

# PLAN: Fast and Approximate Gaussian Kernel Density Visualization in Road Networks

(Edison) Tsz Nam Chan<sup>1</sup>, Hongwei Ye<sup>1</sup>, Bojian Zhu<sup>2</sup>, Leong Hou U<sup>3</sup>,  
Dingming Wu<sup>1</sup>, Ruisheng Wang<sup>1</sup>, Joshua Zhexue Huang<sup>1</sup>

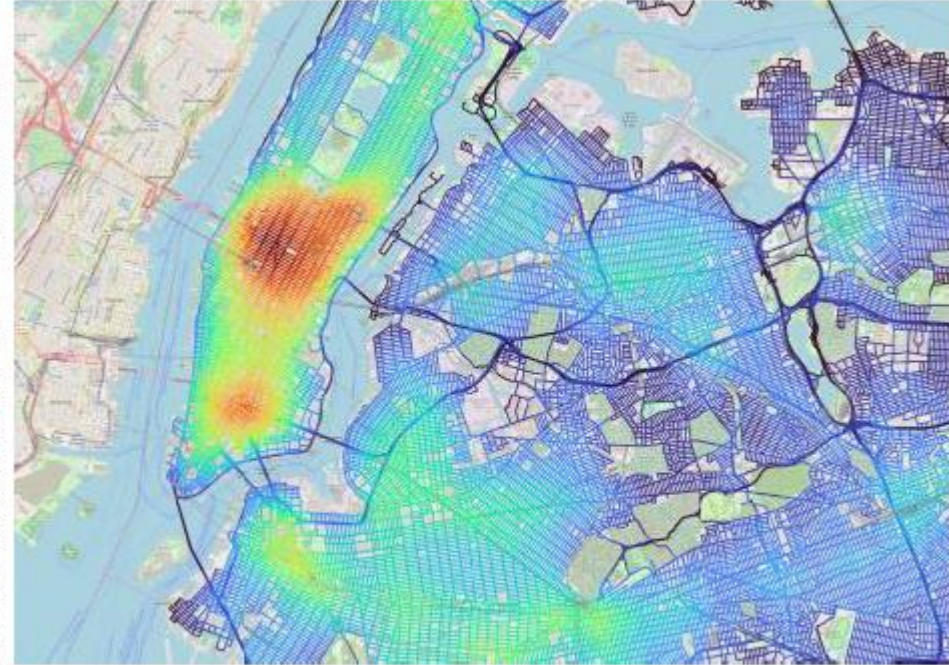
<sup>1</sup>Shenzhen University   <sup>2</sup>Hong Kong Baptist University   <sup>3</sup>University of Macau



# Why Network Kernel Density Visualization (NKDV) ?



(a) New York traffic accident dataset



(b) NKDV

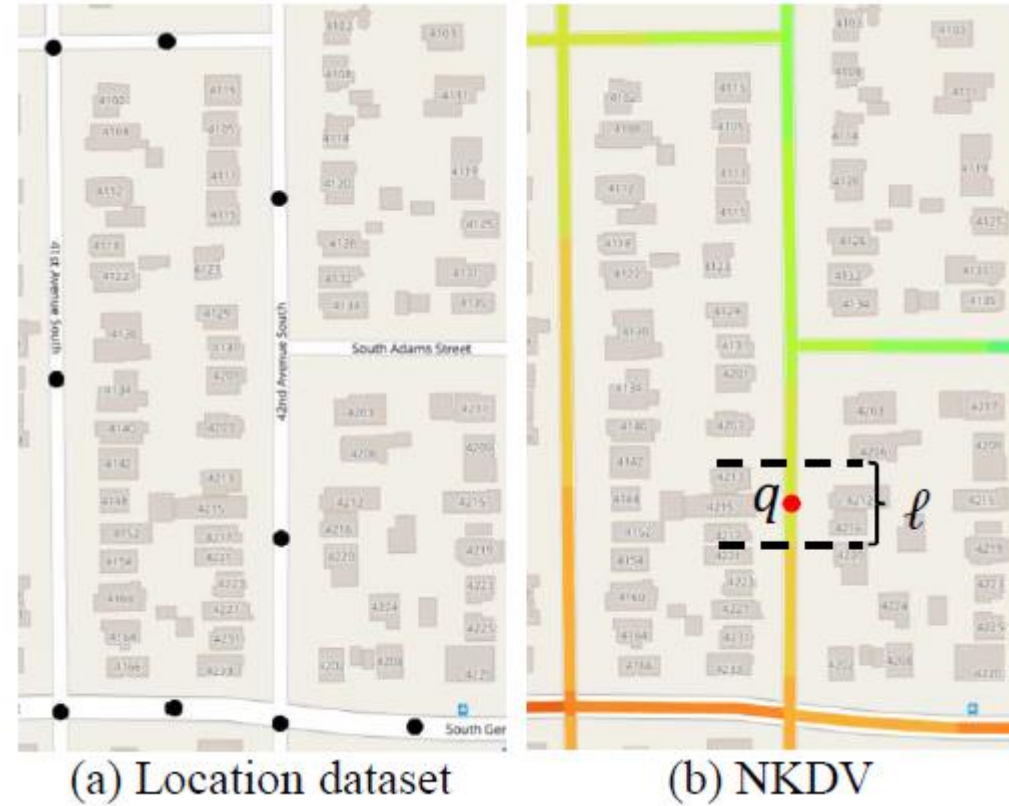
- Some representative applications:
  - Traffic/Traffic accident hotspot detection
  - Crime hotspot detection
  - Urban planning

# What is NKDV?

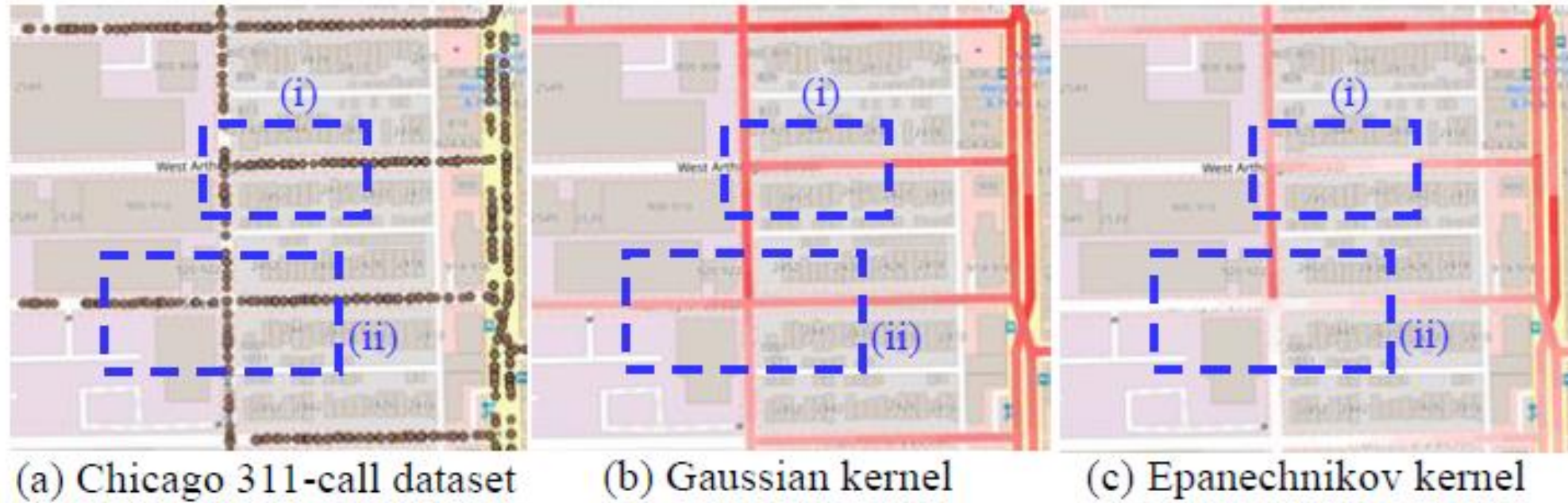
- Given a location dataset  $P = \{p_1, p_2, \dots, p_n\}$  and a road network  $G = (V, E)$ , we need to color each lixel  $q$  based on the network kernel density function  $\mathcal{F}_P(q)$ .

$$\mathcal{F}_P(q) = \frac{1}{|P|} \sum_{p \in P} K_G(q, p)$$

Kernel	Function
Epanechnikov	$\begin{cases} 1 - \frac{1}{b_G^2} d_G(q, p)^2 & \text{if } d_G(q, p) \leq b_G \\ 0 & \text{otherwise} \end{cases}$
Gaussian	$\exp\left(-\frac{1}{b_G^2} d_G(q, p)^2\right)$



# NKDV: Epanechnikov v.s. Gaussian



- Gaussian kernel can provide smoother visualization. 😊
- Gaussian kernel is the default one in many software packages. 😊

# Weakness of Gaussian-based NKDV

- Gaussian-based KDV is computationally expensive. ☹️
- Example (Chicago 311-call dataset):
  - Number of data points: 10.893 million
  - Number of lixels (with 10m): 0.67 million
  - Number of operations: > 7.298 trillion
- Not feasible to generate multiple NKDVs. ☹️

# Weakness of Gaussian-based NKDV

- Cannot directly extend existing fast algorithms for Gaussian-based NKDV. ☹️

$$\begin{cases} 1 - \frac{1}{b_G^2} d_G(q, p)^2 & \text{if } d_G(q, p) \leq b_G \\ 0 & \text{otherwise} \end{cases} \quad \text{v.s.} \quad \exp\left(-\frac{1}{b_G^2} d_G(q, p)^2\right)$$

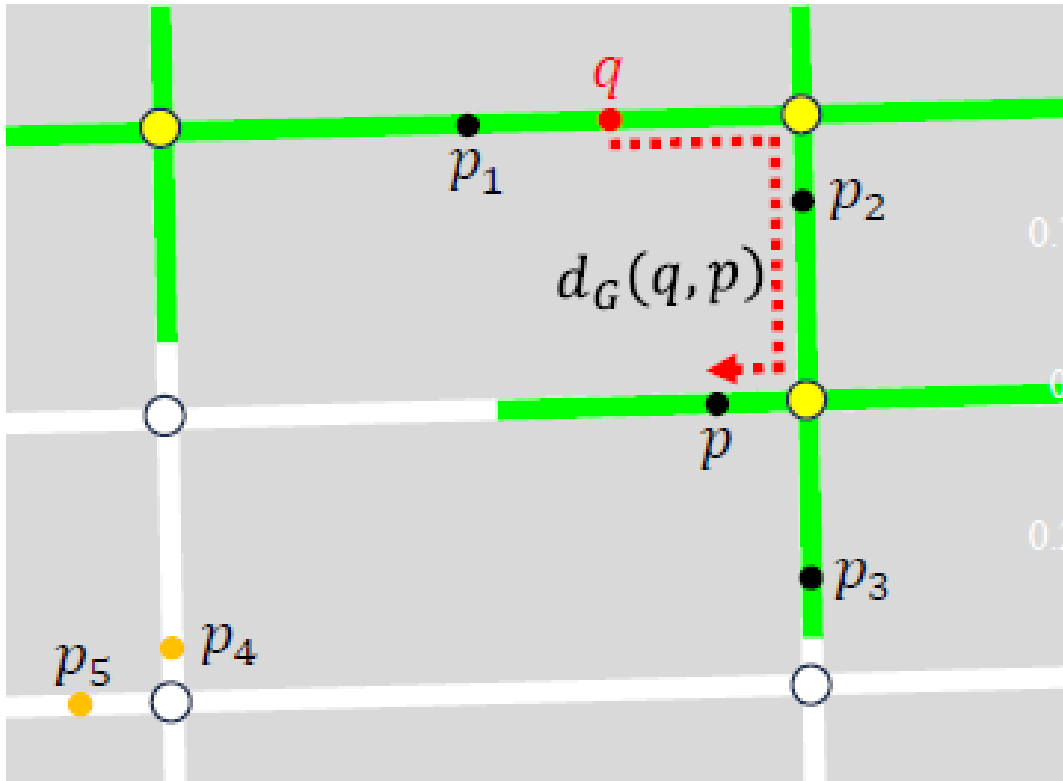
- Does not have short-tail property in Gaussian kernel.
- Cannot decompose the network kernel density function  $\mathcal{F}_P(q)$ .

# Our Contributions

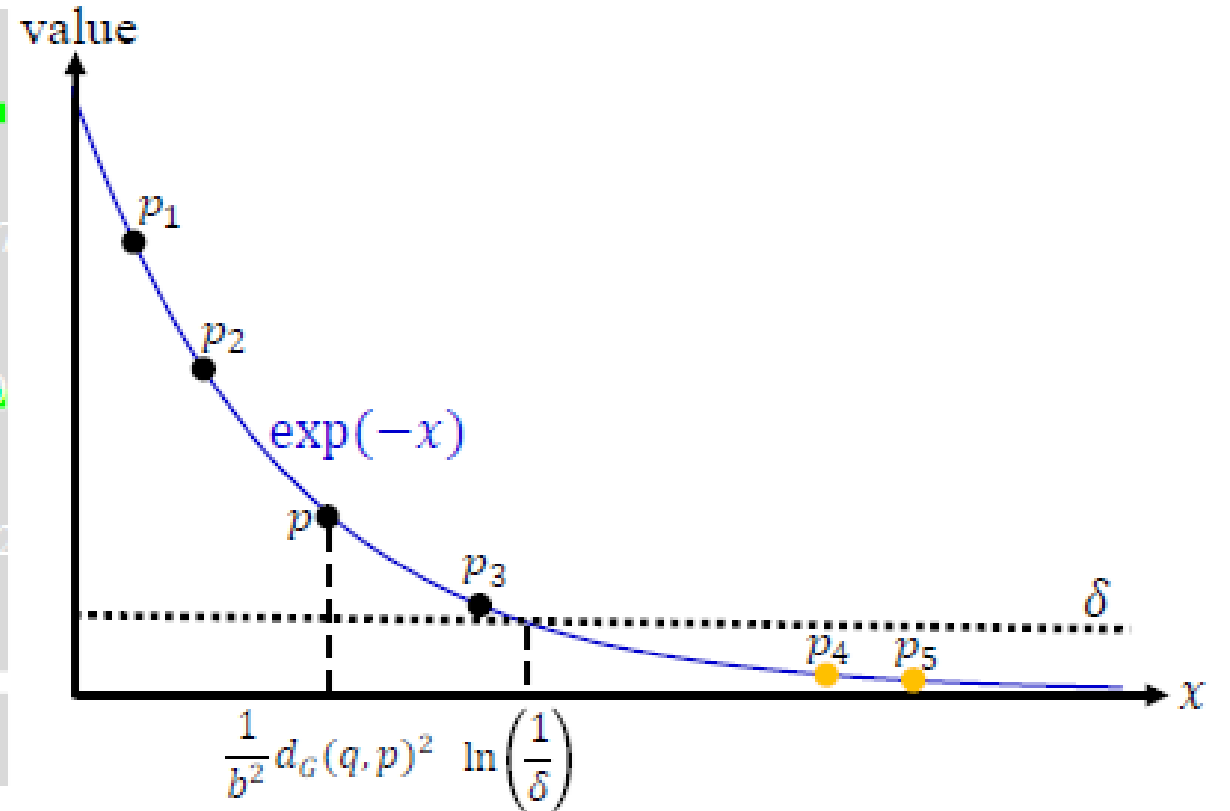
Method	Time complexity	Space complexity	Accuracy guarantee
SOTA	$O( E T_{\text{SP}} +  P L)$	$O( V  +  P  + L + S_{\text{SP}})$	Exact
PLAN	$O\left( E T_{\text{SP}} + LM E  \log\left(\frac{ P }{ E }\right)\right)$ (Theorem 1)	$O( V  +  P  + L + S_{\text{SP}} + M)$ (Theorem 3)	$\epsilon$ -absolute error (Lemma 1)
PLAN+	$O\left( E T_{\text{SP}} + LM E  \log\left(\frac{ P }{M E }\right)\right)$ (Theorem 2)		

- Reduce the time complexity for generating NKDV with (1) a non-trivial approximation guarantee and (2) a slight overhead in terms of space complexity. 😊
- Achieve 32.47x to 2936.88x speedups compared with SOTA.

# Core Idea 1: Enable the Short Tail Property



(a) Data points and a lixel  $q$  in a road network



(b) Exponential function

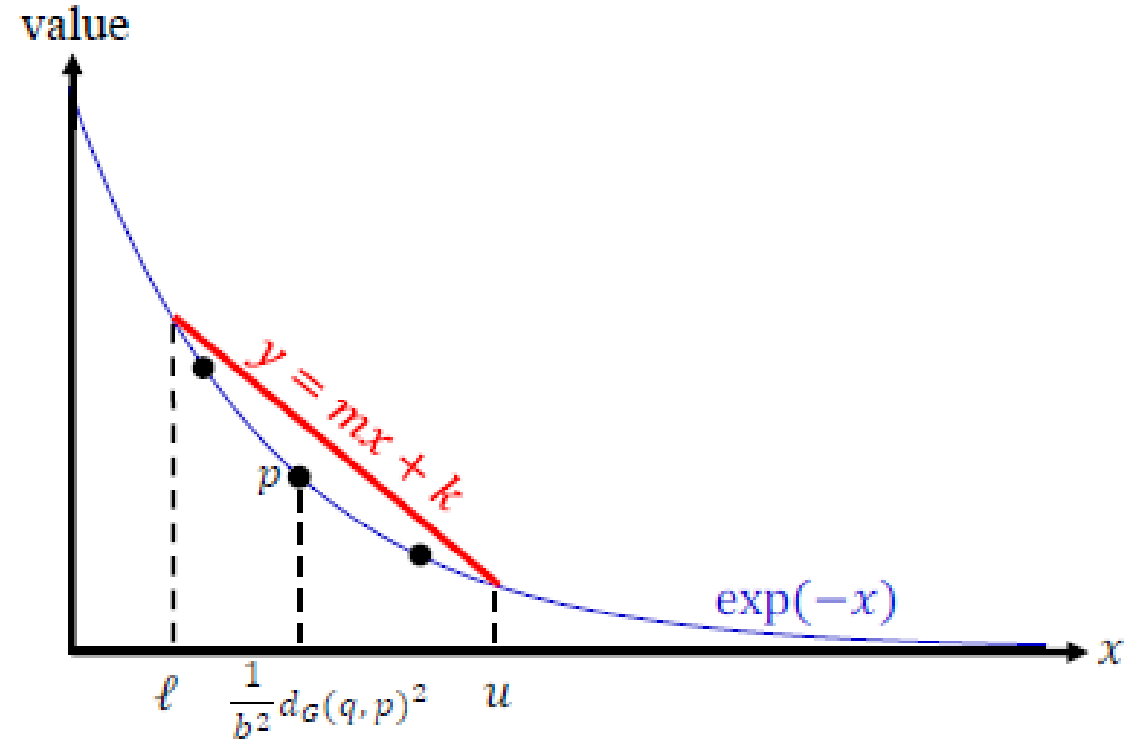
- Data points that are too far away from  $q$  can be discarded.

# Core Idea 2: Function Approximation

- This upper bound function can approximate  $\mathcal{F}_{\mathbb{P}}(q)$ .

$$\begin{aligned}\mathcal{F}_{\mathbb{P}}(q) &= \frac{1}{|\mathbb{P}|} \sum_{p \in \mathbb{P}} \exp\left(-\frac{1}{b^2} d_G(q, p)^2\right) \\ &\leq \frac{1}{|\mathbb{P}|} \sum_{p \in \mathbb{P}} \left(\frac{m}{b^2} d_G(q, p)^2 + k\right)\end{aligned}$$

$$\text{where } \mathbb{P} = \left\{p \in P : \ell \leq \frac{1}{b^2} d_G(q, p)^2 \leq u\right\}$$



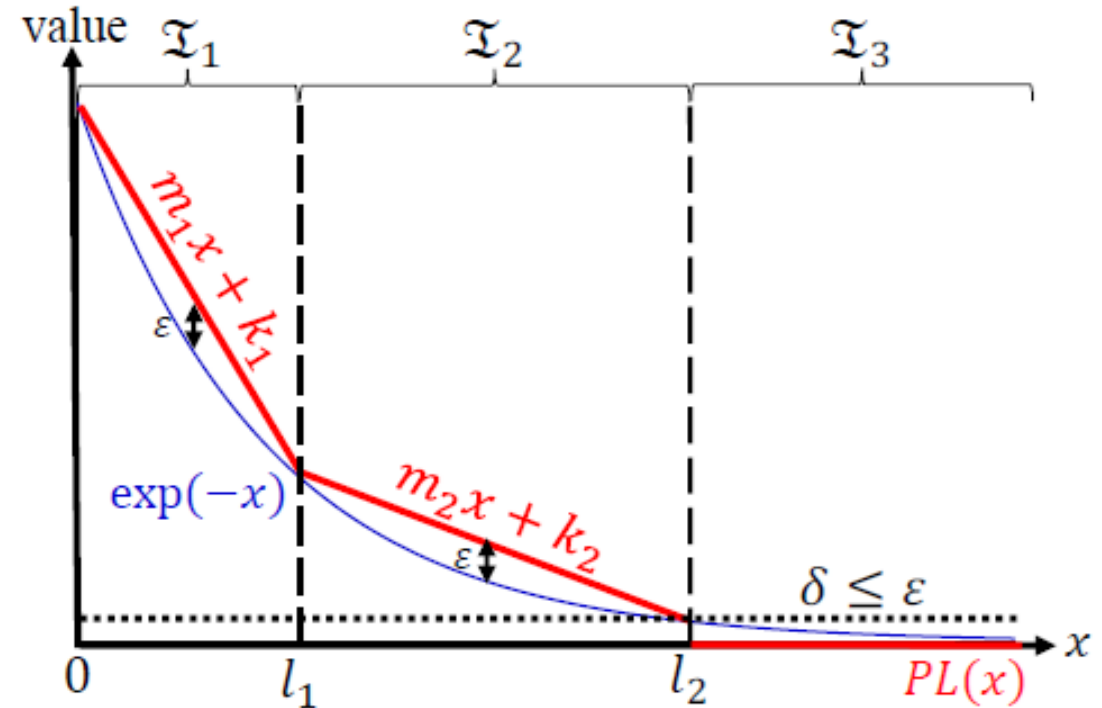
# PLAN: Approximate the Network Kernel Density Function

- Use  $A_P(q)$  to approximate  $\mathcal{F}_P(q)$ .

$$A_P(q) = \frac{1}{|P|} \sum_{p \in P} PL \left( \frac{1}{b^2} d_G(q, p)^2 \right)$$

where

$$PL(x) = \begin{cases} m_1x + k_1 & \text{if } x \in \mathcal{I}_1 \\ m_2x + k_2 & \text{if } x \in \mathcal{I}_2 \\ \vdots & \vdots \\ m_{M-1}x + k_{M-1} & \text{if } x \in \mathcal{I}_{M-1} \\ 0 & \text{if } x \in \mathcal{I}_M \end{cases}$$

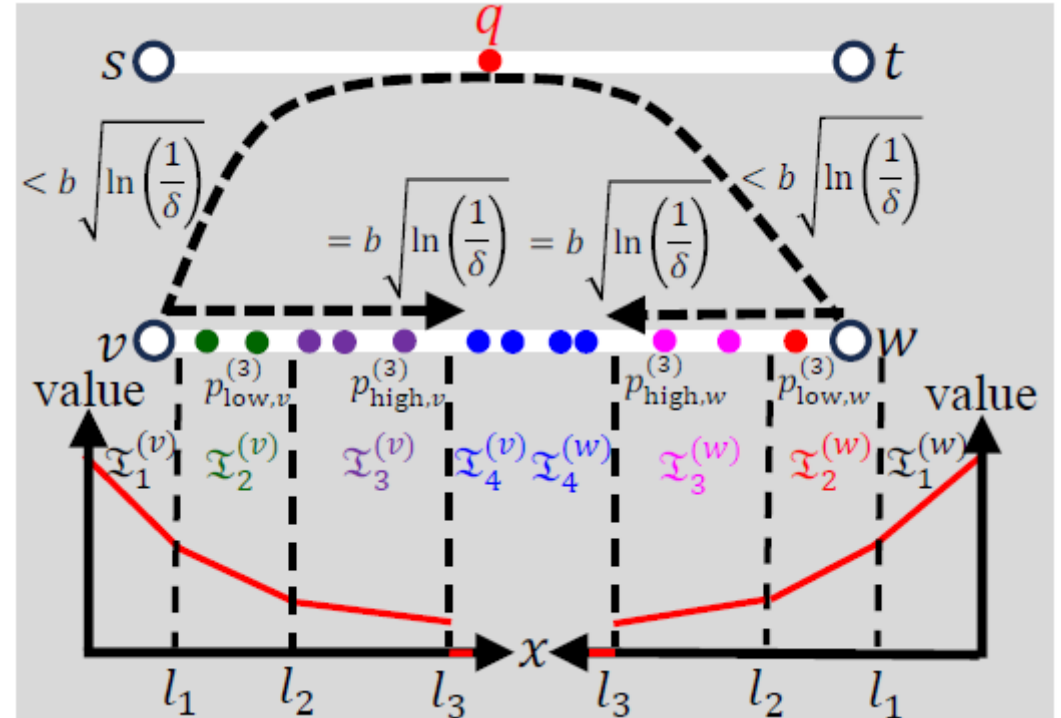


- Can achieve  $\varepsilon$ -absolute error guarantee. 😊

# PLAN: Efficient Computation

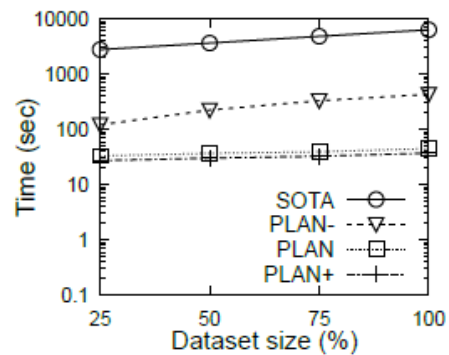
- Similar to ADA [a].
  - Perform augmentation for each data point.
  - Adopt the binary search method.
- Main difference
  - Adopt the binary search method  $M$  times.

Refer to our paper for details.

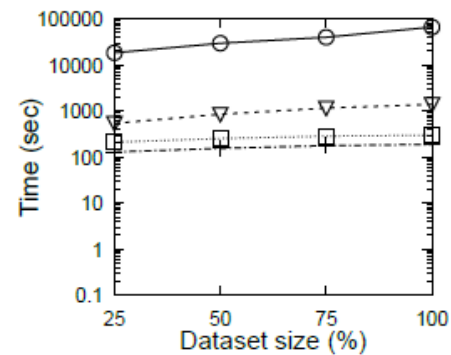


# Experimental Evaluation

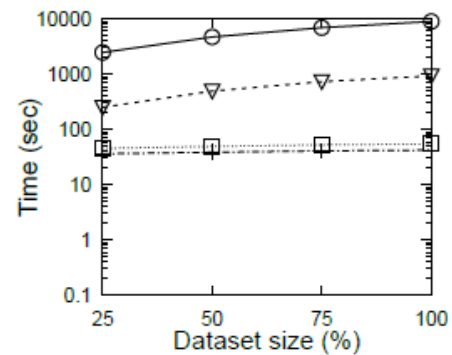
Dataset	$ V $	$ E $	$ P $	Category
Minneapolis [36]	7,388	12,913	1,598,877	911 calls
New York [37]	71,019	120,623	1,897,418	Traffic accidents
San Francisco [34]	6,627	11,593	2,985,502	Crime events
Chicago [25]	41,111	70,067	10,893,129	311 calls



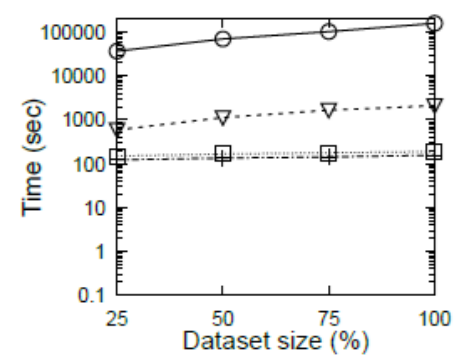
(a) Minneapolis



(b) New York



(c) San Francisco



(d) Chicago



(a) EXACT

(b)  $\epsilon = 0.01$

(c)  $\epsilon = 0.05$

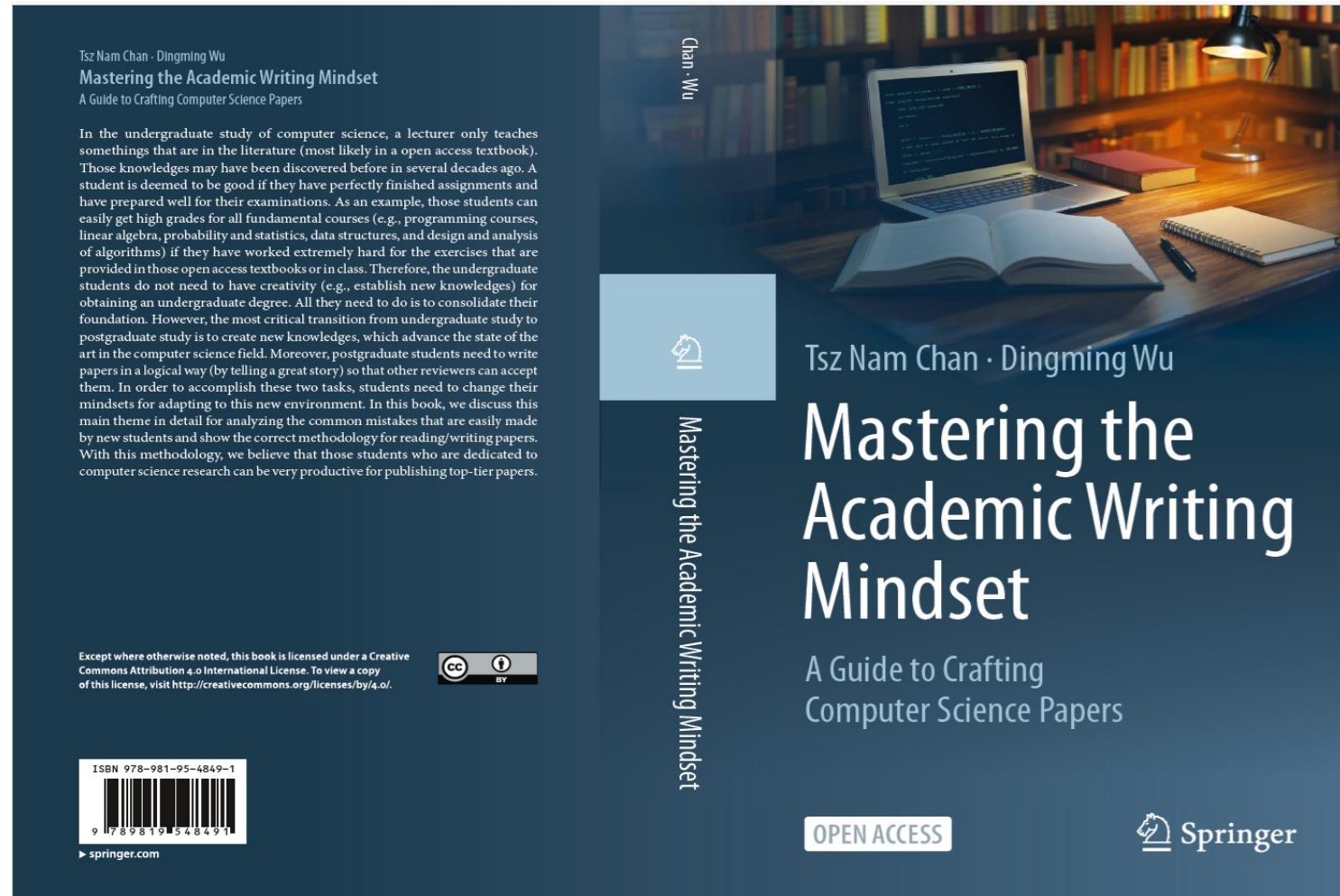
(d)  $\epsilon = 0.09$

(e) Epanechnikov kernel

# Future Opportunities

- Incorporate PLAN into the QGIS plugin, called Fast Density Analysis.
- Support NKDV with adaptive bandwidth.
- Adopt parallel/distributed/hardware-based approaches to improve the efficiency of NKDV.
- Support other inefficient road network analysis tasks, e.g., network Moran's I, network K-function, and network DBSCAN.

# Our New Book



Tsz Nam Chan · Dingming Wu  
**Mastering the Academic Writing Mindset**  
A Guide to Crafting Computer Science Papers

In the undergraduate study of computer science, a lecturer only teaches some things that are in the literature (most likely in an open access textbook). Those knowledges may have been discovered before in several decades ago. A student is deemed to be good if they have perfectly finished assignments and have prepared well for their examinations. As an example, those students can easily get high grades for all fundamental courses (e.g., programming courses, linear algebra, probability and statistics, data structures, and design and analysis of algorithms) if they have worked extremely hard for the exercises that are provided in those open access textbooks or in class. Therefore, the undergraduate students do not need to have creativity (e.g., establish new knowledges) for obtaining an undergraduate degree. All they need to do is to consolidate their foundation. However, the most critical transition from undergraduate study to postgraduate study is to create new knowledges, which advance the state of the art in the computer science field. Moreover, postgraduate students need to write papers in a logical way (by telling a great story) so that other reviewers can accept them. In order to accomplish these two tasks, students need to change their mindsets for adapting to this new environment. In this book, we discuss this main theme in detail for analyzing the common mistakes that are easily made by new students and show the correct methodology for reading/writing papers. With this methodology, we believe that those students who are dedicated to computer science research can be very productive for publishing top-tier papers.

Except where otherwise noted, this book is licensed under a Creative Commons Attribution 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by/4.0/>.

ISBN 978-981-95-4849-1

springer.com

Chan · Wu

Mastering the Academic Writing Mindset

Tsz Nam Chan · Dingming Wu

**Mastering the Academic Writing Mindset**

A Guide to Crafting Computer Science Papers

OPEN ACCESS

Springer

<https://link.springer.com/book/10.1007/978-981-95-4850-7>