}{|\mathrm{OPT}|} \le 3-\frac{1}{6}$。◀

此外，在处理树时，我们还能改进这个界。

▶ **引理 20.** *LA-SWAG* 对于树上的开放式在线旅行商问题具有 $(3-1/3)$-鲁棒性。

证明。考虑一个预测 p 以及一个请求 q 的位置 x。为简单起见，假设 p 和 x 是树中的顶点，否则我们可以直接在这些位置添加顶点。注意，在这种情况下，当从 p 移动到 x 或反向移动时，ALG 会先移动到 p 和 x 的最近公共祖先，然后沿着路径继续移动到目的地。这意味着，在这段移动过程中的任意时刻，它都位于 x 或 p 通往原点的路径之一上。特别地，在 $t=t_f$ 时也是如此。如果它位于通往 p 的路径上，那么它到原点的距离至多是 p 到原点的距离。如果它位于通往 x 的路径上，则意味着它也位于 OPT 将采取的路径上的某个点。因此，它可以先移动到该路径的任一端点，然后复制该路径来完成目标；根据前一种情况的区分，这总共至多需要 $(1+2c)|\mathrm{OPT}|$ 或 $3/2|\mathrm{OPT}|$ 时间，其中 c 是等待 $|\mathrm{OPT}|$ 的比例。因此，我们得到界 2.5、$2+2c$、$3-c$，并通过考虑 $c=1/3$ 得到所需的竞争比。◀

前两个引理给出了我们针对开放变体的主要鲁棒性定理。

▶ **定理 3** (开放式鲁棒性)。 *LA-SWAG* 对于一般度量空间中的开放 *LA-OLTSP* 是 $(3-1/6)$-鲁棒的，而在树中是 $(3-1/3)$-鲁棒的。

▶ 注释 21（开放变体 $\left(3-\frac{1}{3}\right)$-鲁棒性分析的紧确性）。引理 20 中给出的 $\left(3-\frac{1}{3}\right)$-鲁棒性界 LA-SWAG 是紧确的，即使在直线上也是如此。为此，我们在实数直线上将预测放置在 -1 处，并将实际请求放置在 1.5 处，该请求于 $t=1.5$ 时释放。注意，在时间 $t=0.5$，我们的算法开始沿路径前往预测位置。它于 $t=1.5$ 时到达 -1。这意味着它无法在 $t=4$ 之前完成，因为它必须从 -1 出发，于 $t=1.5$ 时开始移动，前往 1.5。然而，最优时间为 1.5，因此竞争比为 $8/3=3-1/3$。

开放变体的一致性-鲁棒性权衡下界。

我们通过证明开放变体的一致性-鲁棒性权衡下界来结束本节。

▶**命题 22.** 设 ALG 是一个用于开放变体的算法，其一致性保证为 $2-\lambda$，其中 $\lambda\in[0,1]$。那么，对于任意 $\epsilon>0$，ALG 都不可能具有 $(2+\lambda-\epsilon)$-鲁棒性。

证明。我们在线上证明这一点。我们在 -1 处放置一个预测，并将实际请求放在 1 处，该请求将在 $t=1$ 时释放。由于 ALG 是 $(2-\lambda)$-一致的，因此它必须位于不严格在 $-\lambda$ 右侧的位置，且时间为 $t=1$；否则，它将无法在 $2-\lambda$ 时到达预测点，这与 ALG 的一致性相矛盾，因为如果预测正确，它会在 $t=1$ 时释放。因此，ALG 只能在 $t=2+\lambda$ 时或更晚到达实际请求。由于 $|\mathrm{OPT}|=1$，证毕。◀

请注意，上述结果意味着，对于任何 1.5-一致算法而言，最优鲁棒性为 2.5，这使我们的算法相对接近底层的帕累托前沿。

5 适用于一般度量空间的单指数时间算法

在预测完美的情况下，利用 SWAG 和引理 11，我们可以降低 [13]中针对一般度量的算法的阶乘级运行时间。我们将证明，只需指数数量的排列即可构造支配预言机。之后，利用 LA-SWAG的方案以及上一节证明的定理，我们将得到一个指数时间算法，该算法具有 $3/2$-一致性、平滑性和鲁棒性。

在每个时间 t，都有一个已释放请求集合 $R(t)$ 和一个未释放请求集合 $U(t)$。为便于说明，这里我们讨论闭合情形，但开放情形与之类似。考虑时间 t时 n 请求的任意排列 σ。可以按如下方式构造与该排列对应的支配巡回路线。首先，找出排列 σ中的第一个未释放请求$u \in U(t)$。排列中位于 u 之前的已释放请求构成已释放请求的一个子集 $R'(t) \subseteq R(t)$。于是，相应的一般支配巡回路线将是：从原点出发、服务 $R'(t)$ 中的所有请求并在请求 u 处结束的最优 TSP 路径，加上从 u 出发、服务 $\big(R(t) \setminus R'(t)\big) \cup \big(U(t) \setminus \{u\}\big)$ 中的所有请求并返回原点的最优 TSP 路径。我们将这些巡回路线称为（对应的）一般支配巡回路线。

在本节中，为便于阐述，我们假设 $t_1 \leq t_2 \leq \cdots \leq t_n$，并使用 $t_0 = 0$ 和 $t_{n+1} = \infty$。在时间 t_i，给定 $i \leq n$，我们根据集合 $R(t_i)$ 和 $U(t_i)$ 计算一般支配巡回路线集合 D_i。随后，将该集合与先前计算得到的集合一起，作为集合 $S(t_i) = \bigcup_{j=0}^{i} D_j$ 提供给 SWAG。

▶ **定义 23** (一般支配排列)。设 σ 是 n 请求的一个排列。给定时间 t，有一个已释放请求集合 $R(t)$ 以及一个集合 $U(t)$（暂且假设 $U(t) \neq \emptyset$）包含未释放请求。设 $u \in U(t)$ 是排列 σ 中第一个未释放请求，且 $R'(t) \subseteq R(t)$ 是排列中位于 u 之前的已释放请求集合。在时间 t，与 σ 对应的一般支配巡回路线由以下部分组成：

1. 一条最优 *TSP* 路径 $TSP_{\mathcal{O}-R'-u}$ 该路径从原点出发，在 $R'(t)$内服务所有请求，并以请求 u结束，**2.** 以及一条最优 *TSP* 路径 $TSP_{u-B-\mathcal{O}}$ 该路径从请求 u出发，服务$B = \big(R(t) \setminus R'(t)\big) \cup \big(U(t) \setminus \{u\}\big)$ 并最终返回原点。

注意，这里的最优 *TSP* 路径是在忽略请求的释放时间的情况下计算的。相应的通用支配巡回路线的长度 ℓ 随后是

$$\ell = \ell_{TSP_{\mathcal{O}-R'-u}} + \ell_{TSP_{u-B-\mathcal{O}}} \ .$$

如果 $U(t) = \emptyset$，那么 $|R(t)| = n$，每个请求都已释放，并且我们将每个 σ 对应的一般支配巡回路线定义为从原点出发、服务所有请求并最终返回原点的最优旅行商问题巡回路线（每个 σ 对应的都是同一条最优旅行商问题巡回路线）。

开放情形下，排列 σ 对应的一般支配路径的定义类似，即路径不再终止于原点，而是终止于来自 $\big(R \setminus R'\big) \cup \big(U \setminus \{u\}\big)$ 的任意请求，因此我们省略其定义。下文我们将讨论一般支配排列（路径或巡回路线）。

▶**命题 24.** 设 σ 为 n 个请求的一个排列。那么，在每个时间 t，都存在一个与之对应的 σ 的一般支配排列 σ'。等价地，有

$$\ell_{\sigma'} \leq \ell_{\sigma} \tag{12}$$

and

$$(1-\alpha_{\sigma'}(t))\ell_{\sigma'} \leq (1-\alpha_{\sigma}(t))\ell_{\sigma} \ . \tag{13}$$

证明。 由于时间 t 在此处是固定的，为方便起见，将省略对时间的依赖。在时间 t，存在一个已释放请求集合 R 以及一个未释放请求集合 U (暂且假设 $U \neq \emptyset$)。令 $u \in U$ 为时间 t 时排列 σ 中的第一个未释放请求，并令 $R' \subseteq R$ 为排列中位于 u 之前的已释放请求集合。因此，我们知道排列 σ 首先服务请求集合 R'，然后服务请求 u，之后再服务其余请求（集合 $B = \big(R \setminus R'\big) \cup \big(U \setminus \{u\}\big)$）。因此，我们可以将其长度 ℓ_σ 写为

$$\ell_\sigma = \ell_{\mathcal{O}-R'-u} + \ell_{u-B-\mathcal{O}} \ .$$

容易看出，数值 $(1-\alpha_\sigma)\ell_\sigma$ 恰好等于从未释放请求 u 开始的巡回路线部分的长度，即 $\ell_{u-B-\mathcal{O}}$。类似地，对于其对应的一般支配排列 σ'，有 $(1-\alpha_{\sigma'})\ell_{\sigma'} = \ell_{TSP_{u-B-\mathcal{O}}}$。显然，$\ell_{TSP_{\mathcal{O}-R'-u}} \leq \ell_{TSP_{\mathcal{O}-R'-u}}$ 且 $\ell_{TSP_{u-B-\mathcal{O}}} \leq \ell_{TSP_{u-B-\mathcal{O}}}$。因此，我们得到 (12) 和 (13)。

若 $U = \emptyset$，则 σ' 是服务所有请求并返回原点的最优旅行商问题巡回路线，且 $\alpha_\sigma = \alpha_{\sigma'} = 1$。因此，我们再次得到 (12) 和 (13) 成立。

因此，在每个时间 t，一个一般支配排列 σ' 支配排列 σ。◀

接下来，我们给出一般支配集的正式定义 D_i。

▶ **定义 25** (一般支配集)。 我们将一般支配集 D_i 定义为一个集合，其中包含由集合 $R(t_i)$、$U(t_i)$ 以及 $R'(t_i) \subseteq R(t_i)$ 和 $u \in U(t_i)$ 的所有可能选取所生成的全部一般支配排列，且其中 $i \in \{0, 1, 2, \ldots, n\}$。

对于集合 S，排列 SWAG 每次都会使用该集合，因此对每个 $t \leq t'$，都有 $S(t) \subseteq S(t')$，因为我们在时间 t_i 将所有 D_k，$k \leq i \leq n$ 的并集提供给算法。此外，一般支配集 D_i 对所有 $t < t_{i+1}$ 的排列均构成支配。因此，我们将证明：预言机 $\mathcal{D}$ 在任意时刻 $t \in [t_i, t_{i+1})$ 输出每个 $i \in \{0, 1, 2, \ldots, n\}$ 对应的一般支配集 $\bigcup_{j=0}^{i} D_j$，就是一个支配预言机。在此之前，我们先计算这种算法的运行时间。首先，我们证明，对于集合 $\bigcup_{j=0}^{i} D_j$ 中满足 $i \leq n$ 的一般支配排列，其数量总是可以用 2^n 作为上界。

▶ **引理 26**。 对于所有 $i \in \{0, 1, \ldots, n\}$，总存在一个集合 $\bigcup_{j=0}^{i} D_j$，其中至多包含 2^n 个不同的一般支配排列。

证明。考虑定义 23 中定义的一般支配排列。这里我们只考虑闭合变体；开放变体的证明类似。假设我们总是选取对应于由（唯一的）字典序最小的最优 TSP 路径 $TSP_{\mathcal{O}-R'-u}$ 和 $TSP_{u-B-\mathcal{O}}$ 组成的巡回路线的排列。对于任意 i 值，这样的排列完全由集合 $Q' = R'(t_i) \cup \{u\}$ 确定，因为 u 是 Q' 中唯一未释放的请求。此外，所有集合 Q' 都是全体 n 请求的子集，因此不同的集合至多有 2^n 个。注意，如果 u 不存在，即所有请求都已释放，那么就只有一个字典序最小的最优 TSP 巡回路线。因此，对于所有 $i \in \{0, 1, \ldots, n\}$，我们总能构造集合 $\bigcup_{j=0}^{i} D_j$，其中至多包含 2^n 个不同的（一般支配）排列。◀

接下来，我们使用引理 26 来证明以下关于 LA-SWAG 运行时间的定理。

▶ **引理 27**。 *LA-SWAG* 使用一般支配集的算法，在 *OLTSP* 的闭合和开放变体中，其时间复杂度均为 $O(n^2 \cdot 2^n)$ 。

证明。 根据引理 26，集合 $N = 2^n$ 中至多有 S 个排列供算法使用。在 $t = 0$时，我们可以预先计算一个集合 D，其中包含大小至多为 2^n 的一般支配排列；根据引理 26，计算时间为 $O(n^2 2^n)$，使用动态规划 [15, 24]。之后，每当 t_j释放一个请求时，我们就可以从预先计算好的所有可能的一般支配排列中选出合适的集合 $D_j \subseteq D$，将其加入之前的集合，并提供给充当预言机的算法。根据推论13，可得 LA-SWAG 使用一般支配集的算法运行时间为 $O(n^2 \cdot 2^n)$ 。 ◀

现在，我们可以陈述并证明本节的主定理了。

▶ **定理 4** （一般度量）。 *LA-SWAG* 使用一般支配集的预言机 D 算法运行时间为单指数，且对于旅行商问题的闭合变体，其竞争比为$\min\{3/2 + 5\eta, 2.75\}$；对于 *LA-OLTSP* 的开放变体，其竞争比为$\min\{3/2 + 5\eta, 3 - 1/6\}$。

证明。 如果一般支配集满足支配预言机的两个条件，那么我们可以使用 LA-SWAG 和引理 11，得到一个针对开放和闭合 LA-OLTSP 的指数时间算法，该算法具有 $3/2$一致性。显然，条件 1 得到满足，因为$S(t) \subseteq S(t')$ 对每个 $t \leq t'$都成立。条件 2 也得到满足，因为对于每个时间 t，对于 $R(t)$ 的每个可能子集以及 u的每种选择，都有相应的支配排列，因此对每个排列 σ也如此。因此，我们应用引理 11，得到 $3/2$的一致性。对于平滑性和鲁棒性，我们直接使用定理 1、2 和 3，从而完成证明。 ◀

6 特定度量空间上的多项式时间/FPT 算法

本节介绍如何基于一般框架，为特定度量空间（即树、环和花）构建多项式时间/FPT 算法。具体而言，我们将展示如何在这些情形下构建一个大小为多项式/FPT 的支配集。这些支配集将在每种情形下支配一个所谓的合理的 排列集，稍后我们会对其进行定义。我们现在聚焦于闭合变体，稍后再解释如何处理开放情形。此外，为了简便起见，我们将定义每当一个请求被释放时，添加到预言机集合中的“新”支配排列。应当理解，预言机会维护所有已生成排列的并集，并将其提供给 LA-SWAG。

我们还需要实现 LA-SWAG 的“清理”步骤，这需要计算经典（离线、不考虑释放时间）旅行商问题路径/回路的最优解。我们将在第 6.5 小节简要介绍如何做到这一点。不过请注意，这些运行时间足够短（事实上，在 $O(n)$ 中），因此 LA-SWAG 的运行时间瓶颈将是支配排列集合的基数。

现在，让我们开始着手 构建我们承诺过的定义和构造 。

▶**定义 28** (安全集)。 设 X 为一个请求位置集合。若对于向 X 中每个位置赋予释放时间的任意赋值 f，都存在一个排列 $\pi \in \Pi$，使其对于由此产生的输入 $Q = \{(x, f(x)) \mid x \in X\}$ 是最优的，则称排列集合 Π 对 X 是安全的。

请注意，所有排列的集合都是安全的。然而，正如我们将看到的，我们可以定义一些遵循特定理想结构但仍然安全的排列集合。我们将这类集合称为合理的。我们将支配这些集合，而非所有排列的集合。回想一下

LA-SWAG 之所以能够提供它所提供的所有保证，是因为它使用的预言机支配一个安全集，这也是本节的全部动机。

一般而言，我们将通过以下步骤证明支配关系。首先，我们会论证，在到达第一个尚未释放的请求 q （在被支配排列中）之前，支配排列能以不超过被支配排列的时间服务其所服务请求的超集。之后，将采用一个为其余请求提供服务的最优解。以上即可证明支配关系。

6.1 树

本节考虑形状如树的度量空间。也就是说，我们有一些连续空间，它们可以嵌入到边上带有长度的树中；这些长度均为正（并对应边的遍历时间），而与叶相连的边可以具有“无限”长度。请求可能出现在树的节点上，也可能出现在边上的某个点。例如，请求可能出现在距 q 1个单位、距 p2个单位的位置，其中 p 和 q 是树上由一条长度为3的边连接的节点。这对应于现实场景中，两处路口之间某条街道上的请求。允许叶边具有无限长度非常重要，这样才能涵盖直线和星度量。

合理集合。

对于这样的度量空间，我们可以专注于一组非常具体的合理排列。先来说明这些定义背后的想法。假设某个算法在时间 t 服务请求 q 。当算法完成时（例如在时间 t_f），它将位于原点 $\mathcal{O}$。从 q 到 $\mathcal{O}$ 只有一条路径$\mathcal{P}_q$ ，而且必须在 t 之后沿此路径行进。因此，路径 $\mathcal{P}_q$ 上的任何请求 q' ，无论如何都会在 t 之后的某个时刻被访问。所以，算法没有理由在时间 t 之前（即在服务 q之前）尝试服务 q' ，因为无论如何它都会在 t 之后返回原点的途中服务 q' 。由此可知，对于沿着通往原点的路径 $\mathcal{P}_q$ 遇到的任意请求序列 S_q ，都可以安全地假设它是最优排列的一个子序列。这便引出了以下定义。

▶ **定义29** (树的合理集合)。 设 T 是一棵以原点 O 为根的树， X 是 T 上一组请求位置。集合 $\Pi_s(T \quad X)$ 由所有满足以下条件的排列 π 构成。设 $\pi = x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$。对于任意 $x_{\pi(i)}$ 以及路径 $P^{x_{\pi(i)}}$ 上从 $x_{\pi(i)}$ 到原点的任意 $x_{\pi(j)}$ ， 均有 $j > i$。

▷ 断言30。对于任意树$\mathcal{T}$和任意请求位置集合 X，集合 $\Pi_s(\mathcal{T} \quad X)$是安全的。

证明。令 π 为任意排列，并令 f 为将释放时间赋值给 X 的任意赋值。我们将证明存在 $\pi_s \in \Pi_s$(T X)，使得 $|\pi_s| \leq |\pi|$，其中 $|\pi|$ 表示在 f 下与排列 π 相关联的巡回路线的完成时间。令 $z(t)$ 表示服务器在时间 t 沿着 π 行进时的位置。令 $l_i = \max\{t \mid z(t) = x_i\}$。现在，我们对 l_i 值排序，并提取排序后的索引，得到 $\pi_s = x_{\pi_s}(1)\ x_{\pi_s}(2)\ \cdots, x_{\pi_s}(n)$。换言之， π_s 是根据服务器沿着 π 行进时最后访问各请求的先后顺序导出的。根据构造， $\pi_s \in \Pi_s$(T X)。现在，让我们看看为什么 $|\pi_s| \leq |\pi|$。令 t^{run} 为 π_s 在请求位置 $x_{\pi_s}(f)$ 等待的最后时刻。显然， $x_{\pi_s}(f)$ 的释放时间至少为 t^{run}。这意味着 π 在 t^{run} 之后仍须完成一条包含 $\mathrm{P}_{run} = x_{\pi_s}(f) \rightarrow x_{\pi_s}(f+1) \cdots \rightarrow x_{\pi_s}(n) \rightarrow$ O 的超路径。令 l_{run} 表示 P $_{run}$ 的长度。显然， $|\pi| \geq t^{run} + l_{run} = |\pi_s|$。 ◀

支配集。

对于上述合理集合，我们将定义一个基数为$O(2^l \cdot n)$数量级的支配集，其中 l 是底层树的叶节点数。这将为树上的该问题带来一种算法，其运行时间关于叶节点数参数化后属于FPT。正如读者可能已经猜到的那样，其思路是考察叶节点的子集，更有趣的是，考察这些叶节点到原点的路径。也就是说，对于叶节点到原点的路径的每个子集，以及每个中间请求 q的选择，我们将定义这样一个排列：首先以最优方式访问所选叶节点，并在 q 处结束，然后从 q 开始并在原点结束，“清理”其余请求。之所以这样就足够了，其直觉在于，任何合理排列只有在已经服务了 q'“下方”的叶节点后，才能在 q 之前服务请求q' 。

现在我们可以正式定义支配排列。为此，我们考虑将树空间限制在请求的张成空间上。也就是说，对于每个叶，如果其相邻边上存在请求，我们就将其替换为一个新节点；该节点是沿着叶的可能无限长边距离最远的请求，并将无限长度替换为该请求与原叶的父节点之间的距离。此外，当通往某个叶的路径上没有其他请求时，我们会在请求之后对树进行“修剪”。现在，每个叶都是一个请求。

▶ **定义 31** ((q, L')-扫描)。 设 T 为一棵有根树， L 为其叶节点集合。设 L' 为 L 的子集，且 q 为树上的某个请求，其位置为 x ，满足 x 位于通往 L' 中某个叶节点的某条路径上（x 也可能位于一条边上）。现在，令 P 为从根出发并以 x 结束、同时沿途访问 L' 中所有叶节点的最优路径。我们称 P 为 T 的(q, L')-扫描。

▶ 定义 *32*（（q, L'）-支配者）。设 R, U 分别为时间 t时已释放和未释放请求的集合。设 $L \subseteq R$ 为已释放叶节点的子集，且设 L' 为这些叶节点的一个子集。设 $q \in U$ 为沿着通往某个叶节点的路径上最远的未释放请求。设 P' 为 T 的（q, L'）-扫描。现在，设 U'为此次扫描后未服务请求的集合。也就是说， $U' = U \cup R'$，其中 R' 是已释放请求中未被 P'服务的子集。构造以 q为根的树 T' ，按照 U' 进行修剪并保留原点。设 L'' 为树 T' 的叶节点。设 P'' 为 T' 的（O L''）-扫描。巡回路线 $P'P''$ 所隐含的排列 π 是 T 的（q, L'）-支配者。我们将使用记号 $d_{R,U,\mathcal{T}}(q, L')$ 来表示它。

▶ **定义 33** (树的支配集). 设 R、U 如定义 *32* 所述，且 T L 如定义 *31* 所述。树支配集为：

$$Dom^{\curlyvee}(R, U, \mathcal{T}, L) = \bigcup_{q \in U^*, L' \subseteq L} d_{R,U,\mathcal{T}}(q, L'),$$

其中， U^* 是沿着通向叶节点的路径上最远的未释放请求集合。

现在我们将证明，该集合确实支配树的合理排列。回想一下，如果沿着从叶到根（原点）的单一路径的所有请求序列，按照距原点的距离递减排序后，在排列中以相同顺序出现，则称该排列对于该树是合理的。

▶**引理 34** (树支配). 设 π 是对树 T（根据输入请求进行修剪）而言合理的排列。设 t 是当前时间， R和U 分别是时间 t时的已释放请求和未释放请求。此外，设 L 是 T 的叶节点集合。那么 $Dom^{\curlyvee}(R$、U、T $L)$包含一个支配 π的排列 π_{dom} 。

证明。设 $\pi = q_{\pi(1)}, q_{\pi(2)}, \ldots, q_{\pi(n)}$。令 $q_{\pi(u)}$ 为 π 中第一个未释放请求。令$R_{served} = \{q_{\pi(1)}, q_{\pi(2)}, \ldots, q_{\pi(u-1)}\}$ 为 π 在 $q_{\pi(u)}$ 之前服务的请求。令 $L' = L \cap R_{served}$。那么，$(q_{\pi(u)}, L')$-支配者 $\pi_{dom} = d_{R,U,\mathcal{T}}(q_{\pi(u)}, L') \in Dom^{\Upsilon}(R, U, \mathrm{T} \quad L)$ 支配 π。由于 π_{dom} 的构造，具体而言，因为会为 $q_{\pi(u)}$ 之前和之后的部分计算一条最优路径，我们只需证明 π_{dom} 在不花费更多时间的情况下，服务的请求是 π 在 $q_{\pi(u)}$ 之前所服务请求的超集。注意，由于 π 是合理的，R_{served} 必须对路径扩展封闭。也就是说，对于每个 $q \in R_{served}$ 以及 q 所在的每条路径 P，路径 P 上比 q 更远的所有请求也必须在 R_{served}中。这意味着 R_{served} 中的所有请求都位于某条通往 L'中某个叶的路径上。但是，π_{dom} 在到达 $q_{\pi(u)}$ 之前会遍历所有这些路径，这证明 π_{dom} 在 $q_{\pi(u)}$ 之前服务的请求是 R_{served} 的超集。还需证明它服务这些请求的速度至少与 π 服务 R_{served} 一样快。这是因为π_{dom} 被选为仅对 L' 进行最优服务并最终到达 $q_{\pi(u)}$，而 π 无论如何也必须这样做。证毕。◀

现在，利用上述引理，我们得到了关于树的主定理。复杂度来自引理 12 中与支配集基数相关的项，因为后文将证明，在树上求解经典旅行商问题需要 $O(n)$ 时间。

▶**定理 5 (树).** 存在一种适用于树的支配预言机 *LA-SWAG*，其时间复杂度为 $O(2^l \cdot n^3)$（闭合变体）和 $O(2^l \cdot n^4)$（开放变体），其中l 为输入树的叶节点数。

6.2 环

合理集合。

环与直线事实证明相当相似。事实上，要定义环上的合理排列，只需注意以下几点。如果一个排列沿环路环行超过一整圈，就可以删去除最后一圈之外的所有环行，因为每一圈都会经过每个请求。由于在完整环行之前服务任何请求都没有意义，因此离开$\mathcal{O}$后，可以先进行环行。如果存在这样的环行，我们可以假设此后排列沿环移动，就像沿直线移动一样，即不会越过原点的对跖点。因此，我们可以根据是否进行环行来定义合理排列，然后将问题归约为树的情形。

▶ **定义 35**（环的合理集合）。设 X 为环上的一组请求位置。排列 $\Pi_s^{\circ}(X)$ 的合理集合由所有排列 π 组成，这些排列是通过拼接 π_{loop} 和 π_{line} 得到的，其中 π_{loop} 是以循环方式覆盖 $X' \subseteq X$ 的一个子排列，而 $\pi_{line} \in \Pi_s(T, X \mid X')$，其中 T 是在与原点相对的中点处分割环后得到的树。

▷ 断言 36. 始终存在一个遵循合理排列的最优解。

证明。我们可以假设环是周长为 1 的圆。现在考虑一个最优解 OPT，并用 σ^* 表示 OPT 服务请求的顺序（请求的一个排列）。回顾一下，不失一般性，OPT 沿着巡回路线/路径 σ^*行进，只在请求所在的位置等待：它在时间 $d(\mathrm{O}\ \sigma^*[1])$ 内从 O 前往（ $\sigma^*[1]$ 的位置），如果请求尚未释放，则在 $\sigma^*[1]$ 的位置等待；然后在时间$d(\sigma^*[1], \sigma^*[2])$内从 $\sigma^*[1]$ 前往 $\sigma^*[2]$，.....接下来我们将证明，对于每个最优解 OPT，都存在另一个最优解 OPT′，它是一个合理排列。

考虑 OPT 最后一次通过原点的对跖点$\mathcal{O}A\mathcal{O}'$ 从一个半圆移动到另一个半圆$\mathcal{O}B\mathcal{O}'$（或反之）的时间$\mathcal{O}'$（图 3）。如果 OPT 在任何时刻都不穿越

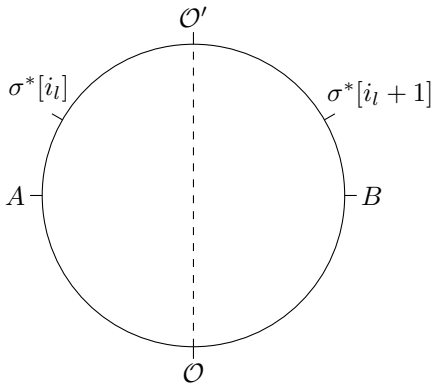


图 3 用于证明断言 36 的环。

$\mathcal{O}'$，那么 OPT′ 可以直接跟随具有之字形移动的 OPT，而这是一种合理排列。否则，在该排列中会存在一对连续的请求 $\sigma^*[i_l]$、$\sigma^*[i_l+1]$，穿越会发生在这两个请求之间。无损失地假设*O O</style}and O*$B\mathcal{O}'$。于是，OPT 必须从 $\sigma^*[i_l+1]$ 移动到原点，且不能再次穿越$\mathcal{O}'$。令 t^* 为 OPT 在处理完请求 $\sigma^*[i_l+1]$后第一次到达原点的时间。OPT′ 不必跟随 OPT，而是可以在原点等待 t^*-1 时间，然后绕整圆$\mathcal{O}A\mathcal{O}'B\mathcal{O}$ 移动一周，中途不停留。在这种情况下，它会在时间t^* 到达$\mathcal{O}$ （与 OPT 相同），并且至少处理完 OPT 截至该时间所处理的那些请求。接下来，OPT′ 只需跟随具有之字形移动的 OPT，因为它之后再也不会穿越$\mathcal{O}'$。因此，OPT′ 是一个合理排列，并且 $|\text{OPT}'| \leq |\text{OPT}|$。

◀

支配集。

正如我们所讨论的，环上的合理排列在开始时至多包含一个完整环行。请注意，如果缺少这个环行，该情形就退化为直线情形，而对于直线情形，我们已经定义了支配器。因此，只需在假设 π 确实在开始时有一个环行的情况下定义支配器，因为这样我们只需取两个集合的并集，即可构造所有支配器组成的集合。

一个关键区别在于，当前请求 q 是否属于 π 的环行。在第一种情况下，π 到目前为止只走过环行的一部分；在第二种情况下，它已经走过一个完整环行，随后又走过了某条可能较为复杂的类直线路径。对于 q的每一种取值，我们定义两个排列，分别对应第一种情况和第二种情况。

第一个排列服务于 π已走过的环行部分内的所有已释放请求（沿相同方向），然后移动到 q。从那里开始，它遵循一个服务其余请求的最优排列（忽略释放时间）；对于环而言，这可以很容易地计算出来。第二个排列服务于包含完整环行的所有已释放请求（如果 q 位于环的左弧上，则按顺时针顺序，反之亦然），然后移动到 q。最后，它以相同的方式，通过计算从 q 到原点的最优路径来完成。现在，我们将更加形式化地定义这个支配排列集合。

▶ **定义 37** （新月排列）。 设 R 为时间 t时环上已释放请求的集合，U 为相应的未释放请求集合。设 $r_1, \ldots, r_m$ 为从原点开始按顺时针顺序排列的已释放请求。对于每个 $q \in U$，左新月排列

of q 是

$$(r_1, \ldots, r_l) \mid\mid (q) \mid\mid opt_q((R \cup U) \setminus \{r_1, \ldots, r_l\}),$$

其中， l 是满足以下条件的最大索引：从原点出发按顺时针顺序观察时， r_l 位于 q 左侧；而 $opt_s(S)$ 表示从 s出发、遍历 S 中的每个点并最终返回原点，且不考虑释放时间的最快路径。 q 的右新月排列定义类似。我们分别用$lc_{R,\,U}(q)$ 和$rc_{R,\,U}(q)$ 表示这些排列。

▶ **定义 38** （满月排列）。 设 R、U如定义 *37* 所述，且 $q \in U$。若 q位于环的左弧上，也就是说，沿环顺时针行进时，它位于原点的对跖点之前，则 q 的满月排列为

$$(r_1, \ldots, r_m) \mid\mid (q) \mid\mid opt_q(U).$$

若 q 位于另一段弧上，则已释放请求的服务顺序相反。我们用$fm_{R,\,U}(q)$ 表示这一排列。

▶ **定义 39** （环的支配集）。 对于定义 *37* 和 *38* 中给定的 R, U， R, U 的环支配集为

$$Dom^{\circ}(R, U) = \bigcup_{q \in U} \{rc_{R,U}(q), lc_{R,U}(q), fm_{R,U}(q)\}.$$

现在我们已经定义了支配者，接下来需要证明它们确实支配包含环路的合理排列集合。

▶**引理 40** （环支配）。 设 π 是一个合理排列，其巡回路线起始处包含一个完整环行。设 t 为当前时间， R, U 分别为时间 t 已释放和未释放的请求。那么， $Dom^{\circ}(R, U)$ 中包含一个支配 π 的排列 π_{dom}。

证明。 设 $\pi = q_1, q_2, \ldots\ldots, q_n$。不失一般性，假设 π 的环路方向为顺时针。此外，设 $q_x \in U$ 是时间 t 时 π 中第一个未释放请求。我们分两种情况讨论：

- 请求 q_x 位于 π 的环路部分上。那么，$lc_{R,\,U}(q_x) \in Dom^{\circ}(R, U)$ 是 π 的一个支配者。这是因为这两个排列到达 q_x （沿顺时针弧前往该处）所行距离相同，而且在到达该处之前，$lc_{R,\,U}(q_x)$ 服务的是 π 所服务请求的超集（该弧上所有已释放的请求），而 π 只服务其中一些已释放请求。此外，到达 q_x 后， $lc_{R,\,U}(q_x)$ 以最优方式服务 π 必须服务的请求的子集，因此在这段路程上也更快，这意味着它支配 π。
- 请求 q_x位于 π 的环路部分之后。于是，$fm_{R,\,U}(q_x) \in Dom^{\circ}(R, U)$ 是 π 的支配者。这是因为 $fm_{R,\,U}(q_x)$ 服务于其环路部分中所有已释放请求，并以可能的最早时间移动到 q_x 环路之后的位置（环路的代价为 1，之后走到 q_x 的最短弧）。因此，在到达 q_x 时，它服务的请求是 π 所服务请求的超集，且所用时间不超过由 π 诱导的巡回路线所需的时间。随后，它以最优方式服务于 π 必须服务的请求的一个子集，因此在该部分也更快，这意味着它支配 π。

◀

因此，我们关于环的主定理得证。

▶**定理 6 (环)**。 存在一个用于 *LA-SWAG* 环的支配预言机，其闭合变体的时间复杂度为 $O(n^3)$ ，开放变体的时间复杂度为 $O(n^5)$ 。

6.3 花

本节将前面关于环和树的小节中采用的推理扩展到所谓的花。花由若干个环（花瓣）组成，它们都连接到原点（花托）。为了在艺术上更加完整，还会在原点处连接一条与花瓣不相交的半直线（主干）[3]。图 4 展示了一个示意图。下文将为这类图定义合理排列和支配排列的集合。

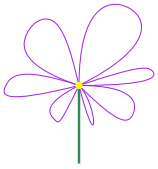


图 4一朵有 7 片花瓣的花

合理集合。

核心思想如下。我们可以假设，任何合理的解都至多以循环方式遍历花的一个花瓣一次，原因与环上这一结论成立的原因相同。此外，这个花瓣不会在此之前被访问。

不失一般性。在进行这样的遍历后（如果确实发生），我们可以安全地在该花瓣的中点处从原点将其“剪开”，并将其视为树的两个不同分支，因为中点之后不会再被穿过。因此，这里的合理集合定义为包含这样的排列：每个花瓣至多有一个环路，这类环路是对相应花瓣的首次访问（但可以在排列的任意连续部分内完整执行），并且树上的合理集合限制也适用于每个花瓣（它在执行这样的环路后会变成一条直线；如果确实存在这样的环路）。

支配集。

利用这一观察，我们可以按如下方式构造支配排列。首先，选择一个未释放请求 q。这将是我们要为其定义支配者的合理排列中的“当前”请求。令 π 表示该排列。注意，π在整个过程中都会以循环方式使用花瓣的一个子集。因此，我们“猜测”这个子集，并将其称为$\mathcal{P}$。此外，在到达 q之前，子集$\mathcal{P}_{done}$ 中的元素将已被完全遍历。我们还猜测$\mathcal{P}_{done}$。这也定义了$\mathcal{P}_{left} = \mathcal{P} \setminus \mathcal{P}_{done}$。然后，按以下方式定义支配者。首先，以任意顺序遍历$\mathcal{P}_{done}$ 中的所有花瓣。接着，“剪去”除$\mathcal{P}$中的花瓣以外的所有花瓣。这样便留下了一棵树$\mathcal{T}$（实际上是一颗星形）以及$\mathcal{P}$中的花瓣，它们彼此不相交。注意，如果我们正确猜出了这些花瓣集合，这样做就是安全的，因为 π 访问$\mathcal{P}$之外的花瓣时，会表现得仿佛它位于一棵树上。

现在，我们还猜测在 q 之前要访问的 $\mathcal{T}$的叶节点子集 $L' \subseteq L$。确定这一点后，我们将从 $\mathcal{O}$ 到 q 的最优路径追加到初始的花瓣环状子回路中；这条路径会经过 L' 中的每个叶节点，并且还可能服务于 π 在 q 的花瓣上执行环路时可能服务过的那条弧。最后，我们追加从 q 出发的最优路径，以“清理”剩余未服务的请求。

注意，为了做出所有这些不同的猜测，每个花瓣最多有六种选择（环路已完成、稍后处理环路，或剪除后两个叶节点的四种子集选择之一）。因此，如果图中有 p 个花瓣，支配器的基数为 $O(6^p \cdot n) = O(2^{2.59p} \cdot n)$。为简单起见，上述计算将主干视为一个花瓣。同样，这为花情形给出了一个固定参数可处理算法。

[3] 事实上，对于由一棵树以及附着在树的单个顶点上的若干边不相交环组成的图，也可以采用同样的方法。因此，甚至可以将花的（地面）根也纳入其中。

下面简要证明其有效性。回想一下，我们只需证明，任意合理排列 π 在任何时刻都被上述某个支配者 π_{dom} 支配。因此，对于 π 时刻 t，我们声称，与 q、P、P_{done} 的正确猜测相对应的 π_{dom} 和 L' 支配 π。我们先聚焦于 P_{done} 的花瓣。显然，π_{dom} 以遍历环路所需的最短时间服务这些花瓣中所有已释放的请求；正确猜测 P_{done} 时，π 也会花费这段时间。现在，我们把注意力转向 π_{dom} 在初始环路之后、到达 q 之前所经过的部分。根据构造，这条路径是访问 L' 以及所有可能请求的最优路径；如果当时正在遍历 q 的花瓣环路，π 可能已经服务了这些请求。不过，依照树小节中的类似推理（π 所服务请求的路径扩展性质），无论是否通过环路访问 P_{done}，π 也必须完成这项任务。这种独立性是因为 L' 在 P_{done} 的花瓣中没有叶节点：那里所有已释放的请求都在 π_{dom} 的初始环路中得到服务；此外，如果 q 属于某个环路，那么它的花瓣不在 P_{done} 中。而且，π_{dom} 会服务 π 到达 q 之前已服务的所有请求，因为 π 服务的每个请求要么位于 P_{done} 的某个花瓣中，要么沿着通往 L' 中某个叶节点的路径，要么沿着通往 q 的（正确选取的）弧。因此，π_{dom} 服务的请求是 π 所服务请求的超集，并且至少同样快地到达 q。根据“清理”路径的最优性，即可推出支配关系。

现在，我们将通过精确定义支配器，并证明它们支配排列的合理集合，使上述内容更加严谨。

▶ **定义 41** (嬉皮排列)。令 R、U 分别为时间 t时已释放请求和未释放请求的集合。令 $q \in U$ 为一个尚未释放的“当前”请求。令 P 为输入花形度量空间中花瓣的子集。令 $P_{done} \subseteq P$。暂且假设 q 不在 P 的某个花瓣上。令 T 为将不属于 P 的花瓣替换为两条“分支”后得到的树，每条分支的长度均为原花瓣的一半（在花瓣的中点处剪开）。T 也可能包含花的主干。最后，令 L 为 T 的叶节点（根据已释放请求进行修剪），并令 $L' \subseteq L$。嬉皮排列 $hp_{R,U}(q, P, P_{done}, L')$ 定义如下：

- 首先，通过依次遍历 P_{done} 的每个花瓣各一次，以任意方式（例如顺时针方向），并按照某个固定顺序优先处理各花瓣，从而服务所有已释放的请求。然后，计算一条最优路径，使其经过 L' 中的每个叶，并最终到达 q。我们将这条路径（更准确地说，与之对应的子排列）追加到排列中。最后，计算并追加一条最优路径：该路径从 q 出发，经过此前尚未服务的任意请求，最终回到原点。

若 q 位于 P 的一个花瓣中，我们将上述定义稍作修改：要求在第二部分中，不仅访问 L' 中的叶节点，还要访问沿着通往 q 的花瓣弧上的所有请求。我们定义两个这样的排列，每条通向 q 的弧各对应一个。因此，$hp_{R,U}(q, P, P_{done}, L')$表示按上述方式生成的排列集合（实际上是一个或两个排列）。

▶ **定义 42** (花的支配集)。我们沿用定义 *41* 中的记号。现在描述花的支配集是如何生成的。我们只需取所有（有效）P、P_{done}、q、L' 选择所对应的嬉皮排列的并集。也就是说，我们定义集合：

$$Dom^{\text{✿}}(R,U) = \bigcup_{\substack{q \in U \\ \mathcal{P} \text{ is a subset of petals,} \\ \mathcal{P}_{done} \subseteq \mathcal{P}, \\ L' \text{ is a subset of the leaves of } \mathcal{T}}} hp_{R,U}(q, \mathcal{P}, \mathcal{P}_{done}, L')$$

最后，我们现在可以证明，上述集合支配花的合理排列集合。

▶ **引理 43** (花形支配)。 设 π 是该花形的合理排列，且 R和U分别是时间 t时已释放和未释放的请求。那么， $Dom^{\text{✿}}(R, U)$ 包含一个支配 π的排列π_{dom} 。

证明。 设 q 为 π 中在时间 t 尚未释放的第一个请求。此外，设 $\mathcal{P}$ 为 π 在某个时刻将要经过（或已经经过）的花瓣集合。设 $\mathcal{P}_{done} \subseteq \mathcal{P}$ 为其中在到达 q 之前已经完成环路的花瓣所构成的子集。设 $\mathcal{T}$ 如定义 41 所述。设 L' 为 π 在 q 之前访问的叶节点子集。我们声称，$hp_{R,\ U}(q,\ \mathcal{P},\ \mathcal{P}_{done},\ L')$ 包含了我们所要寻找的排列 π_{dom}。

- 暂且假设 q不在 P 的某个花瓣上。那么，$hp_{R,U}(q, \mathrm{P}, \mathrm{P}_{done}, L')$ 是单元素集合 $\{\pi_{dom}\}$。原因如下：注意，我们可以将 π_{dom} 到达 q之前的“移动”分为两个相互独立的部分，一个对应初始环路，另一个对应其余部分。第一部分的代价等于 P_{done}中花瓣的总长度，而 π 也必须支付这笔代价，因为在到达 q之前的某个时刻，它已经完整地遍历了这些花瓣。现在注意，根据构造， L' 的叶节点均不位于 P_{done}的花瓣上。因此， π_{dom}会以最优方式访问这些叶节点，而 π 在到达 q之前，也必须额外支付至少相同的代价（这笔代价独立于与 P_{done}中环路相关的代价，且需要在其基础上额外支付）。因此，我们已经证明， π_{dom} 在到达 q之前所遵循的路径不会更长。剩下只需证明，沿着这条路径，它能够服务所有 π 在到达 q之前已经服务过的请求。对于环形花瓣，这显然成立，因为 π_{dom} 会服务那里所有已释放的请求。接下来，对于 P 之外的所有请求，我们可以将问题视为发生在一棵树上的问题。因此，树结构小节中的论证均适用（例如，在路径扩展下的封闭性等）。最后，还需证明，P^{left} 中不存在 π 已经服务但 π_{dom} 尚未服务的请求。这一点显然成立，因为根据 q 不在 P^{left}的任何花瓣上的假设，此类请求不存在。

- 现在我们可以解决 q 位于环形花瓣上的情况。这意味着 π 当前正在经历一个环路。我们选择 π_{dom} 作为沿着对应弧到达 q之后的排列。可以用大致相同的方式证明，在到达 q 之前， π_{dom} 支付的成本不会更高（我们将成本分为最初的环路，然后是 $\mathcal{T}$加上最后一条弧）。我们唯一还需要证明的是， π_{dom} 为 $\mathcal{P}^{left}$ 中 π在到达 q之前处理每个请求。但这些正是正确选择的、通往 q 的弧上的请求，因此结论成立。

◀

同样，上述引理证明了我们关于花的主定理。至于树，复杂度来自与引理 12 中支配集的基数相关的项，因为后文将证明，在花上求解经典旅行商问题需要 $O(n)$ 时间。

▶**定理 7** (花)。 存在一个适用于花的 *Domination* 预言机 `LA-SWAG`，其时间复杂度为 $O(6^p \cdot n^3)$（对于闭合变体），以及 $O(6^p \cdot n^5)$（对于开放变体），其中 p 是输入花的花瓣数量。

6.4 处理开放变体

上述分析大多基于利用问题的闭性质，具体而言，OPT 必须返回$\mathcal{O}$这一事实。在开放情形下，这一点不再成立。不过，我们可以通过枚举所有可能的

OPT 路径的端点，人为施加类似的限制。也就是说，我们假设某个特定请求 q_f 是 OPT 最后服务的，并在此假设下构造支配排列。最后，我们考虑所有这些集合的并集（即遍历最终请求的所有可能选择）；由于任何合理排列都必须以某个 请求结束，因此该并集将支配开放情形下的合理集合。

树。

对于树的情形，问题其实相当简单。固定 q_f 后，我们只需将树重新选根为 q_f。随后，所有以 q_f 结束的合理排列都必须遵循重新选根后的树中通往叶节点的路径上各请求的顺序；这与我们在闭合情形下采用的简化方法非常相似。因此，我们可以通过求出输入树在所有 n 种重新选根方式下的树支配者，并取其并集，来构造支配排列集合。请注意，重新选根最多只会使叶节点数量增加 1。只有当新根恰好位于一条边上时才会发生这种情况，此时我们必须将该边分成两条。该集合支配树上开放情形的合理排列集合，且其大小为 $O(2^l \cdot n^2)$，因此再次得到一个固定参数可处理算法。

环。

对于环的情况，与闭合变体相比，我们采用了完全不同的方法。我们不再指定合理集合并加以支配，而是通过相对简单的分类讨论，直接支配所有排列构成的集合。

- 第一个简单的情况是，要被支配的排列 π 在到达当前请求 q 之前没有越过点 O′, ，即原点的对跖点。在这种情况下，我们将环视为一条直线，而已为树定义的支配器就足够了。这里的对应关系是：排列 π 被支配者π_{dom} 支配，当且仅当在 q 之前，π_{dom} 访问了与 π 相同的一组极端请求（最左侧或最右侧请求）。现在，假设 π 确实在到达 x（即 q 的位置）之前越过了点 O′。我们先排除 π 在到达请求 q 之前移动了至少 $1+d$(*x*, O) 距离的情况，因为此时，先通过环路服务所有已释放请求，然后尽可能早地前往 x 的排列会支配 π。不失一般性地，假设在最后一次经过 O 与之后第一次经过 O′之间，π 沿右弧移动。现在，我们聚焦左弧。设 q_1 是左弧上距离 O′ 最近、且 π 在最后一次经过 O 之前服务过的请求。设 q_2 是左弧上距离 O 最近、且 π 在最后一次经过 O 之后、离开 q 之前服务过的请求（也可能发生 $q_2 = q$）。我们分别用 x_1 和 x_2 表示它们的位置。由于必须在移动距离不超过 $1 + d$(*x*, O) 的过程中经过这些点，我们可推得d (x_1, O) $<$ d(x_2, O)。注意，(x_1, x_2) 左弧上的任何请求都不可能在请求 q之前由 π 服务。

对于这样的 π，我们定义如下支配者。首先， π_{dom} 移动到 x_1 ，并在返回原点的途中服务所有请求。然后，它按照顺序服务沿右弧直至$\mathcal{O}'$, 的所有请求。到达 $\mathcal{O}'$, 后，它继续朝 x_2移动，并沿左弧服务途中的所有请求。最后，它移动到 x，随后以最优方式处理剩余的未服务请求。我们可以看到，在服务 q之前， π_{dom} 服务了位于 (x_1, x_2) 左弧之外的所有请求，即它服务的请求集合包含 π 所服务的请求集合。此外， π_{dom} 的移动可以独立地对应于 π的移动进行计费，因此行进距离不会更大，这意味着支配关系成立。

上述支配者大多数属于最后一种类型，在这种情况下我们必须选择 q，q_1和 q_2（与树的情况不同，我们不需要选择 q_f）。因此共有 $O(n^3)$ 个支配器，由此得到一个多项式时间算法。

花。

对于非平凡花（即不只是单个环），情况稍微复杂一些，但我们仍然可以得到一个固定参数可处理算法。注意，最终请求 q_f 要么位于某个花瓣上，要么不在。如果不在，那么所有花瓣至多仍可按循环方式遍历一次（并且在这样的环路之前完全不会被访问）这一论证依然成立。因此，只需重新选树根，并猜测修改后的树的闭合变体花支配器的参数（之后再附加开放变体的最优解），仍然可行。

现在，如果 q_f 恰好位于某个花瓣上，我们就还需要做一些额外工作。不过请注意，在这种情况下，任何从请求 q 到 q_f 的路径，只要 q 不和 q_f 位于同一个花瓣上，就必须经过原点。因此，我们仍可以假设，合理排列遵循从原点出发、通向（可能经过剪裁的）树的叶节点的路径上各请求的顺序，至少对于不在最终花瓣上的请求是如此。在这种情况下，我们不会重新选取树的根。此外，所有非最终花瓣最多只能沿环形路径遍历一次。因此，我们只需处理与 q_f 位于同一个花瓣上的请求。幸运的是，这些请求相对容易处理，基本上只需将最终花瓣视为开放变体中的单个环。

假设“当前”请求 q 不在最终花瓣上。那么，我们可以像原先那样，通过环路处理最终花瓣，或者将其剪除并猜测叶节点。这是因为在到达 q之前，π 要么已经经过最终花瓣的中点，要么还没有。对于前一种情况，它必定已经完整环行，因为 q 不在该花瓣上。对于后一种情况，我们将最终花瓣视为一棵树。

剩下的情形比较有意思：q 和 q_f 都位于同一片花瓣上。在这种情况下，我们先猜测其他 花瓣和树上的叶节点，并以闭合方式最优地服务它们，即返回原点。现在，我们只考虑 q和 q_f 所在的花瓣，并为该花瓣添加一个环型支配器。

总的来说，我们得到一个基数为 $O(n^3 \cdot 6^p) = O(n^3 \cdot 2^{2.59p})$ 的支配集，其中 p是花的花瓣数。这是因为，与闭合情形相比，我们的支配器数量大致多出一个二次乘法因子，因为我们还要额外选择要么 q_f，要么 q_1 和q_2。第一种情况对应于 q_f 位于树部分的情形（此时无需选择 q_1 和 q_2）；第二种情况下，对 q_1 和 q_2 的选择意味着选定了 q_f 所在的花瓣，而这正是定义我们所添加的环型支配器所需了解的全部信息。

6.5 快速求解经典旅行商问题

这里介绍如何在适当的运行时间界限内计算经典旅行商问题的最优解。这一方法用于算法 2 的“清理”步骤，也用于支配排列的定义。

对于树，我们可以简单地从起点执行一次深度优先搜索（DFS）遍历，同时降低通向 q 的边的优先级，其中 q 是我们希望计算的最优路径的终点。DFS 所遍历的相应路径是最优的，其代价等于边长总和的两倍减去 q 到起点的路径长度。请注意，出于我们的目的，我们可以假设 DFS 的运行时间为 $O(n)$，其中 n 是请求的数量。为此，我们需要证明实际树的规模可以被限制在 n 的常数倍以内，因为这样一来，DFS 的复杂度就归结为 $O(n)$。作为第一步预处理（且只需执行一次），我们进行如下操作：对树进行修剪，使其仅保留从各个请求出发的路径。

到根（可以是任意点，也可以是原点）的边会被保留。这意味着我们有 $l \leq n$，其中 l 是修改后的树的叶节点数。此外，我们假设树中不存在度恰好为 2 的顶点（原点除外），因为这类顶点对在线旅行商问题并非必不可少（可以将与其相连的两条边视为一条边）。现在请注意，除了叶节点外，树中的任何顶点的度至少为 3。令 i 为树中非叶节点的数量。根据握手引理，我们有$2e = \sum_v \deg(v) \geq 3i + l \implies i \leq l$，其中 e 是树的边数，并回想起 $e = i + l - 1$。因此，树的规模确实为 $O(n)$，这表明我们可以在 $O(n)$ 时间内解决经典旅行商问题。

对于花（其中包含环），我们首先确定哪些花瓣更适合沿环遍历，哪些更适合沿两条不同的分支遍历。只需比较完整环的代价与一种之字形遍历的代价即可轻松作出判断，后者会访问所有请求。随后，我们对确定要作为树遍历的花的部分运行深度优先搜索，并在该深度优先搜索遍历中任何经过原点的位置附加花瓣环。如果不存在这样的位置，则属于退化情况，处理起来也很简单（深度优先搜索只遍历半直线）。显然，只要预处理时丢弃不含请求的花瓣，整个过程也能在 $O(n)$ 时间内完成。

我们注意到，对于 LA-SWAG，这些额外开销在渐近意义上并不重要，因为运行时间的瓶颈已来自支配集的基数乘以n^2；而计算支配排列时，每个只需花费 $O(n)$，且总共最多调用支配预言机 $n + 1$ 次。

7 结论

我们研究了结合请求位置预测的在线旅行商问题。在预测完美的假设下，我们的算法 LA-SWAG 达到了 $3/2$ 的竞争比；在我们考察的大多数情形中，这一结果是紧的。此外，该算法具有平滑性，并提供低于 3 的鲁棒性保证，优于先前工作。LA-SWAG 的运行时间为单指数；不过，我们展示了如何在特定度量空间中消除对请求数量的指数依赖。

我们相信，我们的技术也可以推广到其他图类，从而得到固定参数可处理算法；仙人掌图、一般的有界树宽图以及平面图都是值得探索的方向。此外，将该算法扩展到拨号接送问题似乎也是一个合理的研究方向。最后，在略微改进了直线开线路变体的一致性下界后，我们推测可以扩展我们的构造，并在此情形下也证明其紧性；我们将此作为未来工作。

另一个有趣的研究方向是利用欧几里得平面上经典旅行商问题的多项式时间近似方案背后的思想 [6, 7]（或者任何其他能够采用近似方案的空间），从而获得一致性保证，并使其能够随着计算时间的增加而任意接近 3/2。请注意，由于我们的平滑性证明直接源于一致性，而鲁棒性则源于求解一个经典旅行商问题实例，因此这些结果也将自动推广，只是界限会根据近似质量有所恶化。为此，一种可能的方法是将合理集合的概念扩展为 ϵ-合理集合，即保证在该集合内存在一个最优在线旅行商问题解的 $(1 + \epsilon)$-近似解。类似的放宽方式也适用于支配集这一概念。

参考文献 1 Luca Allulli、Giorgio Ausiello 和 Luigi Laura。在线车辆路径问题中前瞻的作用。在 COCOON，2005年。 2 Spyros Angelopoulos、Christoph Dürr、Shendan Jin、Shahin Kamali 和 Marc P. Renault。利用不可信建议进行在线计算。在 ITCS，2020年。 3 Antonios Antoniadis、Christian Coester、Marek Eliás、Adam Polak 和 Bertrand Simon。带有不可信预测的在线度量算法。在 ICML，2020年。 4 Antonios Antoniadis、Peyman Jabbarzade Ganje 和 Golnoosh Shahkarami。在线速度缩放的一种新型预测设定。CoRR abs/2112.03082，2021年。 5 Antonios Antoniadis、Themis Gouleakis、Pieter Kleer 和 Pavel Kolev。带有机器学习建议的秘书问题和在线匹配问题。在 NeurIPS，2020年。 6 Sanjeev Arora。欧几里得旅行商问题及其他几何问题的多项式时间近似方案。在 FOCS，1996年。 7 Sanjeev Arora。欧几里得旅行商问题及其他几何问题的近线性时间近似方案。在 FOCS，1997年。 8 Norbert Ascheuer、Martin Grötschel、Sven O. Krumke 和 Jörg Rambau。组合在线优化。载于 Peter Kall 和 Hans-Jakob Lüthi 编，《1998年运筹学研究会议论文集》，第21–37页，柏林，海德堡，1999年。Springer Berlin Heidelberg。 9 Giorgio Ausiello、Marc Demange、Luigi Laura 和 Vangelis Paschos。在线配额旅行商问题的算法。《信息处理快报》，92(2):89–94，2004年。 10 Giorgio Ausiello、Esteban Feuerstein、Stefano Leonardi、Leen Stougie 和 Maurizio Talamo。在线旅行商问题的算法。Algorithmica，29:560–581，2001年。 11 Étienne Bamas、Andreas Maggiori、Lars Rohwedder 和 Ola Svensson。通过速度缩放实现学习增强型能量最小化。在 NeurIPS，2020年。 12 Étienne Bamas、Andreas Maggiori 和 Ola Svensson。用于学习增强算法的原始对偶方法。在 NeurIPS，2020年。 13 Evripidis Bampis、Bruno Escoffier、Niklas Hahn 和 Michalis Xefteris。已知位置的在线旅行商问题。即将在 WADS 发表，2023年。CoRR abs/2210.14722。 14 Evripidis Bampis、Bruno Escoffier 和 Michalis Xefteris。带预测的加拿大旅行者问题。载于 Parinya Chalermsook 和 Bundit Laekhanukit 编，《近似算法与在线算法》，第116–133页，Cham，2022年。Springer International Publishing。 15 Richard Bellman。旅行商问题的动态规划处理。《ACM学报》，9(1):61–63，1962年1月。 16 Giulia Bernardini、Alexander Lindermayr、Alberto Marchetti-Spaccamela、Nicole Megow、Leen Stougie 和 Michelle Sweering。适用于在线图问题的输入预测通用误差度量。在 NeurIPS，2022年。 17 Antje Bjelde、Jan Hackfeld、Yann Disser、Christoph Hansknecht、Maarten Lipmann、Julie Meißner、Miriam Schlöter、Kevin Schewior 和 Leen Stougie。直线上的在线旅行商问题的紧界。《ACM算法汇刊》，17(1):3:1–3:58，2021年。 18 Michiel Blom、Sven O. Krumke、Willem E. De Paepe 和 Leen Stougie。公平对手下的在线旅行商问题。《INFORMS计算机学报》，13(2):138–148，2001年。 19 Shuchi Chawla 和 Dimitris Christou。带预测的在线时间窗旅行商问题。CoRR，abs/2304.01958，2023年。 arXiv:2304.01958, doi:10.48550/arXiv.2304.01958. 20 Paul Dütting、Silvio Lattanzi、Renato Paes Leme 和 Sergei Vassilvitskii。带建议的秘书问题。在 EC，2021年。 21 Franziska Eberle、Alexander Lindermayr、Nicole Megow、Lukas Nölke 和 Jens Schlöter。在线图探索方法的鲁棒化。《AAAI人工智能会议论文集》，36(9):9732–9740，2022年6月。 22 Sreenivas Gollapudi 和 Debmalya Panigrahi。带专家建议的租或买问题在线算法。在 ICML，2019年。 23 Themis Gouleakis、Konstantinos Lakis 和 Golnoosh Shahkarami。直线上的在线旅行商问题的学习增强算法。即将在 AAAI 发表，2023年。CoRR abs/2206.00655。

32 环、树、花及其他所有地方上的学习增强型在线旅行商问题

24 Michael Held 和 Richard M. Karp。序列问题的动态规划方法。《工业与应用数学学会学报》，10(1):196–210，1962年。25 Hsiao-Yu Hu、Hao-Ting Wei、Meng-Hsi Li、Kai-Min Chung 和 Chung-Shou Liao。带预测的在线旅行商问题。CoRR，abs/2206.15364，2022年。26 Patrick Jaillet 和 Michael R. Wagner。在线路由问题：高级信息对改进竞争比的价值。《交通科学》，40:200–210，2006年5月。27 Vinay A. Jawgal、V. N. Muralidhara 和 P. S. Srinivasan。圆上的在线旅行商问题。载于 T.V. Gopal 和 Junzo Watada（编），《计算模型理论与应用》，第325–336页，Cham，2019年。Springer International Publishing。28 Murali Kodialam 和 T. V. Lakshman。预测增强型分段路由。载于2021年IEEE第22届高性能交换与路由国际会议（HPSR），第1–6页，2021年。29 Tim Kraska、Alex Beutel、Ed H. Chi、Jeffrey Dean 和 Neoklis Polyzotis。采用机器学习索引结构的理由。载于 SIGMOD，2018年。30 Eugene L. Lawler、Jan K. Lenstra、Alexander H. G. Rinnooy Kan 和 David B. Shmoys。《旅行商问题：组合优化导论》。Wiley离散数学与优化系列，1991年。31 Alexander Lindermayr 和 Nicole Megow。Alps。 https://algorithms-with-predictions.github.io/。32 Thodoris Lykouris 和 Sergei Vassilvitskii。利用机器学习建议实现竞争性缓存。《ACM学报》，68(4):24:1–24:25，2021年。33 Michael Mitzenmacher。学习型布隆过滤器模型及其夹逼优化方法。载于 NeurIPS，2018年。34 Michael Mitzenmacher。带预测的调度及错误预测的代价。载于 ITCS，2020年。35 Michael Mitzenmacher 和 Sergei Vassilvitskii。带预测的算法。载于《算法的超越最坏情况分析》。剑桥大学出版社，2020年。36 Benjamin Moseley、Sergei Vassilvitskii、Silvio Lattanzi 和 Thomas Lavastida。通过学习权重实现在线调度。载于 SODA，2020年。37 Harilaos N. Psaraftis、Marius M. Solomon、Thomas L. Magnanti 和 Tai-Up Kim。具有释放时间的海岸线上的路由与调度。《管理科学》，36(2):212–223，1990年。38 Manish Purohit、Zoya Svitkina 和 Ravi Kumar。通过机器学习预测改进在线算法。载于 NeurIPS，2018年。39 Dhruv Rohatgi。利用机器学习建议实现近乎最优的在线缓存界限。载于 SODA，2020年。40 Shufan Wang 和 Jian Li。利用机器学习预测解决多商店滑雪租赁问题的在线算法。载于 AAMAS，2020年。41 Alexander Wei。更优且更简单的学习增强型在线缓存。载于 APPROX/RANDOM，2020年。

开放情形下直线上的一致性下界

▶**命题44**。 对于直线上的开放式 *LA-OLTSP*，没有算法能达到优于 $\frac{1+\sqrt{61}}{6}$ 的一致性。

证明。 设 $\lambda = \frac{1+\sqrt{61}}{6} \approx 1.468$，满足 $\lambda = \frac{5}{3\lambda-1}$。

设 A、O、B 是直线上坐标分别为 $x_A = -1$、$x_O = 0$ 和 $x_B = 1$ 的点。我们考虑在 A 和 B 之间存在连续请求集的情形（可以很容易地将其离散化，从而用有限个请求得到相同的渐近下界）。在 $t = 1$ 之前没有请求被释放。另记 $\delta = 5 - 3\lambda$（注意 δ 略大于 0.5）。

我们考虑 $t = 1$ 处的两种情形。令 s 表示算法在 $t = 1$ 时的位置。不失一般性，假设 $s \geq 0$。

如果 $s \geq 1 - \delta$，那么我们以单位速度释放所有请求，从 $t = 1$ 时位于 A 的位置出发，直到 $t = 3$ 时到达 B。于是 $|OPT| = 3$。如果算法在 A 之前服务 B，则至少需要 5。如果算法在 B 之前服务 A，则至少需要 $1 + (1 - \delta) + 1 + 2 = 3\lambda$。因此，竞争比至少为 λ。

否则，$0 \leq s < 1 - \delta$。然后，我们以单位速度释放来自 A 和来自 B 的请求，直到首次出现满足以下条件的时间 t_0：算法位于某个位置 s'，且 $|s'| = (2 - t_0) - \delta$；也就是说，服务器与请求的“已发布部分”之间的距离为 δ。根据 t_0，我们区分两种情况。不失一般性，假设 $s' \geq 0$ 在 t_0，因此 $s' = (2 - t_0) - \delta$。令 C 和 D 分别为位于 $x_C = -(2 - t_0)$ 和 $x_D = 2 - t_0$ 的位置上的点（也就是说，在 t_0，请求在 $[A, C]$ 和 $[D, B]$ 上释放）。

- 如果 $t_0 \geq 3\lambda - 3$（直观地说，略少于一半的请求已被释放，且 ALG 接近 O），那么我们从 C 到 D 以单位速度释放请求，从 t_0 开始。然后再次 $|OPT| = 3$。如果 ALG 在 B 之前服务 A，那么与之前一样，它需要 $t_0 + (2 - t_0 - \delta) + 3 = 5 - \delta = 3\lambda$。如果 ALG 在 A 之前服务 B，它需要 $t_0 + (1 - (2 - t_0 - \delta)) + 2 = 2t_0 + \delta + 1 = 2t_0 + 6 - 3\lambda \geq 3\lambda$。竞争比至少为 λ。

- 最后，如果 $t_0 \leq 3\lambda - 3 = 2 - \delta$（直观地说，ALG 与原点并不“太接近”）：我们在 t_0 释放除 D 中的请求以外的所有请求。然后直观地说，ALG 要么朝着 A 前进，我们将像之前一样行动；要么朝着 B 前进，我们将在 D 阻止释放。更形式化地，在 $t = t_0 + 1$：如果算法位于位置 $s'' < 0$，那么我们释放最后一个请求（位于 D）。注意，ALG 不可能已经服务了 A（因为我们在 t_0 有 $s' > 0$），也不可能服务了 B（没有时间在 $t_0 + 1$ 前往 $s'' < 0$）。于是，ALG 在 $t = t_0 + 1$ 之后能做的最好选择是前往 A（因为 $s'' < 0$），然后前往 B。由于我们在 t_0 有 $s' = (2 - t_0) - \delta$，因此 ALG $\geq t_0 + (2 - t_0) - \delta + 3 = 5 - \delta = 3\lambda$，而 $|OPT| = 3$，所以竞争比至少为 λ。如果 $s'' \geq 0$，我们在 $t_1 = 3 + (1 - x_D) = 2 + t_0$。那么 $|OPT| = t_1$（在时间 1 位于 A，然后前往 B（时间 3），再返回 D）。如果 ALG 在 D 之前服务 A，由于 $s'' \geq 0$ 在 $t_0 + 1$，它至少需要 $t_0 + 1 + 2 + x_D = 5$。如果它在 A 之前服务 D，由于 D 在 t_1 释放，它至少需要 $t_1 + x_D + 1 = 5$。因此，根据 λ 的定义，竞争比至少为 $\frac{5}{t_1} = \frac{5}{t_0+2} \geq \frac{5}{3\lambda-1} = \lambda$。

◀