An Efficient and Accurate Grid Compression Solution for Point-to-Line Nearest Neighbor Search

Hongwei Ye
Tsz Nam Chan*
Dingming Wu
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
yehongwei2023@email.szu.edu.cn
{edisonchan,dingming}@szu.edu.cn

Leong Hou U
Department of Computer and Information Science, University of Macau
Macao, China
ryanlhu@um.edu.mo

Ruisheng Wang
School of Architecture and Urban Planning, Shenzhen University
Shenzhen, China
ruiswang@szu.edu.cn

Abstract

Point-to-Line Nearest Neighbor Search (PLNNS) aims to project each location data point to the nearest line (or road), which is a prior step for many commonly used road network analysis tools. Therefore, PLNNS has been adopted in many application domains, e.g., transportation science, crime science, and urban planning. However, PLNNS is computationally expensive, which acts as the main bottleneck in a road network analysis pipeline. To address the efficiency issue of PLNNS, we first develop the approximate grid compression method, called $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$, which utilizes a small number of grid cells to represent all data points for efficiently solving the PLNNS problem with a non-trivial approximation guarantee. In order to ensure the exact correctness in road network analysis, we then develop $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$, which further achieves the exact result of solving the PLNNS problem. Our experiment results on four large-scale datasets show that $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ achieve 7.72x to 524.3x and 5.64x to 18.76x speedups, respectively, compared with the state-of-the-art solution. The implementation of all methods can be found in https://github.com/fastnsa/grid-comp-plnns.

CCS Concepts

• **Information systems** → **Geographic information systems**.

Keywords

PLNNS; grid compression; exact; approximation

ACM Reference Format:
Hongwei Ye, Tsz Nam Chan, Dingming Wu, Leong Hou U, and Ruisheng Wang. 2026. An Efficient and Accurate Grid Compression Solution for Point-to-Line Nearest Neighbor Search. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26), August 09–13, 2026, Jeju Island, Republic of Korea.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3770855.3817711

*Corresponding author.

KDD '26, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817711

1 Introduction

Road-network-based spatial analysis has been extensively used in many application domains, including transportation science [27, 33, 39], crime science [16, 44, 49], and urban planning [46, 59, 64]. However, unlike traditional spatial analysis (using the Euclidean distance), each data point must be projected to the correct line (or the correct edge) of a road network in advance before we can adopt those network analysis tools (e.g., network histogram [44], network kernel density visualization [21, 23, 44], and network K-function [22]). Figure 1 shows an example for projecting those data points in the Chicago 311-call dataset [2] to the corresponding road network.

(a) Original data points (b) Projected data points

Figure 1: Projecting those data points in the Chicago 311-call dataset to the corresponding road network.

Therefore, many GIS software packages, e.g., QGIS [9], ArcGIS [1], PostGIS [8], OSMnx [19], and spNetwork [11], offer the point-to-line nearest neighbor search (PLNNS) functionality for this projection task. Using Figure 2 as an example, $\mathbf{p}_1$, $\mathbf{p}_2$, and $\mathbf{p}_3$ are projected to the nearest lines l_1 (with the red data point), l_2 (with the purple data point), and l_1 (with the blue data point), respectively.

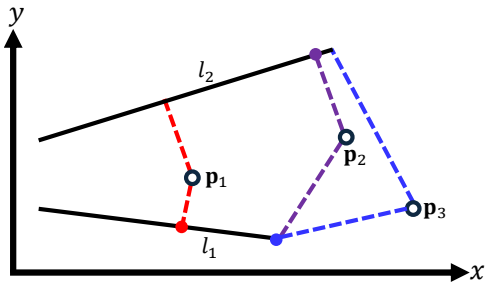


Figure 2: Illustration of point-to-line nearest neighbor search (PLNNS) and the projection of each data point.

However, one major drawback for using PLNNS is that this functionality is inefficient. Consider a set of n location data points and L lines. The time complexity of PLNNS is $O(nL)$. Using the Chicago crime location dataset [3] (with 8,189,459 data points and 41,222 lines) as an example, the naïve approach for solving PLNNS takes at least 337.59 billion operations (8,189,459 × 41,222), which cannot be feasible to handle this dataset. Although many index structures, e.g., R-tree [34], have been adopted for indexing those lines in order to improve the efficiency for solving PLNNS, these approaches independently process each data point, which still suffers from high response time (especially when n is very large).

In this paper, we ask a question. *Can we develop the efficient and accurate solution for handling PLNNS?* To provide an affirmative answer to this question, we have proposed the pioneering grid compression solution, called $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$, which represents a group of data points for each grid cell by its center, so that it can simultaneously (1) reduce accessing the state-of-the-art index structure and (2) achieve non-trivial accuracy guarantees. To further obtain the exact result for solving the PLNNS problem, we have proposed $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$, by investigating the candidate search space. Our experiment results on four large-scale datasets demonstrate that $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ achieve 5.64x to 524.3x speedups compared with the state-of-the-art solutions, without significant accuracy degradation. As a remark, these two solutions have been incorporated into the python library, called FastNSA [6] (released on 24$^{\text{th}}$ January 2026), which has received 1,280 downloads (until 22$^{\text{nd}}$ May 2026).

The rest of the paper is organized as follows. We first formally define the PLNNS problem and discuss the hierarchical indexing framework in Section 2. Then, we present $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ in Section 3 and Section 4, respectively. Next, we discuss our experiment results in Section 5, After that, we provide the literature review in Section 6. Lastly, we conclude this paper in Section 7. Appendix can be found in Section 8.

2 Preliminaries

In this section, we first formally define the PLNNS problem in Section 2.1. Then, we discuss the hierarchical indexing framework for solving this problem in Section 2.2.

2.1 Problem Definition

We first formally define each line l in Definition 1.

Definition 1. *A line l can be represented by two data points $\mathbf{s}$ and $\mathbf{e}$, which is denoted as $l = [\mathbf{s}, \mathbf{e}]$.*

With Definition 1, there are three cases for calculating the distance between a data point $\mathbf{p}$ to each line l (see Figure 3), which is the distance between $\mathbf{p}$ and its nearest position on l. We then define the point-to-line distance in Definition 2.

Definition 2. *Given a data point $\mathbf{p}$ and a line $l = [\mathbf{s}, \mathbf{e}]$, the point-to-line distance is denoted as $d(\mathbf{p}, l)$.*

$$d(\mathbf{p}, l) = \begin{cases} ||\mathbf{p} - \mathbf{s}|| & \text{if } C(\mathbf{p}, l) < 0 \\ \frac{(\mathbf{p}-\mathbf{s})^T(\mathbf{e}-\mathbf{s})}{||\mathbf{e}-\mathbf{s}||} & \text{if } 0 \le C(\mathbf{p}, l) \le 1 \\ ||\mathbf{p} - \mathbf{e}|| & \text{if } C(\mathbf{p}, l) > 1 \end{cases} \quad (1)$$

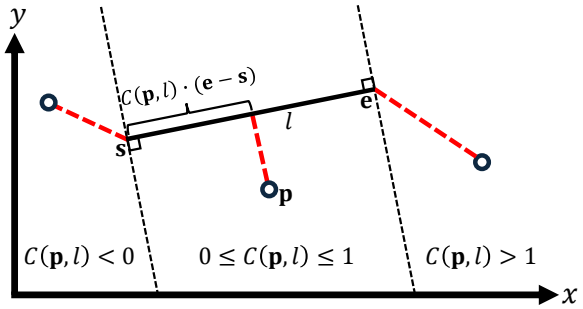


Figure 3: Computation of the point-to-line distance.

where $||\cdot||$ is the Euclidean norm and $C(\mathbf{p}, l)$ is defined as follows.

$$C(\mathbf{p}, l) = \frac{(\mathbf{p} - \mathbf{s})^T(\mathbf{e} - \mathbf{s})}{||\mathbf{e} - \mathbf{s}||^2} \quad (2)$$

As a remark, the term $C(\mathbf{p}, l)$ is obtained based on the concept of orthogonality (from linear algebra [14]).

$$[C(\mathbf{p}, l) \cdot (\mathbf{e} - \mathbf{s})]^T[\mathbf{p} - (\mathbf{s} + C(\mathbf{p}, l) \cdot (\mathbf{e} - \mathbf{s}))] = 0$$

Based on the point-to-line distance $d(\mathbf{p}, l)$ (see Definition 2), we formally state PLNNS in Problem 1.

Problem 1. *Given a set of n data points $\mathbb{P} = \{\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n\}$ and a set of L lines $\mathbb{L} = \{l_1, l_2, ..., l_L\}$, we need to find, for every data point $\mathbf{p}$ in $\mathbb{P}$, (1) the nearest line $l^* = [\mathbf{s}^*, \mathbf{e}^*]$ so that $d(\mathbf{p}, l^*) \le d(\mathbf{p}, l)$ for all l in $\mathbb{L}$ and (2) the projected data point (on l^*) $\mathbf{s}^* + C(\mathbf{p}, l^*) \cdot (\mathbf{e}^* - \mathbf{s}^*)$.*

2.2 Hierarchical Indexing Framework

In order to efficiently solve the PLNNS problem, a representative approach is to filter some unnecessary computations. Observe from Figure 4 that we can avoid processing those lines in the yellow rectangular region R if the smallest distance between the data point $\mathbf{p}$ and R, i.e., $d_E^{(\min)}(\mathbf{p}, R)$, is larger than $d(\mathbf{p}, l)$ (the distance value that has been obtained in advance). The main reason is that $d_E^{(\min)}(\mathbf{p}, R)$ must be smaller than the distance between $\mathbf{p}$ and any position in R, which indicates that any arbitrary line l_a in R must fulfill the inequality $d(\mathbf{p}, l_a) \ge d_E^{(\min)}(\mathbf{p}, R) > d(\mathbf{p}, l)$, i.e., any l_a in R cannot be the nearest line for $\mathbf{p}$ (based on Problem 1).

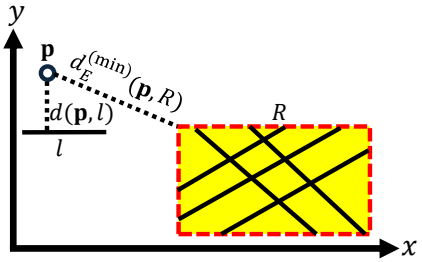


Figure 4: Illustration of how to filter a set of lines (inside the yellow rectangular region R), given that the distance $d(\mathbf{p}, l)$ has been obtained in advance.

With the above concept, we can construct a hierarchical index structure (e.g., R-tree families [17, 28, 34]) for those lines (see Figure 5) in order to solve the PLNNS problem. In this framework, we need to maintain the minimum priority queue (PQ) for each data point $\mathbf{p}$ and the best-so-far nearest neighbor distance b_{NND}. Note

that, for each data point $\mathbf{p}$, each entry of PQ either stores (1) a tree node with its rectangular region R and its corresponding $d_E^{(\min)}(\mathbf{p}, R)$ or (2) a line l and its corresponding $d(\mathbf{p}, l)$. Table 1 shows the steps for using this framework to handle the data point $\mathbf{p}$ in Figure 5. Initially, PQ only stores the node with R_{root} and b_{NND} is set to be ∞ (see step 1). This minimum priority queue iteratively pops the node with the smallest distance $d_E^{(\min)}(\mathbf{p}, R)$ and push its child nodes in PQ (e.g., R_1, with the highest priority in step 2, is popped and its child nodes, R_3 and R_4, are pushed into PQ in step 3). Once the popped node is in the leaf level, those lines l in this node are pushed into PQ (e.g., R_3 is popped and the lines, l_1 and l_2, are pushed into PQ in step 4). The best-so-far nearest neighbor distance b_{NND} is updated if the line l is popped from PQ and its current b_{NND} value is larger than $d(\mathbf{p}, l)$ (e.g., b_{NND} (with ∞ in step 4) is updated to be $d(\mathbf{p}, l_2)$ in step 5). This iterative process terminates and the nearest line l is returned if the popped node R/line l has the distance value $d_E^{(\min)}(\mathbf{p}, R)/d(\mathbf{p}, l)$ larger than b_{NND} (e.g., $d_E^{(\min)}(\mathbf{p}, R_4) > b_{\text{NND}}$ and l_2 is returned as the nearest line in step 6). Note that all entries of PQ are ultimately cleared (see step 6), and this PQ is reused for processing the next data point.

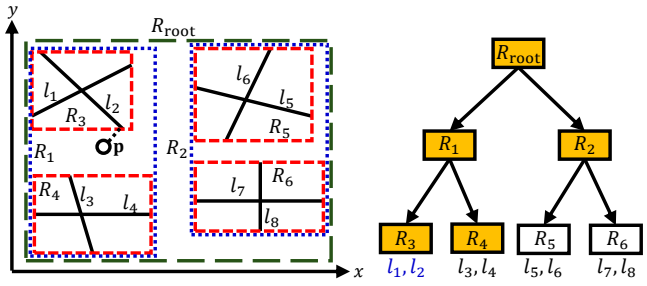


Figure 5: Using the hierarchical indexing framework to solve the PLNNS problem with one data point $\mathbf{p}$, where those orange nodes and the lines, l_1 and l_2 (colored in blue), are accessed for handling $\mathbf{p}$.

Table 1: Steps for solving PLNNS in Figure 5 for each data point $\mathbf{p}$ using the hierarchical indexing framework.

Step	Minimum priority queue (PQ)	b_{NND}	Popped node/line
1	R_{root}	∞	
2	R_1, R_2	∞	R_{root}
3	R_3, R_4, R_2	∞	R_1
4	l_2, R_4, l_1, R_2	∞	R_3
5	R_4, l_1, R_2	$d(\mathbf{p}, l_2)$	l_2
6		$d(\mathbf{p}, l_2)$	R_4

3 Our Grid Compression Solution: An Approximation Approach

We first discuss the core ideas of our grid compression solution in Section 3.1. Then, we illustrate how to tune the optimal grid cell size in Section 3.2. Lastly, we present our grid compression algorithm in Section 3.3.

3.1 Core Ideas

Recall from Section 2.2 that the state-of-the-art solution needs to independently process each data point (i.e., access the index structure for each data point). However, observe from Figure 1a that many data points are spatially close to each other. Hence, we ask a question. *Can we represent multiple data points in a batch in order to efficiently and accurately solve the PLNNS problem (see Problem 1)?* To provide an affirmative answer to this question, we have the following three core ideas, which are (1) grid compression, (2) approximate PLNNS, and (3) feasible grid cell size.

Core idea 1 (Grid compression): Observe from Figure 6 that we can first establish the grid structure on top of those data points. Then, we utilize the center position $\mathbf{c}$ to represent all data points for each grid cell (see the red circle in Figure 6).

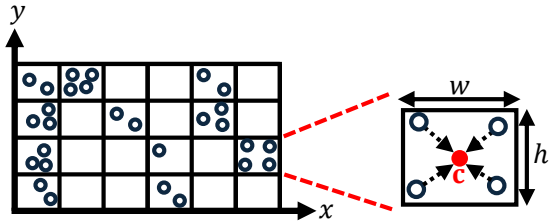


Figure 6: Illustration of grid compression.

Core idea 2 (Approximate PLNNS): With this grid structure (see Figure 6), we can (1) find the nearest line $l_c^* = [\mathbf{s}_c^*, \mathbf{e}_c^*]$ for the center position $\mathbf{c}$ (see the red dotted lines in Figure 7) and (2) find the projected position on l_c^* for each data point $\mathbf{p}$, i.e., $\mathbf{s}_c^* + C(\mathbf{p}, l_c^*) \cdot (\mathbf{e}_c^* - \mathbf{s}_c^*)$ (see those pink points in Figure 7), in this grid cell.

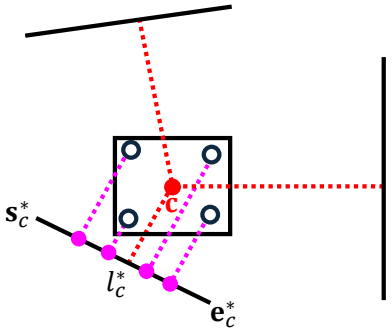


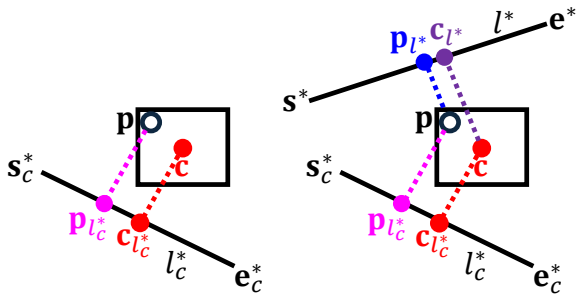
Figure 7: Approximately solving the PLNNS problem for all those data points inside a grid cell.

In order to avoid significant error for using this grid structure, we need to ensure that every data point $\mathbf{p}$ in each grid cell (with the center $\mathbf{c}$) has the absolute error guarantee with respect to the point-to-line nearest neighbor distance (see Problem 2).

Problem 2. *Given an absolute error ϵ, any grid cell with the center $\mathbf{c}$, and any data point $\mathbf{p}$ in this grid cell, we need to find l_c^* and the projected point $\mathbf{s}_c^* + C(\mathbf{p}, l_c^*) \cdot (\mathbf{e}_c^* - \mathbf{s}_c^*)$ for each $\mathbf{p}$ on l_c^* so that the point-to-line nearest neighbor distance $d(\mathbf{p}, l^*)$ fulfills this absolute error guarantee $|d(\mathbf{p}, l^*) - d(\mathbf{p}, l_c^*)| \leq \epsilon$.*

Core idea 3 (Feasible grid cell size): In order to establish the grid (see Figure 6) so that we can solve the approximate PLNNS problem (see Problem 2), we need to consider this expression $|d(\mathbf{p}, l^*) - d(\mathbf{p}, l_c^*)|$.

Observe from Figure 8 that there are two possible cases for projecting any data point $\mathbf{p}$ in the grid cell with the center $\mathbf{c}$ to the corresponding line $l_c^* = [s_c^*, e_c^*]$, which are (1) $l_c^* = l^*$ and (2) $l_c^* \neq l^*$.



(a) Case 1 ($l_c^* = l^*$) (b) Case 2 ($l_c^* \neq l^*$)

Figure 8: There are two possible cases for projecting any data point p in the grid cell with the center c to the line l_c^* (the closest line to c).

Case 1 ($l_c^* = l^*$): This case is the trivial one. As $l_c^* = l^*$, we can conclude that $|d(\mathbf{p}, l^*) - d(\mathbf{p}, l_c^*)| = 0$, which indicates that there is no error for this projection.
Case 2 ($l_c^* \neq l^*$): In this case, we represent the expression $|d(\mathbf{p}, l^*) - d(\mathbf{p}, l_c^*)|$ by the following upper bound (based on the triangle inequality).

$$\begin{aligned} |d(\mathbf{p}, l^*) - d(\mathbf{p}, l_c^*)| &= |d(\mathbf{p}, l^*) - d(\mathbf{c}, l_c^*) + d(\mathbf{c}, l_c^*) - d(\mathbf{p}, l_c^*)| \\ &\leq |d(\mathbf{p}, l^*) - d(\mathbf{c}, l_c^*)| + |d(\mathbf{c}, l_c^*) - d(\mathbf{p}, l_c^*)| \quad (3) \end{aligned}$$

Here, we further show that this upper bound is only based on the Euclidean distance between the center point $\mathbf{c}$ and the data point $\mathbf{p}$, i.e., $||\mathbf{p} - \mathbf{c}||$ (see Lemma 1 and Lemma 2). The proofs of these two lemmas can be found in Section 8.1 and Section 8.2.

Lemma 1. *Given any data point* $\mathbf{p}$ *in a grid cell with its center* $\mathbf{c}$*, the nearest line* l^* *for* $\mathbf{p}$*, and the nearest line* l_c^* *for* $\mathbf{c}$*, we have*

$$|d(\mathbf{p}, l^*) - d(\mathbf{c}, l_c^*)| \leq ||\mathbf{p} - \mathbf{c}|| \quad (4)$$

Lemma 2. *Given any data point* $\mathbf{p}$ *in a grid cell with its center* $\mathbf{c}$*, the nearest line* l^* *for* $\mathbf{p}$*, and the nearest line* l_c^* *for* $\mathbf{c}$*, we have*

$$|d(\mathbf{c}, l_c^*) - d(\mathbf{p}, l_c^*)| \leq ||\mathbf{p} - \mathbf{c}|| \quad (5)$$

Since Lemma 1 and Lemma 2 hold, we note that $|d(\mathbf{p}, l^*) - d(\mathbf{p}, l_c^*)| \leq 2||\mathbf{p} - \mathbf{c}||$ (see Equation 3). Hence, the absolute error guarantee $|d(\mathbf{p}, l^*) - d(\mathbf{p}, l_c^*)| \leq \epsilon$ (see Problem 2) can be fulfilled if $||\mathbf{p} - \mathbf{c}|| \leq \frac{\epsilon}{2}$ holds for all data points $\mathbf{p}$ inside the grid cell with the center $\mathbf{c}$ (see Theorem 1).

Theorem 1. *Given any grid cell with its center* $\mathbf{c}$*, any data point* $\mathbf{p}$ *in this grid cell, and an absolute error* ϵ*, we can solve the approximate PLNNS problem with the absolute error guarantee (Problem 2) if* $||\mathbf{p} - \mathbf{c}|| \leq \frac{\epsilon}{2}$.

Based on this theorem, we can conclude that any rectangle (with its center position $\mathbf{c}$) that is within the circle with radius $\frac{\epsilon}{2}$, e.g., the red, black, and pink dotted rectangles in Figure 9, can be used as a feasible grid cell.

3.2 How to Tune the Optimal Grid Cell Size?

In order to improve the efficiency of solving the PLNNS problem, we need to minimize the number of grid cells that is used to cover a geographical region (see Figure 1). Hence, given a $W \times H$ geographical region, we need to find the grid cell with size $w \times h$ so that the number of grid cells $\lceil \frac{W}{w} \rceil \times \lceil \frac{H}{h} \rceil$ is minimized (see Figure 10).

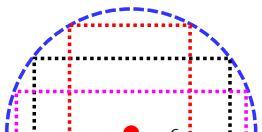


Figure 9: Each rectangular region that is within the circular range $\frac{\epsilon}{2}$ from the center c, e.g., the red dotted rectangle, the black dotted rectangle, and the pink dotted rectangle, can be used as a feasible grid cell.

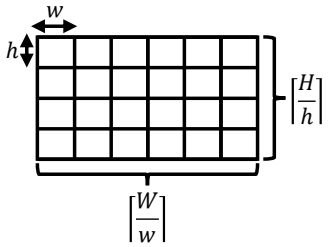


Figure 10: A $W \times H$-size geographical region should be covered by $\lceil \frac{W}{w} \rceil \times \lceil \frac{H}{h} \rceil$ grid cells, where each of them has the size of $w \times h$.

Moreover, observe from Figure 9 that each grid cell must be inside the circle with radius $\frac{\epsilon}{2}$. We can state that the parameters w and h must also fulfill this inequality $(\frac{w}{2})^2 + (\frac{h}{2})^2 \leq (\frac{\epsilon}{2})^2$. Hence, finding the optimal grid cell size $w \times h$ is equivalent to solving the following optimization problem.

$$\begin{aligned} \underset{w,h}{\text{minimize}} \quad & \left\lceil \frac{W}{w} \right\rceil \times \left\lceil \frac{H}{h} \right\rceil \\ \text{such that} \quad & w^2 + h^2 \leq \epsilon^2 \end{aligned} \quad \text{(OPT)}$$

However, we note that the objective function $\lceil \frac{W}{w} \rceil \times \lceil \frac{H}{h} \rceil$ of the optimization problem (OPT) is not continuous (and thus non-differentiable), which is hard to find the correct w and h that can solve this problem. As such, we simplify this optimization problem by removing the ceiling function.

$$\begin{aligned} \underset{w,h}{\text{minimize}} \quad & \frac{W}{w} \times \frac{H}{h} \\ \text{such that} \quad & w^2 + h^2 \leq \epsilon^2 \end{aligned} \quad \text{(OPT-S)}$$

In Lemma 3, we state that we can solve this simplified optimization problem (OPT-S) by setting $w = h = \frac{\epsilon}{\sqrt{2}}$. The proof of this lemma can be found in Appendix (see Section 8.3).

Lemma 3. *When we set* $w = h = \frac{\epsilon}{\sqrt{2}}$*, we can solve the optimization problem (OPT-S).*

Here, we further state in Lemma 4 that $w = h = \frac{\epsilon}{\sqrt{2}}$ can act as the optimal solution for the optimization problem (OPT) if $w << W$

(or $\frac{w}{W}$ tends to 0) and $h << H$ (or $\frac{h}{H}$ tends to 0). The proof of this lemma can be found in Appendix (see Section 8.4).

Lemma 4. *If $\frac{w}{W}$ and $\frac{h}{H}$ tend to 0, we can solve the optimization problem (OPT) by setting $w = h = \frac{\epsilon}{\sqrt{2}}$.*

As a remark, the width and height of the geographical region, i.e., W and H, respectively, should be much larger than the absolute error ϵ (and thus w and h, based on the constraint of (OPT)). Otherwise, it incurs significant practical error. Therefore, the conditions, $\frac{w}{W} \leftarrow 0$ and $\frac{h}{H} \leftarrow 0$, generally hold in practice.

3.3 Grid Compression Algorithm

After we have obtained the optimal size of each grid cell $w \times h$, we maintain the set of grid cells $\mathbb{G}$ in order to solve the approximate PLNNS problem with the ϵ-absolute error guarantee (see Problem 2). Algorithm 1 shows the pseudocode of this method.

Algorithm 1 Grid Compression Algorithm

```
1:  procedure GRID_COMP^APPROX(ℙ = {p_1, p_2, ..., p_n}, absolute error ε)
2:      Obtain w and h of each grid cell (based on ε)   ▷ Lemma 4
3:      Maintain the set of grid cells 𝔾, where each grid cell g stores
        (1) g.c: the center of g
        (2) g.set: the point set that covers those data points in g
4:      for each p ∈ ℙ do
5:          Find the grid cell g that covers p
6:          if g ∈ 𝔾 then
7:              g.set ← g.set ∪ {p}
8:          else
9:              Compute g.c
10:             g.set ← {p}
11:             𝔾 ← 𝔾 ∪ {g}
12:     Solve the approximate PLNNS problem (based on 𝔾)
```

Theorem 2 states the time complexity of using $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ (Algorithm 1) to solve the approximate PLNNS problem (see Problem 2). The proof of this theorem can be found in Section 8.5 of Appendix.

Theorem 2. *The time complexity of $GRID_{COMP}^{APPROX}$, i.e., Algorithm 1, is $O(\mathcal{T}_{SOTA}(|\mathbb{G}|) + n \log |\mathbb{G}|)$, where $\mathcal{T}_{SOTA}(|\mathbb{G}|)$ denotes the time complexity of using the state-of-the-art method for handling $|\mathbb{G}|$ grid cells.*

Hence, instead of calling the state-of-the-art method (e.g., the hierarchical indexing framework in Section 2.2) to solve the PLNNS problem for all data points in $\mathbb{P}$ (i.e., with $O(\mathcal{T}_{\text{SOTA}}(n))$ time), this approach only needs to call this method $|\mathbb{G}|$ times, which can significantly improve the efficiency performance as $|\mathbb{G}| << n$, i.e., $\mathcal{T}_{\text{SOTA}}(|\mathbb{G}|) << \mathcal{T}_{\text{SOTA}}(n)$, in practice.

4 Our Grid Compression Solution: An Exact Approach

Although our grid compression solution can achieve the non-trivial absolute error guarantee for solving the PLNNS problem (see Problem 2), it is possible for projecting some data points to other lines (e.g., the data point $\mathbf{p}$ in Figure 8b is projected to line l_c^* instead of l^*). However, in practice, two positions (in different roads) can be close in terms of Euclidean distance but they can be far away in terms of the shortest path distance. Using Figure 11 as an example, the black point and the pink point are close (based on Euclidean distance) but they are far away from each other in this road network (see the purple dashed line). Therefore, we ask a question. *Can we extend this grid compression solution to achieve the exact result for solving the PLNNS problem (see Problem 1)?* In this section, we provide an affirmative answer to this question by proposing a new method, called $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$.

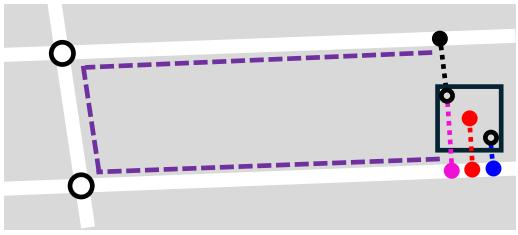


Figure 11: Projecting a data point to an incorrect road (the pink point) can result in a large error (the purple dashed line), as two positions that are close in terms of Euclidean distance can be far away on a road network, which is based on the shortest path distance.

Consider a set of data points, $\mathbf{p}_1$ and $\mathbf{p}_2$, in a grid cell (with the center $\mathbf{c}$) as an example (see Figure 12). Observe that there exists a (yellow) circular candidate space that covers the minimum point-to-line distance from each data point in this grid cell.[1] With this concept, we can determine that those black lines that are not inside this circular candidate space cannot be the nearest line for every data point. Therefore, these black lines can be filtered.

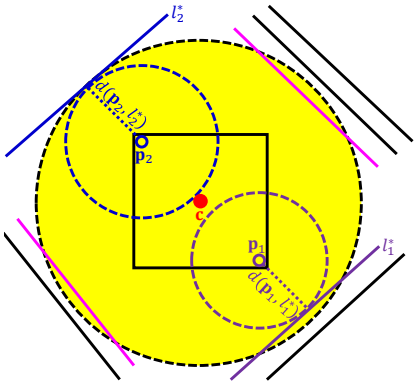


Figure 12: Illustration of the (yellow) circular candidate space with its center c for a grid cell, which covers the minimum point-to-line distance from each data point (see the blue dashed circle and the purple dashed circle). All those black lines that are not in this candidate space can be filtered.

However, one major challenge of obtaining this candidate search space is that it depends on those shortest point-to-line distance values (e.g., $d(\mathbf{p}_1, l_1^*)$ and $d(\mathbf{p}_2, l_2^*)$ in Figure 12), which are supposed to be found for solving the PLNNS problem (see Problem 1), leading to a chicken and egg situation. To get rid of this challenge, Lemma 5

[1] Imagine that we enlarge the circular space with its center $\mathbf{c}$ until it covers the blue dashed circle and the purple dashed circle in Figure 12.

concludes that we can establish this candidate search space for this grid cell with its center $\mathbf{c}$ that is solely based on the radius $d(\mathbf{c}, l_{\mathbf{c}}^*) + \epsilon$, i.e., the shortest point-to-line distance from $\mathbf{c}$ to l_c^* and the absolute error parameter ϵ (see Figure 13). The proof of this lemma can be found in Appendix (see Section 8.6).

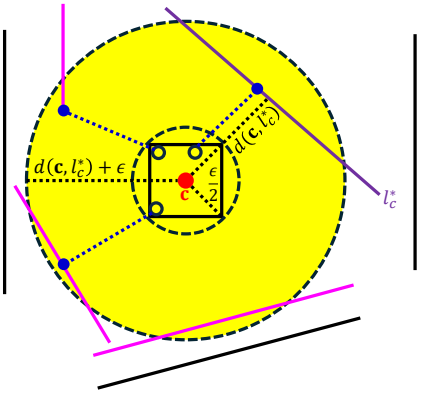


Figure 13: The (yellow) candidate search space can be determined by $d(\mathbf{c}, l_{\mathbf{c}}^*) + \epsilon$.

Lemma 5. *Given any grid cell with its center* $\mathbf{c}$*, an absolute error* ϵ*, any data point* $\mathbf{p}$ *in the grid cell that fulfills* $||\mathbf{p} - \mathbf{c}|| \leq \frac{\epsilon}{2}$*, and the closest line* l_c^* *to* $\mathbf{c}$*, all lines* l_o *with* $d(\mathbf{c}, l_o) > d(\mathbf{c}, l_{\mathbf{c}}^*) + \epsilon$ *must not be the nearest to any* $\mathbf{p}$.

By slightly modifying the hierarchical indexing framework (see Section 2.2), which will be discussed in detail in Section S1 of the supplementary material [63], our method, $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$, first retains all those lines that are inside the candidate search space (e.g., all those pink and purple lines in Figure 13). Then, $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ computes the point-to-line distance between each data point in this grid cell and each retained line for solving the PLNNS problem (see the blue dotted lines in Figure 13).

5 Experimental Evaluation

We first discuss the experimental setup in Section 5.1. Then, we illustrate the effectiveness of grid compression in Section 5.2. Next, we test the time efficiency, space efficiency, and accuracy of all methods for solving the PLNNS problem in Sections 5.3, 5.4, and 5.5, respectively. Lastly, we conduct a case study to demonstrate how our methods for PLNNS can be useful for one of the road network analysis tasks, called network kernel density visualization (NKDV), in Section 5.6. Additional experiments for Section 5.6 are presented in Section 8.7. Some experiments, including a case study for network histogram, parallelization of $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$, and comparisons with other software packages, are provided in Sections S2, S3, and S4, respectively, of the supplementary material [63].

5.1 Experimental Setup

Datasets. To conduct our experiments, we adopt four large-scale point datasets for testing. In order to obtain the corresponding line dataset for each point dataset, we (1) utilize the OSMnx python library [19] to extract the road network from the city that this point dataset belongs to and (2) regard each edge to be a line. Table 2 summarizes all these datasets, where n and L denote the number of data points and the number of lines, respectively (see Problem 1).

Table 2: Datasets.

Dataset	n	L	Category
San Francisco [10]	6,951,409	11,964	911 calls
Chicago [4]	8,189,459	41,222	Crime events
New $\text{York}_{\text{Taxi}}$ [7]	33,948,485	73,697	Pick-up locations
New $\text{York}_{\text{Bike}}$ [5]	35,107,030	73,697	Starting locations

Settings. We compare our methods, $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$, with the state-of-the-art solution, $\text{HIF}_{\text{R-tree}}$, which integrates the advanced R-tree structure [34][2] in the hierarchical indexing framework (see Section 2.2). We implemented all these methods with C++ and conducted experiments on a machine with an Intel i5-12400F 2.5GHz CPU with 64GB RAM. By default, we set the absolute error parameter ϵ to be 10m. As a remark, we have compared our methods with more competitors, which are consistently inferior to $\text{HIF}_{\text{R-tree}}$, in the supplementary material [63] (see Section S4). Hence, we omit them in this section.

5.2 Effectiveness of Grid Compression

In this section, we investigate the effectiveness of using our grid compression methods, $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$, to compress each point dataset with respect to the absolute error ϵ. Observe from Figure 14 that the larger the ϵ, the smaller the number of grid cells for representing each point dataset. The main reason is that the grid cell size becomes larger if we adopt a larger ϵ (see Lemma 4). In addition, the number of grid cells is significantly smaller than the size of the original point dataset, no matter which ϵ (from 5m to 25m) we adopt. Consider the largest dataset, New $\text{York}_{\text{Bike}}$, as an example. When ϵ is set to be 5m, the number of grid cells is 586784, which is only 1.67% of the size in the original dataset. Based on the high effectiveness of grid compression, we expect that $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ can achieve significant speedups compared with the state-of-the-art $\text{HIF}_{\text{R-tree}}$ method.

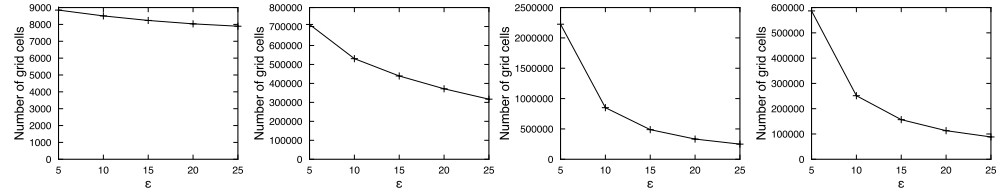
5.3 Time Efficiency of All Methods

In this section, we measure the response time of each method by conducting the following two experiments.

Varying the dataset size. We proceed to investigate how the point dataset size affects the response time of each method. To conduct this experiment, we first randomly sample each dataset with four sampling rates, which are 25%, 50%, 75%, and 100% (original one), and then measure the response time of each method. Figure 15 shows the results of all methods. Observe that the larger the dataset size, the longer the response time of each method. Since $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ utilize a smaller number of grid cells to compress each point dataset (i.e., reduce the number of using the R-tree structure), these two methods can outperform $\text{HIF}_{\text{R-tree}}$ by 5.64x to 524.3x. Note that $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ can further improve the efficiency by 1.16x to 51.47x compared with $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ since $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ needs to process additional lines for each grid cell (see the pink lines in Figure 13) to ensure the exact result for solving the PLNNS problem.

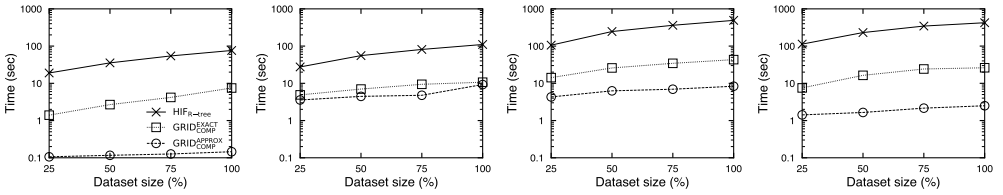
Varying the absolute error parameter ϵ. Here, we further investigate how the absolute error ϵ affects the response time of each method by choosing five values, which are 5m, 10m, 15m, 20m, and 25m. Figure 16 shows the results of all methods. Observe that

[2]Since all those lines are available in advance, we utilize the advanced bulk-loading technique for constructing the R-tree.



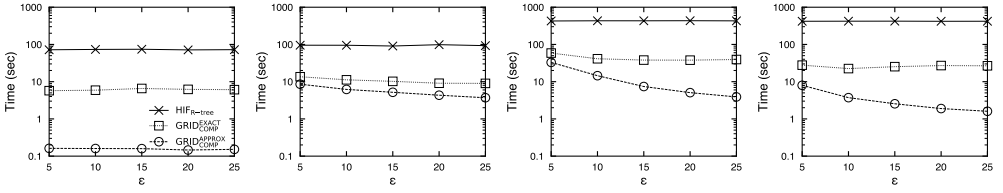
(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 14: Number of grid cells in our grid compression methods, $\text{GRID}^{\text{APPROX}}_{\text{COMP}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$, varying the absolute error ϵ.



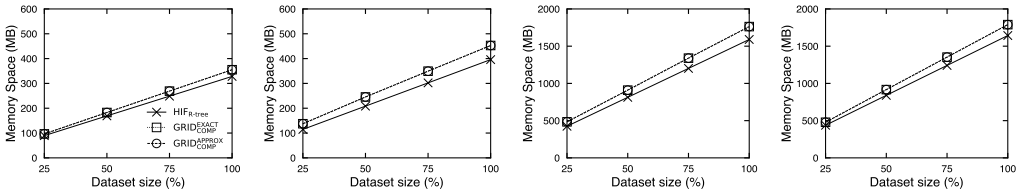
(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 15: Response time for solving the PLNNS problem, varying the point dataset size.



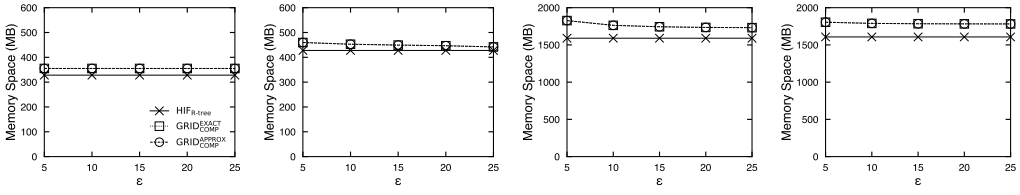
(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 16: Response time for solving the PLNNS problem, varying the absolute error ϵ.



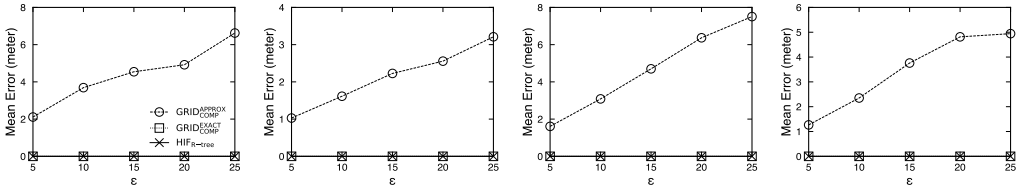
(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 17: Memory space consumption (MB) of all methods, varying the point dataset size.



(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 18: Memory space consumption (MB) of all methods, varying the absolute error ϵ.



(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 19: Accuracy (Mean Error, *ME*) for solving the PLNNS problem, varying the absolute error ϵ.

the state-of-the-art $\text{HIF}_{\text{R-tree}}$ method is insensitive to ϵ since the R-tree structure does not depend on this parameter. In addition, the larger the ϵ value, the smaller the response time of $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ (based on the reduced number of grid cells in Figure 14). Moreover, the response time of $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ is not significantly reduced with a large ϵ although the number of grid cells can also be reduced (like $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$). The main reason is that a large ϵ can result in a large candidate search space (see Figure 13), which increases the processing time for handling additional lines. As a remark, our advanced methods, $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$, can achieve 11.15x to 488.83x and 7.03x to 18.76x speedups, respectively, compared with the state-of-the-art $\text{HIF}_{\text{R-tree}}$ method.

5.4 Space Efficiency of All Methods

In this section, we further investigate the memory space consumption of each method.

Varying the dataset size. In this experiment, we first randomly sample each dataset with different sampling rates, which are 25%, 50%, 75%, and 100% (original one). Then, we measure the memory space consumption of each method. Observe from Figure 17 that the larger the sampling rate, the higher the memory space consumption of each method. In addition, since $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ need to maintain the set of grid cells $\mathbb{G}$ (see Section 3.3), the memory space consumption of these two methods is normally higher than the one of $\text{HIF}_{\text{R-tree}}$. Furthermore, the additional space should not be very large since $|\mathbb{G}| << n$ (see Figure 14 and Table 2). As an example, $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ only consume 1.03x to 1.15x additional space more than the one of $\text{HIF}_{\text{R-tree}}$.

Varying the absolute error parameter ϵ. We proceed to examine how the absolute error ϵ affects the memory space consumption of each method. To conduct this experiment, we first choose five ϵ values, which are 5m, 10m, 15m, 20m, and 25m, and then measure the memory space consumption of each method with respect to each ϵ. Figure 18 shows the results of all methods. Since $|\mathbb{G}|$ is normally much smaller compared with n in practice (see Figure 14 and Table 2), the memory space consumption of all methods is not very sensitive to ϵ. Note that $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ only consume 1.08x to 1.2x additional space compared with $\text{HIF}_{\text{R-tree}}$.

5.5 Accuracy of All Methods

In this section, we proceed to examine the practical accuracy of all methods for solving the PLNNS problem with respect to the absolute error ϵ. Here, we utilize the following mean error (*ME*) for measuring the accuracy (based on the notations in Problem 2).

$$ME = \frac{1}{|\mathbb{P}|} \sum_{\mathbf{p} \in \mathbb{P}} d_G\big(\mathbf{s}_c^* + C(\mathbf{p}, l_c^*) \cdot (\mathbf{e}_c^* - \mathbf{s}_c^*), \mathbf{s}^* + C(\mathbf{p}, l^*) \cdot (\mathbf{e}^* - \mathbf{s}^*)\big) \tag{6}$$

where $\mathbf{s}_c^* + C(\mathbf{p}, l_c^*) \cdot (\mathbf{e}_c^* - \mathbf{s}_c^*)$ denotes the projected point returned by the evaluated method on the line segment $l_c^* = [\mathbf{s}_c^*, \mathbf{e}_c^*]$, and $\mathbf{s}^* + C(\mathbf{p}, l^*) \cdot (\mathbf{e}^* - \mathbf{s}^*)$ denotes the exact projected point on the ground-truth nearest line segment $l^* = [\mathbf{s}^*, \mathbf{e}^*]$. The function $d_G(\cdot, \cdot)$ represents the shortest path distance on the road network.

Figure 19 shows the accuracy of all methods. Since both $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ and $\text{HIF}_{\text{R-tree}}$ are the exact methods, each projected data point $\mathbf{s}_c^* + C(\mathbf{p}, l_c^*) \cdot (\mathbf{e}_c^* - \mathbf{s}_c^*)$ must be the same as $\mathbf{p}$. Therefore, *ME* must be zero for these two methods regardless of which ϵ we adopt. In addition, the larger the ϵ, the larger the chance for the data point p to be projected to another line (i.e., $l_C^* \neq l^*$ in Problem 2), which increases the *ME* value. As a remark, the *ME* values of $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ are range from 1.03m to 7.51m in these four datasets once we set ϵ from 5m to 25m.

5.6 A Case Study for Road Network Analysis: Network Kernel Density Visualization

In this section, we investigate why efficiently and accurately solving the PLNNS problem can benefit for supporting road network analysis. Here, we utilize the network kernel density visualization (NKDV) [21, 44] tool, which discovers hotspot regions of a location dataset, as a case study.

Efficiency of computing NKDV with PLNNS. Recall from Figure 1 that we need to project all data points to the road network in advance so that we can perform the road network analysis tasks (e.g., NKDV). Therefore, we test how $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ can improve the end-to-end efficiency performance, i.e., computing (1) PLNNS and (2) NKDV, compared with $\text{HIF}_{\text{R-tree}}$. Here, we utilize the efficient algorithm in [21], which can compute an exact NKDV for each dataset and set the default ϵ = 10m for $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$.

Table 3 shows the response time of each method. Although the end-to-end response time includes both the time of solving the PLNNS problem and computing NKDV, $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ can still achieve 5.67x to 43.04x and 5.31x to 10.61x speedups, respectively, compared with $\text{HIF}_{\text{R-tree}}$. Note that these two methods can still significantly outperform the $\text{HIF}_{\text{R-tree}}$ method because solving the PLNNS problem occupies the majority of the end-to-end response time. More details can be found in Section 8.7.

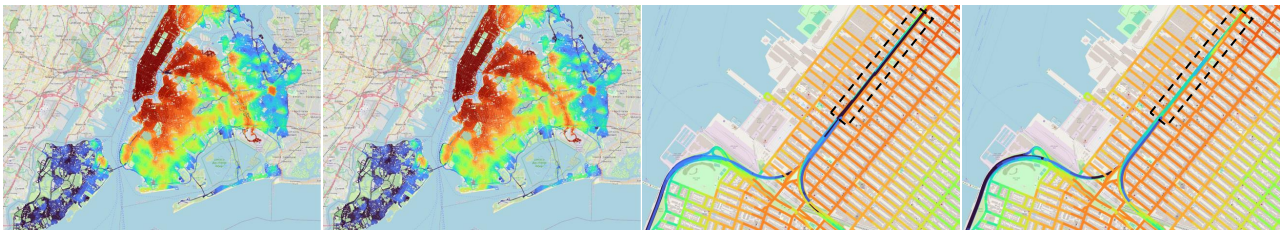
Table 3: The end-to-end response time (sec) for supporting NKDV with different methods of PLNNS.

Dataset	$\text{HIF}_{\text{R-tree}}$	$\text{GRID}_{\text{COMP}}^{\text{APPROX}}$	$\text{GRID}_{\text{COMP}}^{\text{EXACT}}$
San Francisco	77.91	**1.81**	9.15
Chicago	122.83	**21.68**	23.12
$\text{New York}_{\text{Taxi}}$	513.37	**30.96**	65.86
$\text{New York}_{\text{Bike}}$	439.29	**17.73**	41.42

Accuracy of computing NKDV with PLNNS. Here, we investigate the subjective accuracy of generating NKDV with the exact ($\text{HIF}_{\text{R-tree}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$) and approximate ($\text{GRID}_{\text{COMP}}^{\text{APPROX}}$) PLNNS methods using the $\text{New York}_{\text{Taxi}}$ location dataset. Observe from Figures 20a and b that NKDVs with the exact and approximate PLNNS methods have the similar visualization quality. Therefore, in practice, once we perform analysis in a large region (e.g., city-level), we can utilize the $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$ method to further improve the efficiency without significantly degrading the visualization quality. However, once we zoom in to the Sunset Park region of New York, NKDV with the approximate PLNNS method, $\text{GRID}_{\text{COMP}}^{\text{APPROX}}$, can provide the incorrect result compared with the exact methods, $\text{HIF}_{\text{R-tree}}$ and $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ (see the black dashed rectangles in Figures 20c and d). Therefore, suppose that we perform detailed analysis in a small region (e.g., street-level), it is better to utilize $\text{GRID}_{\text{COMP}}^{\text{EXACT}}$ for solving the PLNNS problem. Some experiments related to objective accuracy can also be found in Section 8.7.

6 Related Work

In this section, we review four camps of research studies, which are closely related to the PLNNS problem.



(a) Exact (New York) (b) Approximation (New York) (c) Exact (Sunset Park) (d) Approximation (Sunset Park)

Figure 20: Subjective accuracy of generating NKDVs using the exact ($\text{HIF}_{\text{R-tree}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$) and approximate ($\text{GRID}^{\text{APPROX}}_{\text{COMP}}$, with $\epsilon = 10$m) PLNNS methods for mapping data points in the New York$_{\text{Taxi}}$ location dataset to the corresponding road network, where (a) and (b) are in New York and (c) and (d) are in the Sunset Park region of New York.

Map matching. Recall from Problem 1 that PLNNS aims to find the nearest line for each data point and project (or map) this data point to the corresponding line, which can be regarded as a category of map matching [32, 36, 43]. However, most of the existing methods [24, 32, 36, 38, 41, 43, 47, 48, 52, 57, 65, 67] aim to utilize the machine learning models to map trajectories onto a road network, by leveraging the relationship (or properties) between different data points in these trajectories. However, in many spatial analysis tasks on road networks, those location datasets (e.g., the 311-call data points in Figure 1) may not be trajectories, which, therefore, do not exhibit these properties. To handle such analysis tasks, many software packages, e.g., QGIS [9] and OSMnx [19], also provide the functionality of PLNNS for mapping those data points on road networks.

Tree-based indexing structures. In the literature, many tree-based indexing structures, e.g., kd-tree [18, 51], ball-tree/m-tree [25, 42, 51, 66], R-tree [17, 28, 51], and range-tree [26], have been proposed for tackling different types of spatial queries (e.g., kNN and range queries [45, 50]). However, most of these tree structures, e.g., kd-tree, ball-tree/m-tree, and range-tree, can only be used for indexing data points, instead of lines, which, therefore, cannot be used for handling the PLNNS problem.[3] In addition, although some tree structures, e.g., R-tree, can be used for indexing lines, this approach needs to access the tree structure for finding the nearest line for every data point, which results in huge response time. Note that our grid compression solution aims to use one grid cell to represent multiple data points, which can significantly reduce the number of accesses no matter which tree structure we adopt.

Grid-based indexing structures. To tackle different types of spatial queries, many grid-based indexing structures [12, 13, 20, 53–55, 60, 61, 68] have been proposed in the last few decades, which can be used to boost the efficiency of solving these queries. However, unlike this work, all of these research studies do not consider any point-to-line distance property for compression. Furthermore, most of these research studies [12, 13, 20, 53–55, 60, 68] only focus on handling spatial queries for supporting point datasets. Therefore, all these research studies cannot be easily extended to solve our PLNNS problem (see Problem 1) with non-trivial approximation and exact guarantees (see Theorem 1 and Lemma 5).

Point-to-hyperplane nearest neighbor search. In the literature, many research studies [15, 29–31, 35, 37, 40, 58] have developed efficient algorithms for handling the point-to-hyperplane nearest neighbor search problem. However, unlike our PLNNS problem, which finds the nearest line (from a set of lines) for each data point (see Problem 1), these research studies aim to find the nearest data point in a point dataset from each hyperplane query. With this main difference, all these methods cannot support our setting.

[3] Since we need to find the nearest line for each data point (see Problem 1), building a tree-based indexing structure for data points cannot solve this problem.

7 Conclusion

In this paper, we study the point-to-line nearest neighbor search (PLNNS) problem, which is the prior step before using the road-network-based spatial analysis tools (e.g., NKDV). However, PLNNS is computationally expensive, which cannot be scalable to handle a large-scale point dataset and a large number of line segments (e.g., roads). Therefore, PLNNS can be regarded as the main bottleneck of the road network analysis pipeline. Although the state-of-the-art hierarchical indexing framework can improve the efficiency of solving the PLNNS problem, this approach needs to invoke the hierarchical index structure (e.g., R-tree) for every data point, which can still be inefficient in practice. To tackle this efficiency issue, we first develop the approximate grid compression method, called $\text{GRID}^{\text{APPROX}}_{\text{COMP}}$, which utilizes a small number of grid cells to represent those data points in a point dataset in order to efficiently solve the PLNNS problem with a non-trivial approximation guarantee. By investigating the property of a candidate search space that is established by all data points in a grid cell, we then develop the exact grid compression method, called $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$, for solving this PLNNS problem. Our experiments on four large-scale datasets verify that $\text{GRID}^{\text{APPROX}}_{\text{COMP}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$ can achieve 7.72x to 524.3x and 5.64x to 18.76x speedups, respectively, compared with the state-of-the-art $\text{HIF}_{\text{R-tree}}$ method, without significantly degrading the accuracy of solving the PLNNS problem. Our case studies also demonstrate that $\text{GRID}^{\text{APPROX}}_{\text{COMP}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$ can substantially improve the efficiency and retain the similar accuracy for supporting road network analysis tools. These two algorithms have also been integrated into a python library, called FastNSA [6], for public use.

In the future, we plan to establish the QGIS/ArcGIS plugins for supporting these two grid compression methods. Moreover, we will develop efficient algorithms for handling other map matching tools.

Acknowledgments

This work was supported by the Natural Science Foundation of China under grants 23IAA00610, 62572326, and 62372308, Scientific Foundation for Youth Scholars of Shenzhen University, and the Science and Technology Development Fund Macau SAR (0003/2023/RIC, 0052/2023/RIA1, 0011/2025/RIC, 001/2024/SKL for SKL-IOTSC). This work was performed in part at SICC which is supported by SKL-IOTSC, University of Macau.

References

[1] [n. d.]. ArcGIS. https://www.arcgis.com/index.html.
[2] [n. d.]. Chicago 311 Service Requests. https://data.cityofchicago.org/Service-Requests/311-Service-Requests/v6vf-nfxy/about_data.
[3] [n. d.]. Chicago Crimes - 2001 to Present. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2/about_data.
[4] [n. d.]. Chicago Data Protal. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2/about_data.
[5] [n. d.]. Citi Bike Trip Histories. https://citibikenyc.com/system-data.
[6] [n. d.]. FastNSA. https://pypi.org/project/fastnsa/.
[7] [n. d.]. NYC Yellow Taxi Trip Data. https://www.kaggle.com/datasets/elemento/nyc-yellow-taxi-trip-data.
[8] [n. d.]. PostGIS. https://postgis.net/.
[9] [n. d.]. QGIS. https://qgis.org/.
[10] [n. d.]. San Francisco Data Portal. https://data.sfgov.org/Public-Safety/Fire-Department-and-Emergency-Medical-Services-Dis/nuek-vuh3/about_data.
[11] [n. d.]. spNetwork: Spatial Analysis on Network - R Project. https://cran.r-project.org/web/packages/spNetwork/spNetwork.pdf.
[12] Charu C. Aggarwal and Philip S. Yu. 2000. The IGrid index: reversing the dimensionality curse for similarity indexing in high dimensional space. In *SIGKDD*. ACM, 119–129. https://doi.org/10.1145/347090.347116
[13] Pritom Ahmed, Ahmed Eldawy, Vagelis Hristidis, and Vassilis J. Tsotras. 2023. Reverse spatial top-k keyword queries. *VLDB J.* 32, 3 (2023), 501–524. https://doi.org/10.1007/S00778-022-00759-9
[14] H. Anton. 2010. *Elementary Linear Algebra.* John Wiley & Sons.
[15] Martin Aumüller, Tobias Christiani, Rasmus Pagh, and Francesco Silvestri. 2018. Distance-Sensitive Hashing. In *PODS*. ACM, 89–104. https://doi.org/10.1145/3196959.3196976
[16] Adrian Baddeley, Gopalan Nair, Suman Rakshit, Greg McSwiggan, and Tilman M. Davies. 2021. Analysing point patterns on networks — A review. *Spatial Statistics* 42 (2021), 100435.
[17] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-Tree: An Efficient and Robust Access Method for Points and Rectangles. In *SIGMOD*. 322–331.
[18] Jon Louis Bentley. 1975. Multidimensional Binary Search Trees Used for Associative Searching. *Commun. ACM* 18, 9 (1975), 509–517.
[19] Geoff Boeing. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65 (2017), 126 – 139.
[20] Panagiotis Bouros, Shen Ge, and Nikos Mamoulis. 2012. Spatio-textual similarity joins. *Proc. VLDB Endow.* 6, 1 (2012), 1–12. https://doi.org/10.14778/2428536.2428537
[21] Tsz Nam Chan, Zhe Li, Leong Hou U, Jianliang Xu, and Reynold Cheng. 2021. Fast Augmentation Algorithms for Network Kernel Density Visualization. *Proc. VLDB Endow.* 14, 9 (2021), 1503–1516.
[22] Tsz Nam Chan, Leong Hou U, Yun Peng, Byron Choi, and Jianliang Xu. 2022. Fast Network K-function-based Spatial Analysis. *Proc. VLDB Endow.* 15, 11 (2022), 2853–2866.
[23] Tsz Nam Chan, Rui Zang, Bojian Zhu, Leong Hou U, Dingming Wu, and Jianliang Xu. 2024. LION: Fast and High-Resolution Network Kernel Density Visualization. *Proc. VLDB Endow.* 17, 6 (2024), 1255–1268. https://www.vldb.org/pvldb/vol17/p1255-chan.pdf
[24] Minxiao Chen, Haitao Yuan, Nan Jiang, Zhihan Zheng, Sai Wu, Ao Zhou, and Shangguang Wang. 2025. RLOMM: An Efficient and Robust Online Map Matching Framework with Reinforcement Learning. *Proc. ACM Manag. Data* 3, 3 (2025), 209:1–209:26. https://doi.org/10.1145/3725346
[25] Paolo Ciaccia, Marco Patella, and Pavel Zezula. 1997. M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. In *VLDB*. 426–435.
[26] Mark De Berg, Otfried Cheong, Marc Van Kreveld, and Mark Overmars. 2008. *Computational geometry: algorithms and applications.* Springer.
[27] Zidong Fang, Ci Song, Hua Shu, Jie Chen, Tianyu Liu, Xi Wang, Xiao Chen, Xiaorui Yan, and Tao Pei. 2023. Length L-function for network-constrained point data. *Transactions in GIS* 27, 2 (2023), 476–493. https://doi.org/10.1111/tgis.13035 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/tgis.13035
[28] Antonin Guttman. 1984. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD*. 47–57.
[29] Qiang Huang, Yifan Lei, and Anthony K. H. Tung. 2021. Point-to-Hyperplane Nearest Neighbor Search Beyond the Unit Hypersphere. In *SIGMOD*. ACM, 777–789. https://doi.org/10.1145/3448016.3457240
[30] Qiang Huang and Anthony K. H. Tung. 2023. Lightweight-Yet-Efficient: Revitalizing Ball-Tree for Point-to-Hyperplane Nearest Neighbor Search. In *ICDE*. IEEE, 436–449. https://doi.org/10.1109/ICDE55515.2023.00040
[31] Prateek Jain, Sudheendra Vijayanarasimhan, and Kristen Grauman. 2010. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. In *NIPS*. Curran Associates, Inc., 928–936. https://proceedings.neurips.cc/paper/2010/hash/470e7a4f017a5476afb7eeb3f8b96f9b-Abstract.html
[32] Linli Jiang, Chaoxiong Chen, and Chao Chen. 2023. L2MM: Learning to Map Matching with Deep Models for Low-Quality GPS Trajectory Data. *ACM Trans. Knowl. Discov. Data* 17, 3 (2023), 39:1–39:25. https://doi.org/10.1145/3550486
[33] Jalal Khalil, Da Yan, Lyuheng Yuan, Mostafa Jafarzadehfadaki, Saugat Adhikari, Jiao Han, Virginia Sisiopiku, and Zhe Jiang. 2025. Urban Traffic Simulation with Shared Mobility Services: An Approach Using Spatiotemporal Network Kernel Density Estimation and MATSim. *ACM Transactions on Spatial Algorithms and Systems* 11, 4 (2025), 1–31.
[34] Scott T. Leutenegger, Mario Alberto López, and J. M. Edgington. 1997. STR: A Simple and Efficient Algorithm for R-Tree Packing. In *ICDE*. 497–506. https://doi.org/10.1109/ICDE.1997.582015
[35] Wei Liu, Jun Wang, Yadong Mu, Sanjiv Kumar, and Shih-Fu Chang. 2012. Compact Hyperplane Hashing with Bilinear Functions. In *ICML*. icml.cc / Omnipress. http://icml.cc/2012/papers/16.pdf
[36] Xuemei Liu, James Biagioni, Jakob Eriksson, Yin Wang, George Forman, and Yanmin Zhu. 2012. Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In *SIGKDD*. ACM, 669–677. https://doi.org/10.1145/2339530.2339637
[37] Xianglong Liu, Xinjie Fan, Cheng Deng, Zhujin Li, Hao Su, and Dacheng Tao. 2016. Multilinear Hyperplane Hashing. In *CVPR*. IEEE, 5119–5127. https://doi.org/10.1109/CVPR.2016.553
[38] Yu Liu, Qian Ge, Wei Luo, Qiang Huang, Lei Zou, Haixu Wang, Xin Li, and Chang Liu. 2024. GraphMM: Graph-Based Vehicular Map Matching by Leveraging Trajectory and Road Correlations. *IEEE Trans. Knowl. Data Eng.* 36, 1 (2024), 184–198. https://doi.org/10.1109/TKDE.2023.3287739
[39] Jianglin Lu, Chunjiao Dong, Xuedong Yan, Jingjun Li, and Zhenzhen Lu. 2026. Exploring spatiotemporal heterogeneity and nonlinear effects in electric vehicle crash risk prediction: A hybrid modeling approach. *Accident Analysis & Prevention* 226 (2026), 108357. https://doi.org/10.1016/j.aap.2025.108357
[40] Kejing Lu, Yoshiharu Ishikawa, and Chuan Xiao. 2022. MQH: Locality Sensitive Hashing on Multi-level Quantization Errors for Point-to-Hyperplane Distances. *Proc. VLDB Endow.* 16, 4 (2022), 864–876. https://doi.org/10.14778/3574245.3574269
[41] Huajian Mao, Wuman Luo, Haoyu Tan, Lionel M. Ni, and Nong Xiao. 2012. Exploration of ground truth from raw GPS data. In *UrbComp@KDD*. ACM, 118–125. https://doi.org/10.1145/2346496.2346515
[42] Andrew W. Moore. 2000. The Anchors Hierarchy: Using the Triangle Inequality to Survive High Dimensional Data. In *UAI*. 397–405.
[43] Paul Newson and John Krumm. 2009. Hidden Markov map matching through noise and sparseness. In *SIGSPATIAL*. ACM, 336–343. https://doi.org/10.1145/1653771.1653818
[44] A. Okabe and K. Sugihara. 2012. *Spatial Analysis Along Networks: Statistical and Computational Methods.* Wiley. https://books.google.com.hk/books?id=48GRqj51_W8C
[45] Guido Proietti and Christos Faloutsos. 2000. Analysis of Range Queries and Self-Spatial Join Queries on Real Region Datasets Stored Using an R-Tree. *IEEE Trans. Knowl. Data Eng.* 12, 5 (2000), 751–762. https://doi.org/10.1109/69.877506
[46] I. Gede Brawiswa Putra, Pei-Fen Kuo, and Dominique Lord. 2024. Estimating the effectiveness of marked sidewalks: An application of the spatial causality approach. *Accident Analysis & Prevention* 206 (2024), 107699. https://doi.org/10.1016/j.aap.2024.107699
[47] Mohammed A. Quddus, Washington Y. Ochieng, and Robert B. Noland. 2007. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies* 15, 5 (2007), 312–328. https://doi.org/10.1016/j.trc.2007.05.002
[48] Huimin Ren, Sijie Ruan, Yanhua Li, Jie Bao, Chuishi Meng, Ruiyuan Li, and Yu Zheng. 2021. MTrajRec: Map-Constrained Trajectory Recovery via Seq2Seq Multi-task Learning. In *SIGKDD*. ACM, 1410–1419. https://doi.org/10.1145/3447548.3467238
[49] Gabriel Rosser, Toby O. Davies, Kate. Bowers, Shane D. Johnson, and T. Cheng. 2017. Predictive Crime Mapping: Arbitrary Grids or Street Networks? *Journal of Quantitative Criminology* 33 (2017), 569 – 594.
[50] Nick Roussopoulos, Stephen Kelley, and Frédéic Vincent. 1995. Nearest Neighbor Queries. In *SIGMOD*. ACM, 71–79. https://doi.org/10.1145/223784.223794
[51] H. Samet. 2006. *Foundations of Multidimensional and Metric Data Structures.*
[52] Weijie Shi, Jiajie Xu, Junhua Fang, Pingfu Chao, An Liu, and Xiaofang Zhou. 2023. LHMM: A Learning Enhanced HMM Model for Cellular Trajectory Map Matching. In *ICDE*. IEEE, 2429–2442. https://doi.org/10.1109/ICDE55515.2023.00187
[53] Jaewoo Shin, Ahmed R. Mahmood, and Walid G. Aref. 2019. An Investigation of Grid-enabled Tree Indexes for Spatial Query Processing. In *SIGSPATIAL*. ACM, 169–178. https://doi.org/10.1145/3347146.3359384
[54] Darius Sidlauskas, Simonas Saltenis, Christian W. Christiansen, Jan M. Johansen, and Donatas Saulys. 2009. Trees or grids?: indexing moving objects in main memory. In *SIGSPATIAL*. ACM, 236–245. https://doi.org/10.1145/1653771.1653805
[55] Darius Sidlauskas, Simonas Saltenis, and Christian S. Jensen. 2014. Processing of extreme moving-object update and query workloads in main memory. *VLDB J.* 23, 5 (2014), 817–841. https://doi.org/10.1007/S00778-014-0353-2

[56] Brian S Thomson, Judith B Bruckner, and Andrew M Bruckner. 2008. *Elementary real analysis*. Vol. 1. ClassicalRealAnalysis.com.
[57] Wei Tian, Jieming Shi, and Man Lung Yiu. 2025. Efficient Methods for Accurate Sparse Trajectory Recovery and Map Matching. In *ICDE*. IEEE, 363–375. https://doi.org/10.1109/ICDE65448.2025.00034
[58] Sudheendra Vijayanarasimhan, Prateek Jain, and Kristen Grauman. 2014. Hashing Hyperplane Queries to Near Points with Applications to Large-Scale Active Learning. *IEEE Trans. Pattern Anal. Mach. Intell.* 36, 2 (2014), 276–288. https://doi.org/10.1109/TPAMI.2013.121
[59] Teng Wang, Yandong Wang, Xiaoming Zhao, and Xiaokang Fu. 2018. Spatial distribution pattern of the customer count and satisfaction of commercial facilities based on social network review data in Beijing, China. *Computers, Environment and Urban Systems* 71 (2018), 88–97. https://doi.org/10.1016/j.compenvurbsys.2018.04.005
[60] Roger Weber, Hans-Jörg Schek, and Stephen Blott. 1998. A Quantitative Analysis and Performance Study for Similarity-Search Methods in High-Dimensional Spaces. In *VLDB*. 194–205.
[61] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. 2012. Constructing popular routes from uncertain trajectories. In *SIGKDD*. ACM, 195–203. https://doi.org/10.1145/2339530.2339562
[62] Zhixiao Xie and Jun Yan. 2008. Kernel Density Estimation of traffic accidents in a network space. *Computers, Environment and Urban Systems* 32, 5 (2008), 396 – 406. https://doi.org/10.1016/j.compenvurbsys.2008.05.001
[63] Hongwei Ye, Tsz Nam Chan, Dingming Wu, Leong Hou U, and Ruisheng Wang. 2026. Supplementary Material for "An Efficient and Accurate Grid Compression Solution for Point-to-Line Nearest Neighbor Search". https://github.com/fastnsa/grid-comp-plnns/blob/main/supplementary_material.pdf.
[64] Wenhao Yu, Tinghua Ai, and Shiwei Shao. 2015. The analysis and delimitation of Central Business District using network kernel density estimation. *Journal of Transport Geography* 45 (2015), 32–47. https://doi.org/10.1016/j.jtrangeo.2015.04.008
[65] Ayman Zeidan, Eemil Lagerspetz, Kai Zhao, Petteri Nurmi, Sasu Tarkoma, and Huy T. Vo. 2020. GeoMatch: Efficient Large-scale Map Matching on Apache Spark. *Trans. Data Sci.* 1, 3 (2020), 21:1–21:30. https://doi.org/10.1145/3402904
[66] P. Zezula, G. Amato, V. Dohnal, and M. Batko. 2006. *Similarity Search: The Metric Space Approach*. Springer US. https://books.google.com.hk/books?id=KTkWXsiPXR4C
[67] Kai Zhao, Jie Feng, Zhao Xu, Tong Xia, Lin Chen, Funing Sun, Diansheng Guo, Depeng Jin, and Yong Li. 2019. DeepMM: Deep Learning Based Map Matching with Data Augmentation. In *SIGSPATIAL*. ACM, 452–455. https://doi.org/10.1145/3347146.3359090
[68] Baihua Zheng, Jianliang Xu, Wang-Chien Lee, and Dik Lun Lee. 2006. Grid-partition index: a hybrid method for nearest-neighbor queries in wireless location-based services. *VLDB J.* 15, 1 (2006), 21–39. https://doi.org/10.1007/S00778-004-0146-0

8 Appendix

8.1 Proof of Lemma 1

Proof. Here, we let $\mathbf{p}_{l^*}$ and $\mathbf{p}_{l_c^*}$ be the projected data points for $\mathbf{p}$ to the nearest line l^* and the line l_c^*, respectively. Moreover, we also let $\mathbf{c}_{l^*}$ and $\mathbf{c}_{l_c^*}$ be the projected data points for $\mathbf{c}$ to the line l^* and the nearest line l_c^*, respectively. Therefore, we have

$$d(\mathbf{p}, l^*) = ||\mathbf{p} - \mathbf{p}_{l^*}|| \quad \text{and} \quad d(\mathbf{p}, l_c^*) = ||\mathbf{p} - \mathbf{p}_{l_c^*}|| \tag{7}$$

$$d(\mathbf{c}, l^*) = ||\mathbf{c} - \mathbf{c}_{l^*}|| \quad \text{and} \quad d(\mathbf{c}, l_c^*) = ||\mathbf{c} - \mathbf{c}_{l_c^*}|| \tag{8}$$

Based on the property of the point-to-line distance (see Figure 3), we know that $d(\mathbf{c}, l^*)$ (see Equation 8) must be smaller than the distance from $\mathbf{c}$ to any position on l^* (see Figure 8b). Therefore, we have

$$||\mathbf{c} - \mathbf{p}_{l^*}|| \geq ||\mathbf{c} - \mathbf{c}_{l^*}|| \tag{9}$$

Moreover, since l_c^* is the nearest line to $\mathbf{c}$, we have

$$||\mathbf{c} - \mathbf{c}_{l^*}|| \geq ||\mathbf{c} - \mathbf{c}_{l_c^*}|| \tag{10}$$

With Equations 9 and 10, we conclude that

$$||\mathbf{c} - \mathbf{p}_{l^*}|| \geq ||\mathbf{c} - \mathbf{c}_{l_c^*}|| \tag{11}$$

Based on the triangular inequality and Figure 8b, we can observe that the expression holds.

$$||\mathbf{p} - \mathbf{p}_{l^*}|| + ||\mathbf{p} - \mathbf{c}|| \geq ||\mathbf{c} - \mathbf{p}_{l^*}|| \tag{12}$$

By substituting Equations 7, 11, and 8 into Equation 12, we have

$$d(\mathbf{p}, l^*) - d(\mathbf{c}, l_c^*) \geq -||\mathbf{p} - \mathbf{c}|| \tag{13}$$

On the other hand, we also have (based on triangular inequality and Figure 8b)

$$||\mathbf{c} - \mathbf{c}_{l_c^*}|| + ||\mathbf{p} - \mathbf{c}|| \geq ||\mathbf{p} - \mathbf{c}_{l_c^*}|| \tag{14}$$

Moreover, based on similar ideas, the following inequality holds (see Figure 8b)

$$||\mathbf{p} - \mathbf{c}_{l_c^*}|| \geq ||\mathbf{p} - \mathbf{p}_{l_c^*}|| \geq ||\mathbf{p} - \mathbf{p}_{l^*}|| \tag{15}$$

By substituting Equation 15 into Equation 14 and considering Equations 7 and 8, we have

$$d(\mathbf{p}, l^*) - d(\mathbf{c}, l_c^*) \leq ||\mathbf{p} - \mathbf{c}|| \tag{16}$$

Based on Equations 13 and 16, we conclude that Equation 4 holds. □

8.2 Proof of Lemma 2

Proof. Observe from Figure 8b that we have (based on the triangle inequality)

$$||\mathbf{c} - \mathbf{c}_{l_c^*}|| + ||\mathbf{p} - \mathbf{c}|| \geq ||\mathbf{p} - \mathbf{c}_{l_c^*}|| \tag{17}$$

In addition, we also have (based on the point-to-line distance property in Figure 3)

$$||\mathbf{p} - \mathbf{c}_{l_c^*}|| \geq ||\mathbf{p} - \mathbf{p}_{l_c^*}|| \tag{18}$$

Based on Equation 17, Equation 18, Equation 7, and Equation 8, we conclude that

$$d(\mathbf{c}, l_c^*) - d(\mathbf{p}, l_c^*) \geq -||\mathbf{p} - \mathbf{c}|| \tag{19}$$

By adopting the similar concepts in Equations 17 and 18, we also have (see Figure 8b)

$$||\mathbf{p} - \mathbf{p}_{l_c^*}|| + ||\mathbf{p} - \mathbf{c}|| \geq ||\mathbf{c} - \mathbf{p}_{l_c^*}|| \geq ||\mathbf{c} - \mathbf{c}_{l_c^*}|| \tag{20}$$

Based on Equations 7 and 8, we have

$$d(\mathbf{c}, l_c^*) - d(\mathbf{p}, l_c^*) \leq ||\mathbf{p} - \mathbf{c}|| \tag{21}$$

With Equations 19 and 21, we conclude that Equation 5 holds. □

8.3 Proof of Lemma 3

Proof. Consider the Lagrange multiplier of this optimization problem. We have

$$\mathcal{L}(w, h, \lambda) = \frac{W}{w} \times \frac{H}{h} + \lambda(w^2 + h^2 - \epsilon^2)$$

By setting the partial derivatives to zero and solving the resulting equations, we can conclude that $w = h = \frac{\epsilon}{\sqrt{2}}$.

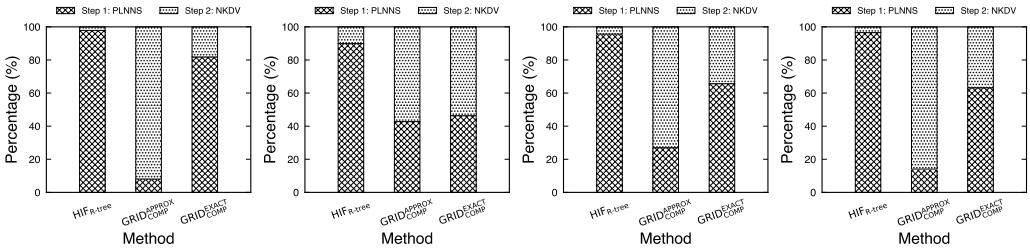
□

8.4 Proof of Lemma 4

Proof. Since we know that $\frac{W}{w} \times \frac{H}{h} \leq \lceil \frac{W}{w} \rceil \times \lceil \frac{H}{h} \rceil \leq (\frac{W}{w} + 1) \times (\frac{H}{h} + 1)$, the relative error can be represented by the following lower bound and upper bound.

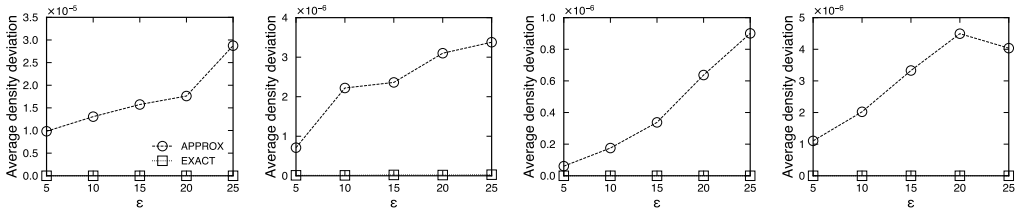
$$0 \leq \frac{\lceil \frac{W}{w} \rceil \times \lceil \frac{H}{h} \rceil - \frac{W}{w} \times \frac{H}{h}}{\frac{W}{w} \times \frac{H}{h}} \leq \frac{\left(\frac{W}{w} + 1\right) \times \left(\frac{H}{h} + 1\right) - \frac{W}{w} \times \frac{H}{h}}{\frac{W}{w} \times \frac{H}{h}} = \frac{h}{H} + \frac{w}{W} + \frac{wh}{WH}$$

Therefore, once $\frac{w}{W}$ and $\frac{h}{H}$ tend to 0, the upper bound of this relative error also tends to 0, which, based on the squeeze theorem [56], can conclude that the relative error tends to 0, i.e., the objective function $\lceil \frac{W}{w} \rceil \times \lceil \frac{H}{h} \rceil$ is nearly the same as $\frac{W}{w} \times \frac{H}{h}$ (indicating that $w = h = \frac{\epsilon}{\sqrt{2}}$ can also solve the (OPT) problem). □



(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 21: Distribution of the response time for the step 1, solving the PLNNS problem, and the step 2, generating NKDV.



(a) San Francisco (b) Chicago (c) New York$_{\text{Taxi}}$ (d) New York$_{\text{Bike}}$

Figure 22: Objective accuracy, average density deviation (ADD), of generating NKDV, using the exact ($\text{HIF}_{\text{R-tree}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$) and approximate ($\text{GRID}^{\text{APPROX}}_{\text{COMP}}$) methods for solving the PLNNS problem.

8.5 Proof of Theorem 2

Proof. Since (1) many grid cells have zero data point[4] and (2) the number of grid cells is large (especially for a large geographical region, i.e., large $W \times H$) in practice, we maintain the priority queue for $\mathbb{G}$ to store only those grid cells (using the x-coordinate and y-coordinate of each center as the priority) with at least one data point in Algorithm 1. Therefore, after this algorithm finds the grid cell g that covers each data point $\mathbf{p}$ in line 5 with $O(1)$ time, it takes $O(\log |\mathbb{G}|)$ time for checking whether g is in $\mathbb{G}$ (see line 6) and inserting g in $\mathbb{G}$ (see line 11), meaning that line 5 to line 11 takes $O(n \log |\mathbb{G}|)$ time throughout the loop in line 4. Note that solving the approximate PLNNS problem in line 12 takes $O(\mathcal{T}_{\text{SOTA}}(|\mathbb{G}|))$ time. We can conclude that the time complexity of Algorithm 1 is $O(\mathcal{T}_{\text{SOTA}}(|\mathbb{G}|) + n \log |\mathbb{G}|)$. □

8.6 Proof of Lemma 5

Proof. Based on the similar concept in the proof of Lemma 2 and Equation 19, these two inequalities, $d(\mathbf{c}, l_o) \leq d(\mathbf{p}, l_o) + ||\mathbf{p} - \mathbf{c}||$ and $d(\mathbf{c}, l_c^*) \geq d(\mathbf{p}, l_c^*) - ||\mathbf{p} - \mathbf{c}||$, hold, respectively. By substituting them into the condition $d(\mathbf{c}, l_o) > d(\mathbf{c}, l_{\mathbf{c}}^*) + \epsilon$, we have

$$d(\mathbf{p}, l_o) > d(\mathbf{p}, l_c^*) - 2||\mathbf{p} - \mathbf{c}|| + \epsilon$$

Since $||\mathbf{p} - \mathbf{c}|| \leq \frac{\epsilon}{2}$, we conclude that $d(\mathbf{p}, l_o) > d(\mathbf{p}, l_c^*)$. Therefore, l_o cannot be the nearest line to $\mathbf{p}$. □

8.7 Additional Experiments of Section 5.6

In this section, we further conduct two additional experiments for the case study of generating NKDV with two PLNNS methods (see Section 5.6).

Distribution of response time of computing NKDV. Here, we measure the distribution of the response time for generating NKDV, which can be divided into two steps, which are (1) solving the PLNNS problem for projecting each data point on the nearest edge of the road network and (2) generating NKDV. Observe from Figure 21 that the main bottleneck of computing NKDV is in step 1, which occupies at least 89.9% of the end-to-end response time, when we adopt $\text{HIF}_{\text{R-tree}}$ as the PLNNS method. With our efficient $\text{GRID}^{\text{APPROX}}_{\text{COMP}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$ methods, step 1 only occupies 8.04% to 81.82% of the end-to-end response time. Therefore, these results can further illustrate why our methods can significantly improve the efficiency compared with the $\text{HIF}_{\text{R-tree}}$ method.

Objective accuracy of computing NKDV. We proceed to measure the objective accuracy of computing NKDV based on the exact ($\text{HIF}_{\text{R-tree}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$) and approximate ($\text{GRID}^{\text{APPROX}}_{\text{COMP}}$) PLNNS methods. In order to generate NKDV, previous research studies [21, 23, 62] first divide a road network into a set of lixels[5] $\mathbb{L}$ and then color each lixel q based on the network kernel density function value $\mathcal{F}_{\widehat{\mathbb{P}}}(q)$, where $\widehat{\mathbb{P}}$ denotes the set of (exact) projected data points on a road network. Here, we utilize the average density deviation (ADD) as the objective accuracy.

$$\text{ADD} = \frac{1}{|\mathbb{L}|} \sum_{q \in \mathbb{L}} |\mathcal{F}_{\widehat{\mathbb{P}_A}}(q) - \mathcal{F}_{\widehat{\mathbb{P}}}(q)| \tag{22}$$

where $\widehat{\mathbb{P}_A}$ denotes the set of projected data points based on the algorithm A of the PLNNS problem.

Figure 22 shows the ADD results of all datasets. Since the exact PLNNS methods ($\text{HIF}_{\text{R-tree}}$ and $\text{GRID}^{\text{EXACT}}_{\text{COMP}}$) provide the same network kernel density value for each q, i.e., $\mathcal{F}_{\widehat{\mathbb{P}_A}}(q) = \mathcal{F}_{\widehat{\mathbb{P}}}(q)$, the ADD values of these methods must be zero. Observe that the approximate PLNNS method, i.e., $\text{GRID}^{\text{APPROX}}_{\text{COMP}}$, can still provide the accurate performance for generating NKDV (with up to 3×10^{-5} in terms of the ADD results among these four datasets).

[4] As an example, some regions (see the upper right region and the lower right region in Figure 1a) contain zero data point.

[5] Lixel is a small road segment on each edge, which can be regarded as a "pixel" in a road network.