Hindawi
Discrete Dynamics in Nature and Society
Volume 2021, Article ID 5550315, 19 pages
https://doi.org/10.1155/2021/5550315

Research Article

Detection and Optimization of Traffic Networks Based on Voronoi Diagram

Rui Tao,[1] Jian Liu,[2] Yuqing Song,[3] Rui Peng,[1] Dali Zhang,[4] and Jiangang Qiao [1]

[1]*School of Civil and Transportation, Hebei University of Technology, 5340 Xiping Road, Beichen District, Tianjin 300401, China*
[2]*Zhong Dian Jian Ji Jiao Expressway Investment Development Co., LTD., 672 Chengjiao Street, Qiaoxi District, Shijiazhuang 050090, Hebei, China*
[3]*Tianjin University of Technology and Education, 1310 Dagu South Road, Tianjin 300222, China*
[4]*Yanchong Management Center of Hebei Expressway Group Co., LTD., Jianshe Road, Zhangjiakou 075400, Hebei, China*

Correspondence should be addressed to Jiangang Qiao; qiaojg369@126.com

Received 1 February 2021; Revised 29 July 2021; Accepted 6 November 2021; Published 13 December 2021

Academic Editor: Rui Wang

Traffic peak is an important parameter of modern transport systems. It can be used to calculate the indices of road congestion, which has become a common problem worldwide. With accurate information about traffic peaks, transportation administrators can make better decisions to optimize the traffic networks and therefore enhance the performance of transportation systems. We present a traffic peak detection method, which constructs the Voronoi diagram of the input traffic flow data and computes the prominence of candidate peak points using the diagram. Salient peaks are selected based on the prominence. The algorithm takes $O(n \log n)$ time and linear space, where n is the size of the input time series. As compared with the existing algorithms, our approach works directly on noisy data and detects salient peaks without a smoothing prestep and thus avoids the dilemma in choosing an appropriate smoothing scale and prevents the occurrence of removing/degrading real peaks during smoothing step. The prominence of candidate peaks offers the subsequent analysis the flexibility to choose peaks at any scale. Experiments illustrated that the proposed method outperforms the existing smoothing-based methods in sensitivity, positive predictivity, and accuracy.

1. Introduction

Accurate information about traffic peaks is important in planning, maintenance, and control of modern transport systems. Traffic congestion has become widespread in large metropolitan areas. Peak-hour congestion is also a problem in many suburban areas. Traffic peaks can be used to compute congestion indices, helping individuals and traffic management units to avoid traffic jam and minimize travel time [1–4]. By monitoring traffic peaks, transportation administrators can make better decisions to optimize the traffic networks and therefore enhance the performance of transportation systems [5–7]. Automatic traffic event detection is important in intelligent transportation systems (ITSs). The detection can be performed with streaming data clustering [8], marking points clustering [9], and deep learning methods [10].

Effective peak detection remains a challenging problem in many fields, including asteroseismology [11, 12], analytical chemistry [13, 14], disease diagnosis [15, 16], and traffic analysis [7, 17]. Mathematically, a peak of a function is a local maximum point where the height of the function is the greatest in an interval around the point. Billauer developed an algorithm that detects peaks using the local maxima (peak) and minima (valley) values [18, 19]. In [20], the first-order derivative of chromatographic signal is computed to automatically extract peaks in the data. In [21], overlapped peaks are detected by examination of the second derivative of the raw mass spectrum. In existing algorithms, smoothing techniques are needed to eliminate or reduce

noise [7, 13–15]. The major problems with the existing peak detection methods are related to the smoothing step:

(i) The smoothing often removes the fine details and makes the peaks vanish or degrade. The effect is unrecoverable; that is to say, if a real peak is removed/degraded during the smoothing step, it is not recovered in the subsequent analysis.

(ii) The goal of smoothing is to eliminate or reduce the fluctuation and noise to avoid weak and false peaks, and it is crucial to select an appropriate smoothing scale. Choosing such a scale for a given traffic data requires prior knowledge about the data, which is often not available before the data analysis is finished.

For traffic flow data, most fluctuation and noise have smaller amplitude than the peaks we are interested in. Based on this property, we introduce a Voronoi diagram-based traffic peak detection method. The new method has the following advantages. Firstly, it works on noisy data directly and avoids the difficulty in choosing an appropriate smoothing scale. Secondly, it computes the prominence of candidate peak points and offers the subsequent analysis step the flexibility to choose peaks at any scale. The algorithm takes $O(n \log n)$ time and linear space, where n is the size of the input time series.

The rest of this paper is organized as follows. We introduce the related work in Section 2 and describe in detail the proposed method in Section 3. Following that in Section 4, we give the experiment results. We conclude the paper with a brief discussion in Section 5.

2. Related Work

Traffic network optimization improves a traffic network, maximizing its performance. The problem is formulated as an NP-hard integer program, and there exists no exact polynomial-time algorithm; heuristic methods such as genetic algorithms (GAs) are common ways to solve the problem [22]. An enhanced GA was introduced in [22], which performs the mutation in a goal-oriented manner by local optimization: changing an individual into the best one in its neighborhood. The enhanced GA performs better than a standard GA with respect to the trip duration.

Mao et al. [23] proposed a traffic signal control optimization method using GAs and machine learning models, minimizing the total travel time in urban networks. They assume that the O-D demands are predefined and use the traffic assignment model to get the deterministic link traffic flows. In case study, they applied a static traffic assignment modelling scenario to obtain the initial link flows assigned to each road section during morning peak. Comparison shows that the proposed method is faster than the original GAs and can be successfully applied under nonrecurrent incident conditions.

Paricio et al. proposed evolutionary algorithms for the traffic weighted multimap (TWM) technique to solve the traffic assignment problem, minimizing total travel time and providing the best-cost routes to vehicles [5]. The complexity of optimal TWM distributions depends on several factors, like the O/D directionality and traffic flow volumes. Obtained results show that TWM performs good in optimal routing at the system level, avoiding the need for continuous route calculus based on traffic status data streaming.

Bianchin and Pasqualetti [6] studied the problem of controlling intersections with the goal of optimizing network-wide congestion. They cast an optimization problem to describe the goal of optimally controlling automated intersections and relate congestion objectives with the problem of optimizing a metric of controllability of the associated dynamical system. The system performance is characterized in relation to optimal configurations, and conditions are identified to guarantee network stability. With respect to established models, the framework relies on a continuous time and discrete space characterization of the traffic flows and allows for a more tractable network analysis.

When assessing and optimizing traffic networks, traffic flow peaks are an important parameter. However, peak detection is still a challenge since peaks in real applications are sensitive to local variation and noise. Simple maxima or derivative methods yield poor results when the signal contains noise. Smoothing techniques are applied to eliminate or reduce noise. Common smoothing methods used in peak detection include average [7, 24], Gaussian [13, 25], Savitzky–Golay [15, 16], and wavelet smoothing [11, 14].

A rise/fall analysis was used to detect air traffic peaks [7]. A peak exists where there is a change from rising to falling. To smooth out the noises, a 10th order rolling average is applied before the peaks are identified. Zhao et al. [24] presented a stripe peak detection method for structured light-based 3D reconstruction. In this approach, a weighted average method is applied to determine accurate peak position in stripe areas after they are segmented.

Lu et al. [25] introduced an adaptive ion-connecting network-based strategy for ion chromatogram (IC) extraction based on which a multiscale Gaussian smoothing-based peak detection method is applied after bad ICs are removed. Fu et al. [13] proposed a multiscale Gaussian smoothing-based approach for chromatographic peak extraction, where peaks appear as local maximum values under various smoothing window widths and can be detected through the ridge lines of maximum values under these window scales and of signal-to-noise ratios greater than a threshold.

The work in [15] proposed a technique for T-wave peak detection using simple root mean square-based decision rule. A Savitzky–Golay smoothing filter is applied for preprocessing. The filter essentially performs a local polynomial regression on the signal to determine the smoothed value for each point. The approach in [16] detects R peaks in ECG signals, where the Shannon energy envelope with the Savitzky–Golay filter was designed to suppress noise and enhance the QRS-complex.

The algorithm in [11] uses a continuous wavelet transform (CWT)-based pattern-matching approach. The CWT serves as a pattern-matching function where the signal is compared with a wavelet function specifically chosen to have similar features as the most common peaks that contain signal. Zheng et al. [14] proposed a peak detection algorithm

based on continuous wavelet transform, which identifies peaks by drawing ridges based on the movement of particles in the CWT coefficient matrix.

3. Proposed Method

Daily traffic volume data are a time series or mathematically a discrete function, consisting of a list of 2D points, and each point is a time-volume pair. With the point list as input, our method takes 4 main steps, as shown in Figure 1.

Given a collection of geometric primitives, its Voronoi diagram is a subdivision of space into cells such that all points in a cell are closer to one primitive than to any other [26, 27]. For a generalized Voronoi diagram, the geometric primitives can be points, line segments, or circular arcs. In our algorithm, we compute the Voronoi diagram of a polyline, where the geometric primitives are either line segments or points (endpoints of line segments). Based on the Voronoi diagram, we give our definitions (Section 3.1) and report the algorithm (Section 3.2).

3.1. Definitions. We first define function curve.

Definition 1. Given a discrete function, we connect the adjacent points by line segments and get a curve, called the function curve.

A function curve is a polyline. From the example in Figure 2, we can see that the Voronoi diagram divides the plane into cells, each of which is associated with a line segment or an endpoint of a line segment.

We define Voronoi tree.

Definition 2. The Voronoi diagram of a function curve is partitioned by the function curve into two parts, upper and lower parts; for each Voronoi edge e,

(i) If e crosses the function curve, then e is partitioned into two parts, e_u and e_l, which are, respectively, above and below the function curve. We put e_u and e_l, respectively, to the upper and lower parts of the Voronoi diagram.

(ii) Otherwise, e is put to the upper/lower part of the Voronoi diagram if it is above and below the function curve.

The edges in the lower part of the Voronoi diagram make a forest; we add a virtual root node to make it rooted. The virtual root represents an infinite point. The rooted tree is called the Voronoi tree of the function (see Figure 3).

To define the prominence of a Voronoi edge and of a function point, we use as an example a time series (Figure 4) of traffic flow data in vehicles per 15 minutes over 24 hours.

Definition 3. In a Voronoi tree, the subtree led by a Voronoi edge represents a hill in the function curve (Figure 5(b)). We find a top point in the hill and record it as the peak site of the Voronoi edge; we compute the height difference between the peak site and the higher of the left and right bottom points of the hill and record it as the prominence of the Voronoi edge (Figure 5(b)).

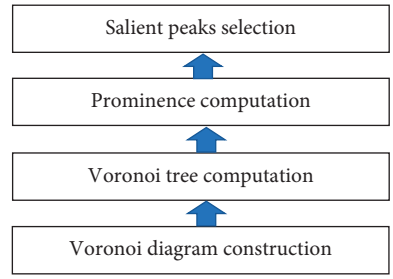


Figure 1: The main steps of the proposed method.

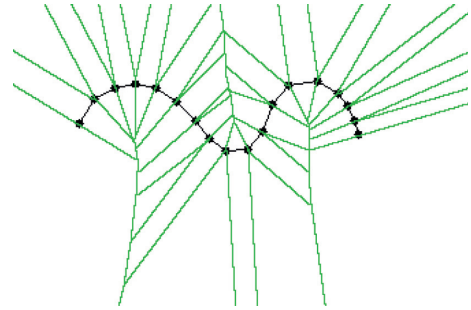


Figure 2: Voronoi diagram of a polyline of 18 points in the plane.

Definition 4. For a function f, we define the prominence of a function point p as the greatest prominence of a Voronoi edge with p as its peak site. The function points of positive prominence are the candidate peak points.

In computing the Voronoi diagram, since the x and y values of time series data are of different features (time and traffic volume, respectively), we need to unify them by a feature normalization process. In our algorithm, we adopt a commonly used normalization approach: scaling the time and traffic volume values of a time series to the range [0, 1].

3.2. Algorithm. Our algorithm consists of 4 main steps (Figure 1), and among them, the first 2 steps are straightforward by definition. We address the 3rd and 4th steps.

We first compute the prominences of the Voronoi edges. For any Voronoi edge, if its left/right Voronoi cell is associated with an endpoint of a line segment, we let its left/right input site be the endpoint; otherwise (it is associated with a line segment), we let its left/right input site be the right/left endpoint of the line segment.

For each Voronoi edge e in a Voronoi tree, we denote its input sites as p_i and p_j with $i \leq j$. We compute the following properties of e. The algorithm to compute these properties is given in Algorithm 1.

The prominences of the points are obtained straightforwardly by definition. We collect the points of positive prominences as the candidate peak points (Figure 6(a)) and sort their prominences (Figure 6(b)), making the sorted prominence curve. We then connect the first and last prominence dots by a line (the green dash line) and find in

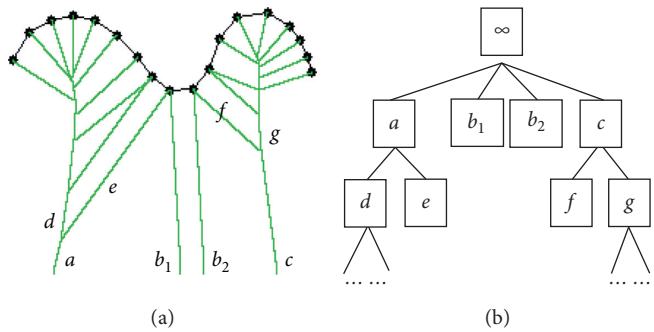


Figure 3: (a) The edges in the lower part of the Voronoi diagram in Figure 2. (b) The 3 top levels of the Voronoi tree.

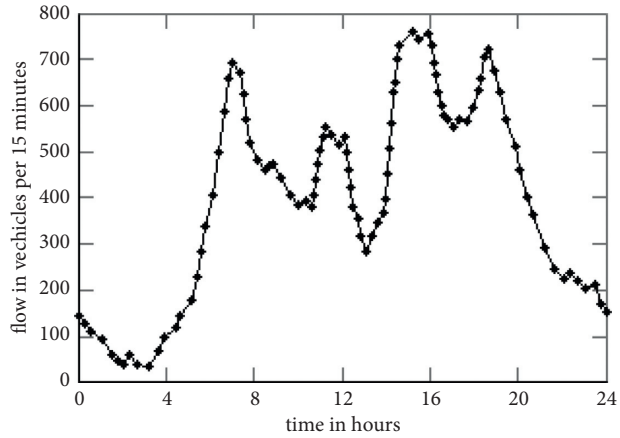


Figure 4: Traffic flow in vehicles per 15 minutes over 24 hours.

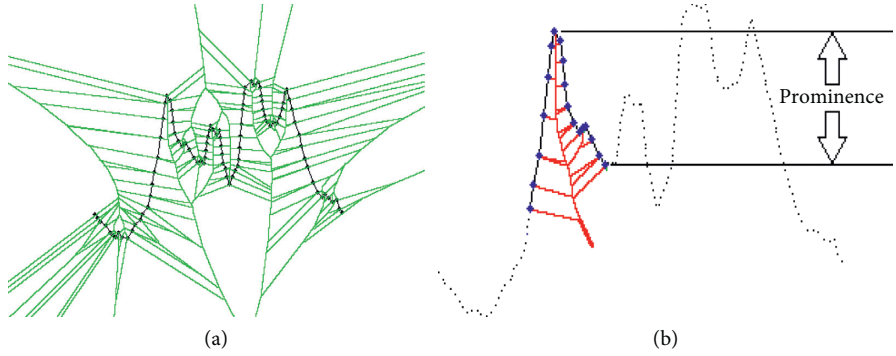


Figure 5: (a) The Voronoi diagram of the data in Figure 4. (b) A Voronoi edge (thick red line) and its descendent edges (thin red lines), representing a hill in the function curve. The height difference between the two black horizontal lines is the prominence of the Voronoi edge.

the prominence curve the dot (the yellow dot) farthest from the line. This dot is called the prominence bending dot. The prominence dots before the bending dot represent the peak points detected (Figure 7(a)).

For a peak point, the hill represented by the Voronoi edge of the greatest prominence is called the supporting hill. The supporting hills of the peak points are not exclusive from each other; they are nested instead (Figure 7(b)).

3.3. Computation Time and Space. In the following discussion, we assume that the input time series has n points. The total computation time is $O(n \log n)$, where

(i) We first construct the Voronoi diagram, taking $O(n \log n)$ time [26]. The diagram has $O(n)$ edges.

(ii) We then examine each Voronoi edge and split the diagram into 2 parts (partitioned by the function curve) and make the lower part the Voronoi tree, taking $O(n)$ time.

(iii) We visit the Voronoi tree bottom-up and compute the prominences of the nodes (Voronoi edges) in the tree, taking $O(n)$ time. The prominence of a function point p is the greatest prominence of a Voronoi edge with p as its peak site. The prominences of the function points are computed in $O(n)$ time.

Input: a function f and its Voronoi tree T
Output: the left valley site of each Voronoi edge, represented by Valley^{L} (•), the right valley site of each Voronoi edge, represented by Valley^{R} (•), the peak site of each Voronoi edge, represented by Peak (•), and the prominence of each Voronoi edge, represented by Prominence (•)
Begin
(1)**for** each Voronoi edge e in T **do** //T is visited in a bottom-up order.
(2) Let the two input sites be p_i and p_j with $i \le j$.
(3) **if** $j = i$ **then** //in this case e is a leaf node.
(4) Peak(e) = Valley^{L}(e) = Valley^{R}(e) = p_i
(5) else
(6) let the first and second child nodes of e be, respectively, e_1 and e_2
(7) **if** y(Peak (e_1)) < y(Peak (e_2)) **then**
(8) Peak (e) = Peak (e_2)
(9) Valley^{L} (e) = the lowest of Valley^{L} (e_1), Valley^{R} (e_1), and Valley^{L} (e_2)
(10) Valley^{R} (e) = Valley^{R} (e_2)
(11) else
(12) Peak (e) = Peak (e_1)
(13) Valley^{L} (e) = Valley^{L} (e_1)
(14) Valley^{R} (e) = the lowest of Valley^{R} (e_1), Valley^{L} (e_2), and Valley^{R} (e_2)
(15) Prominence(e) = y(Peak (e))−max(y(Valley^{L} (e)), y(Valley^{R} (e)))
(16)**return** Valley^{L} (•), Valley^{R} (•), Peak (•), and Prominence (•).
End

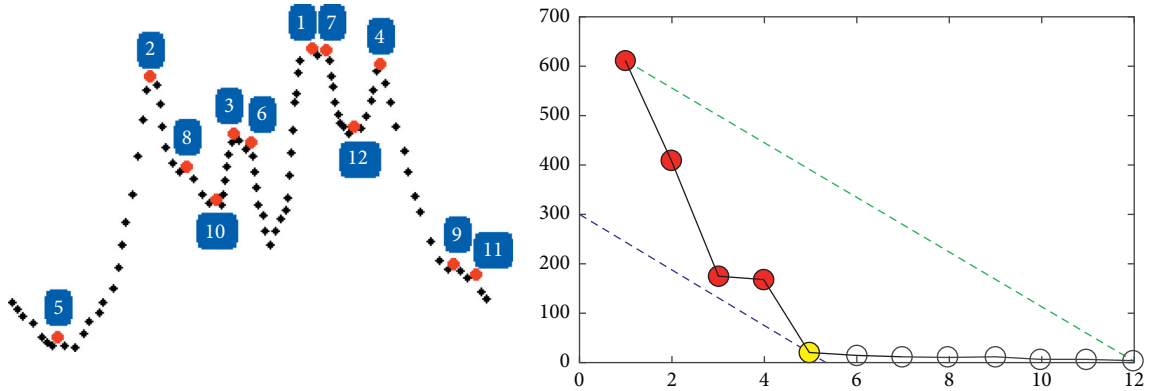
Algorithm 1: Compute prominence of VoronoiEdges (f, T).

(i) The peak site, p_k, is a highest point among p_i, p_{i+1}, ..., and p_j
(ii) The left valley site, p_u, is a lowest point among p_i, p_{i+1}, ..., and p_k

(iii) The right valley site, p_v, is a lowest point among p_k,p_{k+1}, ..., and p_j
(iv) The prominence, equals to $y(p_k) - \max(y(p_u), y(p_v))$, where y(•) is the y coordinate of a point

Table 1: The results of the average filter of different sizes.

Filter size	The roads where the average filter of the size detects the peaks successfully
9	Road1, Road2, Road3, Road4
19	Road3, Road4
29	Road3
39	
49	



(a) (b)

Figure 6: (a) The candidate peak points of the data in Figure 4, each labeled by its number in the sorted prominences. (b) The sorted prominences in dots, where the x-axis represents the ordinal number, the y-axis represents the prominence, the green dash line connects the first and last dots, the blue dash line is a tangent line parallel to the green line, the yellow dot is the bending dot, and the 4 red dots correspond to the 4 peak points detected.

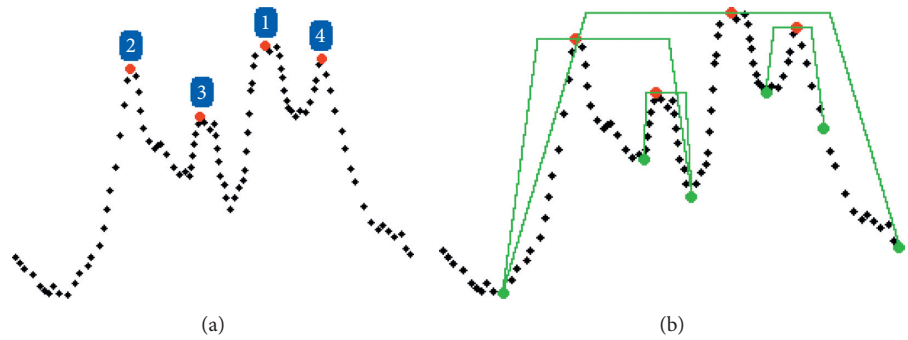


Figure 7: (a) The 4 peak points detected. (b) The supporting hills of the peak points, where each supporting hill is illustrated by a bottom-open trapezoid with the top line passing the peak point and the 2 base points specifying the range of the hill.

(iv) We sort the positive prominences of the function points, taking $O(n \log n)$ time; connect the first and last prominence dots by a line, taking $O(1)$ time; find the prominence bending dot, taking $O(n)$ time; output the peak points detected (the prominence dots before the bending dot in the sorted prominence curve), taking $O(n)$ time.

The computation takes total $O(n)$ space:

(i) The Voronoi diagram takes $O(n)$ space [26]

(ii) The Voronoi tree takes $O(n)$ space

(iii) The prominences of the Voronoi edges and the function points take $O(n)$ space

(iv) The peak points detected take $O(n)$ space

4. Experiments

We implemented our algorithm in C++ and ran experiments on a ThinkPad X270 laptop with 2.80 GHz i7-7500U CPU and 8 GB RAM. We performed experiments on the data provided by the Traffic Management Bureau of Hebei Provincial Public Security Department in China. The bureau maintains a road traffic congestion index system by integrating the real-time traffic data from taxi GPS, traffic patrols, and road detectors. The system dynamically monitors traffic volume peaks, which are fused with occupancy and speed data to compute the traffic congestion indices.

We report experiments on 6 selected series in Section 4.1 and on a set of series with quantitative performance in Section 4.2. The experiments in Section 4.1 show that noise in the traffic data has different scales, and the existing smoothing-based methods fail on some of the series since there does not exist a uniform smoothing scale for all of the series. On the other hand, the proposed method adapts itself to different noise scales by finding the bending dot in the sorted prominence curve. The statistical test in Section 4.2 uses performance indices of sensitivity, positive predictivity, and accuracy. For each existing smoothing-based method, we choose a best parameter such that the overall accuracy is maximized. In the comparison, our method outperformed the smoothing-based methods on all the performance indices. The experiments verify the advantages of the proposed algorithm, which avoids the difficulty in choosing an appropriate smoothing scale and offers the subsequent analysis step the flexibility to choose peaks at any scale.

4.1. Experiments on Examples. We choose 6 time series of traffic volume data, namely, Road1 to Road6 data (Figure 8), to demonstrate the effectiveness of our method. The traffic volume was measured in vehicles per 5 minutes.

For each time series, we computed the prominences of the data points, got the candidate peak points (points with positive prominences), sorted their prominences, and selected the peak points by finding the prominence bending dot. The supporting hills of the peak points are nested, and their prominence dots form a steep section in the prominence curve. Variation and noise in the time series often have repeated patterns, and their prominence dots form a slowly declining section. The two sections are separated at the bending dot. We report the sorted prominences, the peak points detected, and their supporting hills of the Road1 to Road6 data, respectively, in Figures 9–14. In all tested time series data, our method successfully detected the salient peaks, even in Road5 and Road6 data, where the prominence of the last detected peak (the third one in both data) is not much greater than the greatest prominence of the noise. The results demonstrate the effectiveness of our method in traffic data with noise.

For comparison, the average, Gaussian, Savitzky–Golay, and Wavelet smoothing methods were applied to the data, followed by Billauer's peak picking method. Billauer's method has the ability to skip peaks of amplitude less than a threshold. We set the amplitude threshold as 2, which is obtained through trial and error.

The average, Gaussian, and Savitzky–Golay methods use a smoothing filter of a given size. Savitzky–Golay filter requires the filter size to be odd. We chose 5 filter sizes, 9, 19, . . ., 49. The results of the average, Gaussian, and Savitzky–Golay smoothing are shown in Figures 15–17, respectively. In the figures, the x-coordinates represent the time in unit of 5 minutes. We list the detected results of the 3 filters of the 5 sizes in Tables 1–3, respectively, and see that there does not exist a single size with which the 3 filters work successfully on all data. The best results are achieved by Savitzky–Golay filter of sizes 19 and 29, each of which only fails on only one series.

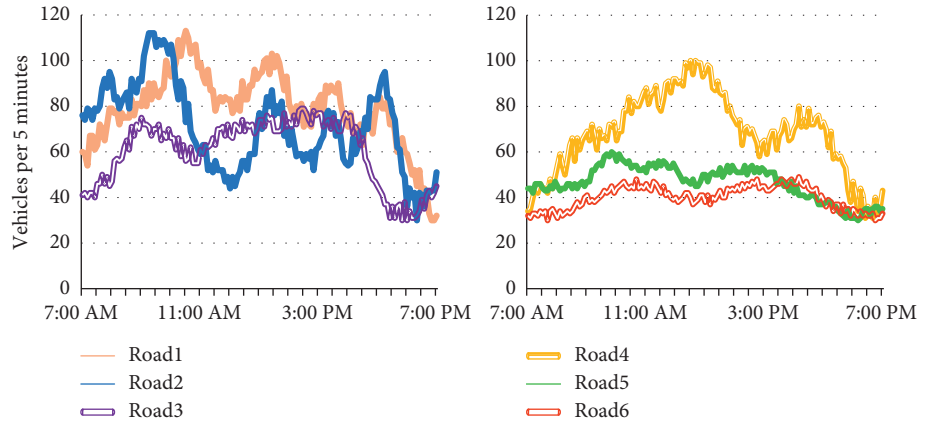


Figure 8: Six time series of traffic volume data.

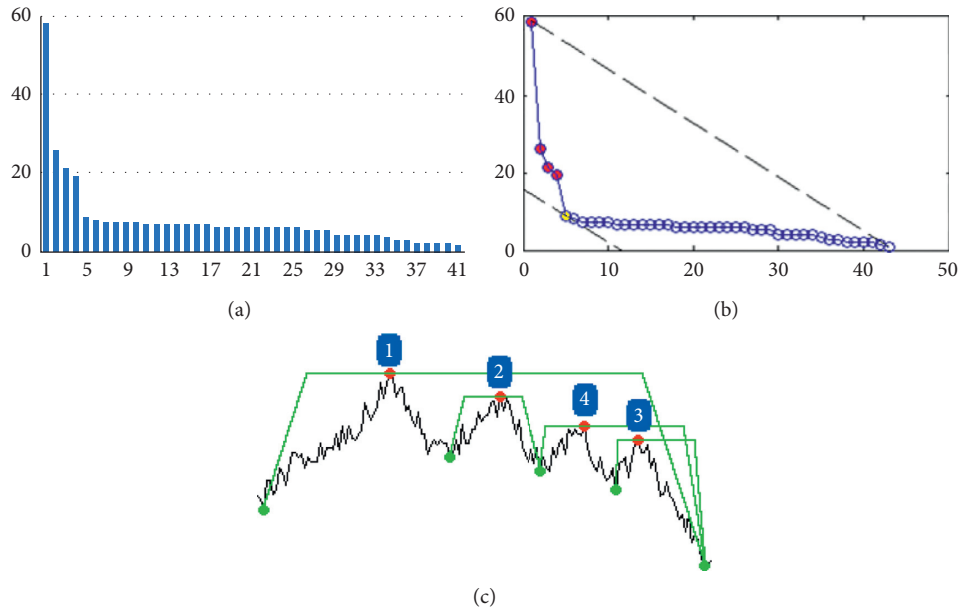
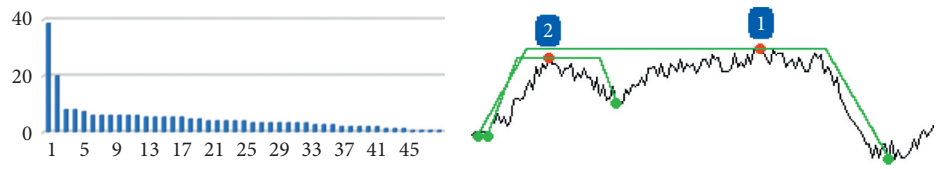


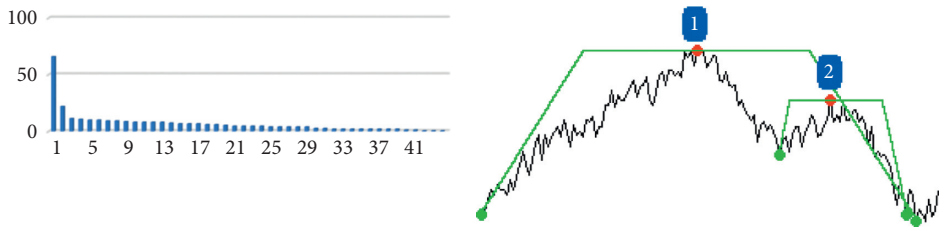
Figure 9: (a) Sorted prominences of the candidate peak points of Road1 data in bars, where the ratio of the 4th (the prominence of the last detected peak) to the 5th (the greatest prominence of the noise) is 223%. (b) Sorted prominences in dots, where the long dash line connects the first and last dots, the short dash line is a tangent line parallel to the long dash line, the yellow dot is the bending dot, and the 4 red dots correspond to the 4 peak points detected. (c) The detected peak points (each marked by its prominence ordinal number) and the supporting hills.

Figure 10: (a) Sorted prominences of Road2 data, where the ratio of the 5th (the prominence of the last detected peak) to the 6th (the greatest prominence of the noise) is 201%. (b) The detected peak points (each marked by its prominence ordinal number) and the supporting hills.



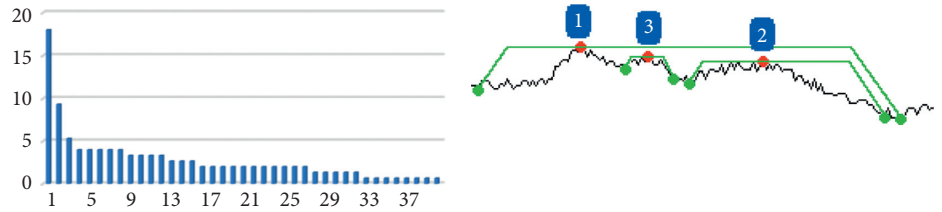
(a) (b)

Figure 11: (a) Sorted prominences of Road3 data, where the ratio of the second (the prominence of the last detected peak) to the third (the greatest prominence of the noise) is 250%. (b) The detected peak points (each marked by its prominence ordinal number) and the supporting hills.



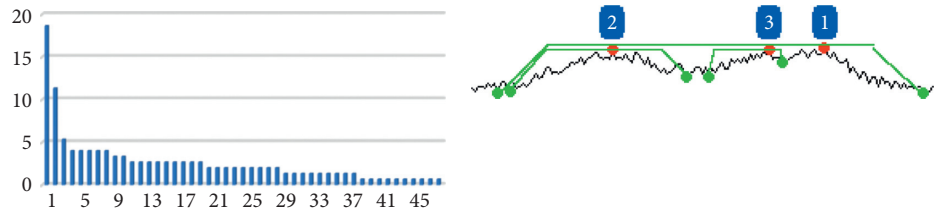
(a) (b)

Figure 12: (a) Sorted prominences of Road4 data, where the ratio of the second (the prominence of the last detected peak) to the third (the greatest prominence of the noise) is 194%. (b) The detected peak points (each marked by its prominence ordinal number) and the supporting hills.



(a) (b)

Figure 13: (a) Sorted prominences of Road5 data, where the ratio of the third (the prominence of the last detected peak) to the fourth (the greatest prominence of the noise) is 131%. (b) The detected peak points (each marked by its prominence ordinal number) and the supporting hills.

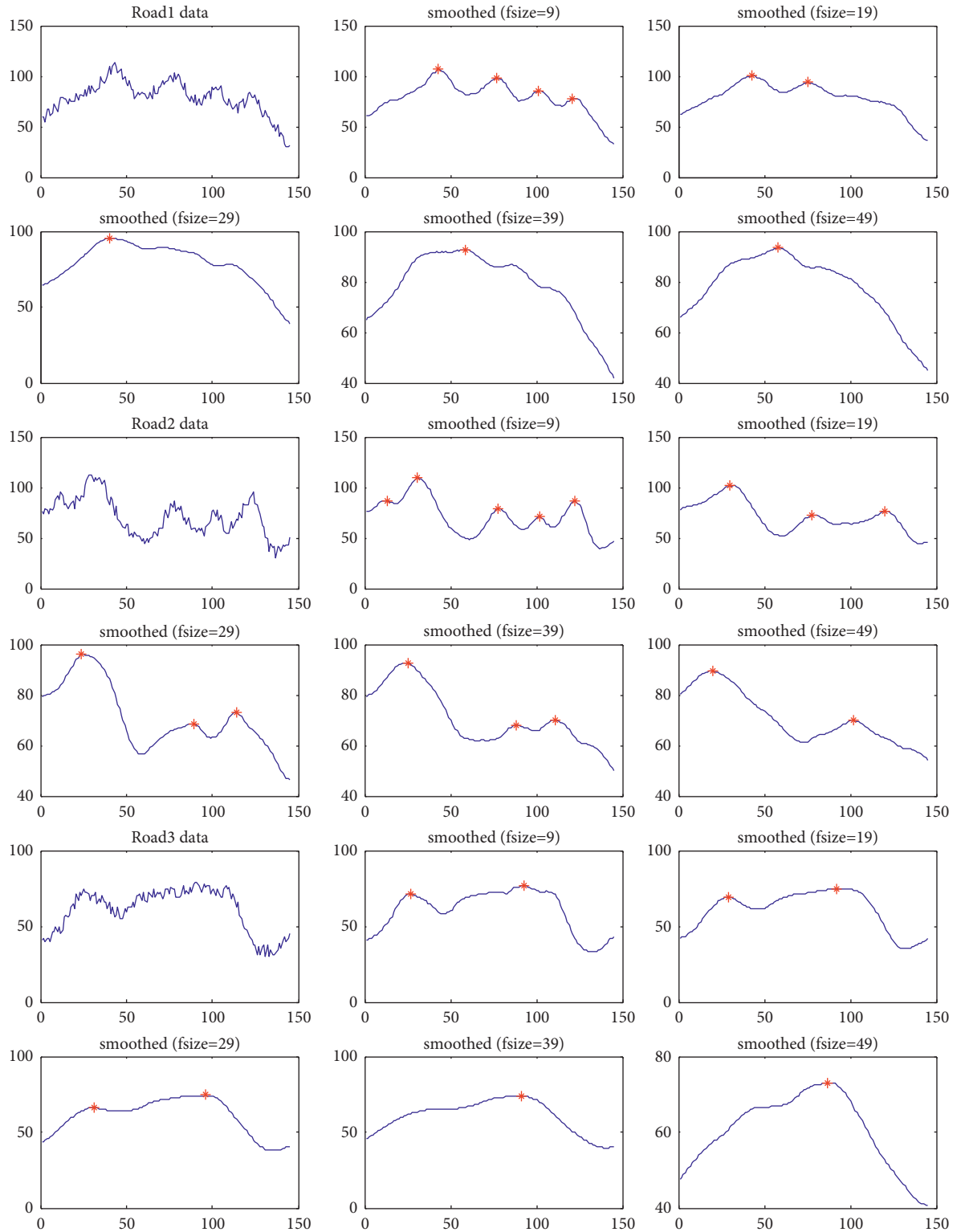


(a) (b)

Figure 14: (a) Sorted prominences of Road6 data, where the ratio of the third (the prominence of the last detected peak) to the fourth (the greatest prominence of the noise) is 132%. (b) The detected peak points (each marked by its prominence ordinal number) and the supporting hills.

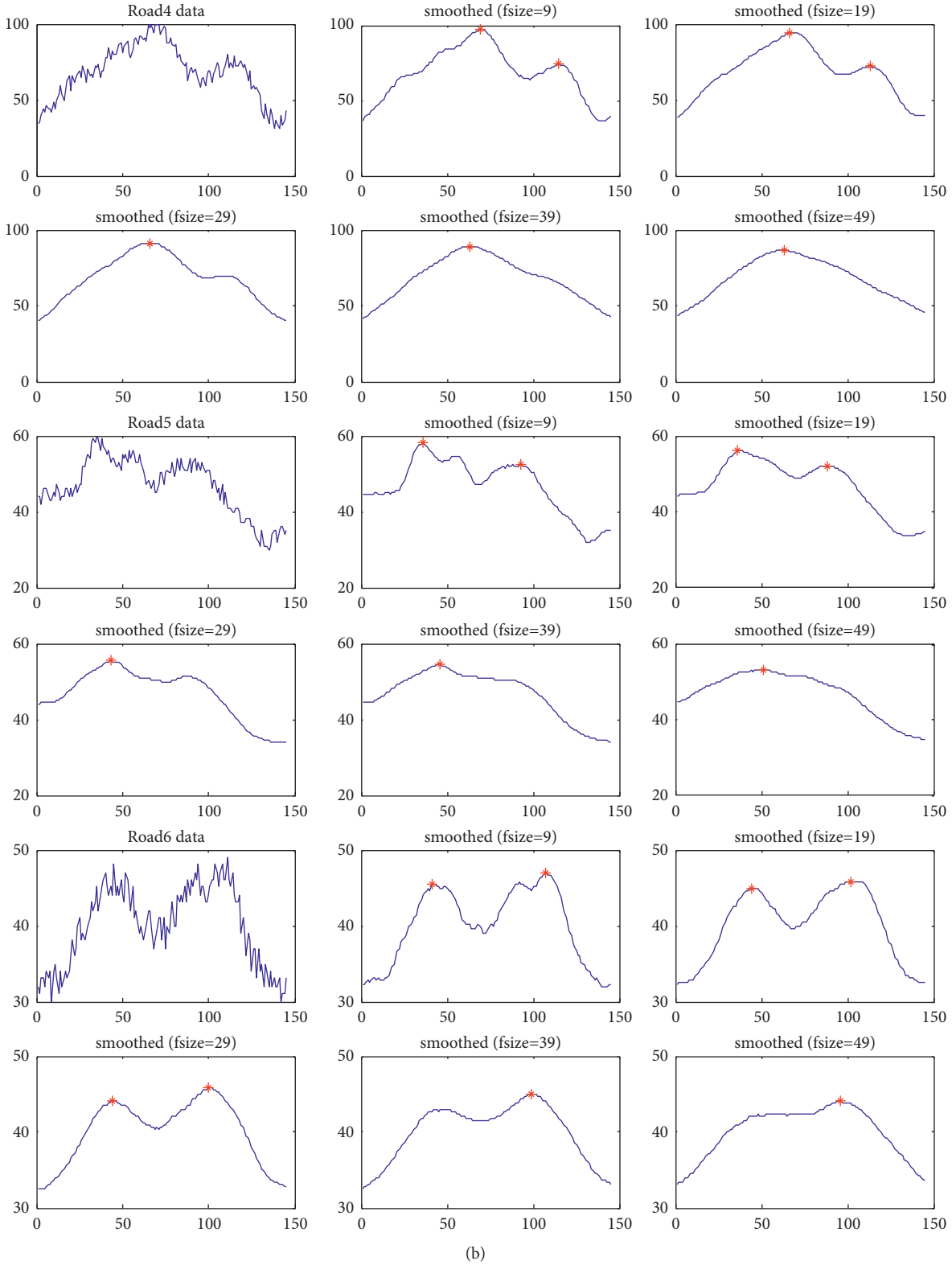
The noise in the 6 series has different scales, and we cannot use one uniform smoothing scale for all of them. The proposed method adapts itself to different noise scales by finding the bending dot in the sorted prominence curve, which is then separated into rapidly and slowly declining sections.

Wavelet smoothing transforms the data into wavelet space and thresholds small coefficients and performs the inverse wavelet transform to get the smoothed data. We used 3 wavelets, db4, db6, and sym5 in the experiments and report the results in Figures 18–20. In the experiments, the wavelet db4 failed on Road3 and Road4 data, db6 failed on Road2 and



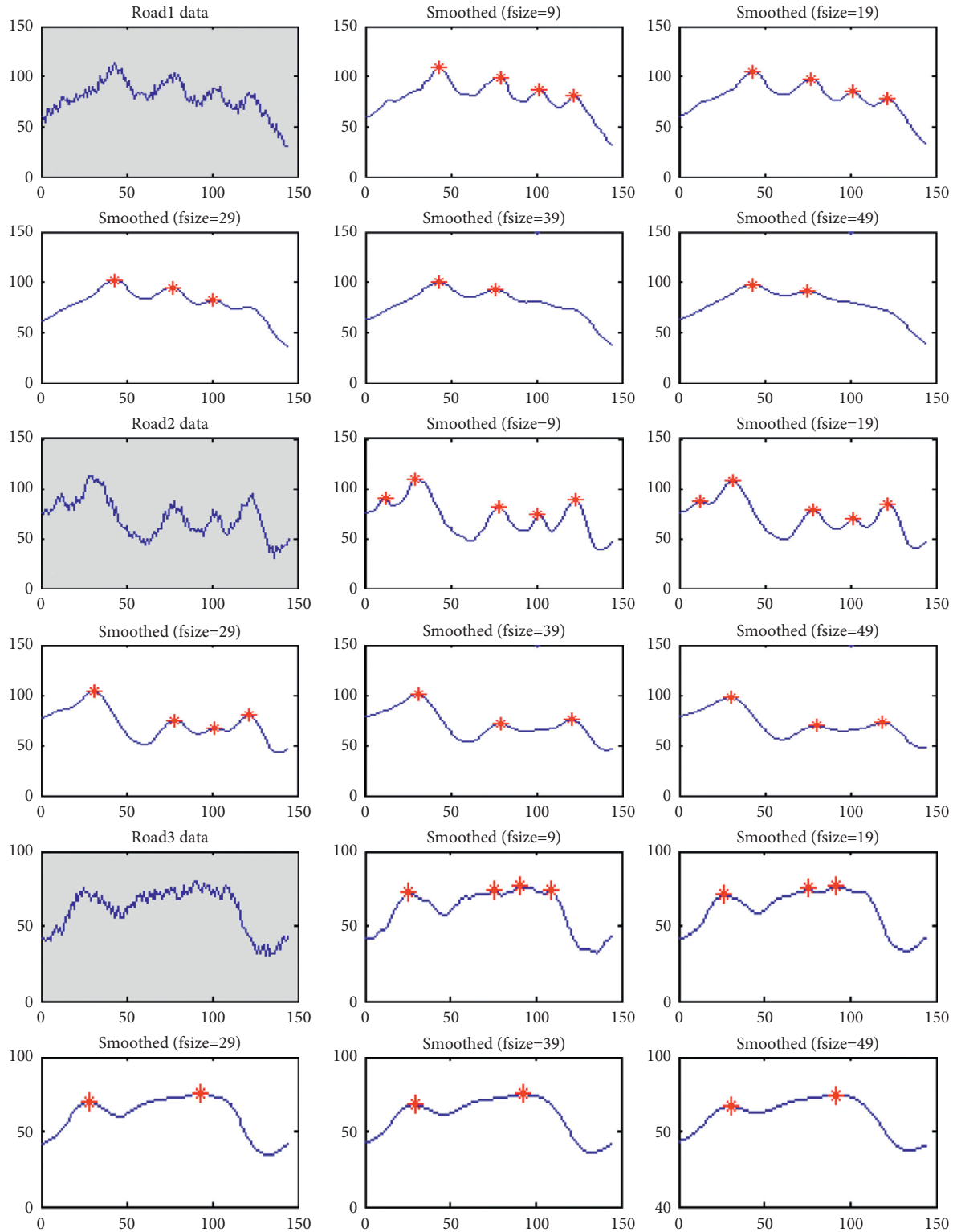
(a)

Figure 15: Continued.



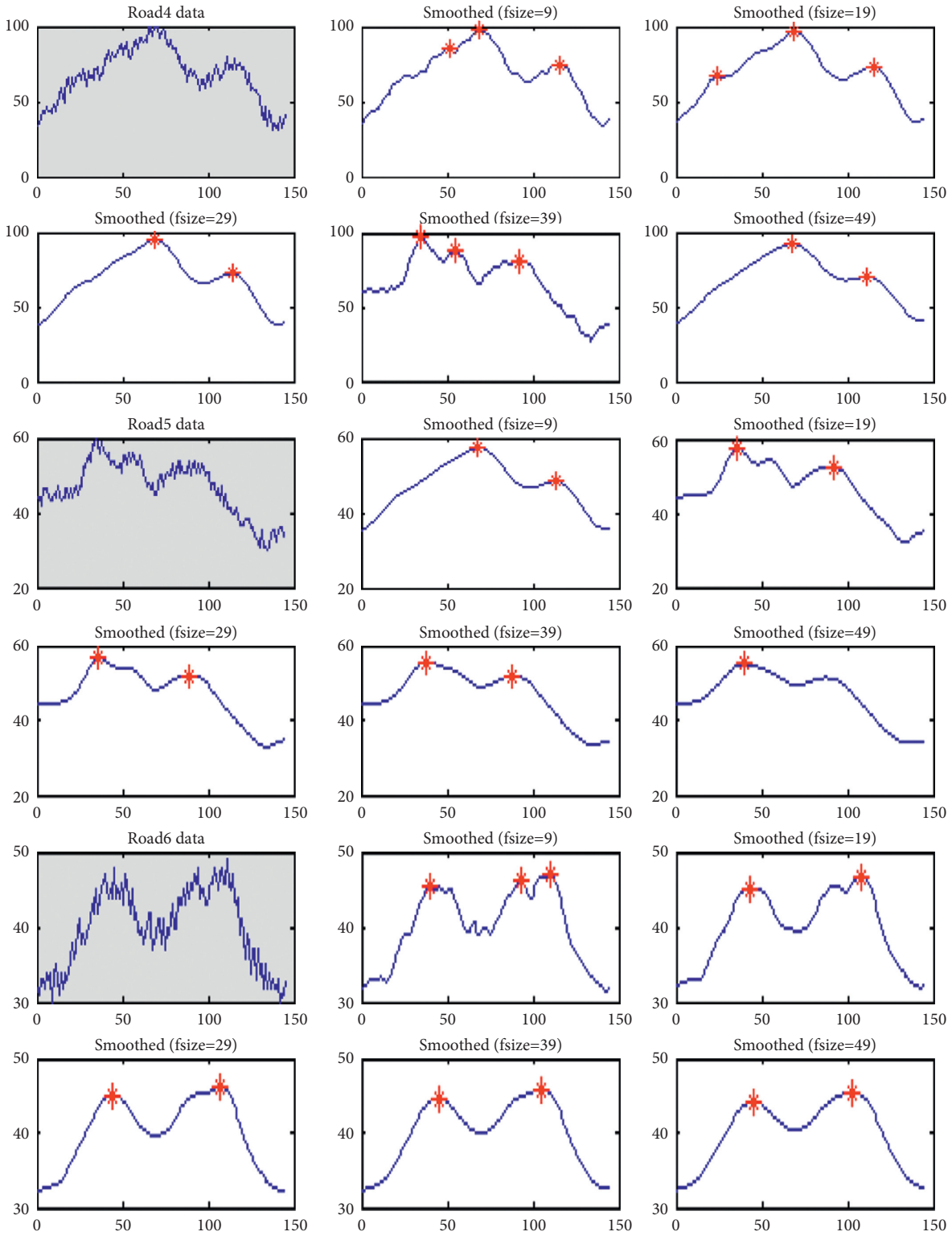
(b)

Figure 15: The data smoothed by the average filter and the peaks picked by Billauer's method on the smoothed data. For each time series, we list the original data, followed by the smoothed data with the 5 filter sizes (fsize). In this and the following figures, the x-coordinates represent the time in unit of 5 minutes.



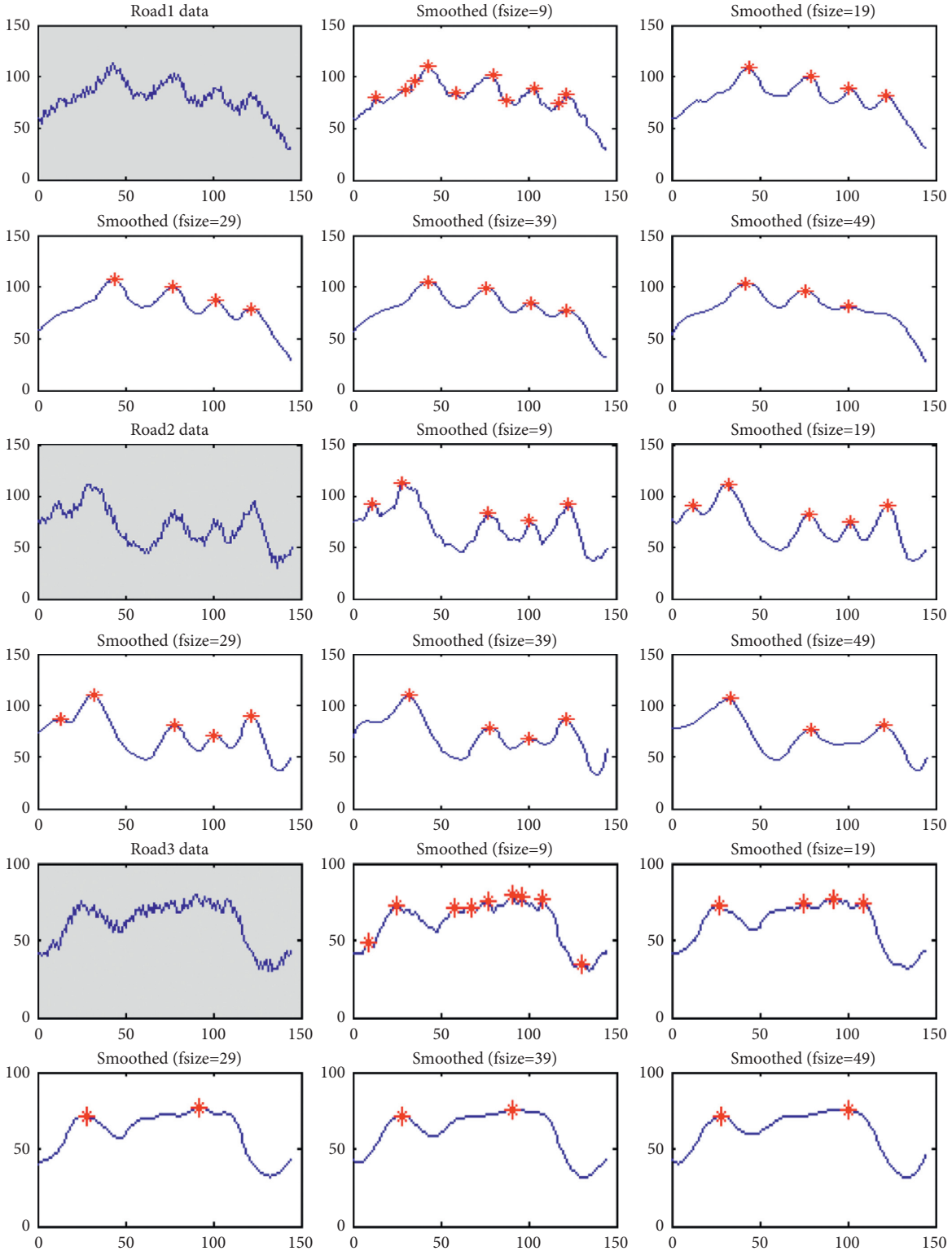
(a)

Figure 16: Continued.



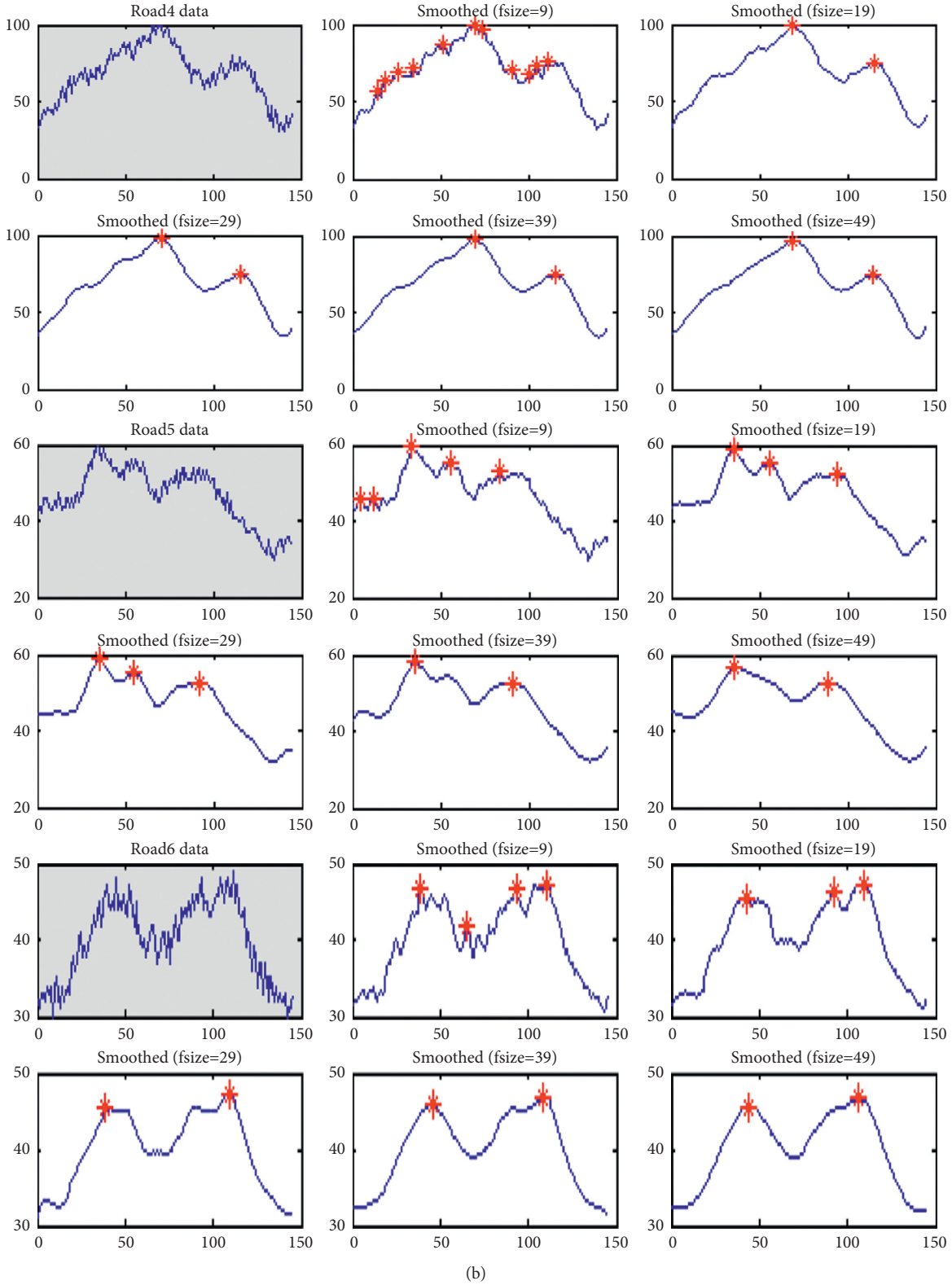
(b)

Figure 16: The data smoothed by Gaussian filter and the peaks picked.



(a)

Figure 17: Continued.



(b)

Figure 17: The data smoothed by Savitzky–Golay filter and the peaks picked.

Table 2: The results of Gaussian filter of different sizes.

Filter size	The roads where the Gaussian filter of the size detects the peaks successfully
9	Road1, Road2, Road5, Road6
19	Road1, Road2
29	Road3, Road4
39	Road3, Road4
49	Road3, Road4

Table 3: The results of Savitzky–Golay filter of different sizes.

Filter size	The roads where Savitzky–Golay filter of the size detects the peaks successfully
9	Road2
19	Road1, Road2, Road4, Road5, Road6
29	Road1, Road2, Road3, Road4, Road5
39	Road1, Road3, Road4
49	Road3, Road4

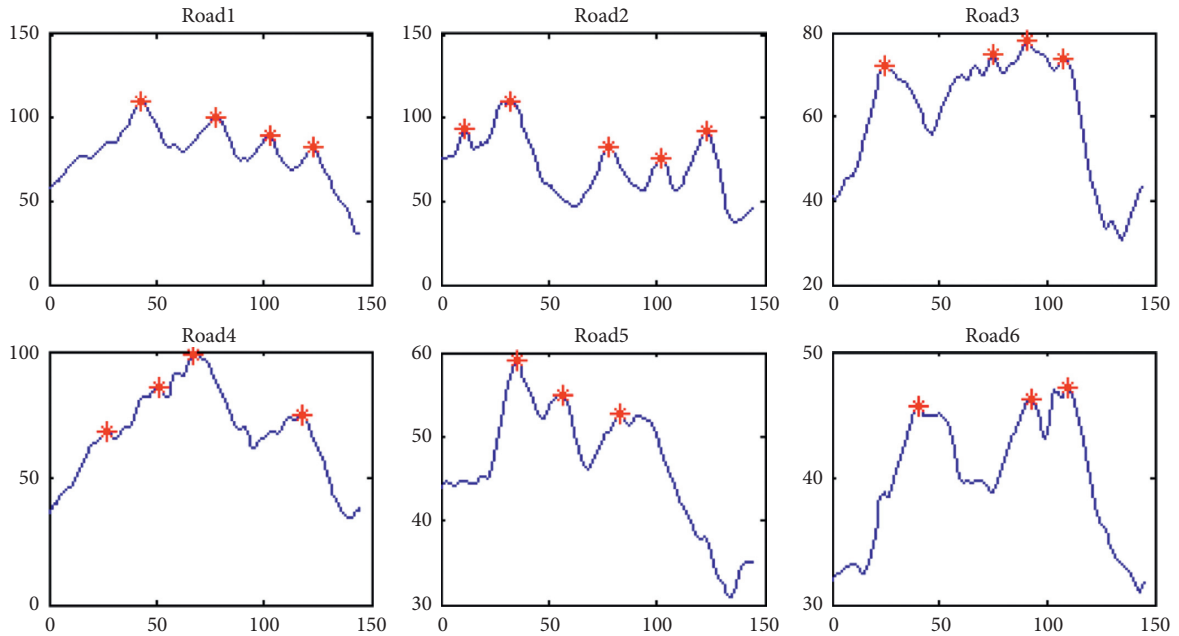


Figure 18: The data smoothed with the wavelet db4 and the peaks picked.

Road6, and sym5 failed on Road1, Road2, and Road6. Our method transforms the noisy data to sorted prominences, similar to a wavelet transform decomposing the signal spectrum into different frequency ranges. The experiments demonstrated the superiority of the sorted prominences to the wavelet transforms.

4.2. Quantitative Comparison. To compare quantitatively, we use one week (2018) daily data, from 12 midnight to 12 midnight, of different roadways, including 10 national highways, 10 provincial highways, and 10 urban roads. The data were sampled every 10 minutes, and each data series contains $24 * 6 = 144$ points.

For the average, Gaussian, and Savitzky–Golay smoothing methods, we used 20 filter sizes, 5, 7, 9, . . ., 43. For wavelet smoothing, we used 3 wavelets, db4, db6, and sym5. Sensitivity (SE), positive predictivity (PP), and accuracy (AC) were used as performance indices. For a method m with a parameter p on a series data d,

$$\begin{aligned} \mathrm{SE}(m,p,d) &= \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FN})}, \\ \mathrm{PP}(m,p,d) &= \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FP})}, \\ \mathrm{AC}(m,p,d) &= \frac{(\mathrm{SE}(m,p,d)+\mathrm{PP}(m,p,d))}{2}, \end{aligned} \tag{1}$$

where the true positive (TP) is the number of correctly detected peaks in the series, the false negative (FN) is the number of ground-truth peaks undetected, and false positive (FP) is the number of nonground-truth peaks detected as peaks.

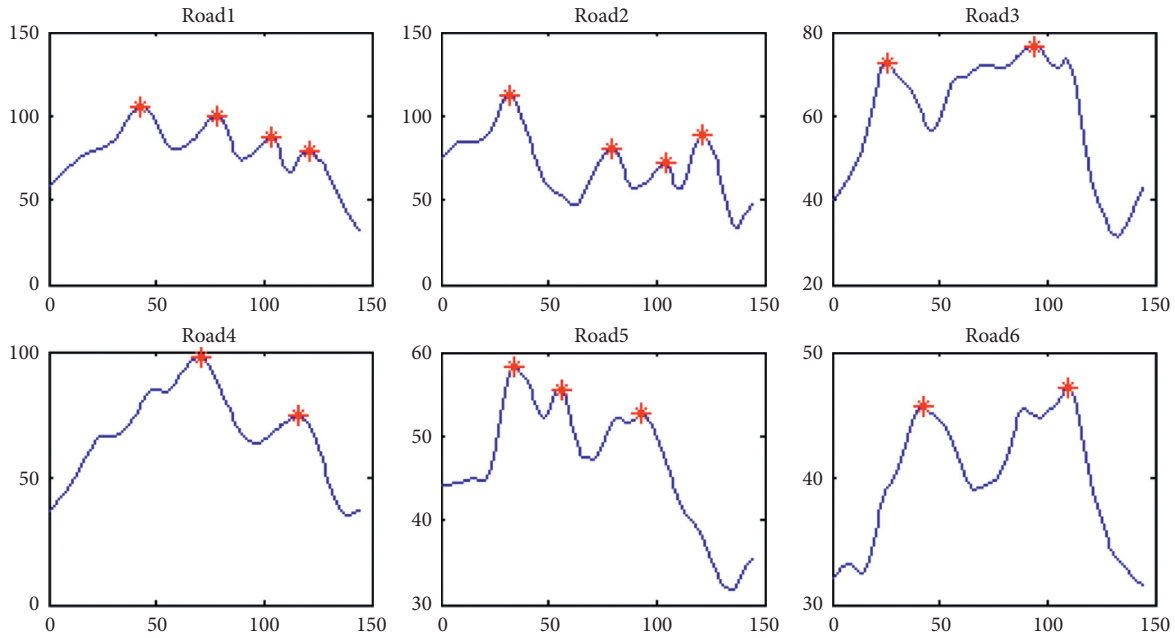


Figure 19: The data smoothed with the wavelet db6 and the peaks picked.

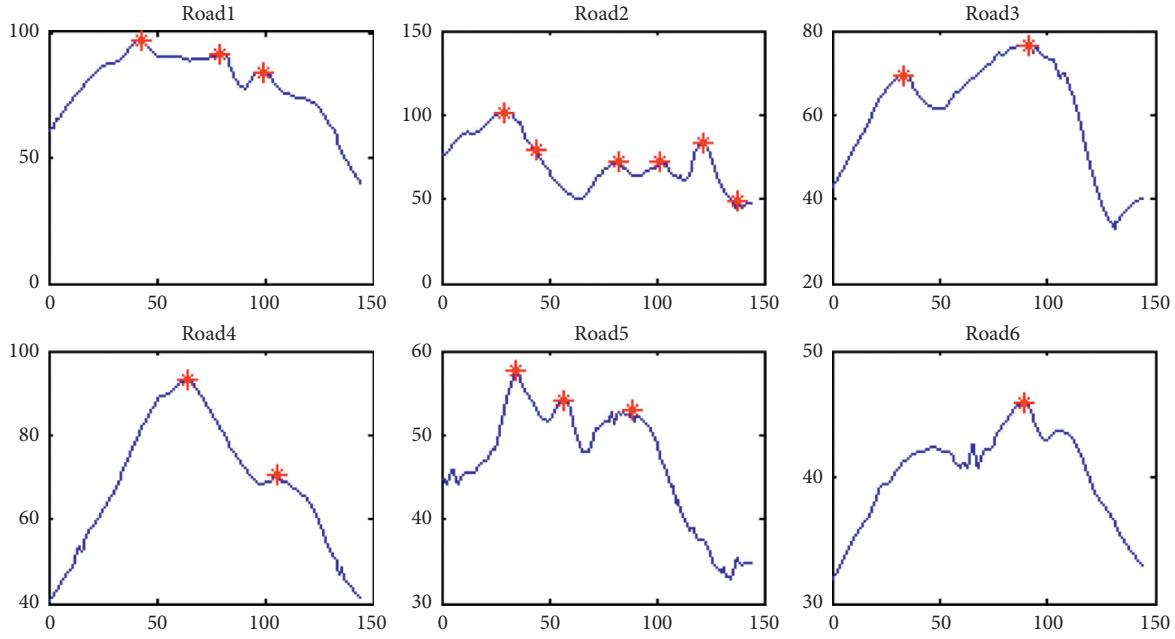


Figure 20: The data smoothed with the wavelet sym5 and the peaks picked.

Table 4: Comparing to the smoothing methods with a best parameter.

Method	SE (%)	PP (%)	AC (%)	Time (ms)
Average smoothing (fsize = 11)	81.4	73.4	77.4	34
Gaussian smoothing (fsize = 9)	82.7	74.2	78.4	50
Savitzky–Golay smoothing (fsize = 19)	85.8	69.1	77.5	39
Wavelet smoothing (with db4)	82.7	70.5	76.6	62
Voronoi diagram	96.8	86.4	91.6	42

Table 5: Comparing to the smoothing methods with an oracle.

Method	SE^o (%)	PP^o (%)	AC^o (%)
Average smoothing	93.4	89.6	91.5
Gaussian smoothing	94.9	88.9	91.9
Savitzky–Golay smoothing	95.5	89.3	92.4
Wavelet smoothing	91.1	86.6	88.9

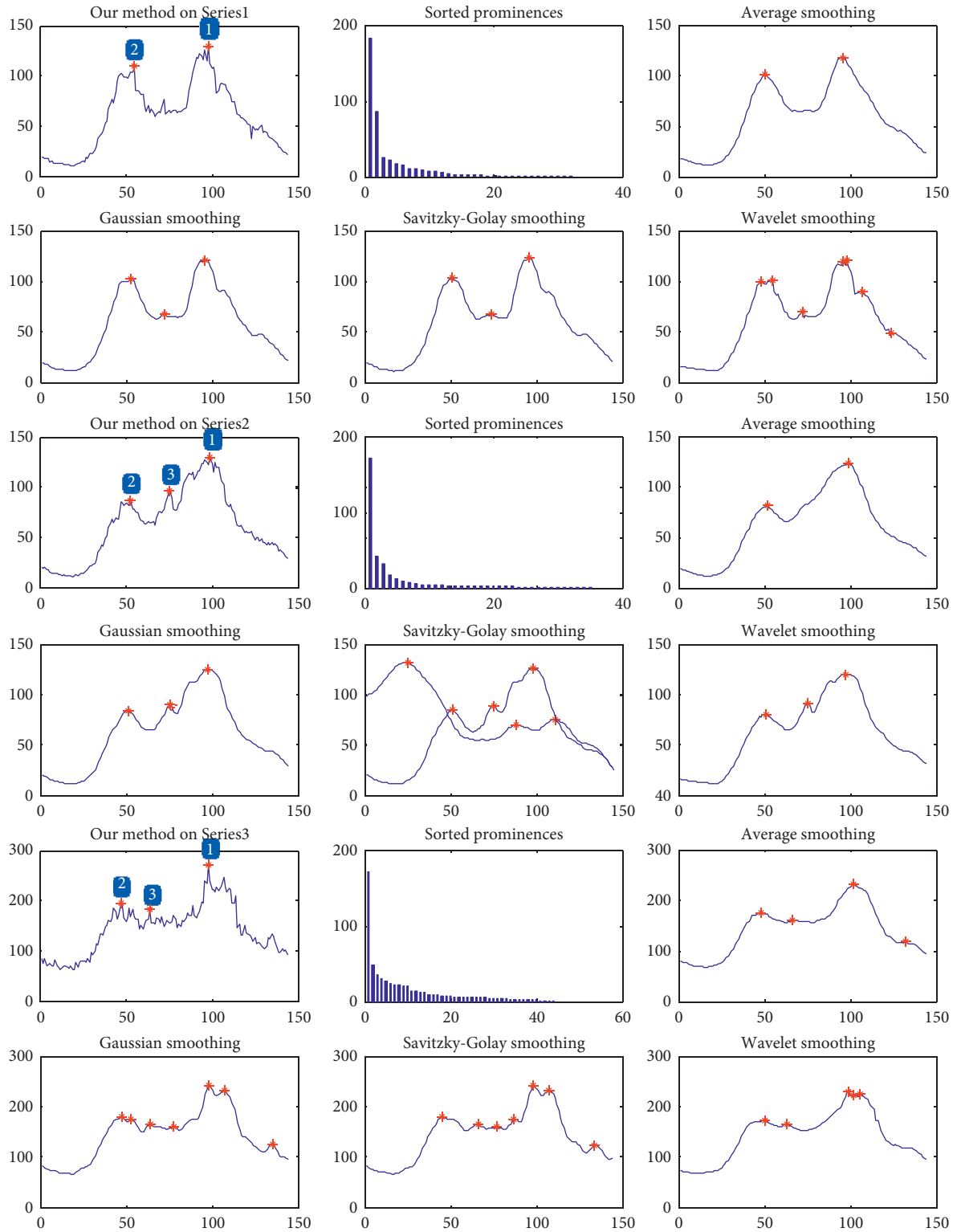


Figure 21: The detected peaks on 3 series by our method and the smoothing-based methods with a best parameter (as listed in Table 4). For each series, the sorted prominences are also depicted. The peaks by our method are marked and labeled on the raw data, and the peaks by the smoothing methods are marked on the smoothed data.

We computed the overall performance of a method m with a parameter p on all series as the average performance of m with p on all series:

$$\mathrm{SE}(m,p) = \frac{\left(\sum_{i=1}^{n} \mathrm{SE}(m,p,d_i)\right)}{n}, \tag{2}$$

$$\mathrm{PP}(m,p) = \frac{\left(\sum_{i=1}^{n} \mathrm{PP}(m,p,d_i)\right)}{n}, \tag{3}$$

$$\mathrm{AC}(m,p) = \frac{(\mathrm{SE}(m,p) + \mathrm{PP}(m,p))}{2}. \tag{4}$$

For each existing smoothing-based method, we choose a best parameter p such that the overall accuracy $\mathrm{AC}(m,p)$ is maximized. The overall performance of the 4 smoothing methods with a best parameter is compared with ours, as reported in Table 4. In the table, the best parameters are given in parentheses. In the comparison, our method outperformed the 4 others in sensitivity, positive predictivity, and accuracy. In Table 4 (last column), we also list the average computation time in milliseconds, which shows that the computation time of the proposed method is comparable to that of the smoothing-based methods.

To verify that the proposed method can effectively avoid the dilemma in choosing an appropriate smoothing scale, we extended the existing smoothing-based methods with an "oracle" which "knows" how to choose a best parameter for each series. We then compared the proposed method with the idealized smoothing-based methods. For a smoothing-based method on a series, we choose a parameter with the best accuracy, and let the SE and PP of the parameter be the SE and PP of the method on the series. We then compute the average of the SE and PP on all series, which are used as the overall SE and PP of the idealized method, respectively. Formally, for a method m and a series d_i, let $p(m,i)$ be a parameter p which maximizes the accuracy $\mathrm{AC}(m,p,d_i)$. Then, we define

$$\mathrm{SE}^o(m) = \frac{\left(\sum_{i=1}^{n} \mathrm{SE}(m,p(m,i),d_i)\right)}{n}, \tag{5}$$

$$\mathrm{PP}^o(m) = \frac{\left(\sum_{i=1}^{n} \mathrm{PP}(m,p(m,i),d_i)\right)}{n}, \tag{6}$$

$$\mathrm{AC}^o(m) = \frac{\left(\mathrm{SE}^o(m) + \mathrm{SE}^o(m)\right)}{2}. \tag{7}$$

Notice that the parameter p in formulas (5) and (6) is series-dependent as is different in formulas (2) and (3). The overall performance of the idealized smoothing methods is reported in Table 5, revealing that our method is comparable in accuracy to the smoothing-based methods with an oracle, which of course does not exist in reality. This comparison verifies again the effectiveness of the proposed method.

We select 3 series to discuss the strength and the weakness of the proposed method (Figure 21). The 3 series belong to one road on 3 different days. The road is a provincial 2-way 4-lane highway. In all 3 series, the first 2 peaks are visually salient and reported as detected peaks by all methods. In series 1, the 3rd peak (the thorn shape between the first 2 peaks, not labeled) has a much lower prominence than the first 2 and is correctly classified as not detected by our method and the average smoothing. In series 2, the 3rd peak has a prominence comparable to the 2nd peak and is correctly detected by all methods except the average smoothing. In series 3, the 3rd peak is a false positive though its prominence is also comparable to that of the 2nd peak. If we compare the 3rd peaks in series 2 and 3, we find that the peak in series 3 is sharper and covers a shorter time period, and it is more likely a noise or fluctuation. In this case, a simple shape-based outlier removal process can be applied to remove this false peak. In general cases, whether a salient sharp peak is a false peak cannot be determined solely by traffic volume. Other traffic parameters, such as speed, density, and headway, are needed. As a prestep, our method provides salient candidates for further comprehensive analysis where multiple traffic parameters are involved.

5. Conclusion

We introduced a traffic peak detection method based on Voronoi diagram. The method computes the prominence of candidate peak points and selects the salient peaks by finding the bending dot in the sorted prominence curve. It works directly on noisy data and avoids the difficulty in choosing an appropriate smoothing scale. The prominence of candidate peak points also offers the subsequent analysis step the flexibility to choose peaks at any scale.

The introduced method relies on the assumption that noise has smaller amplitude than the salient peaks. For data not satisfying the assumption, an outlier removal process is needed. Other future work includes improving the traffic network model by computational intelligence approaches, such as monarch butterfly optimization [28], earthworm optimization [29], and elephant herding optimization algorithms [30].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Our work was supported by Key R&D Project of Hebei Province 19211210D.

References

[1] N. Cohn, E. Kools, and P. Mieth, "The tomtom congestion index," *Its World Congress*, 2012.

[2] H. Zhao, J. Xia, F. Li, Z. Li, and Q. Li, "A peak traffic congestion prediction method based on bus driving time," *Entropy*, vol. 21, no. 7, p. 709, 2019.

[3] M. M. Shirmohammadi and M. Esmaeilpour, "The traffic congestion analysis using traffic congestion index and artificial neural network in main streets of electronic city (case study: Hamedan city)," *Programming and Computer Software*, vol. 46, no. 6, pp. 433–442, 2020.

[4] Y. Cheng, X. Ye, and Z. Wang, "A forecasting model of the proportion of peak-period boardings for urban mass transit system: a case study of osaka prefecture," in *Proceedings of the Transportation Research Board 95th Annual Meeting*, Washington, DC, USA, January 2016.

[5] A. Paricio and M. A. Lopez-Carmona, "Application of traffic weighted multi-map optimization strategies to traffic assignment," *IEEE Access*, vol. 9, pp. 28999–29019, 2021.

[6] G. Bianchin and F. Pasqualetti, "A network optimization framework for the analysis and control of traffic dynamics and intersection signaling," in *Proceedings of the 2018 IEEE Conference on Decision and Control (CDC)*, pp. 1017–1022, Miami, FL, USA, December 2018.

[7] G. Öttl, P. Böck, N. Werpup, and M. Schwarze, "Derivation of representative air traffic peaks as standard input for airport related simulation," *Journal of Air Transport Management*, vol. 28, no. Complete, pp. 31–39, 2013.

[8] Y. Ji, J. Wang, Y. Niu, and H. Ma, "Reliable event detection via multiple edge computing on streaming traffic social data," *IEEE Access*, p. 1, 2021.

[9] C. Lin, Y. Guo, W. Li, H. Liu, and D. Wu, "An automatic lane marking detection method with low-density roadside LiDAR data," *IEEE Sensors Journal*, vol. 21, no. 8, pp. 10029–10038, 2021.

[10] S. Ahmed, U. Kamal, and M. K. Hasan, "DFR-TSD: a deep learning based framework for robust traffic sign detection under challenging weather conditions," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2021.

[11] A. García Saravia Ortiz de Montellano, S. Hekker, and N. Themeßl, "Automated asteroseismic peak detections," *Monthly Notices of the Royal Astronomical Society*, vol. 476, no. 2, pp. 1470–1496, 2018.

[12] E. Corsaro, J. D. Ridder, and R. A. García, "Bayesian peak bagging analysis of 19 low-mass low-luminosity red giants observed with Kepler," *Astronomy & Astrophysics*, vol. 579, 2015.

[13] H. Y. Fu, J. W. Guo, Y. J. Yu et al., "A simple multi-scale gaussian smoothing-based strategy for automatic chromatographic peak extraction," *Journal of Chromatography A*, vol. 1452, 2016.

[14] Y. Zheng, R. Fan, C. Qiu, Z. Liu, and D. Tian, "An improved algorithm for peak detection in mass spectra based on continuous wavelet transform," *International Journal of Mass Spectrometry*, vol. 409, pp. 53–58, 2016.

[15] L. D. Sharma and R. K. Sunkaria, "Novel T-wave detection technique with minimal processing and RR-Interval based enhanced efficiency," *Cardiovascular Engineering and Technology*, vol. 10, no. 2, pp. 367–379, 2019.

[16] L. Miran, P. Dajeong, S. Y. Dong, and I. Youn, "A novel r peak detection method for mobile environments," *IEEE Access*, vol. 6, pp. 51227–51237, 2018.

[17] P. Cortés, J. R. Fernández, J. Guadix, and J. Muñuzuri, "Fuzzy logic based controller for peak traffic detection in elevator systems," *Journal of Computational & Theoretical Nanoscience*, vol. 9, no. 2, pp. 310–318, 2012.

[18] E. Billauer, *Peak Detection Using Matlab*, Eli Billauers Home Page, 2012, http://www.billauer.co.il/peakdet.html.

[19] D. J. Kelley, T. R. Oakes, L. L. Greischar et al., "Automatic physiological waveform processing for fMRI noise correction and analysis," *PLoS One*, vol. 3, no. 3, Article ID e1751, 2008.

[20] Y. J. Yu, Q. L. Xia, S. Wang et al., "Chemometric strategy for automatic chromatographic peak detection and background drift correction in chromatographic data," *Journal of Chromatography A*, vol. 1359, pp. 262–270, 2014.

[21] J. Lu, M. J. Trnka, S. H. Roh et al., "Improved peak detection and deconvolution of native electrospray mass spectra from large protein complexes," *Journal of the American Society for Mass Spectrometry*, vol. 26, no. 12, pp. 2141–2151, 2015.

[22] L. Fogelstaller, "Traffic network optimization, an approach combining genetic algorithms and nonlinear programming," Master's thesis, Technische Universität München, Munich, Germany, 2014.

[23] T. Mao, A. S. Mihaita, F. Chen, and H. L. Vu, "Boosted genetic algorithm using machine learning for traffic control optimization," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–30, 2021.

[24] D. Zhao, Z. Yu, Y. Yao, and S. Du, "A novel peak detection method of structured light stripes for 3D reconstruction," in *Proceedings of the International Conference on Intelligent Human-Machine Systems and Cybernetics*, pp. 43–46, Hangzhou, China, August 2011.

[25] P. Lu, M. J. Fan, Q. Zhang et al., "A novel strategy for extracted ion chromatogram extraction to improve peak detection in UPLC-HRMS," *Analytical Methods*, vol. 10, no. 42, 2018.

[26] F. Aurenhammer, "Voronoi diagrams-a survey of a fundamental geometric data structure," *ACM Computing Surveys*, vol. 23, no. 3, pp. 345–405, 1991.

[27] The Boost. Polygon library https://www.boost.org/doc/libs/1_57_0/libs/polygon/doc/index.htm.

[28] Y. Feng, S. Deb, G. G. Wang, and A. H. Alavi, "Monarch butterfly optimization: a comprehensive review," *Expert Systems with Applications*, vol. 168, 2020.

[29] G. G. Wang, S. Deb, and L. D. S. Coelho, "Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems," *International Journal of Bio-Inspired Computation*, vol. 12, no. 1, p. 1, 2018.

[30] W. Li and G. G. Wang, "Elephant herding optimization using dynamic topology and biogeography-based optimization based on learning for numerical optimization," *Engineering with Computers*, vol. 168, no. 1, 2021.