STARS: A Sampling and Threshold Sharing Solution for Network K-function Analytics

Hongwei Ye
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
yehongwei2023@email.szu.edu.cn

Tsz Nam Chan*
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
edisonchan@szu.edu.cn

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

Dingming Wu
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
dingming@szu.edu.cn

Wei Tu
Ruisheng Wang
School of Architecture and Urban Planning, Shenzhen University
Shenzhen, China
{tuwei,ruiswang}@szu.edu.cn

Joshua Zhexue Huang
College of Computer Science and Software Engineering, Shenzhen University
Shenzhen, China
zx.huang@szu.edu.cn

Abstract

Network K-function analytics is an important tool for different application domains, including transportation science, criminology, and urban planning. Despite this, this tool is computationally demanding, which does not scale to support large datasets. Although the state-of-the-art method, called neighbor sharing (NS), can successfully reduce the time complexity for supporting network K-function analytics, this method is still very slow. To address the efficiency issue, we propose the sampling and threshold sharing solution (STARS), which is the first work that can simultaneously (1) reduce the time complexity, (2) retain the similar space complexity, and (3) achieve the non-trivial absolute error (accuracy) guarantee. Experimental results with four large-scale datasets (up to 8.19 million data points) show that STARS achieves (1) 1.52x to 21.57x speedups, (2) negligible additional space overhead, and (3) negligible practical accuracy degradation compared with NS. The implementation of this paper can be found in https://github.com/fastnsa/STARS/.

CCS Concepts

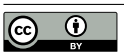
• **Information systems → Geographic information systems**.

Keywords

Network K-function, Sampling, Approximation Algorithms, Efficiency

ACM Reference Format:
Hongwei Ye, Tsz Nam Chan, Leong Hou U, Dingming Wu, Wei Tu, Ruisheng Wang, and Joshua Zhexue Huang. 2026. STARS: A Sampling and Threshold Sharing Solution for Network K-function Analytics. 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/10.1145/3770855.3817709

*Corresponding author.



KDD '26, August 09–13, 2026, Jeju Island, Republic of Korea

ACM ISBN 979-8-4007-2259-2/2026/08
https://doi.org/10.1145/3770855.3817709

1 Introduction

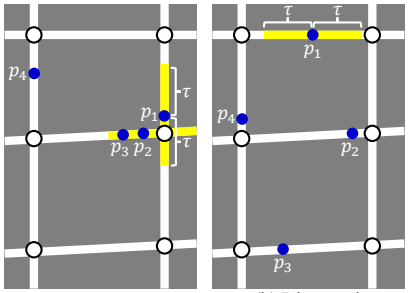
Spatial point pattern analysis [12, 44, 56] is a fundamental field for analyzing location datasets. Among most of the tools in this field, the network K-function [20] has recently received significant attention due to its wide range of applications, including transportation science, criminology, and urban planning. Transportation scientists [10, 16, 27, 38, 68] and criminologists [11, 26, 39–41] utilize the network K-function to determine whether a location dataset consists of meaningful traffic accident and crime hotspots/clusters. Urban planners [15, 46, 57, 64, 65, 67] adopt the network K-function to understand whether the points of interest (e.g., facilities in a city) or urban events (e.g., fire events) exhibit the clustering tendency. Due to the popularity of the network K-function, many software packages have been developed for supporting this network analysis tool. Some representative examples include SANET [7] (a plugin for QGIS [54] and ArcGIS [1]), spNetwork [8] (an R package), and NetPointLib [33] (a recently developed python package).

Figure 1 illustrates how the network K-function is used for analyzing a location dataset. This tool aims to **first count all data points that are within the threshold τ from each data point and then normalize this value by the number of all pairs of data points.** Observe from Figure 1a that $\{p_2, p_3\}$, $\{p_1, p_3\}$, and $\{p_1, p_2\}$ are within τ from p_1, p_2, and p_3, respectively, and no data point is within τ from p_4. Therefore, the network K-function for the dataset in Figure 1a is $\frac{6}{4\times 3} = 0.5$. In contrast, the network K-function for the dataset in Figure 1b is 0 since no data point is within τ for each data point.

To determine whether a location dataset P (with size n) has meaningful clusters/hotspots with respect to each threshold τ, domain experts need to generate a network K-function plot (see Figure 2) by (1) specifying $L-1$ random datasets, which are R_1, R_2,..., R_{L-1}, with the same size n as P and D thresholds, which are τ_1, τ_2,..., τ_D, (2) obtaining the network K-function value for the original dataset P with respect to each threshold (i.e., the black curve in Figure 2), and

Table 1: Theoretical results of different methods, where n, L, D, $|E|$, T_{SP}, and S_{SP} denote the dataset size, the number of datasets, the number of thresholds, the number of edges, the time complexity of the shortest path algorithm, and the space complexity of the shortest path algorithm, respectively.

Method	Time complexity	Space complexity	Accuracy guarantee
NS (Section 2.2 [20])	$O(\|E\|T_{\text{SP}} + LnD\|E\| + Ln\log n)$	$O(\|E\| + Ln + S_{\text{SP}})$	Exact
STARS (Section 3.2)	$O\big(\|E\|T_{\text{SP}} + \frac{LD\|E\|^2}{\epsilon}\log\big(\frac{1}{\epsilon}\big) + Ln\|E\| + Ln\log n\big)$ (Theorem 3)	$O\big(\|E\| + Ln + \frac{D}{\epsilon} + S_{\text{SP}}\big)$ (Theorem 4)	ϵ-absolute error (Theorem 1)



(a) Clustered (b) Dispersed

Figure 1: Illustration of the network K-function, where the zone that is within the threshold τ from the data point p_1 is colored by yellow.

(3) obtaining the smallest and largest network K-function values for those $L-1$ randomly generated datasets with respect to each threshold (i.e., the blue dotted curve and red dotted curve, respectively, in Figure 2). With this network K-function plot, domain experts can claim that the location dataset P exhibits the cluster/hotspot properties for small thresholds (see the yellow region in Figure 2) and may not contain meaningful clusters/hotspots if the threshold is set to be large (see the green region in Figure 2). With this additional information, domain experts can (1) avoid analyzing those clusters/hotspots (detected by clustering/hotspot detection algorithms) with large sizes [34, 45], e.g., larger than τ_3 in Figure 2, and (2) set the proper threshold parameter for the analytic tools, e.g., set the bandwidth value of the network kernel density visualization tool [18, 22, 44] (another point pattern analysis tool) to be smaller than τ_3 (see Figure 2) for detecting hotspots.

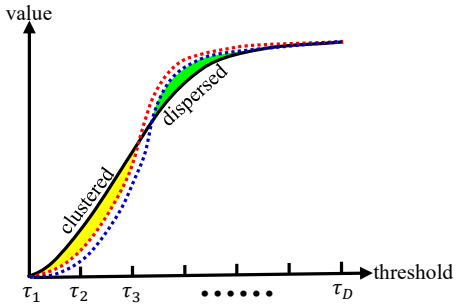


Figure 2: An example of the network K-function plot, where the black curve represents the network K-function values of the original dataset P with D thresholds, τ_1, τ_2,..., τ_D, and the blue dotted curve and the red dotted curve represent the smallest and the largest network K-function values, respectively, of $L-1$ randomly generated datasets with D thresholds.

However, generating a network K-function plot is computationally expensive, which needs to evaluate $L \times D$ network K-functions. Consider the New York 911-call location dataset [5] as an example, which consists of 5.43 million data points. Using a naïve approach to compute a single network K-function already takes at least 29.48 trillion operations,[1] let alone to compute multiple network K-functions for generating a network K-function plot.

To tackle this efficiency issue, Chan et al. [20] have proposed the state-of-the-art exact solution, called Neighbor Sharing (NS), which successfully reduces the time complexity of generating a network K-function plot to $O(|E|T_{\text{SP}} + LnD|E| + Ln\log n)$, where $|E|$ and T_{SP} denote the number of edges of a road network and the time complexity of the shortest path algorithm, respectively. Despite this, the time complexity still depends on $LnD|E|$, which can still be large in practice. Consider the San Francisco 311-call location dataset [6] (with 6.05 million data points) as an example. Generating the network K-function plot with $L = 5$ datasets and $D = 100$ thresholds **takes more than 10 hours**. The main reason for incurring the high computational time (or having high time complexity) is that the existing solution, NS, aims to retain the exact network K-function plot. Therefore, we ask a question in this paper. *Can we develop a new approximate solution for generating a network K-function plot, which can simultaneously achieve the non-trivial accuracy guarantee, further reduce the time complexity, and retain the similar space complexity?* To provide a definite answer to this question, we develop a new Sampling and Threshold shARing Solution (STARS), which is **the first work that successfully reduces the time complexity for generating the network K-function plot with a non-trivial ϵ-absolute error guarantee and a slight overhead for the space complexity.** Table 1 compares the theoretical results between NS and STARS. Experimental results with four large-scale location datasets show that STARS achieves 1.52x to 21.57x speedups, negligible additional space overhead, and negligible accuracy degradation compared with NS. As a remark, we have also integrated STARS into the python library, called fastnsa [3] (until 22$^{\text{nd}}$ May 2026, **the cumulative number of downloads has reached 1,280**, based on the statistics in https://pepy.tech/), for public use.

The rest of the paper is structured as follows. We discuss the preliminaries in Section 2. Then, we illustrate our solution, STARS, in Section 3. Next, we present the experimental results in Section 4. After that, we provide the literature review in Section 5. Lastly, we conclude this paper in Section 6. Appendix is in Section 7. Supplementary material can be found in [69].

2 Preliminaries

We first define the network K-function and formulate the problem of generating a network K-function plot in Section 2.1. Then, we

[1] For each location data point, this naïve approach needs to access all data points on a road network. Therefore, the total number of operations is at least 5.43 million × 5.43 million, excluding the operations of calculating shortest path distances.

overview the state-of-the-art solution, called neighbor sharing (NS), in Section 2.2. Lastly, we discuss the data sampling solution for another statistical model, called kernel density estimation [73], which is closely related to network K-function, in Section 2.3. Some commonly used notations can be found in Section S1 of [69].

2.1 Problem Formulation

Here, we formally define the network K-function in Definition 1.

Definition 1. *[12] Given a road network $G = (V, E)$, a location dataset $P = \{p_1, p_2, ..., p_n\}$ (with size n), where each data point resides on one and only one edge in G, and a threshold τ, the network K-function is expressed as follows.*

$$K_P(\tau) = \frac{1}{n(n-1)} \sum_{p_i \in P} \sum_{\substack{p_j \in P \\ p_j \neq p_i}} \mathbb{I}(d_G(p_i, p_j) \leq \tau) \quad (1)$$

where $d_G(p_i, p_j)$ and $\mathbb{I}$ denote the shortest path distance between p_i and p_j in G and the indicator function, respectively.

Based on Definition 1, we further formulate the problem of generating a network K-function plot (see Figure 2) in Problem 1.

Problem 1. *Given a road network $G = (V, E)$, a location dataset P (with size n), $L-1$ randomly generated datasets (with size n), i.e., R_1, R_2,..., R_{L-1}, and D thresholds, τ_1, τ_2,..., τ_D, generating a network K-function plot involves the computation of $K_P(\tau_\rho), \mathbb{L}(\tau_\rho)$, and $\mathbb{U}(\tau_\rho)$ for each threshold τ_ρ, where $1 \leq \rho \leq D$.*

$$\mathbb{L}(\tau_\rho) = \min_{1 \leq \lambda \leq L-1} K_{R_\lambda}(\tau_\rho) \text{ and } \mathbb{U}(\tau_\rho) = \max_{1 \leq \lambda \leq L-1} K_{R_\lambda}(\tau_\rho) \quad (2)$$

2.2 Neighbor Sharing (NS)

In order to efficiently generate a network K-function plot, Chan et al. [20] recently propose a new method, called neighbor sharing (NS). By considering $P(e)$ to be the set of data points on the edge e, they decompose the network K-function (see Equation 1) into the following expression (see Equation 3).

$$K_P(\tau) = \frac{1}{n(n-1)} \sum_{\widehat{e} \in E} \sum_{e \in E} \sum_{p_i \in P(\widehat{e})} C^{(p_i)}_{P(e)}(\tau) \quad (3)$$

where $C^{(p_i)}_{P(e)}(\tau)$ is the $(p_i, P(e))$-count function (see Equation 4).

$$C^{(p_i)}_{P(e)}(\tau) = \sum_{\substack{p_j \in P(e) \\ p_j \neq p_i}} \mathbb{I}(d_G(p_i, p_j) \leq \tau) \quad (4)$$

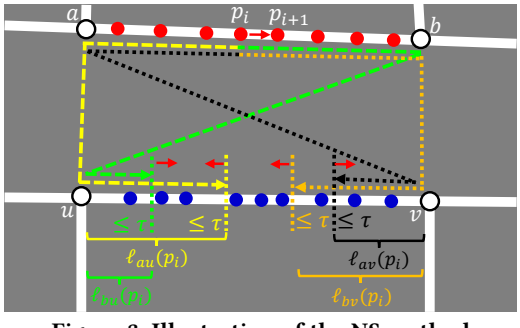


Figure 3: Illustration of the NS method.

To efficiently evaluate $C^{(p_i)}_{P(e)}(\tau)$ on an edge $e = (u, v)$ for a data point p_i on $\widehat{e} = (a, b)$ (see Figure 3), they propose to maintain four sets of data points, which are $\ell_{au}(p_i)$ (yellow), $\ell_{av}(p_i)$ (black), $\ell_{bu}(p_i)$ (green), and $\ell_{bv}(p_i)$ (orange). With these four sets of data points, they can evaluate $C^{(p_i)}_{P(e)}(\tau)$ based on the following conditions, which are C1 and C2.

C1 ($\max(|\ell_{au}(p_i)|, |\ell_{bu}(p_i)|) + \max(|\ell_{av}(p_i)|, |\ell_{bv}(p_i)|) < |P(e)|$):

$$C^{(p_i)}_{P(e)}(\tau) = \max(|\ell_{au}(p_i)|, |\ell_{bu}(p_i)|) + \max(|\ell_{av}(p_i)|, |\ell_{bv}(p_i)|)$$

C2 ($\max(|\ell_{au}(p_i)|, |\ell_{bu}(p_i)|) + \max(|\ell_{av}(p_i)|, |\ell_{bv}(p_i)|) \geq |P(e)|$):

$$C^{(p_i)}_{P(e)}(\tau) = |P(e)|$$

Once NS processes the next data point on $\widehat{e} = (a, b)$ (i.e., moving from p_i to p_{i+1}), each dotted line on $e = (u, v)$ only moves according to one direction (see the red arrows in Figure 3). Based on this concept, Chan et al. [20] show that these dotted lines only scan all blue data points on $e = (u, v)$ at most four times in total (for updating $\ell_{au}(p_i)$, $\ell_{av}(p_i)$, $\ell_{bu}(p_i)$ and $\ell_{bv}(p_i)$) once NS processes all red data points on $\widehat{e} = (a, b)$. Therefore, they show that computing $C^{(p_i)}_{P(e)}(\tau)$ for all red data points on $\widehat{e} = (a, b)$ takes $O(|P(\widehat{e})| + |P(e)|)$ time (instead of $O(|P(\widehat{e})||P(e)|)$ time), which further achieves $O(|E|T_{\text{SP}} + LnD|E| + Ln \log n)$ time[2] for generating a network K-function plot.

2.3 A Sampling Solution for Kernel Density Estimation with One-dimensional Data

In this section, we discuss the concept of another statistical model, called kernel density estimation (KDE), with one-dimensional data, which is closely related to network K-function, and the optimal sampling solution for computing approximate KDE. Problem 2 formally defines KDE with the uniform kernel function.

Problem 2. *[50] Given a set of one-dimensional data points $X = \{x_1, x_2, ..., x_{|X|}\}$, where $x_1 \leq x_2 \leq ... \leq x_{|X|}$, a positive bandwidth value σ, an arbitrary position q, the KDE function for q should be computed based on Equation 5.*

$$KDE_X(q) = \frac{1}{|X|} \sum_{x \in X} \mathbb{I}(d(q, x) \leq \sigma) \quad (5)$$

where $d(q, x)$ and $\mathbb{I}$ denote the Euclidean distance and the indicator function, respectively.

Given an absolute error δ, Zheng et al. [73] have proposed the sorted selection (SS) sampling approach for finding the subset $S = \{s_1, s_2, ..., s_{|S|}\}$ of X, where

$$s_l = x_{\lceil (l - \frac{1}{2})\delta |X| \rceil} \text{ and } |S| = \left\lceil \frac{1}{\delta} \right\rceil \quad (6)$$

Figure 4 illustrates how SS works. Observe that only those yellow points are retained for computing $KDE_S(q)$ in this example (with $|X| = 1000000$ and $\delta = 0.001$).

They further show that SS can produce a sampled dataset S with $O(|X| \log |X|)$ time so that $KDE_S(q)$ can be used to approximate $KDE_X(q)$ with a non-trivial accuracy guarantee (see Lemma 1).

Lemma 1. *[73] Given an absolute error δ, a location dataset X, the SS method produces a sampled dataset S with size $|S| = O(\frac{1}{\delta})$*

[2] Chan et al. [20] also propose the advanced shortest path sharing (ASPS) approach to reduce the time complexity for generating a network K-function plot by sharing the shortest path information with multiple location datasets on a road network. Due to space limitations, we omit the details. However, NS in this paper has already been combined with ASPS, which achieves the lowest worst-case time complexity.

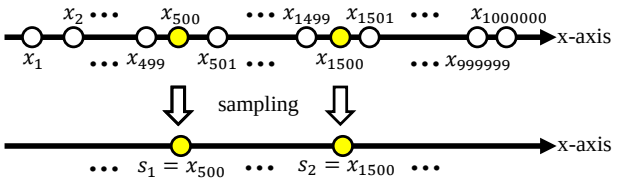


Figure 4: Illustration of using the sampling approach, called sorted selection (SS), where the size $|X| = 1000000$ and the approximation error $\delta = 0.001$ in this example.

in $O(|X| \log |X|)$ time so that the following guarantee holds for an arbitrary position q.

$$|KDE_X(q) - KDE_S(q)| \leq \delta \tag{7}$$

As a remark, Phillips [50] has pointed out that the size of the sampled dataset $|S| = \Omega(\frac{1}{\delta})$ no matter which sampling algorithm is adopted. Therefore, SS has already achieved the smallest (i.e., optimal) sample size with δ absolute error guarantee for $KDE_X(q)$.

3 Our Solution

We first illustrate the connection between network K-function and KDE in Section 3.1. With this concept, we propose a Sampling and Threshold shARing Solution, called STARS, which simultaneously reduces the time complexity for generating a network K-function plot (see Problem 1), achieves the similar space complexity, and retains an absolute error approximation guarantee, in Section 3.2.

3.1 Connection between Network K-function and KDE

Observe that the KDE function (see Equation 5) and the $(p_i, P(e))$-count function $C^{(p_i)}_{P(e)}(\tau)$ (see Equation 4) of the network K-function (see Equation 3) are similar to each other. Therefore, we ask a question. *Can we represent $C^{(p_i)}_{P(e)}(\tau)$ by the KDE function?*

To provide a positive answer to this question, we first project the nodes u and v and all blue data points on $e = (u, v)$ into the x-axis (see Figure 5). Here, we regard the node u as the origin, i.e.,

$$x_u = 0 \text{ and } x_{p_j} = d_G(u, p_j) \text{ and } x_v = d_G(u, v) \tag{8}$$

Note that p_i is on another edge $\widehat{e} = (a, b)$, which has two routes, i.e., from p_i to u and from p_i to v, to reach the edge e. Therefore, we can project p_i into two positions, which are

$$x_{u,p_i} = -d_G(p_i, u) \text{ and } x_{v,p_i} = d_G(p_i, v) + d_G(u, v) \tag{9}$$

With this projection, we can then have the following conclusion for computing $C^{(p_i)}_{P(e)}(\tau)$ (see Lemma 2 and Section 7.1 for its proof).

Lemma 2. *Given a data point p_i on the edge $\widehat{e} = (a, b)$, the shortest path distances from p_i to the nodes u and v on the edge $e = (u, v)$, i.e., $d_G(p_i, u)$ and $d_G(p_i, v)$, respectively, the $(p_i, P(e))$-count function $C^{(p_i)}_{P(e)}(\tau)$ can be computed based on the following cases.*

Case 1 ($2\tau - d_G(p_i, u) - d_G(p_i, v) < d_G(u, v)$):

$$C^{(p_i)}_{P(e)}(\tau) = |P(e)|(KDE_{P(e)}(x_{u,p_i}) + KDE_{P(e)}(x_{v,p_i}))$$

Case 2 ($2\tau - d_G(p_i, u) - d_G(p_i, v) \geq d_G(u, v)$):

$$C^{(p_i)}_{P(e)}(\tau) = |P(e)|$$

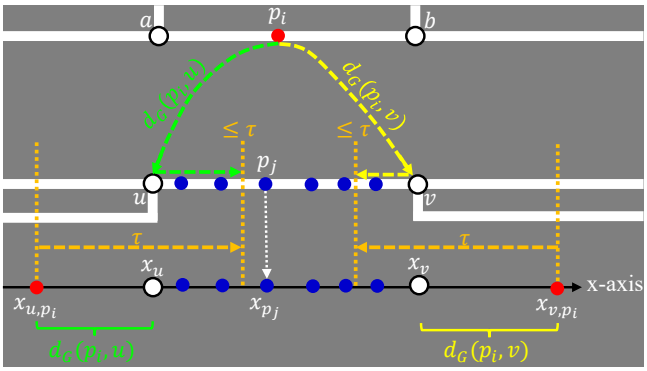


Figure 5: By projecting each blue data point p_j into the x-axis (see the white dotted line), the nodes u and v into x_u and x_v, respectively, and the data point p_i (on $\widehat{e} = (a, b)$) into two positions, namely x_{u,p_i} and x_{v,p_i}, computing $C^{(p_i)}_{P(e)}(\tau)$ can be converted to evaluating KDE for x_{u,p_i} and x_{v,p_i}.

3.2 STARS: A Sampling and Threshold Sharing Solution

In this section, we propose the solution, called STARS, which is based on these two components, namely (1) sampling and (2) threshold sharing. Here, we first discuss these two components and then illustrate how STARS can be used for generating the network K-function plot (i.e., solving Problem 1).

Sampling. Recall from Section 3.1 that $C^{(p_i)}_{P(e)}(\tau)$ (see Equation 4) of the network K-function $K_P(\tau)$ (see Equation 3) can be connected to the KDE function (see Equation 5). In addition, the SS data sampling method can significantly reduce the dataset size for efficiently computing the approximate KDE function with the non-trivial approximation guarantee (see Equation 7). Therefore, we ask a question. *Can we transfer SS for efficiently computing the network K-function with the non-trivial accuracy guarantee?* To provide an affirmative answer to this question, we investigate how to accurately estimate the approximate value of $C^{(p_i)}_{P(e)}(\tau)$ based on SS.

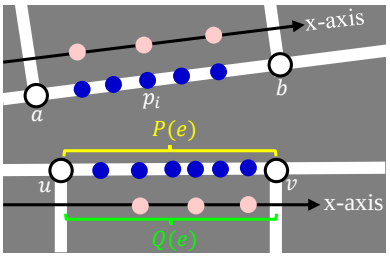


Figure 6: Using the SS data sampling method (with the absolute error parameter δ) on each edge of the original dataset P (blue data points) in order to construct the sampled dataset Q (light pink data points).

Observe from Figure 6 that we first adopt SS with an absolute error δ (see Section 2.3) on each edge of the original dataset P (see the blue data points) to construct the sampled dataset Q (see the light pink data points). Since both $C^{(p_i)}_{P(e)}(\tau)$ and $C^{(p_i)}_{Q(e)}(\tau)$ count the data points on the edge $e = (u, v)$ with the sets of $P(e)$ and $Q(e)$, respectively (see Figure 6), our conjecture is that the ratio between $C^{(p_i)}_{P(e)}(\tau)$ and $C^{(p_i)}_{Q(e)}(\tau)$ should be similar to $\frac{|P(e)|}{|Q(e)|}$. Hence, we aim to

approximate $C^{(p_i)}_{P(e)}(\tau)$ by $\frac{|P(e)|}{|Q(e)|}C^{(p_i)}_{Q(e)}(\tau)$ for every data point p_i in the original dataset P. In Lemma 3, we state that this approximation can result in the absolute error with at most $2\delta|P(e)|$. The proof of this lemma is in Section 7.2.

Lemma 3. *Given a data point p_i of the dataset P on an edge $\widehat{e} = (a, b)$, the sampled dataset Q (obtained from SS with an absolute error δ), and the edge $e = (u, v)$, this error bound holds.*

$$\left|C^{(p_i)}_{P(e)}(\tau) - \frac{|P(e)|}{|Q(e)|}C^{(p_i)}_{Q(e)}(\tau)\right| \leq 2\delta|P(e)| \tag{10}$$

Since we adopt $\frac{|P(e)|}{|Q(e)|}C^{(p_i)}_{Q(e)}(\tau)$ as the estimation of $C^{(p_i)}_{P(e)}(\tau)$, $A_{P,Q}(\tau)$ (see Equation 11) can act as the approximation of $K_P(\tau)$ (see Equation 3).

$$A_{P,Q}(\tau) = \frac{1}{n(n-1)}\sum_{\widehat{e}\in E}\sum_{e\in E}\sum_{p_i\in P(\widehat{e})}\frac{|P(e)|}{|Q(e)|}C^{(p_i)}_{Q(e)}(\tau) \tag{11}$$

In Theorem 1, we state that $A_{P,Q}(\tau)$ can achieve the ϵ-absolute error guarantee if we choose $\delta = \frac{(n-1)\epsilon}{2n}$ in the SS data sampling method. The proof of this theorem is in Section 7.3.

Theorem 1. *Given an absolute error ϵ and a location dataset P, we need to choose $\delta = \frac{(n-1)\epsilon}{2n}$ for SS to construct Q so that*

$$|A_{P,Q}(\tau) - K_P(\tau)| \leq \epsilon \tag{12}$$

Furthermore, we also state in Theorem 2 that the size of Q is $O(\frac{|E|}{\epsilon})$ (instead of $O(n)$ in P) and the size of Q on each edge e, i.e., $|Q(e)|$, is $O(\frac{1}{\epsilon})$ if we adopt $\delta = \frac{(n-1)\epsilon}{2n}$ for SS to construct Q. The proof of this theorem is in Section 7.4.

Theorem 2. *Given an absolute error ϵ and a location dataset P, the size of the sampled dataset Q is $O(\frac{|E|}{\epsilon})$ and the size of Q on each edge e, i.e., $|Q(e)|$, is $O(\frac{1}{\epsilon})$ if we choose $\delta = \frac{(n-1)\epsilon}{2n}$ for SS to construct Q.*

Moreover, using SS (with $\delta = \frac{(n-1)\epsilon}{2n}$) to construct Q from P in a road network G only takes $O(|E| + n\log n + \frac{|E|}{\epsilon})$ time (see Lemma 4), which is a small time overhead. The proof of this lemma is in Section 7.5.

Lemma 4. *Given a road network $G = (V, E)$, an absolute error ϵ, a location dataset P in G, the time complexity of SS (with $\delta = \frac{(n-1)\epsilon}{2n}$) for constructing Q is $O(|E| + n\log n + \frac{|E|}{\epsilon})$.*

Threshold sharing. In order to generate an approximate network K-function plot (like Figure 2) based on $A_{P,Q}(\tau)$ (or $C^{(p_i)}_{Q(e)}(\tau)$), we propose the threshold sharing method, which can reduce the time complexity of computing $C^{(p_i)}_{Q(e)}(\tau)$ with D thresholds, $\tau_1, \tau_2, ..., \tau_D$. This method is based on three observations, namely (1) evaluation order reversal, (2) establishment of sorted threshold positions, and (3) sweep line processing.

Evaluation order reversal. Recall from Equation 11 that we need to compute $\mathbb{C}^{P(\widehat{e})}_{Q(e)}(\tau)$ (see Equation 13) for all $(\widehat{e}, e)$-pairs in order to obtain $A_{P,Q}(\tau)$.

$$\mathbb{C}^{P(\widehat{e})}_{Q(e)}(\tau) = \sum_{p_i\in P(\widehat{e})} C^{(p_i)}_{Q(e)}(\tau) \tag{13}$$

However, instead of evaluating $C^{(p_i)}_{Q(e)}(\tau)$ for each p_i on $\widehat{e} = (a, b)$ (see those blue data points in Figure 7) for obtaining $\mathbb{C}^{P(\widehat{e})}_{Q(e)}(\tau)$, we reverse the evaluation order by considering each p_k on $e = (u, v)$ (see those light pink data points in Figure 7) for computing $C^{(p_k)}_{P(\widehat{e})}(\tau)$ in order to obtain $\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau)$, where

$$\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau) = \sum_{p_k\in Q(e)} C^{(p_k)}_{P(\widehat{e})}(\tau) \tag{14}$$

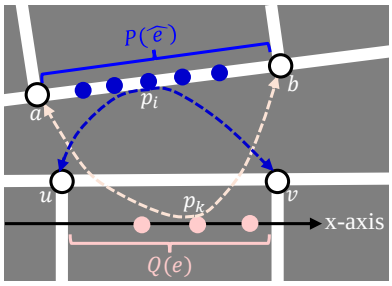


Figure 7: Reverse the evaluation order from p_i in $P(\widehat{e})$ to the evaluation order from p_k in $Q(e)$.

Here, we show that reversing the evaluation order does not change the final result (see Lemma 5), i.e., $\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau)$ is the same as $\mathbb{C}^{P(\widehat{e})}_{Q(e)}(\tau)$. The proof of this lemma is in Section 7.6.

Lemma 5. *Consider two sets of data points, P and Q, and two edges, $\widehat{e}$ and e. We have $\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau) = \mathbb{C}^{P(\widehat{e})}_{Q(e)}(\tau)$.*

Establishment of sorted threshold positions. To compute $\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau)$ (see Equation 14) with D thresholds, $\tau_1, \tau_2, ..., \tau_D$, we count data points in $P(\widehat{e})$ on $\widehat{e} = (a, b)$ in order to evaluate $C^{(p_k)}_{P(\widehat{e})}(\tau_\rho)$ for each (p_k, τ_ρ)-pair, where p_k is in $Q(e)$ and $1 \leq \rho \leq D$. Observe from Figure 8 that each (p_k, τ_ρ)-pair corresponds to two end positions on $\widehat{e} = (a, b)$.

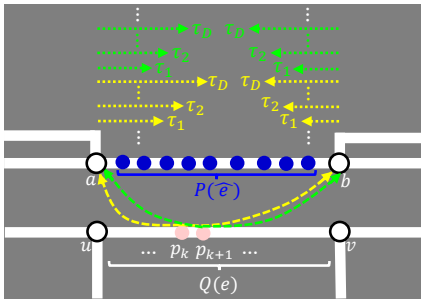


Figure 8: Two end positions (of the dotted arrows) of each (p_k, τ_ρ)-pair can be established on $\widehat{e} = (a, b)$ if the shortest path distances from p_k to a and b have been computed.

By considering the dotted arrows on $\widehat{e} = (a, b)$ from a, i.e., the left dotted arrows in Figure 8, we aim to maintain the sorted order of these end positions based on the distances with respect to a. Observe from Figure 9 that the D end positions, which correspond to D thresholds, must be initially sorted with respect to a for each p_k of $Q(e)$ (e.g., D red/blue/purple/green dotted arrows) since we have $\tau_1 \leq \tau_2 \leq ... \leq \tau_D$. Therefore, by iteratively merging two (consecutive) sorted lists into one sorted list in each level, we obtain the sorted list of $D|Q(e)|$ end positions in $O(D|Q(e)|\log|Q(e)|)$ time (see Lemma 6 and its proof in Section 7.7).

Lemma 6. *Consider each data point p_k in $Q(e)$ and its D end positions from the node a in the edge $\widehat{e} = (a, b)$. Obtaining the sorted order of these $D|Q(e)|$ end positions with respect to the distance values to a takes $O(D|Q(e)|\log|Q(e)|)$ time.*

Similarly, we can also obtain the sorted end positions for those $D|Q(e)|$ dotted arrows from b (see Figure 8) with the same time complexity.

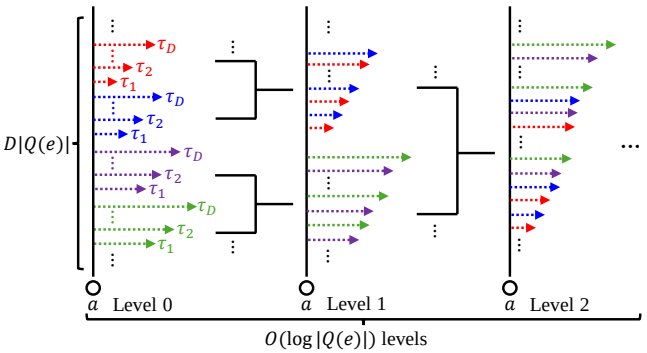


Figure 9: Iteratively merging $D|Q(e)|$ thresholds into the sorted order based on the distances with respect to a.

Sweep line processing. After the end positions are sorted on $\widehat{e} = (a, b)$ based on the distances to a and the distances to b, we can maintain two sweep lines, l_a and l_b, which scan, from a to b and from b to a, respectively, those blue data points and the end positions (see Figure 10). Each sweep line maintains the count value of blue data points during the scanning process. Once the sweep line l_a (or l_b) reaches the end position that corresponds to the (p_k, τ_ρ)-pair, it stores the value $l_a(p_k, \tau_\rho)$ (or $l_b(p_k, \tau_\rho)$) for that pair. Using Figure 10 as an example, $l_a(p_k, \tau_\rho) = 5$ and $l_b(p_k, \tau_\rho) = 4$.

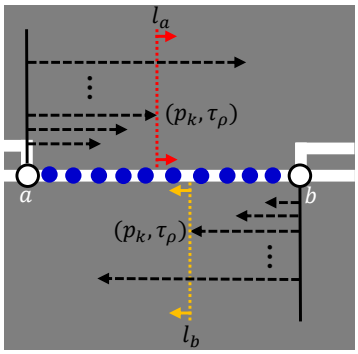


Figure 10: Using two sweep lines from a and b, which are l_a and l_b, respectively, to evaluate $\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau)$ with D thresholds, $\tau_1, \tau_2, ..., \tau_D$.

With these two values, $l_a(p_k, \tau_\rho)$ and $l_b(p_k, \tau_\rho)$, we can compute $C^{(p_k)}_{P(\widehat{e})}(\tau_\rho)$ based on the following equation.

$$C^{(p_k)}_{P(\widehat{e})}(\tau_\rho) = \begin{cases} l_a(p_k, \tau_\rho) + l_b(p_k, \tau_\rho) & \text{if } l_a(p_k, \tau_\rho) + l_b(p_k, \tau_\rho) < |P(\widehat{e})| \\ |P(\widehat{e})| & \text{otherwise} \end{cases} \tag{15}$$

Based on Equation 14, we can further conclude that this sweep line processing approach can compute $\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau)$ with D thresholds, $\tau_1, \tau_2, ..., \tau_D$, in $O(D|Q(e)| + |P(\widehat{e})|)$ time if the end positions are sorted (see Lemma 7 and its proof in Section 7.8).

Lemma 7. *If the end positions in the edge $\widehat{e} = (a, b)$ are sorted, the time complexity of computing $\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau)$ is $O(D|Q(e)| + |P(\widehat{e})|)$.*

STARS for generating the network K-function plot. With the core ideas of sampling and threshold sharing, we illustrate how we can efficiently generate an approximate network K-function plot.

Consider a location dataset P and D thresholds, $\tau_1, \tau_2, ..., \tau_D$. We first obtain the sampled dataset Q for P in a road network G (see Figure 6), which takes $O(|E| + n \log n + \frac{|E|}{\epsilon})$ (see Lemma 4). Then, we compute the approximate network K-function value $A_{P,Q}(\tau)$ with Equation 16 (based on Equations 11, 13, and 14 and Lemma 5).

$$A_{P,Q}(\tau) = \frac{1}{n(n-1)} \sum_{e \in E} \sum_{\widehat{e} \in E} \frac{|P(e)|}{|Q(e)|} C^{Q(e)}_{P(\widehat{e})}(\tau) \tag{16}$$

By calling the shortest path distances from u and v on $e = (u, v)$ to other nodes, e.g., a and b on $\widehat{e} = (a, b)$, which takes $O(T_{\text{SP}})$ time for each edge (i.e., $|E|T_{\text{SP}}$ in total), we can compute $C^{Q(e)}_{P(\widehat{e})}(\tau)$ by first establishing the threshold positions (see Figures 8 and 9) and then adopting the sweep line processing approach (see Figure 10) in $\widehat{e}$, which takes $O(D|Q(e)| \log |Q(e)| + |P(\widehat{e})|)$ time (based on Lemmas 6 and 7). Hence, we conclude that computing $A_{P,Q}(\tau)$ with D thresholds takes $O\big(|E|T_{\text{SP}} + \frac{D|E|^2}{\epsilon} \log\big(\frac{1}{\epsilon}\big) + n|E| + n \log n\big)$ time (see Lemma 8 and its proof in Section 7.9).

Lemma 8. *Given a road network $G = (V, E)$, a location dataset P with size n, the time complexity for computing $A_{P,Q}(\tau)$ with D thresholds, $\tau_1, \tau_2, ..., \tau_D$, is $O\big(|E|T_{SP} + \frac{D|E|^2}{\epsilon} \log\big(\frac{1}{\epsilon}\big) + n|E| + n \log n\big)$.*

Since we need to handle L datasets in order to compute the network K-function plot, the time complexity of generating this plot is $O\big(|E|T_{\text{SP}} + \frac{LD|E|^2}{\epsilon} \log\big(\frac{1}{\epsilon}\big) + Ln|E| + Ln \log n\big)$ (see Theorem 3).[3] Due to its simplicity, we omit this proof.

Theorem 3. *The time complexity of generating an approximate network K-function plot is $O\big(|E|T_{SP} + \frac{LD|E|^2}{\epsilon} \log\big(\frac{1}{\epsilon}\big) + Ln|E| + Ln \log n\big)$.*

Here, we further state in Theorem 4 that the space complexity of STARS for generating an approximate network K-function plot is $O\big(|E| + Ln + \frac{D}{\epsilon} + S_{\text{SP}}\big)$, which is slightly larger than the existing NS method (i.e., $O(|E| + Ln + S_{\text{SP}})$ in Table 1). The proof of this theorem can be found in Section 7.10.

Theorem 4. *The space complexity of generating an approximate network K-function plot is $O\big(|E| + Ln + \frac{D}{\epsilon} + S_{SP}\big)$.*

As a remark, the detailed pseudocode of STARS and further discussion of NS and STARS can be found in Sections S2 and S3 of [69], respectively.

4 Experimental Evaluation

In this section, we first discuss the experimental settings in Section 4.1. Then, we test the time efficiency of all methods in Section 4.2. Next, we measure the practical accuracy of all methods in Section 4.3. Lastly, we conduct a case study in Section 4.4. Additional experiments, including additional time efficiency experiments, space efficiency experiments, and additional accuracy experiments, are provided in Sections S4 to S6 of [69].

[3] By adopting the advanced shortest path sharing approach in [20], we do not need to multiply $|E|T_{\text{SP}}$ by L.

4.1 Experimental Settings

In our experiments, we compare our method, STARS, with the state-of-the-art NS method (see Section 2.2) as well as the traditional RQS and SPS methods (as mentioned in [20]). All of these methods were implemented in C++, and experiments were conducted on a machine equipped with a 12$^{\text{th}}$-generation Intel Core i5 2.5 GHz CPU and 32GB of RAM.

Datasets. Table 2 summarizes four large-scale location datasets in different regions. To obtain each dataset, we utilize the OSMnx python library [14] to first extract the corresponding road network and then map those data points on it.

Table 2: Datasets.

Dataset	$\|V\|$	$\|E\|$	n	Category
Maryland [4]	32,100	46,221	1,830,715	Traffic violations
New York [5]	71,268	121,030	5,430,525	911 calls
San Francisco [6]	47,983	74,856	6,053,150	311 calls
Chicago [2]	23,670	41,226	8,189,459	Crime events

Parameter settings. To generate a network K-function plot with D thresholds, we utilize Equation 17 to obtain those thresholds, τ_1, τ_2,..., τ_D, in the range $[0, r]$ meters, where

$$\tau_\rho = \frac{\rho r}{D} \tag{17}$$

By default, we choose $L = 5$ datasets (with four randomly generated datasets and one original dataset), $D = 20$ thresholds in the range of $[0, 2000]$ meters (i.e., $r = 2000$), and $\epsilon = 0.1$ for STARS in order to conduct the experiments.

Measurement. We measure the response time (sec) to test the time efficiency of all methods. Note that we omit the results that take more than 86,400 sec (i.e., one day). In addition, we also use two commonly used metrics, Absolute Error (AE) (see Equation 18) and Relative Error (RE) (see Equation 19), for comparing the accuracy between the approximation method, STARS, and the exact methods.

$$AE(\tau) = |A_{P,Q}(\tau) - K_P(\tau)| \tag{18}$$

$$RE(\tau) = \frac{|A_{P,Q}(\tau) - K_P(\tau)|}{K_P(\tau)} \tag{19}$$

4.2 Time Efficiency Experiments

Here, we conduct two experiments to test the time efficiency of all methods. Other experiments for varying the number of datasets L and the maximum range of τ are provided in Section S4 of [69].

Varying the number of thresholds D**.** We test how the number of thresholds D affects the response time of each method. To conduct this experiment, we choose five D values, which are 20, 40, 60, 80, and 100, for testing. Figure 11 shows the results of all methods. Due to the lower time complexity of STARS, this method achieves 2.89x to 13.7x speedups compared with the state-of-the-art NS method. Furthermore, the larger the thresholds D, the larger the time gap between NS and STARS. The main reason is that the time complexity of STARS only depends on $Ln|E|$ (see Theorem 3) but not $LDn|E|$ (like NS in Table 1), which implies that the response time of STARS does not increase significantly with respect to D.

Varying the error parameter ϵ**.** We proceed to examine how the error parameter ϵ affects the response time of each method, by choosing five ϵ values (i.e., 0.05, 0.1, 0.15, 0.2, and 0.25). Figure 12 shows the results of all methods. Since RQS, SPS, and NS are exact, they are not sensitive to this parameter. In addition, the smaller the ϵ value, the larger the response time of STARS (based on Theorem 3). Note that STARS can still achieve 1.52x to 21.57x speedups compared with NS in these error parameters.

4.3 Practical Accuracy Experiments

In this section, we conduct the following experiments to compare the subjective accuracy and the objective accuracy of our approximation method, STARS, with the exact methods (i.e., RQS, SPS, and NS). Additional experiments can also be found in Section S6 of [69].

Subjective accuracy. Here, we compare the network K-function plot of the exact and approximation methods with respect to different thresholds in the range of [0, 2000] (meters). Observe from Figure 13 that the curves of the approximation method overlap with the curves of the corresponding exact method. This verifies that STARS does not degrade the result quality of generating a network K-function plot because STARS can achieve non-trivial error guarantee (see Theorem 1).

Objective accuracy. We proceed to investigate the objective accuracy of generating a network K-function plot with respect to each dataset. Figure 14 shows the absolute error (see Equation 18) and the relative error (see Equation 19) of using STARS for generating the network K-function plot with respect to different threshold values τ in the range [0, 2000] (meters) for each original dataset P. Since STARS exhibits the non-trivial error guarantee (see Theorem 1), this method only incurs a small absolute error and a small relative error for each dataset, ranging from 3.874×10^{-9} to 3.507×10^{-6} and 1.118×10^{-6} to 5.197×10^{-3}, respectively, no matter which τ we choose. These results further explain why the approximate network K-function values that are computed by STARS in Figure 13 are nearly the same as the ones of the exact method.

4.4 Case Study

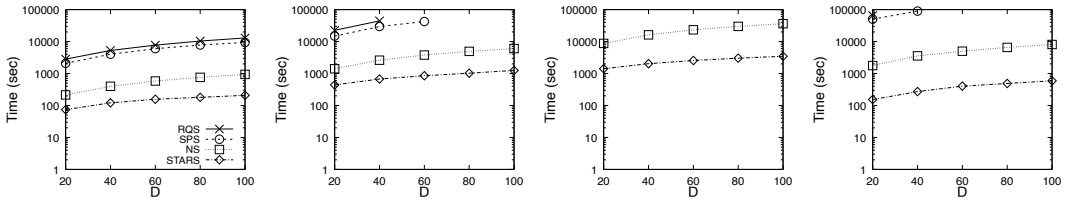
We conduct a case study to (1) demonstrate how accurately the approximation method (STARS) can generate the network K-function plot compared with any exact method (e.g., RQS, SPS, and NS) and (2) simultaneously adopt the approximate network K-function plot with the network kernel density visualization (NKDV) [8, 18], which is another spatial point pattern analysis tool, to understand the potential hotspots in Chicago crime events (see Table 2).

Figure 15 shows the network K-function plots (see Problem 1) that are generated based on the exact and approximation methods under the default settings (see Section 4.1). Observe that the approximation method, STARS, can generate nearly the same curves as the ones based on any exact method, e.g., RQS, SPS, and NS, due to its accuracy guarantee (see Theorem 1). With the approximate network K-function plot, we conclude that Chicago crime events tend to be clustered when the threshold τ_ρ is set to be within 2000m.

Here, we further utilize NKDV to generate the hotspot map for Chicago crime events. To adopt this tool, we first divide each edge in this road network G into a set of small lixels q (or line segments) and then color each lixel q based on the network kernel density function $\mathcal{F}_P(q)$ (see Equation 20).

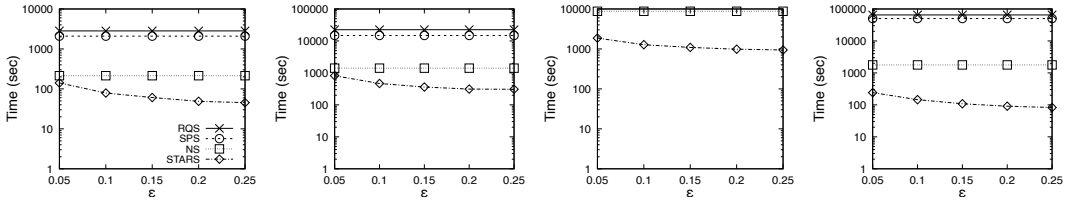
$$\mathcal{F}_P(q) = \frac{1}{n} \sum_{p \in P} \begin{cases} 1 - \frac{1}{b_G^2} d_G(q,p)^2 & \text{if } d_G(q,p) \le b_G \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

where b_G denotes the bandwidth parameter.



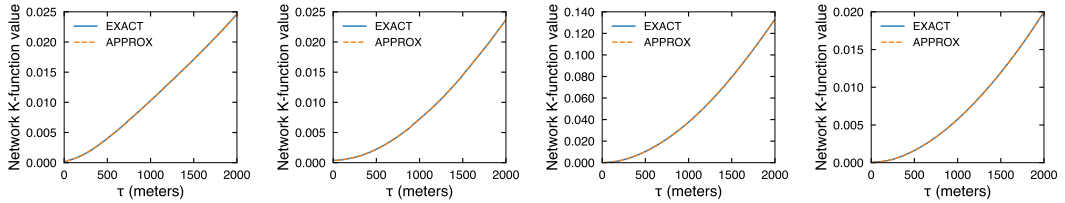
(a) Maryland (b) New York (c) San Francisco (d) Chicago

Figure 11: Response time for computing the network K-function plot, varying the number of thresholds D.



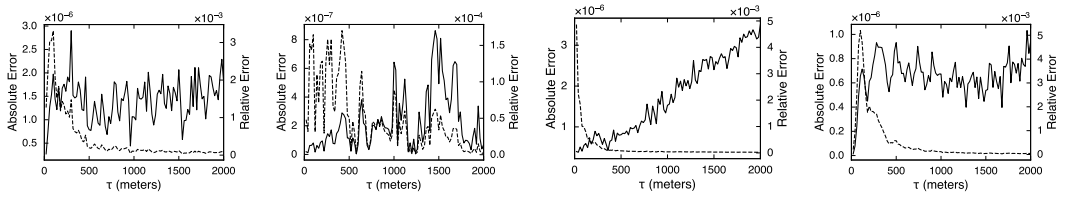
(a) Maryland (b) New York (c) San Francisco (d) Chicago

Figure 12: Response time for computing the network K-function plot, varying the error parameter ϵ.



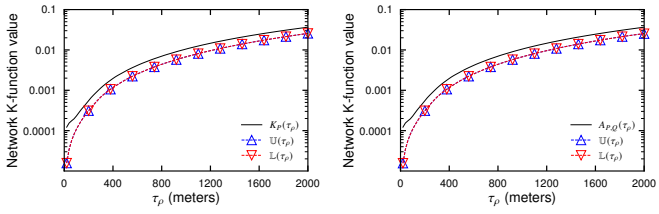
(a) Maryland (b) New York (c) San Francisco (d) Chicago

Figure 13: Generating a network K-function plot with the original location dataset P using the exact and approximation methods, varying the threshold τ.



(a) Maryland (b) New York (c) San Francisco (d) Chicago

Figure 14: Absolute error (solid line) and relative error (dashed line) of STARS for generating a network K-function plot with the original dataset P using different threshold values in the range of $[0, 2000]$ (meters).



(a) Exact (b) Approximation

Figure 15: Generating the network K-function plots, based on the exact and approximation methods, for Chicago crime events.

Since Chicago crime events tend to have meaningful clusters within 2000m. we also set the bandwidth parameter b_G to be 200m (within this range) for detecting hotspots (see Figure 16). Observe that those hotspot regions (red regions) tend to have sizes that are within 2000m, which indicates that they are meaningful (or worth for further investigation).

Figure 16: Generating a hotspot map (based on NKDV) for Chicago crime events, where we zoom in to the West Lakeview region and adopt b_G = 200m.

5 Related Work

In this section, we discuss four camps of research studies that are mostly related to this work.

Efficient algorithms for network K-function. Although network K-function [12, 42–45] has been proposed for more than two decades, those research studies that tackle the efficiency issues of this tool are only proposed recently. Rakshit et al. [55] utilize the shortest path sharing method (SPS) to reduce the number of shortest path computations. Chan et al. [20] further develop the fastest solution, called neighbor sharing (NS), which can achieve the best performance for generating an exact network K-function plot. Compared with these two methods, this is the first work that develops the sampling and threshold sharing solution (STARS) in order to further reduce the time complexity for supporting this tool with a non-trivial accuracy guarantee (see Table 1). Therefore, STARS can significantly improve the efficiency of generating the network K-function plot without degrading its result (see Section 4, and Sections S4 and S6 of [69]) compared with these two methods.

Efficient algorithms for network kernel density visualization. Recently, Chan et al. [18, 22] also develop efficient algorithms for another point pattern analysis tool, called network kernel density visualization (NKDV), which aims to generate a visualization based on the network kernel density function value for each lixel (i.e., linear pixel) on a road network. Note that the network kernel density function is also based on the aggregation of kernel functions, which is similar to the network K-function (see Equation 1). However, there are several weakness of using these methods [18, 22] for handling the network K-function. First, Chan et al. [20] have already attempted to develop the count augmentation (CA) method, which is based on the ADA method in [18], for improving the efficiency of computing the network K-function. However, this CA method is inferior compared with the NS method in terms of theoretical complexity result and practical performance. Second, generating the network K-function plot does not involve the concept of lixels in NKDV. As such, LION [22], which is based on this concept for efficiency improvement, cannot be extended for supporting the network K-function.

Efficient algorithms for kernel density estimation. Recall from Lemma 2 that computing $C_{P(e)}^{(p_i)}(\tau)$ (see Equation 4) is equivalent to evaluating one-dimensional KDE functions. Therefore, fast algorithms for computing the KDE function can improve the efficiency of generating the network K-function plot, which can be categorized into two classes, namely (1) function approximation and (2) data sampling. Function approximation approach [17, 19, 21, 28, 30] utilizes some simple functions to approximate the more complicated KDE function with non-trivial accuracy guarantees. However, this approach cannot reduce the time complexity for computing this function. Data sampling approach [31, 32, 50–53, 62, 73, 74] aims to obtain the sampled dataset for the original dataset in order to improve the efficiency for computing the KDE function using the (smaller) sampled dataset. Since this approach can theoretically (i) reduce the dataset size and (ii) achieve the non-trivial accuracy guarantee (see Lemma 1), we adopt this approach for computing $C_{P(e)}^{(p_i)}(\tau)$. In particular, we utilize the data sampling method in [73] as this method can theoretically achieve the smallest sample size for the KDE function with one-dimensional data. As a remark, this is the first work that observes the connection between the network K-function and the KDE function. In addition, simply combining the data sampling method with the state-of-the-art NS method cannot reduce the time complexity of generating the network K-function plot, while we need to further develop the threshold sharing approach in order to achieve this goal (see Section S3 of [69]).

Efficient shortest path algorithms. Observe from Equation 1 that calculating the network K-function involves computing the shortest path distances. Therefore, fast shortest path algorithms can possibly improve the efficiency of supporting this tool, which can be divided into two categories, namely index-free algorithms and index-based algorithms. Index-free algorithms, which do not build any index on a road network, include the Dijkstra's algorithm [24], A* search [9, 25, 58], and bidirectional search [60, 61]. Index-based algorithms aim to build some index structures on a road network, e.g., contraction hierarchies [13, 29, 47–49, 59, 66, 75], hub labeling [23, 36, 37, 47, 48, 72], and shortest path caching [35, 63, 71]. However, most of these methods (e.g., A* search, bidirectional search, hub labeling, and shortest path caching) only improve the efficiency of solving the single-source-single-destination shortest path problem, which cannot be used for computing the network K-function (as it is based on range search). In addition, as shown in Figure 11, the time gap between SPS and NS/STARS is much larger than the time gap between RQS and SPS, which indicates that the main bottleneck of generating the network K-function plot is to process those data points on a road network (instead of processing the shortest path distances). Therefore, we still adopt the Dijkstra's algorithm for finding the shortest path distances in this work.

6 Conclusion

In this paper, we study the network K-function analysis tool, which has been widely adopted in many application domains, including transportation science, criminology, and urban planning. However, this analysis tool is computationally demanding, which does not scale to large location datasets. Although the state-of-the-art NS method can reduce the time complexity for supporting this tool, this complexity result is still high, leading to high response time for using this tool. To tackle this issue, this work proposes the first approximate solution, called STARS, which (1) further lowers the time complexity, (2) retains the similar space complexity, and (3) achieves a non-trivial ϵ-absolute error guarantee. Our experimental results show that STARS can further achieve (1) 1.52x to 21.57x speedups, (2) negligible additional space overhead, and (3) negligible accuracy degradation compared with NS.

In the future, we will develop the QGIS/ArcGIS plugin for this solution. In addition, we will extend STARS to handle other types of network analysis tools, e.g., network clustering [44, 70].

Acknowledgments

This work was supported by the Natural Science Foundation of China under grants 62572326, 23IAA00610, 62372308, and 42471496, Scientific Foundation for Youth Scholars of Shenzhen University, the Science and Technology Development Fund Macau SAR (0003/2023/RIC, 0052/2023/RIA1, 0011/2025/RIC, 001/2024/SKL for SKL-IOTSC), and the Center for Scientific Research and Development in Higher Education Institutes, MOE (2024HT013). 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 Data Protal. https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2/about_data.

[3] [n. d.]. FastNSA. https://pypi.org/project/fastnsa/.

[4] [n. d.]. Maryland Traffic Violations Data Protal. https://data.montgomerycountymd.gov/Public-Safety/Traffic-Violations/4mse-ku6q/about_data.

[5] [n. d.]. New York Data Protal. https://data.cityofnewyork.us/Public-Safety/NYPD-Calls-for-Service-Year-to-Date-/n2zq-pubd/about_data.

[6] [n. d.]. San Francisco Data Protal. https://data.sfgov.org/City-Infrastructure/311-Cases/vw6y-z8j6/about_data.

[7] [n. d.]. SANET. http://sanet.csis.u-tokyo.ac.jp/.

[8] [n. d.]. spNetwork: Spatial Analysis on Network - R Project. https://cran.r-project.org/web/packages/spNetwork/spNetwork.pdf.

[9] Saman Ahmadi, Andrea Raith, Guido Tack, and Mahdi Jalili. 2024. Resource Constrained Pathfinding with Enhanced Bidirectional A* Search. *CoRR* abs/2412.13888 (2024). https://doi.org/10.48550/ARXIV.2412.13888 arXiv:2412.13888

[10] Amira K. Al-Aamri, Graeme Hornby, Li-Chun Zhang, Abdullah A. Al-Maniri, and Sabu S. Padmadas. 2021. Mapping road traffic crash hotspots using GIS-based methods: A case study of Muscat Governorate in the Sultanate of Oman. *Spatial Statistics* 42 (2021), 100458.

[11] 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.

[12] Adrian Baddeley, Ege Rubak, and Rolf Turner. 2015. *Spatial Point Patterns: Methodology and Applications with R.* Chapman and Hall/CRC Press, London. https://www.routledge.com/Spatial-Point-Patterns-Methodology-and-Applications-with-R/Baddeley-Rubak-Turner/9781482210200/

[13] Reinhard Bauer, Daniel Delling, Peter Sanders, Dennis Schieferdecker, Dominik Schultes, and Dorothea Wagner. 2010. Combining hierarchical and goal-directed speed-up techniques for dijkstra's algorithm. *ACM J. Exp. Algorithmics* 15 (2010). https://doi.org/10.1145/1671970.1671976

[14] Geoff Boeing. 2017. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems* 65 (2017), 126 – 139.

[15] Noli Brazil. 2025 (Online). School Closures and the Spatial Ecology of Education Access in 10 U.S. Cities. *Geographical Analysis* (2025 (Online)). https://doi.org/10.1111/gean.12414

[16] Jiannan Cai, Min Deng, Qiliang Liu, Zhanjun He, Jianbo Tang, and Xuexi Yang. 2019. Nonparametric Significance Test for Discovery of Network-Constrained Spatial Co-Location Patterns. *Geographical Analysis* 51, 1 (2019), 3–22. https://doi.org/10.1111/gean.12155 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/gean.12155

[17] Tsz Nam Chan, Reynold Cheng, and Man Lung Yiu. 2020. QUAD: Quadratic-Bound-based Kernel Density Visualization. In *SIGMOD*. 35–50. https://doi.org/10.1145/3318464.3380561

[18] 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.

[19] Tsz Nam Chan, Leong Hou U, Reynold Cheng, Man Lung Yiu, and Shivansh Mittal. 2022. Efficient Algorithms for Kernel Aggregation Queries. *IEEE Trans. Knowl. Data Eng.* 34, 6 (2022), 2726–2739. https://doi.org/10.1109/TKDE.2020.3018376

[20] 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.

[21] Tsz Nam Chan, Man Lung Yiu, and Leong Hou U. 2019. KARL: Fast Kernel Aggregation Queries. In *ICDE*. 542–553. https://doi.org/10.1109/ICDE.2019.00055

[22] 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

[23] Edith Cohen, Eran Halperin, Haim Kaplan, and Uri Zwick. 2003. Reachability and Distance Queries via 2-Hop Labels. *SIAM J. Comput.* 32, 5 (2003), 1338–1355. https://doi.org/10.1137/S0097539702403098

[24] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, 3rd Edition.* MIT Press. http://mitpress.mit.edu/books/introduction-algorithms

[25] Rina Dechter and Judea Pearl. 1985. Generalized Best-First Search Strategies and the Optimality of A*. *J. ACM* 32, 3 (1985), 505–536. https://doi.org/10.1145/3828.3830

[26] Matthias Eckardt and Jorge Mateu. 2018. Point Patterns Occurring on Complex Structures in Space and Space-Time: An Alternative Network Approach. *Journal of Computational and Graphical Statistics* 27, 2 (2018), 312–322.

[27] Suoyao Feng, Aobo Wang, Zong Tian, and Seri Park. 2024. Exploring the correlation between hard-braking events and traffic crashes in regional transportation networks: A geospatial perspective. *Multimodal Transportation* 3, 2 (2024), 100128. https://doi.org/10.1016/j.multra.2024.100128

[28] Edward Gan and Peter Bailis. 2017. Scalable Kernel Density Classification via Threshold-Based Pruning. In *ACM SIGMOD*. 945–959.

[29] Robert Geisberger, Peter Sanders, Dominik Schultes, and Daniel Delling. 2008. Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In *WEA*. 319–333. https://doi.org/10.1007/978-3-540-68552-4_24

[30] Alexander G. Gray and Andrew W. Moore. 2003. Nonparametric Density Estimation: Toward Computational Tractability. In *SDM*. 203–211.

[31] Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. 2012. A Kernel Two-Sample Test. *J. Mach. Learn. Res.* 13 (2012), 723–773. https://doi.org/10.5555/2503308.2188410

[32] Sarang C. Joshi, Raj Varma Kommaraju, Jeff M. Phillips, and Suresh Venkatasubramanian. 2011. Comparing distributions and shapes using the kernel distance. In *SCG*. ACM, 47–56. https://doi.org/10.1145/1998196.1998204

[33] Yunfan Kang, Fangzheng Lyu, and Shaowen Wang. 2024. NetPointLib: Library for Large-Scale Spatial Network Point Data Fusion and Analysis. In *PEARC*. ACM, 43:1–43:4. https://doi.org/10.1145/3626203.3670615

[34] David S. Lamb, Joni A. Downs, and Chanyoung Lee. 2016. The network K-function in context: examining the effects of network structure on the network K-function. *Transactions in GIS* 20, 3 (2016), 448–460. https://doi.org/10.1111/tgis.12157 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/tgis.12157

[35] Lei Li, Mengxuan Zhang, Wen Hua, and Xiaofang Zhou. 2020. Fast Query Decomposition for Batch Shortest Path Processing in Road Networks. In *ICDE*. 1189–1200. https://doi.org/10.1109/ICDE48307.2020.00107

[36] Wentao Li, Miao Qiao, Lu Qin, Ying Zhang, Lijun Chang, and Xuemin Lin. 2019. Scaling Distance Labeling on Small-World Networks. In *SIGMOD*. ACM, 1060–1077. https://doi.org/10.1145/3299869.3319877

[37] Ye Li, Leong Hou U, Man Lung Yiu, and Ngai Meng Kou. 2017. An Experimental Study on Hub Labeling based Shortest Path Algorithms. *Proc. VLDB Endow.* 11, 4 (2017), 445–457. https://doi.org/10.1145/3186728.3164141

[38] Becky P.Y. Loo and Shenjun Yao. 2013. The identification of traffic crash hot zones under the link-attribute and event-based approaches in a network-constrained environment. *Computers, Environment and Urban Systems* 41 (2013), 249–261. https://doi.org/10.1016/j.compenvurbsys.2013.07.001

[39] Yongmei Lu and Xuwei Chen. 2007. On the false alarm of planar K-function when analyzing urban crime distributed along streets. *Social Science Research* 36, 2 (2007), 611–632. https://doi.org/10.1016/j.ssresearch.2006.05.003

[40] Mehdi Moradi and Ali Sharifi. 2024. Summary statistics for spatio-temporal point processes on linear networks. *Spatial Statistics* 61 (2024), 100840. https://doi.org/10.1016/j.spasta.2024.100840

[41] Luke Muggy, Max Griswold, Florentine Eloundou Nekoul, Sean McKenna, Rosanna Smart, and Priscillia Hunt. 2024. Accounting for socio-economic context in quantifying the attractive and repellent influence of built environment on firearms violence in multiple cities. *Journal of Quantitative Criminology* 40, 1 (2024), 1–32.

[42] Atsuyuki Okabe, Kei-Ichi Okunuki, and Shino Shiode. 2006. SANET: A Toolbox for Spatial Analysis on a Network. *Geographical Analysis* 38, 1 (2006), 57–66.

[43] Atsuyuki Okabe, Kei-Ichi Okunuki, and Shino Shiode. 2006. The SANET Toolbox: New Methods for Network Spatial Analysis. *Transactions in GIS* 10, 4 (2006), 535–550.

[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] Atsuyuki Okabe and Ikuho Yamada. 2001. The K-Function Method on a Network and Its Computational Implementation. *Geographical Analysis* 33, 3 (2001), 271–290. https://doi.org/10.1111/j.1538-4632.2001.tb00448.x

[46] E. Okwori, M. Viklander, and A. Hedström. 2021. Spatial heterogeneity assessment of factors affecting sewer pipe blockages and predictions. *Water Research* 194 (2021), 116934. https://doi.org/10.1016/j.watres.2021.116934

[47] Dian Ouyang, Lu Qin, Lijun Chang, Xuemin Lin, Ying Zhang, and Qing Zhu. 2018. When Hierarchy Meets 2-Hop-Labeling: Efficient Shortest Distance Queries on Road Networks. In *SIGMOD*. ACM, 709–724. https://doi.org/10.1145/3183713.3196913

[48] Dian Ouyang, Dong Wen, Lu Qin, Lijun Chang, Xuemin Lin, and Ying Zhang. 2023. When hierarchy meets 2-hop-labeling: efficient shortest distance and path queries on road networks. *VLDB J.* 32, 6 (2023), 1263–1287. https://doi.org/10.1007/S00778-023-00789-X

[49] Dian Ouyang, Long Yuan, Lu Qin, Lijun Chang, Ying Zhang, and Xuemin Lin. 2020. Efficient Shortest Path Index Maintenance on Dynamic Road Networks with Theoretical Guarantees. *Proc. VLDB Endow.* 13, 5 (2020), 602–615. https://doi.org/10.14778/3377369.3377371

[50] Jeff M. Phillips. 2013. ϵ-Samples for Kernels. In *SODA*. 1622–1632. https://doi.org/10.1137/1.9781611973105.116

[51] Jeff M. Phillips and Wai Ming Tai. 2018. Improved Coresets for Kernel Density Estimates. In *SODA*. 2718–2727. https://doi.org/10.1137/1.9781611975031.173

[52] Jeff M. Phillips and Wai Ming Tai. 2018. Near-Optimal Coresets of Kernel Density Estimates. In *SOCG*. 66:1–66:13. https://doi.org/10.4230/LIPIcs.SoCG.2018.66

[53] Jeff M. Phillips and Wai Ming Tai. 2020. Near-Optimal Coresets of Kernel Density Estimates. *Discret. Comput. Geom.* 63, 4 (2020), 867–887. https://doi.org/10.1007/S00454-019-00134-6

[54] QGIS Development Team. 2009. *QGIS Geographic Information System.* Open Source Geospatial Foundation. http://qgis.osgeo.org

[55] Suman Rakshit, Adrian Baddeley, and Gopalan Nair. 2019. Efficient Code for Second Order Analysis of Events on a Linear Network. *Journal of Statistical Software, Articles* 90, 1 (2019), 1–37. https://doi.org/10.18637/jss.v090.i01

[56] Brian D Ripley. 2005. *Spatial statistics.* John Wiley & Sons.

[57] Yikang Rui, Zaigui Yang, Tianlu Qian, Shoaib Khalid, Nan Xia, and Jiechen Wang. 2016. Network-constrained and category-based point pattern analysis for Suguo retail stores in Nanjing, China. *International Journal of Geographical Information Science* 30, 2 (2016), 186–199.

[58] Stuart J Russell and Peter Norvig. 2016. *Artificial intelligence: a modern approach.* Pearson.

[59] Peter Sanders and Dominik Schultes. 2012. Engineering highway hierarchies. *ACM J. Exp. Algorithmics* 17, 1 (2012). https://doi.org/10.1145/2133803.2330080

[60] Shahaf S. Shperberg, Ariel Felner, Nathan R. Sturtevant, Solomon Eyal Shimony, and Avi Hayoun. 2019. Enriching Non-Parametric Bidirectional Search Algorithms. In *AAAI*. 2379–2386. https://doi.org/10.1609/AAAI.V33I01.33012379

[61] Nathan R. Sturtevant and Ariel Felner. 2018. A Brief History and Recent Achievements in Bidirectional Search. In *AAAI*. 8000–8007. https://doi.org/10.1609/AAAI.V32I1.12218

[62] Wai Ming Tai. 2022. Optimal Coreset for Gaussian Kernel Density Estimation. In *SoCG (LIPIcs, Vol. 224)*. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 63:1–63:15. https://doi.org/10.4230/LIPICS.SOCG.2022.63

[63] Jeppe Rishede Thomsen, Man Lung Yiu, and Christian S. Jensen. 2012. Effective caching of shortest paths for location-based services. In *SIGMOD*. 313–324. https://doi.org/10.1145/2213836.2213872

[64] 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

[65] Zhensheng Wang and Ke Nie. 2019. Measuring Spatial Patterns of Health Care Facilities and Their Relationships with Hypertension Inpatients in a Network-Constrained Urban System. *International Journal of Environmental Research and Public Health* 16, 17 (2019). https://www.mdpi.com/1660-4601/16/17/3204

[66] Lingkun Wu, Xiaokui Xiao, Dingxiong Deng, Gao Cong, Andy Diwen Zhu, and Shuigeng Zhou. 2012. Shortest Path and Distance Queries on Road Networks: An Experimental Evaluation. *Proc. VLDB Endow.* 5, 5 (2012), 406–417. https://doi.org/10.14778/2140436.2140438

[67] Zelong Xia, Hao Li, Yuehong Chen, and Wenhao Yu. 2019. Detecting urban fire high-risk regions using colocation pattern measures. *Sustainable Cities and Society* 49 (2019), 101607. https://doi.org/10.1016/j.scs.2019.101607

[68] Ikuho Yamada and Jean-Claude Thill. 2004. Comparison of planar and network K-functions in traffic accident analysis. *Journal of Transport Geography* 12, 2 (2004), 149–158. https://doi.org/10.1016/j.jtrangeo.2003.10.006

[69] Hongwei Ye, Tsz Nam Chan, Leong Hou U, Dingming Wu, Wei Tu, Ruisheng Wang, and Joshua Zhexue Huang. 2026. Supplementary Material for "STARS: A Sampling and Threshold Sharing Solution for Network K-function Analytics". https://github.com/fastnsa/STARS/blob/main/supplementary_material.pdf.

[70] Man Lung Yiu and Nikos Mamoulis. 2004. Clustering Objects on a Spatial Network. In *SIGMOD*. 443–454. https://doi.org/10.1145/1007568.1007619

[71] Long Yuan, Kongzhang Hao, Xuemin Lin, and Wenjie Zhang. 2024. Batch Hop-Constrained s-t Simple Path Query Processing in Large Graphs. In *ICDE*. IEEE, 2557–2569. https://doi.org/10.1109/ICDE60146.2024.00201

[72] Mengxuan Zhang, Lei Li, Wen Hua, Rui Mao, Pingfu Chao, and Xiaofang Zhou. 2021. Dynamic Hub Labeling for Road Networks. In *ICDE*. IEEE, 336–347. https://doi.org/10.1109/ICDE51399.2021.00036

[73] Yan Zheng, Jeffrey Jestes, Jeff M. Phillips, and Feifei Li. 2013. Quality and efficiency for kernel density estimates in large data. In *SIGMOD*. 433–444.

[74] Yan Zheng and Jeff M. Phillips. 2015. L∞ Error and Bandwidth Selection for Kernel Density Estimates of Large Data. In *SIGKDD*. 1533–1542. https://doi.org/10.1145/2783258.2783357

[75] Andy Diwen Zhu, Hui Ma, Xiaokui Xiao, Siqiang Luo, Youze Tang, and Shuigeng Zhou. 2013. Shortest path and distance queries on road networks: towards bridging theory and practice. In *SIGMOD*. 857–868. https://doi.org/10.1145/2463676.2465277

7 Appendix

7.1 Proof of Lemma 2

Proof. Observe from Figure 5 that there are only two cases for the shortest paths (i.e., the green dashed line and the yellow dashed line) from p_i to the edge $e = (u, v)$.

Case 1 ($2\tau - d_G(p_i, u) - d_G(p_i, v) < d_G(u, v)$): In this case, two shortest paths do not intersect with each other. Based on Equation 4,[4] we have

$$\begin{aligned} C^{(p_i)}_{P(e)}(\tau) &= \sum_{p_j \in P(e)} \mathbb{I}(d_G(p_i, u) + d_G(u, p_j) \le \tau) + \\ &\quad \sum_{p_j \in P(e)} \mathbb{I}(d_G(p_i, v) + d_G(v, p_j) \le \tau) \\ &= \sum_{p_j \in P(e)} \mathbb{I}(d_G(p_i, u) + d_G(u, p_j) \le \tau) + \\ &\quad \sum_{p_j \in P(e)} \mathbb{I}(d_G(p_i, v) + d_G(u, v) - d_G(u, p_j) \le \tau) \end{aligned}$$

By substituting Equation 8 and Equation 9 into $C^{(p_i)}_{P(e)}(\tau)$, we have

$$\begin{aligned} C^{(p_i)}_{P(e)}(\tau) &= \sum_{p_j \in P(e)} \mathbb{I}(-x_{u,p_i} + x_{p_j} \le \tau) + \sum_{p_j \in P(e)} \mathbb{I}(x_{v,p_i} - x_{p_j} \le \tau) \\ &= \sum_{p_j \in P(e)} \mathbb{I}(d(x_{u,p_i}, x_{p_j}) \le \tau) + \sum_{p_j \in P(e)} \mathbb{I}(d(x_{v,p_i}, x_{p_j}) \le \tau) \end{aligned}$$

By regarding $P(e)$ as the set of one-dimensional data points, we have (based on Equation 5)

$$C^{(p_i)}_{P(e)}(\tau) = |P(e)|(KDE_{P(e)}(x_{u,p_i}) + KDE_{P(e)}(x_{v,p_i}))$$

Case 2 ($2\tau - d_G(p_i, u) - d_G(p_i, v) \ge d_G(u, v)$): In this case, two shortest paths must intersect with each other. Therefore, we have $C^{(p_i)}_{P(e)}(\tau) = |P(e)|$. □

7.2 Proof of Lemma 3

Proof. Based on Lemma 2, we consider these two cases, which are (1) $2\tau - d_G(p_i, u) - d_G(p_i, v) < d_G(u, v)$ and (2) $2\tau - d_G(p_i, u) - d_G(p_i, v) \ge d_G(u, v)$.

Case 1 ($2\tau - d_G(p_i, u) - d_G(p_i, v) < d_G(u, v)$): In this case, the following equations hold.

$$\begin{aligned} C^{(p_i)}_{P(e)}(\tau) &= |P(e)|(KDE_{P(e)}(x_{u,p_i}) + KDE_{P(e)}(x_{v,p_i})) \\ C^{(p_i)}_{Q(e)}(\tau) &= |Q(e)|(KDE_{Q(e)}(x_{u,p_i}) + KDE_{Q(e)}(x_{v,p_i})) \end{aligned}$$

Based on these two equations, we have

$$\begin{aligned} &\left| C^{(p_i)}_{P(e)}(\tau) - \frac{|P(e)|}{|Q(e)|} C^{(p_i)}_{Q(e)}(\tau) \right| \\ &= \left| |P(e)|(KDE_{P(e)}(x_{u,p_i}) + KDE_{P(e)}(x_{v,p_i})) - \right. \\ &\quad \left. \frac{|P(e)|}{|Q(e)|} |Q(e)|(KDE_{Q(e)}(x_{u,p_i}) + KDE_{Q(e)}(x_{v,p_i})) \right| \\ &= |P(e)||(KDE_{P(e)}(x_{u,p_i}) - KDE_{Q(e)}(x_{u,p_i})) + \\ &\quad (KDE_{P(e)}(x_{v,p_i}) - KDE_{Q(e)}(x_{v,p_i}))| \\ &\le |P(e)|(|KDE_{P(e)}(x_{u,p_i}) - KDE_{Q(e)}(x_{u,p_i})| + \\ &\quad |KDE_{P(e)}(x_{v,p_i}) - KDE_{Q(e)}(x_{v,p_i})|) \le 2\delta |P(e)| \end{aligned}$$

As a remark, the last inequality is based on Lemma 1 since $Q(e)$ is obtained by the SS method with the absolute error δ.

Case 2 ($2\tau - d_G(p_i, u) - d_G(p_i, v) \ge d_G(u, v)$): In this case, we have

$$C^{(p_i)}_{P(e)}(\tau) = |P(e)| \text{ and } C^{(p_i)}_{Q(e)}(\tau) = |Q(e)|$$

[4] Since p_i is not on the edge $e = (u, v)$, we omit the term $p_j \ne p_i$ in Equation 4.

Therefore, we can conclude that

$$\left|C^{(p_i)}_{P(e)}(\tau) - \frac{|P(e)|}{|Q(e)|}C^{(p_i)}_{Q(e)}(\tau)\right| = |P(e)| - \frac{|P(e)|}{|Q(e)|}|Q(e)| = 0$$

With these two cases, we can show that the error bound holds. □

7.3 Proof of Theorem 1

Proof. Based on Equation 11 and Equation 3, we have

$$\begin{aligned}
&|A_{P,Q}(\tau) - K_P(\tau)| \\
=\ & \frac{1}{n(n-1)}\Big|\sum_{\widehat{e}\in E}\sum_{e\in E}\sum_{p_i\in P(\widehat{e})}\Big(C^{(p_i)}_{P(e)}(\tau) - \frac{|P(e)|}{|Q(e)|}C^{(p_i)}_{Q(e)}(\tau)\Big)\Big| \\
\le\ & \frac{1}{n(n-1)}\sum_{\widehat{e}\in E}\sum_{e\in E}\sum_{p_i\in P(\widehat{e})}\left|C^{(p_i)}_{P(e)}(\tau) - \frac{|P(e)|}{|Q(e)|}C^{(p_i)}_{Q(e)}(\tau)\right|
\end{aligned}$$

With Lemma 3, we further have

$$\begin{aligned}
&|A_{P,Q}(\tau) - K_P(\tau)| \\
\le\ & \frac{1}{n(n-1)}\sum_{\widehat{e}\in E}\sum_{e\in E}\sum_{p_i\in P(\widehat{e})} 2\delta|P(e)| = \frac{2\delta n^2}{n(n-1)} = \frac{2\delta n}{n-1}
\end{aligned}$$

Hence, once we set $\delta = \frac{(n-1)\epsilon}{2n}$, we can conclude that $|A_{P,Q}(\tau) - K_P(\tau)| \le \epsilon$. □

7.4 Proof of Theorem 2

Proof. Based on Lemma 1, the size of the sampled dataset Q on each edge e, $|Q(e)|$, is:

$$|Q(e)| = O\Big(\frac{1}{\delta}\Big) = O\Big(\frac{2n}{(n-1)\epsilon}\Big) = O\Big(\frac{1}{\epsilon}\Big)$$

Therefore, the size of the sampled dataset Q, i.e., $\sum_{e\in E}|Q(e)|$, is

$$\sum_{e\in E}|Q(e)| = O\Big(\frac{|E|}{\epsilon}\Big)$$

□

7.5 Proof of Lemma 4

Proof. Note that we need to access every edge e and adopt SS for constructing $Q(e)$ from $P(e)$. Based on Lemma 1 (by setting $X = P(e)$ and $\delta = \frac{(n-1)\epsilon}{2n}$), the time complexity is:

$$O\Big(|E| + \sum_{e\in E}|P(e)|\log|P(e)| + \sum_{e\in E}|Q(e)|\Big) = O\Big(|E| + n\log n + \frac{|E|}{\epsilon}\Big)$$

□

7.6 Proof of Lemma 5

Proof. Based on Equation 14 and Equation 4, we have

$$\begin{aligned}
\mathbb{C}^{Q(e)}_{P(\widehat{e})}(\tau) &= \sum_{p_k\in Q(e)}\sum_{p_i\in P(\widehat{e})}\mathbb{I}(d_G(p_k,p_i)\le\tau) \\
&= \sum_{p_i\in P(\widehat{e})}\sum_{p_k\in Q(e)}\mathbb{I}(d_G(p_i,p_k)\le\tau) \\
&= \sum_{p_i\in P(\widehat{e})} C^{(p_i)}_{Q(e)}(\tau) = \mathbb{C}^{P(\widehat{e})}_{Q(e)}(\tau)
\end{aligned}$$

As a remark, the third equality and the fourth equality are based on Equation 4 and Equation 13, respectively. □

7.7 Proof of Lemma 6

Proof. Observe from Figure 9 that there are $|Q(e)|$ sorted lists and each list contains D end positions (thresholds) in level 0. In addition, each level needs to merge every two sorted lists together into one sorted list. Therefore, there are $O(\log|Q(e)|)$ levels in total. Since each level needs to access $D|Q(e)|$ end positions for merging those sorted lists, the time complexity is $O(D|Q(e)|\log|Q(e)|)$. □

7.8 Proof of Lemma 7

Proof. Recall from Figure 10 that the sweep line l_a (or l_b) only needs to scan those data points on $\widehat{e} = (a,b)$ (with $O(|P(\widehat{e})|)$ time) and the end positions (with $O(D|Q(e)|)$ time). In addition, this sweep line also needs to store $l_a(p_k,\tau_\rho)$ and $l_b(p_k,\tau_\rho)$ for each (p_k,τ_ρ)-pair (with $O(D|Q(e)|)$ time). With all pairs of $l_a(p_k,\tau_\rho)$ and $l_b(p_k,\tau_\rho)$, we can first compute $C^{(p_k)}_{P(\widehat{e})}(\tau_\rho)$ using Equation 15 and then evaluate $C^{Q(e)}_{P(\widehat{e})}(\tau)$ for all D thresholds using Equation 14 (with $O(D|Q(e)|)$ time). Hence, we conclude that the total time complexity is $O(D|Q(e)| + |P(\widehat{e})|)$. □

7.9 Proof of Lemma 8

Proof. Based on the discussion in the main content, the time complexity of computing $A_{P,Q}(\tau)$ with D thresholds is

$$O\Big(|E| + n\log n + \frac{|E|}{\epsilon} + |E|T_{\mathrm{SP}} + \sum_{e\in E}\sum_{\widehat{e}\in E}(D|Q(e)|\log|Q(e)| + |P(\widehat{e})|)\Big)$$

Since $|Q(e)| = O(\frac{1}{\epsilon})$ (see Theorem 2), we can further conclude that the above expression is:

$$O\Big(|E|T_{\mathrm{SP}} + \frac{D|E|^2}{\epsilon}\log\Big(\frac{1}{\epsilon}\Big) + n|E| + n\log n\Big)$$

□

7.10 Proof of Theorem 4

Proof. Since STARS needs to access all edges and data points in L datasets and call the shortest path algorithm, the space complexity is at least $O(|E|+Ln+S_{\mathrm{SP}})$. In addition, STARS needs to maintain the sampled dataset for each dataset. However, these sampled datasets must be smaller than the original ones (i.e., the total additional space complexity must be smaller than $O(Ln)$). Furthermore, STARS also maintains $D|Q(e)|$ end positions for an edge e (see Figure 8), which can be cleared for processing next edges. Therefore, the additional space is $O(D|Q(e)|) = O(\frac{D}{\epsilon})$ (based on Theorem 2). Hence, we can conclude that the space complexity is $O\big(|E| + Ln + \frac{D}{\epsilon} + S_{\mathrm{SP}}\big)$. □