廣東工業大學

QG 最终考核详细报告书

题　　目　文献《Sequential Trajectory Data Publishing with Adaptive grid-based Weighted Differential Privacy》的仿真复现

学　　院　计算机学院

专　　业　计算机类

年级班别　24 级 1 班

学　　号　3124004052

学生姓名　陈英锐

2025 年 4 月 24 日

目录

一、 引言

1.1 研究背景及目的

1.1.1 研究目的

随着无线通信与定位技术的快速发展，轨迹数据（如出租车轨迹、用户移动路径）的收集变得日益便捷，这些数据在智慧城市、交通规划、个性化推荐等领域具有重要价值。然而，轨迹数据中常包含用户敏感信息（如家庭住址、工作地点），直接发布原始数据可能导致隐私泄露。如何在保证数据隐私的前提下，发布高可用性的轨迹数据，成为当前研究的核心挑战。

本研究旨在提出一种新型的自适应网格加权差分隐私（AWDP）模型，解决现有差分隐私轨迹发布方法中存在的两大问题：

1、 区域分布不均导致的噪声分配不合理：传统方法采用固定网格结构，无法适应轨迹密度的空间异质性，导致稀疏区域噪声过大或密集区域隐私保护不足。

2、 时空连续性破坏：现有方法在合成轨迹时忽略方向与密度的连续性，生成轨迹存在不合理的跳跃或扭曲，降低数据实用性。

1.1.2 研究背景

1、轨迹数据包含高度敏感信息，攻击者可通过背景知识（如居住地、通勤路线）重新识别个体身份。传统匿名化方法（如 k-匿名）无法抵御此类攻击。

2、差分隐私（DP）作为严格的数学隐私框架，能抵抗背景知识攻击，但直接应用于轨迹数据会因噪声添加过多而显著降低数据效用。

3、固定网格结构：如 DP-Star 采用两级网格，无法适应真实轨迹的密度差异，导致稀疏区域信息丢失或密集区域噪声不足。

4、连续性缺失：合成轨迹的起点、中间点和终点独立生成，导致方向突变或密度异常，影响轨迹的真实性。

1.1.3 主要贡献

1、 多分辨率自适应网格结构：根据轨迹密度动态划分网格，高密度区域采用细粒度网格，低密度区域合并为粗粒度网格，使噪声分配更合理，提升数据效用。

2、 区域加权差分隐私机制：提出密度相关的隐私预算分配：高密度区域分配更多预算以减少噪声影响，稀疏区域分配较少预算以增强隐私保护；通过加权矩阵实现预算的动态调整，显著提升隐私预算的利用率。

3、 时空连续性维护方法：方向修正，检测并修正合成轨迹中不合理的角度偏离（如急转弯），通过映射或删除异常点保证方向连续性。密度修正，基于网格密度过滤离群点，确保轨迹通过热点区域（如商业区）的合理性。

二、 项目正文

2.1 理论学习

2.1.1 轨迹序列数据

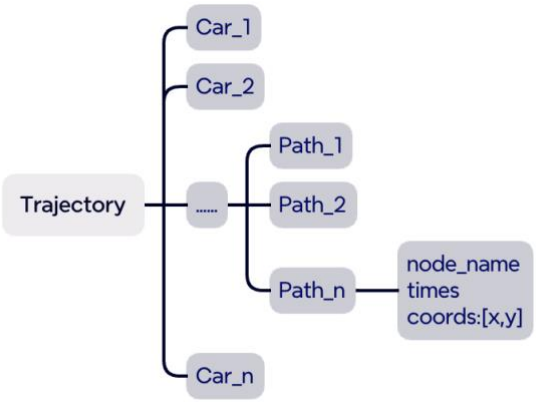
定义 1. 轨迹序列 T 是一个按照时间顺序排列的位置序列，可以表示为：

$$T = l_1 l_2 \cdots l_{|T|}$$

其中l可以表示为一个位置，由轨迹点的经度和纬度的二维坐标组成，|T|表示的是轨迹 T 的长度，在轨迹序列中，我们可以用l_1和$l_{|T|}$表示某一条轨迹的起点和终点；而轨迹数据集 D 可以看作是由|D|条轨迹序列构成的集合，可以表示为：

$$D = \{T_1, T_2, \cdots, T_{|D|}\}$$

其中|D|表示轨迹的数量。



我们举一个简单的例子来理解轨迹序列的定义，上面的一段 json 数据展示的是广州出租车数据的其中一条轨迹数据，结合轨迹序列的定义，我们可以用$l_1 = (26.875, 29.947)$和$l_{|T|} = (44.925, 8.938)$分别表示这条轨迹的起点和终点，在后续的应用中，我们可以循环读取 json 数据从而构建轨迹序列和轨迹数据集。

2.1.2 差分隐私介绍

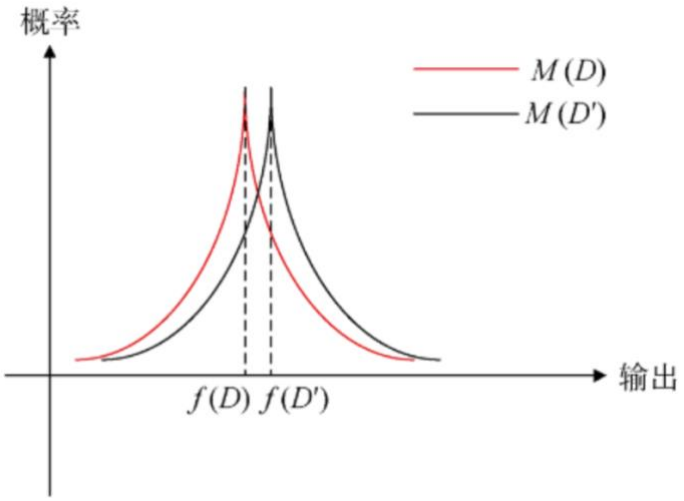
差分隐私是一种严格的隐私保护机制，它保证在两个仅相差一条记录的数据库上进行查询会得到非常相似的结果。

定义 2. 非交互式随机化机制 M 提供ϵ-差分隐私，如果对于任意两个最多相差一条记录的数据库D_1和D_2，以及对于任意输出$\tilde{D} \in Range(M)$，满足

$$Pr[M(D_1) = \tilde{D}] \leq \exp(\epsilon) \times Pr[M(D_2) = \tilde{D}]$$

其中ϵ为正数，数值越小，隐私保护程度越高。

那么我们同样举一个简单的例子来理解差分隐私的概念：



图表 1 相邻数据集的概率分布

如图，D和D'是一对相邻数据集，通过算法 M 将其映射到某一个值域上，图中是这对相邻数据集的概率分布，当我们调整参数ϵ时，可以改变这对相邻数据集在概率分布上的接近程度，ϵ越接近 0，两者越靠近，输出的结果越不可区分，隐私保护程度越高，但是数据的可用性越低，也就是说ϵ越接近 0，数据收集者越难分辨两组数据集，也就无法推敲某一组数据集的内容。

2.2 数据结构设计

为了复现文献的内容，程序语言并没有直接提供现成的数据结构可用，我们需要根据文献提供的信息自己设计一套可行、实用性较强的数据结构，这是本次复现实验的基础步骤。下面将从两个最主要的数据结构进行分析和设计。

2.2.1 轨迹数据集结构设计

轨迹数据集的设计我们可以由浅入深，逐一分析，根据轨迹序列的定义，我们知道轨迹点l所表示的位置可以由其经度和纬度构成的二维坐标表示，此外，为了后续可视化的方便，我们可以在二维坐标的基础上添加时间戳参数，于是，一个轨迹点的数据结构可以用一个元组(Tuple)或者列表(List)表示，即：

$$l = (x, y, TimeStamp) 或 l = [x, y, TimeStamp]$$

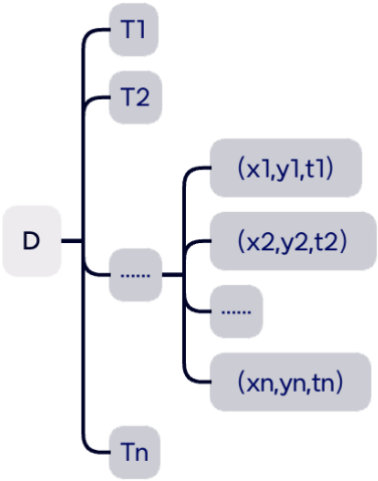
单条轨迹由多个轨迹点组成，我们可以将每个轨迹点存入一个列表，或者存入一个类型为 object 对象的 numpy 数组，就得到了单条轨迹的数据结构：

$$T = [[x\_1, y\_1, TimeStamp\_1], \cdots, [x_n, y_n, TimeStamp_n]]$$

那么轨迹数据集的构建也类似于单条轨迹的构建，将每条轨迹存入列表或者 numpy 数组就得到了轨迹数据集 D：

$$D = [T_1, T_2, \cdots, T_n]$$

根据我们设计的轨迹数据集数据结构，我们可以得到对应的示意图，实际上轨迹数据集是一个多维度的数组，通过合适的下标我们就可以访问数组获取我们所需要的信息。



图表 2 轨迹数据集结构示意图

2.2.2 网格结构设计

考虑到网络结构在实验过程中会根据数据点调整变化，我们需要选择一个能够方便我们及时对变化及时处理的数据结构，于是我们利用树结构来设计网格。

首先我们对树的节点类进行定义，也就是单个网格类，我们为其设置了四个属性：网格边界(bounds)、网格分辨率(resolution)、子网格列表(children)、网格内轨迹点(trajectory_points)。基于树结构的网格结构设计能够方便我们遍历网格以及对网格进行分割：

网格遍历：通过递归函数，逐一遍历每一个网格的子节点，最终获取整个网格的信息，在后续的处理中我们可以遍历网络获取分辨率向量，进行其他一系列操作等；

网格分割：输入参数网格划分大小 d，计算横向和纵向的步长，生成新的节点类，循环添加到当前节点的孩子节点中，实现对网格的分割。

基于此，我们为网格结构设计了三个主要的方法和其余辅助方法，分别是网格划分(self.split)、获取网格所有叶子节点(self.get_all_leaf_cells)、映射点到网格(self.map_point_to_cell)。

下图展示的是网格的分割示意图，如果符合某一条件就对当前节点执行分割函数，根据输入的参数 d 将网格平均分成 d 份，这样我们就可以得到一个多分辨率的网络

图表 3 网格结构分割示意图

获取网格所有叶子节点类似于树的递归遍历，最后函数返回一个 object 列表，包含网络的所有节点类，在实验中可以用于对图的边界等性质进行分析。

而映射函数可以快速将轨迹的坐标点映射到网络之中，这个功能在复现实验中起到关键作用，可以让点的实际坐标和网格结构之间实现快速转换。

2.2.3 数据结构优劣势分析

（1）轨迹数据集结构

优势：多维的 numpy 数组具有高效的内存管理和计算性能，适合大规模的轨迹数据处理；加入时间戳参数可以方便后续的复现实验分析；代码结构符合直觉，简单容易想到。

劣势：固定维度的数组在轨迹发生变化时难以进行快速的处理。

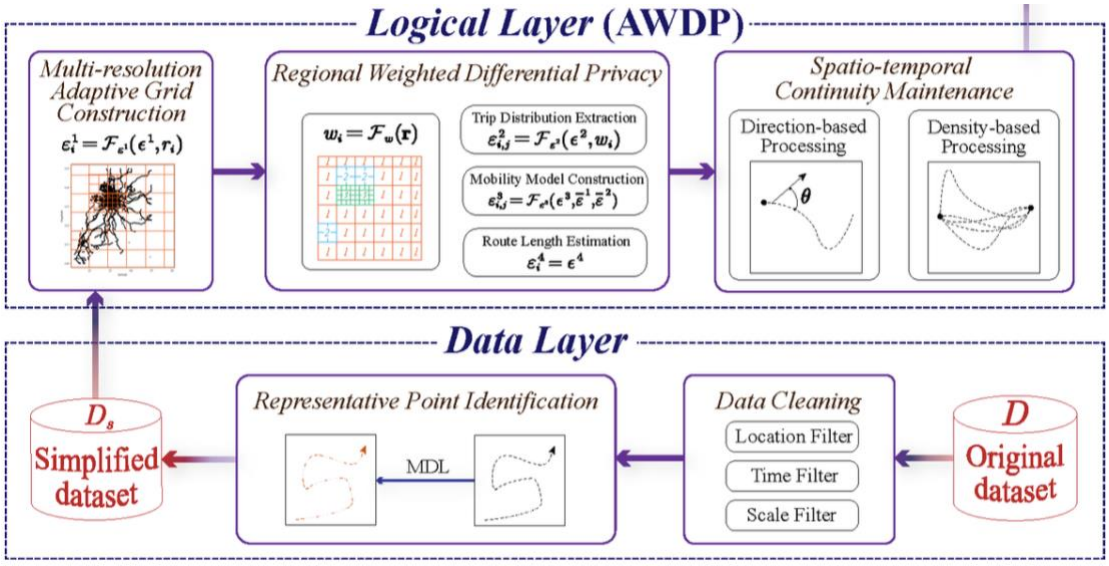
（2）网格结构

优势：树结构的设计能够实现动态的多分辨率划分，可以自适应不同密度的轨迹分布，结构比较灵活；通过递归遍历可以简化复杂网络的分析和操作；映射方法的设计让点的实际坐标与网格之间构建联系，在复现实验中可以高效处理点与网格之间的关系。

劣势：当分辨率较高时，网格的内存消耗较大；部分网格在稀疏区域可能变成空网格导致部分空间的浪费。

2.3 算法设计

AWDP 算法的设计，自然是本次复现实验的核心，算法的设计路线基本按照文献所提供的顺序，依次是：数据预处理、多分辨率自适应网络的构建、区域加权差分隐私、时空连续性维护。关键的算法流程也可以参考文献框架图中的 Logical Layer 逻辑层。



图表 4 关键算法流程图(参考文献)

2.3.1 数据预处理

数据预处理包括三个主要步骤：时空筛选、数据采样、MDL 轨迹简化

（1） 时空筛选：函数接受时间和空间经纬度范围参数，遍历轨迹数据集，对于不在范围内的数据进行删除，最后返回用户所需要的轨迹数据集；

（2） 数据采样：函数接受参数 sample_ratio，从数据集中随机筛选部分数据用于后续的处理，采样百分比可以根据数据集的数量自行调节，例如在本次复现实验所提供的数据

（3） MDL 轨迹简化：MDL 轨迹简化部分采用递归算法实现轨迹压缩，通过计算点到线段垂直距离来保留关键点。算法首先保留轨迹起点和终点，然后递归寻找偏离当前线段最大的点，若偏离距离超过阈值 epsilon 则保留该点作为分割点。最终按原始顺序连接所有关键点形成简化轨迹，在保证形状精度的同时有效减少数据量。对于短于 3 个点的轨迹直接返回不作处理。整个过程通过 epsilon 参数控制简化力度，实现精度与效率的平衡。

2.3.2 构建多分辨率自适应网络

构建多分辨率的自适应网络算法设计我们可以直接依照文献提供的思路进行代码设计，首先需要明确的是函数的输入输出：

表格 1 MultiResolutionAg **函数** IO **设计**

输入	预处理后的简化数据集 D、隐私预算参数ϵ_1、网格分割区间 d，分割阀值ρ_0
输出	处理后的数据集 D、图结构 G、分辨率向量 r、预算矩阵 E1、补充预算矩阵 E1’

下面我们结合函数输入，结合文献设计算法得到函数的输出：

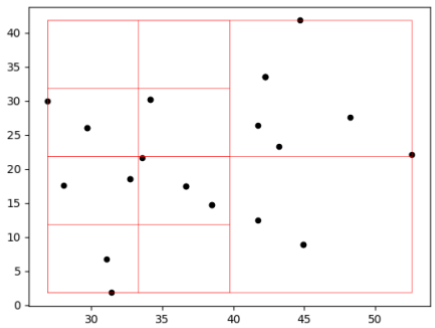
(1) 参数h_{max}计算：$h_{max} = \left\lfloor \frac{1}{2} \times log_d |D_s| \right\rfloor$，该参数用于确认网络划分的最深深度；

(2) 隐私预算ϵ_r计算，用于噪声的添加，$\epsilon_r = \frac{\epsilon_1}{h_{max}}$；

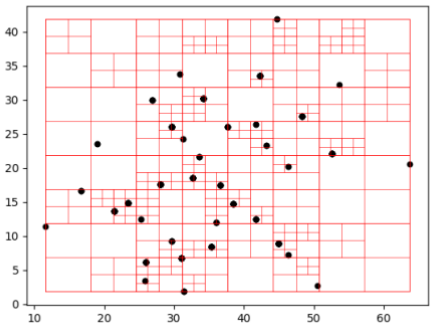
(3) 如果当前深度$h < h_{max}$：计算噪声参数n_i，和预设值ρ_0进行对比并递归划分网络；

(4) 根据分割后的网络 G，计算隐私分配矩阵 E1 和隐私补充矩阵 E1’ 以及返回值 r；

(5) 利用网格类的映射函数将数据集中的点和网络中的网格相对应。



图表 5 **单量出租车自适应网格示意图**



图表 6 所有轨迹点及对应自适应网络示意图

如图 5 所示的是某一辆出租车的轨迹点图以及对应的网格划分示意图，可以看到，在轨迹点比较密集的地方，网格划分的更加密集；这一点在图 6 中可以更加明显的看到，此外，网格的划分还受到预设值ρ_0的影响，当该值越小时，网格更容易被分割，所以要得到较好的实验数据，我们需要设定合适的预设值，才能使网络的划分符合预期，在本次的复现实验中，我们的预设值为默认值 2。

2.3.3 区域加权差分隐私

完成网络结构的构建之后，我们根据上一个函数的输出，结合文献的代码框架，进行区域加权差分隐私的 IO 设计：

表格 2WeightDP 函数 IO 设计

输入	处理后的数据集 D、网络结构 G、分辨率向量 r、补充预算矩阵 E1、隐私预算参数
输出	处理后的数据集 D，网络索引表 g_dict

在设计函数的输出时，这里我们添加了一个新的返回值 g_dict，即网络索引表，这样就避免了在后续计算时对网络的重复搜索，降低程序的运行效率，我们可以直接根据这张索引表定位到目标网络位置。

该部分由三大子部分组成：行程分布提取（分配预算ϵ_2）、移动模型构建（分配预算ϵ_3）、路线长度估计（分配预算ϵ_4），最后结合前面的三个结果生成合成轨迹。下面是算法的具体步骤：

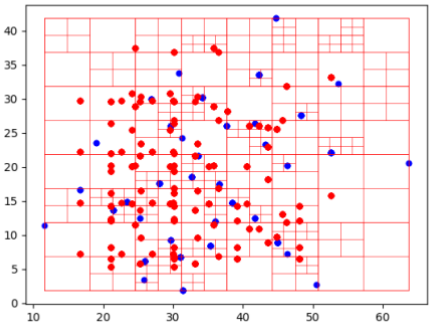
(1) 权重初始化：$w_i = \frac{r_i}{\sum r}$；遍历网络 G，收集所有的叶子节点并且建立索引表 grid_indices；

(2) 行程分布提取（分配预算ϵ_2）：根据文献提供的公式构建预算矩阵，并且矩阵的参数满足条件$\varepsilon_{i,j}^2 = min(\varepsilon_{i,i}^2, \varepsilon_{j,j}^2), \varepsilon^{2\prime} = \varepsilon^2 - \varepsilon_{i,j}^2$，提取行程分布，根据得到的隐私预算矩阵为行程分布添加噪声；

(3) 移动模型构建（分配预算ϵ_3）：利用马尔科夫模型构建移动模型，并且根据隐私预算参数添加噪声，其中隐私预算的参数满足：$\epsilon_{i,j}^3 = \epsilon_{i,j}^{1\prime} + \epsilon_{i,j}^{2\prime} + \epsilon_3$；

(4) 路线长度估计（分配预算ϵ_4）：对每一条轨迹长度都平均分配隐私预算 ε^4，然后对路线长度数据添加噪声；

(5) 合成轨迹生成：根据添加噪声后的行程分布矩阵采样起止点，在根据移动模型从起点到终点随机游走，轨迹长度由路线长度向量中随机抽取，将每一条这样的轨迹 T 存入 D 得到最终的合成轨迹数据集。



图表 7 Dr 数据集可视化效果图

图 7 是对处理后的数据集 D 的可视化，蓝色散点代表的是原始数据点，红色散点代表的是添加噪声后的数据点，为了可视化方便，我们在选取散点时对 D 中的每个网格边界取中点作为最后展示的轨迹点，可以发现添加噪声后的轨迹点分布基本上和原始轨迹点保持一致，并且满足差分隐私条件，在实际应用场景中可以对数据进行保护。

2.3.4 时空连续性维护

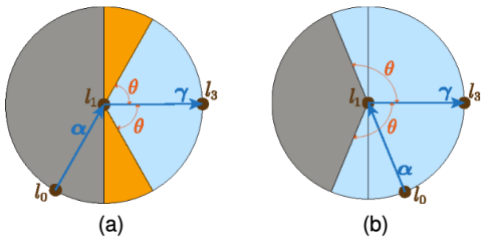
同样按照前面的思路，我们首先要设计好函数的 IO，才能明确我们的目标，编写合理的算法

表格 3ContinuityMaintenance 函数 IO 设计

输入	合成轨迹数据集 D、网格大小 g
输出	处理完成的数据集 D

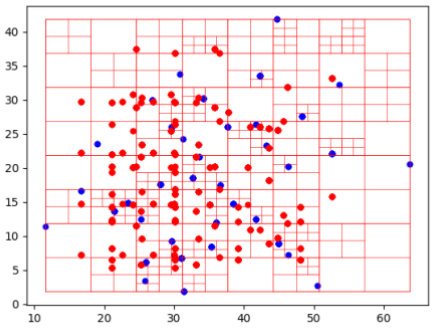
时空连续性的维护包括两大模块：方向性处理和密度修正，具体的步骤如下：

(1) 方向性处理：获取路径的最后四个轨迹点，计算向量和角度，然后根据文献的方法，将异常的轨迹点移除



图表 8 方向修正示意图（参考文献）

(2) 密度修正：首先统计每条轨迹的起点-终点组合出现的次数，并记录每个中间点在这些组合中的出现频率。然后对于每条轨迹，只保留那些出现频率不低于该轨迹起点-终点组合总次数的中间点。这样能确保保留的中间点都是在该路径上频繁出现的合理位置点，而过滤掉那些可能是噪声或异常的低频点。最终每条轨迹保持原起点和终点，中间只保留高频通过的点，从而在保持轨迹整体走向的同时提高数据的合理性。



图表 9 经过时空维护后的轨迹点图

类似于图 7 中对加噪后的点的处理方法，图 9 中的红色散点采用了同样的处理方法方便可视化，图中展示的就是经过时空修正后的轨迹散点图，仍然是符合预期的，并且满足差分隐私预算，同样能够在实际应用中保护轨迹的隐私。

2.3.5 主函数设计

经过前面三个主要函数的铺垫，主函数的设计就比较简单，回顾图 4，主函数的任务实际上就是 LogicLayer 的实现，将数据集依次输入三个重要函数，我们就得到了预期结果，具体步骤就很显然了：

(1) D= PreProcess (D)

(2) D= MultiResolutionAg(D)

(3) D= WeightedDp(D)

(4) D= ContinuityMaintenance(D)

2.3.6 模型评估算法设计

（一）RE

1.随机采样区域：在原始轨迹数据覆盖的空间范围内，随机选择一个矩形区域（由起点和终点坐标确定），并在该区域内随机生成一个中心点；

2. 半径覆盖统计：以中心点为圆心、给定半径（如 0.2）画圆，统计原始轨迹数据（original_T）和合成轨迹数据（synthetic_T）中至少有一个点落在圆内的轨迹数量，分别记为 count_d 和 count_sd；

3.计算基准值：设定基准值 b 为原始轨迹总数的 0.1%（防止分母为零）

4.误差计算：通过公式$R = \frac{|count_sd - count_d|}{max(count_d, b)}$计算相对误差，反映合成数据与原始数据在空间覆盖上的差异。

（二）FPS

1.空间网格划分：将轨迹覆盖的空间划分为固定数量（如 6×6）的网格；

2.轨迹网格化：将每条轨迹的坐标转换为网格编号序列，并过滤连续重复的网格编号；

3. 频繁模式提取：从原始和合成轨迹中分别提取长度在 min_pattern_length（如 3）到 10 之间的子序列模式，统计每种模式的频次；

4. 模式匹配与误差计算：选取原始数据中 Top-K（如 50）的频繁模式，计算合成数据中对应模式的频次相对误差，取平均得到 FPS 值，值越小相似度越高。

（三）KT

1. 共同模式筛选：基于 FPS 步骤中提取的原始和合成数据的频繁模式，筛选两者共有的模式；

2. 模式频次排序：对共有模式按原始数据中的频次排序，并记录合成数据中的对应频次；

3. 一致性检验：比较所有模式对的频次大小关系：若原始和合成数据中模式 A 和 B 的频次高低关系一致，则为一致对（concordant）；否则为不一致对（discordant）；

4. 计算 KT 系数：通过公式$K = \frac{concordant - discordant}{valid_pairs}$计算肯德尔相关系数，值越接近 1 表示排序一致性越高。

（四）DE

1. 空间网格划分：与 FPS 类似，将空间划分为网格（如 6×6）；

2. 网格分布统计：统计原始和合成轨迹中每条轨迹访问的唯一网格的频次，并归一化为概率分布；

3. JS 散度计算：将两分布的网格概率向量输入 JS 散度公式$JS(P||Q) = \frac{1}{2}[KL(P||M) + KL(Q||M)]$，得到 DE 值，衡量分布差异，值越小表示分布越接近。

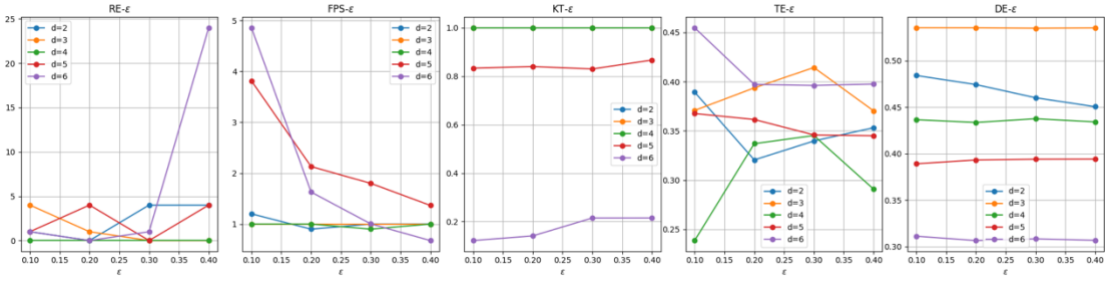
（五）TE

1. 起止点网格划分：将空间划分为较粗的网格（如 2×2），记录每条轨迹的起点和终点所在网格编号；

2. 起止点对统计：统计原始和合成数据中所有可能的起止点网格组合（如 4×4=16 种）的频次，并归一化为概率分布；

3. JS 散度计算：通过 JS 散度比较两分布的差异，得到 TE 值，反映起止点分布的相似性，值越小说明合成数据保留原始起止模式的效果越好。

综合上述五点模型评估，结合复现实验的数据得到以下五组指标与差分隐私参数的关系图：



图表 10 各评估指标与差分隐私参数关系图

2.4 结果复现与分析

【注】复现结果参考图 10

（1） 首先观察 RE、FPS、TE 指标图像，发现这三个指标在 d=2 时都达到最低，而对于 KT 指标，在 d 逐步增加时，KT 值越来越低，也就是随着 d 的增加前后的一致性越来越低。

（2） d=2 的指标，随着差分隐私预算增加，各指标大致呈现稳定的趋势，只有轻微的波动，这也说明了 d=2 是自适应结构的最优选择，同时这也文献给出的结果基本一致。

2.5 项目总结与反思

（1）本次复现实验较上次中期考核既有优点又有缺点；

（2）时间分配问题：前面有很多时间花在一些小问题代码的调试上导致最后整体时间分配不均匀，从而导致部分任务没有完成，并且在后期文案的编写上没有给足充裕的时间；

（3）代码模块化：本次实验代码较以前的代码做了模块化的处理，每个功能都分模块编写，在后期对代码错误进行调试的时候比较方便，部分出现严重错误的代码也可以将对应的部分直接重写，不会影响整体程序的运行；

（4）代码鲁棒性：这是在程序调试过程中逐步发现的，在调试程序的时候，有时因为更换函数参数，随机种子，轨迹数据等都会导致程序异常的终止，这也是在最终调试阶段逐步完善函数的代码，让代码最终足以对抗各类数据，不会因为修改参数而报错；

（5）论文的总结与归纳：结合上次的文献阅读经历，弥补上次的不足，这次在阅读文献的时候边阅读边记录重要公式，整理文章大体思路，在读完整篇论文以后有一个比较明确的方向，而且也可以比较快的定位到文章的段落，不会像第一次阅读时一样读完一遍之后满头雾水。

三、 QG 训练营心得总结

（1） 从寒假开始加入 QG 训练营第一次 Push 文件到 GitHub 至今大概已经过去三个月时间，本学期也基本上过半，训练营的三～四个月对我来说影响还是很深刻的！在代码编写能力上也有很大的提升，具体我认为有以下几点：

1. 学会模块化管理代码，对于大型代码的调试可以更加从容的应对。
2. 学习了部分代码规范，现在对比以前编写的代码，代码规范上也有很大的提升，代码可读性更强，也记住了代码添加注释的重要性！这同样也是适用于各类开发场景的，代码规范对于多人协作开发、团队合作来说是相当重要的一点内容。
3. 代码的错误处理能力，经常因为几个参数的调整或是在其他地方调用函数的时候导致函数报错，不能正常使用，但是在几轮代码编写之后学会了对各类情况都能进行正确处理，对于各种情况的输入都能够正确应对，让函数输出用户所需要的内容。

（2） 学习了问题的解决，包括问题的各种搜索渠道，了解各种开源代码的来源，一些公开的数据集等等各种网上资源，视野比以前更加广阔，相较于以前可以独立处理更加复杂的问题了。

（3） 学会对大量的文件进行分类整理，学会了记录学习笔记，会记录在学习过程中重要的点，并将它们分类和归纳到文件夹中，再翻阅查看的时候拥有自己的知识数据库，对于想要检索的知识很快就能在自己整理的文件夹中找到。

（4） 对于部分事情的处理上更加严谨，最终呈现的效果也更加完美，例如文档的编写，文件夹的排列与文件的规范格式，都尽量按照标准完成，其实也隐隐之中让我学会了用 word 这个看似很简单的软件。

（5） 在相关领域的学习更是值得一提的，无论是大组训练的基础数据结构还是小组任务的 AI 领域的相关知识，我的基础代码能力都能够提升，并且手推代码（故事要从手推多元线性回归说起）之类的烂活（bushi）让我在数学推导方面，数学知识方面也有很大的提升，对于很多未知的数学专业知识都是现学现用的，在每次学习的积累中逐步构建起属于自己的数学体系，此外也初步学习了很多模型，它们在不同的领域都有各自的优势，并且在训练营中学习的 LSTM 模型让我在参加的全国大学生统计建模大赛中用上，顺利的完成了比赛，很多训练营的基础知识经过拓展都可以应用到现实生活当中的！

（6） 印象最深刻的一定是论文的复现实验了，从第一次接触时的手足无措到现在有自己一定的阅读方法，中间有很多的挑战与成长！这让我了解到当下最前沿的研究问题，无限拓宽我的眼界，并且提升我的抗压能力，无数个为论文熬的夜晚怎么不是成长呢？

图表 11 2025/4/25ricckker 为 QG 的第不知道第几个凌晨

附录

4.1 参考文献

[1] Y. Giyn, Y. Zhang, and Y. Li, "Sequential Trajectory Data Publishing With Adaptive Grid-Based Weighted Differential Privacy," IEEE Transactions on Intelligent Transportation Systems, vol. 25, no. 12, pp. 9249-9262, 2024.